UNIVERSITY OF OSLO
Department of Informatics

Biometric Authentication and Identification using Keystroke Dynamics with Alert Levels

Master thesis

Aleksander Sveløkken Andersen
Oslo University College
AleksanderS.Andersen@stud.iu.hio.no

May 23, 2007

Abstract

Proper authentication is needed to avoid theft and destruction of data by unauthorized parties. This research proposes to add software biometrics to the authentication and verification process, using keystroke dynamics. The goal is to grant or revoke users privileges by distinguishing one user's typing pattern from another. Login samples were recorded by monitoring keystroke parameters such as *KeyDown* time and *KeyWait* time, in an 8 character password.

A system to generate user profiles, with the ability to accommodate continuous growth, was developed. By using appropriate *Alert Levels*, a 2.25% False Acceptance Rate and 4.17% False Rejection Rate was achieved. A method to recognize and identify users, based on one login sample, was able to trace 65% of the samples back to the original user. The work in this thesis was unable to find an applicable method for statistical pattern recognition. It concludes that by enabling a biometric keystroke dynamic authentication system, 97.75% of all false login attempts, with stolen but valid credentials, can be prevented, although with a potential downside of increasing the number of falsely rejected logins by 4.17%. However, as users grow accustomed to their password, the false rejection rate will go down and the system will increase in reliability. Password DoS attacks as well as automated dictionary attacks will be prevented, but a potential increase in administration cost is probable because of altered user behavior due to physical or environmental changes.

Acknowledgements

To my *mother*; for making me believe in the Lock Ness monster.

To my *father*; who taught me how to play chess at an early age, and not letting me win.

To my *sister*; for constantly reminding me of the value and importance of honesty and justice.

To my *brother*; for always challenging my every idea. for pushing me to improve. and for everything unsaid but recognized between us.

You have all made a huge impact on who I am today, and for that;
I am forever grateful.

- *To Simen Hagen*- I would like to thank my adviser for his interest, passion and dedication to my research. He has given me praise when praise was needed, and criticism, when appropriate. He gave inspiration when I had none, and tremendous motivation for improving and living up to my potential. I especially want to thank him for opening doors, physically and mentally, when I was seeing only a wall. I want to express my gratitude for all the time he set off for me, for his constructive thinking, for listening to my ideas, for always keeping an open door, but above all; for not letting me travel astray. His calmness, humor and casual attitude has been just what I needed. He understands people, in a way I hope to someday do.

- *To Mari Mehlen*- who has given me so much more than I deserve, in terms of time and information. Her input on the statistical aspects of my work was invaluable to me.

- *To Kyrre Begnum*- for without knowing, leading me to my thesis idea during his lectures. For proving to me research could be fun, all depending on the perspective. And for making me believe I could walk my own way.

- *To Mark Burgess*- for being an endless source of inspiration. He has meant more than he will ever know.

- *To Asbjørn, Kristoffer, Terje, Stein Erik, Stephan and Øyvind*- Your help, ideas and constructive feedback has been extremely useful. It has made my work better in many many ways. Thank you for being my friends.

- *To Matt, Edson and Ingard*- You have all become very dear to me. It has been a great pleasure to study with you. I will miss the discussions in and outside the Master-room. I hope we will continue those after we finish. From being total strangers, you now add sunshine to every single day of my life. I only hope I can give you, what you have given me.

- *To my Guinea Pigs*- Without your eager participation, patience and willingness to help, needlessly to say, none of this work would have been possible. Unfortunately I cannot acknowledge you by name, but I will not forget. Thank you all.

Preface

A paper titled "Using Alert Levels to enhance Keystroke Dynamic Authentication" based on this thesis is currently being written. It will be submitted to NIK (Norsk Informatikkonferanse) by 14th of June 2007. The conference will be held in Oslo 19th to 21st of November 2007.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Authentication of users have been conducted for as long as control of access has been needed. The use of computers and computer systems has been rapidly growing over the last decade, and the need for proper authentication and identification has increased accordingly. Every day vast amounts of sensitive data is maintained by, and transmitted between computer systems. Protection and confidentiality has become more and more important. Compromised user accounts lead to unwanted unauthorized access. The Gartner IT Security Summit 2006 raises insider threats from employees and partners as the number one security threat. Increased focus has been dedicated to attacks coming from inside the organization; behind security features like firewalls and intrusion detection systems. It has been shown that attacks occurring from inside the layers of security protection, are the ones causing highest financial damage. Internal threats, whether they are malicious or simply the result of human error, are the hardest to detect, most financially damaging and the most commonly occurring type of security violation [2].

Internal intruders often have some authority, but seek to gain additional access, or to take action without legitimate authorization [3]. This implies the usage of non-acceptable or uncommon use of privileges compared to the historical profile of the user or the users group/position. Spoofing attacks, password stealing or session high-jacking are all examples of authorization misuse.

Authentication is the process of validating a user (for example verifying the users identity), in order to set privileges based on a policy. This process is often conducted before a session is created, using a user-name 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. This is a problem [4].

1.1 Motivation

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. The trust relationship between a user and a system is based on the selecting the appropriate username, assuming that the password is secret.

When selecting a password policy both security and user convenience is taken into account due to several issues. A password can be guessed and thus enabling unauthorized access. There are features created to face these issues (for example locking access after 3 failed attempts, short password expiration time, single-use passwords and password restrictions). Generating passwords or implementing policies which makes the password hard to remember can lead to a security breach [5, 6], as users choose convenience over security (for example putting the password on a sticker easily available) [7, 6].

There are multiple techniques an attacker could use to gain a password.

- Encrypting a password is a feature used to prevent network password sniffing, where malicious users take use of tools to look at packet content in hope of finding passwords in clear text sent between host and authenticator [8].
- Even encrypted passwords is exploited by attackers using password cracking techniques, such as dictionary attacks to generate encrypted code and in so finding the password. Weak passwords suffer from successful dictionary attacks.

- Brute force attacks, where the attacker tries every possible password, is time consuming, but will inevitably succeed in finding the password.
- Social engineering is when an attacker exploit human beings into giving a password on false basis. For example posing as someone else over the phone or e-mail.

Authentication through user-password schemes is widely used and has a high security breach potential. Biometric methods using unique personal features have been implemented to strengthen the process. In the area of physical access control, biometrics have been found to be successful. Biometric systems (such as iris scans, fingerprints, voice recognition, physical chips or cards, blood samples and face recognition) have been developed with promise of closing this security gap. Some of the problems these technologies have faced are scalability, deployability and cost. A fair bit of administration is also included in such solutions. Electronic access control have not been as successful as physical access control using biometrics.

When a session is successfully authenticated most systems have no approach to re-authenticate a user. This enables scenarios like session highjacking and perimeter possession (for example taking over a terminal which has been authenticated). The trust in the user knowing the password does not catch a password theft scenario. As the threat of inside attacks (users already falsely authenticated) increase the demand for better and more secure techniques during and after authentication is sought after.

Using biometrics to add an extra layer to the existing security setup, may catch anomalous behavior in situations where one previously had no means of discovery. Issues like session highjacking, password theft and perimeter possession, are all such threats. Research presented to affront especially scalability, but also cost, include (but does not exclude to), software biometric systems. The idea is to find specific user patterns through machine learning processes, and add this to a verification process. Keystroke dynamics is one area where research has been showing positive results. It could be one way to accurately identify or verify users, using statistical methods to compare the typing pattern of an authenticating user with a known user profile. Keystroke dynamics could be beneficial in user identification during sessions, not only under a login sequence. Studies indicate that users given proper data, could be accurately identified without the use of username and passwords.

This study makes an effort in using keystroke dynamic in the phases of authentication and verification of users. It also makes a comparison to previous research.

1.2 Goals

This thesis will focus on keystroke dynamics as a feasible feature in the existing security policy by using techniques to identify users in an authentication process to harden passwords and reduce security breaches. The next step would be to re-authenticate a user during an authenticated session, determining if the user behavior is likely to belong to the authenticated user. Finally to make a good identification of the real user falsely authenticated through another account.

Using a combination of elements in keystroke authentication through properties like active and passive listening, keystroke latency, known keystroke combination times and general typing behavior. Creating user profiles on single stroke, password input phase (*content known typing pattern*) and free text typing, enables statistical studies making joint parameter user identification to give accurate detection rates.

1.3 Hypothesis

- Every user has a specific typing pattern that can be profiled and used for comparison to strengthen authentication

- The authentication process could be implemented using biometrics: *It is not what you do, but how you do it, that defines who you are*

- Keystroke dynamics can be used to strengthen the authentication of user login sequences

- Users conducting out-of-profile, can be detected and identified during authentication, using profile mining techniques

1.4 Contribution

The novelty of this work consists of several parts. The parameter KeyWait is the time span between one key is pressed down, until the next key is pressed down. This makes it impossible to get negative numbers, and will be more stable and easier to work with then earlier definitions of the waiting time between keys in related work.

One key aspect of this work is the usage of *Alert Levels*, which enables administrators of the system to adjust the strictness by altering one parameter setting. As profiles grow or users get more consistent, the *Alert Level* can be decreased, making the implementation flexible and able to more easily handle anomalies in user behavior. No previous work known to the author have had a flexible, and granular, solution at this level using keystroke dynamics.

1.5 Document Structure

This document will be structured as follows: The *Background* chapter contains previous related work, a literature survey of the topic, and explanations of the terminology. The *Method* chapter typically describe the setup of tests conducted, as well as explaining why solutions were chosen or discarded. It also contains sections referring to the application, the parameters and profiling. The *Results* chapter go through the steps taken in the tests and alterations done in order to improve results.

This document will discuss and conclude the findings in the *Conclusion* chapter, discussing interesting elements in the findings and suggests some future work. Hereafter follows an *Appendix* and *Bibliography*.

Chapter 2

Background

The growth of the Internet has changed computer confidentiality, integrity and availability severely, as the number of users and amount of sensitive data has grown rapidly. Areas such as authentication, cryptography and general security has received an increasing amount of scientific focus to address these matters.

2.1 Terms

The section A in the Appendix presents an overview of technical terms for computer security and biometrics used in this research and the referenced papers. The section 4.8.3 define result parameters used for describing results and accuracy of tests.

2.2 Classification - Security Measures

Dieter Gollmann defines integrity as: “No user of the system, even if authorized, may be permitted to modify data items in such a way that assets or accounting records are lost or corrupted”. and “prevention of unauthorized disclosure of information”, as the definition of confidentiality [3].

These basic objects were all founding stones in the early childhood of Intrusion

Detection first introduced by Denning [9] in 1985. It is desirable for administrators to monitor the activity in their networks, to detect and counteract unauthorized activity, but due to the burden of logfile and auditdata file sizes, automated monitoring system tools emerged. Tools designed to fight these issues are referred to as Intrusion Detection Systems(IDS). IDS can be separated into two categories: Host Intrusion Detection Systems (HIDS) and Network Intrusion Detection Systems (NIDS).

2.2.1 Intrusion Detection, Anomaly Detection and Authentication

There are many types of NIDS, but they generally focus on packets pattern recognition, either in packet headers, packet data or packet inter-arrival time. The firewall is a general term used for this kind of an easy NIDS, however a firewall differs as it does not analyze log-files, but does run actively while packets arrive.

HIDS have recently been given increasing emphasis due to the growth of inside network threats [10] and unauthorized activity from internal intruders such as disgruntled employees, people abusing their privileges for personal gain and various types of industrial espionage. With External Intruders one often refer to users with no authorized access to the system they attack. Internal intruders have some authority (either physically or in the system), but seek to gain additional ability to take action without legitimate authorization.

HIDS faces the challenge of detecting abnormal system behavior based on prior knowledge of system behavior/activity. Within a host-based Intrusion System a distinction can be made between approaches that model human behavior versus indirect interaction. The interaction uses have with the host can be monitored and profiled in order to re-authenticate a user. Especially important and useful in situations where abnormal user behavior and/or activity can point toward a security breach.

IDS are countermeasures assigned to protect and recover your assets and/or recover from the hazardous and vulnerable state. Anomaly detection will be referred to as a method to detect not nominal or normal behavior of traffic [11].

2.2.2 Authentication and Verification

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. Introducing methods to authenticate users after an authentication process has been successfully made, is called re-authentication.

Situations where, for some reason, authentication has been given to someone else than assumed (for example password hack, session hijack or console/perimeter hijack), a re-authentication system would ideally detect a different user pattern than expected from the user. If the deviation is strong enough, reactions to this can be put into place, such as forced logout, quarantine user or privilege removal.

IDS techniques are Anomaly based detection / Statistical Approach and Signature based detection / Specification approach [12, 13, 14]. They differ as anomaly detection systems have the promise to detect new kind of attacks, while signature based recognizes already identified treats.

Profiles can be built (or grown) through a machine learning process, and current behavior is used to compare against the defined behavior in this profile.

2.3 Historical Results

Nulluri and Kar [13] define the four objectives of a IDS as *Monitor and Analyze*, *Recognize*, *Compare data* and *Raise alerts*. Historically the second objective has been the primary attribute of research. Another vital motivation for research is the ability to reduce the amount of false positives and false negatives. A false positive is a raised alert which turns out to be legitimate behavior, and a false negative is illegal behavior which is not detected. The high rate of administration follows an implementation of a IDS[15, 16, 17, 9], problems with a high rate of false positives has also been mentioned by many [18, 13, 10, 19]. Implementing a behavior re-authentication model into the IDS, may ease the statistical uncertainly and reduce the false positive alerts. One goal of such systems come from biometrics where the hypothesis is; *it is not what you do, rather how you do it, that defines who you are*.

Authentication has historically been conducted using biometrics by the human eye, even if technology have not been available for long. Biometrics are commonly referred to as: *physical traits and behavioral characteristics that makes each of us unique*. Biometrical technologies has seen an increasing popularity to provide an extra security level to existing facilities.

The known existing verification models can be categorized as [4]:

(a) know: something a person knows (a password)

(b) have: something a person possesses (an id card)

(c) are: unique characteristics of a person (biometrics)

Because of cost, (a) and (b) have grown to be quite superior techniques of authentication, but it has been facing some vast weaknesses. The measures falling under the category (a) and (b) has weaknesses in the fact that possession or knowledge may be compromised without discovery. Spoofing, session hijacking, password cracking and theft of identification cards are all well known problems. Because of the increasing trend, the old class (c) biometric has received more interest. But there are some problems related to category (c) as well. The theft of a personal characteristic like a finger or an eye, are the most extreme examples. And the fact that most biometrics have some failure in enrollment, where samples are invalid. These systems are known as behavioral authentication systems, where the ideas were first introduced by Denning in 1985[9]. Their promise is to solve the authentication of users after a valid first authentication process has been successfully completed with access into a system.

Multilevel Security (or multi-level security - MLS), makes an effort to combine multiple factors from categories (a), (b) and (c), to prevent users from obtaining access to information or resources in which they lack authorization. Some systems require high security clearances, or have need for a high degree security. Examples are; online banking services and military systems. Most usually is combining category (b) or (c), with category (a).

2.3.1 Misuse Detection

Snort[20, 18, 13] is a pattern and signature tool with downloadable and re-sharable rule sets for new attacks, popular as a misuse pattern matching tools in small networks. It can perform tasks including content matching, protocol flag analysis and can also be used to detect a variety of attacks and probes (For example buffer overflow attacks, stealth and open port scans and OS fingerprinting attempts, to name but a few.). As it can be setup to do runtime tests, it is frequently used intrusion prevention purposes, by dropping previous illegal attacks on a system.

The snort community is providing newfound rule sets and patterns to stay ahead of new attack types according to [14]. The misuse detection and pattern matching approach to ID is unable to catch new attacks, and is in so failing to comply as profile machine learning technique.

Distributed collectiveness of rule making has made this solution extremely popular, even though the process of mining is resource consuming. Especially the dynamic firewall rule interaction implementation has made such systems very usable[18].

2.3.2 Host Based Intrusion Detection Systems

The importance of detecting abnormal activity on single computers or servers grows daily, as attackers get more cunning and professional. Studies have proved that almost 80% of all companies experienced some sort of inside attack[17]. This indicates a growing trend, which must be handled by the ID systems.

Current Host-based Intrusion Detection systems can be categorized under four areas of workings [21]:

(a) filesystem monitoring

(b) logfile analysis

(c) connection analysis

(d) kernel-based intrusion detection

Filesystem monitoring (a) basically deals with file and directory integrity checking for illegal creation, deletion and/or alteration of the monitored objects, changed permissions, owner and group. When detection abbreviates from the original stored data, it raises alerts. Well known tools are Tripwire [22], AIDE [23] and MTree. Some of the drawbacks however are lack of dynamics as tests are often ran at scheduled times, a high false positive rate without explicit and well planned schemes (a new installation of an application can create a huge numbers of alerts), attackers would also be able to cover their tracks before the comparison takes place. Never the less, this kind of IDS is particularly good at picking up most changes done on the filesystem and detecting unwanted activity.

Logfile analysis (b) is effectively used on more or less static logs, having a built-in pattern matching criteria, which can generate real-time alerts when abnormalities occur.

In connection analysis (c), applications such as PortSentry[24] listens for illegal traffic coming to the host (such as attempts to connect to unwanted services), illegal ports, portscan detection and port bindings. These tools usually have built in firewall support, to dynamically block traffic, found to be illegal. There can be a high administration cost to these systems to keep up to date, and also manage rare, but not illegal, traffic which theoretically could be blocked.

The last approach (d) is kernel-based intrusion detection which includes aspects such as users system usage, user system calls (order and argument based), kernel model loading and unloading prevention (and more). The major drawback for these kinds of systems is the high administration cost.

2.3.3 Data Mining

LaPadulla [25] found that scalability was one of the aspects of available tools that had improved most with the Sensor/Agent/Director architecture (the sensor gathers information, agent filters and reports to the director which make the decisions). Having multiple sensors and local agents reporting to a central director service auditing data, seems to be the most common approach to large scale data collection. LaPadulla concluded that even WAN scanners were now available to provide a wide data collection platform. However he also points out that the lack of traceback functionality these tools provide, makes it almost impossible to identify the origin of the attack. Bala et. al [26] has developed a state-of-the- art distributed data collection with inter-agent communications, in so using TCPDump to collect large-scale datasets in which to build profiles and more or less dynamically search for attacks. The false positive rate was however fairly high. LaPadullas Anomaly Detection survey anno 1999 [25] concludes the trend of going primary from network and system monitoring capabilities, into including more and more AD functionality and modules.

Specification approach or pattern matching are specification-based similar to anomaly detection in many ways, as they are based on detecting deviations of a norm. Instead of relying on a machine learning technique, they base on manually developed specifications that capture legitimate (rather than previously seen) system behaviors. The downside is that the development of detailed specifications and pattern creation can be time consuming, and the likelihood of a false negatives can be high if the policy is not updated often enough.

2.3.4 Creating Profiles

In order for an anomaly detector to be able to differentiate the sampled data and the 'known' behavior or profile, sufficient amount of data to do the statistical analysis has to be collected. One often says that the statistical approach is characterized by two phases: A training phase and a detection phase. In the training phase, the behavior of the system is observed in the absence of attacks and machine learning techniques are used to create a profile of such normal behavior. In the detection phase, this profile is compared against the current behavior of the system, and any deviations are flagged as potential attacks[14]. Unfortunately, systems often exhibit legitimate but previously unseen behavior, which again leads AD techniques to produce a high degree of false alarms, and hence reduce the effectiveness of the systems [13, 14, 27, 28]. Much effort is made trying to reduce the high rate of false alarms since HIDS is becoming more and more important.

2.3.5 Biometrics

A look at some types of behavioral authentication techniques reveal some different attempts. Most modern biometric methods rely on characteristics of the body and its behavior. Some examples of identifying biometric features used for identification include hand geometry[29], thermal pattern in the face, blood vessel patterns in the retina and hand, finger and voice prints[30], and handwritten signatures[31], not to mention blood samples and DNA tests[29]. A common factor for most of

System	FER	FQR	FAR
Face Recognition	0.0	0.0	N/A
Fingerprint - Chip	1.0	2.8	9
Fingerprint - Optical	2.0	0.4	8
Hand	0.0	0.0	4
Iris scan	0.5	0.0	3
Vein	0.0	0.0	N/A
Voice recognition	0.0	2.5	9

Table 2.1: An overview of biometrics system-reliability in percent(%) on Failed Enrollment Rate, Fail Aquire Rate and False Acceptance Rate.

these methods, is that they very rarely include behavioral characteristics. Most often only physical traits.

Research on physical biometric authentication methods are many. Especially important are the success rates of different biometrics, for comparison with non-physical biometric authentication methods. Fingerprint verification is of the cheapest to implement. Work with different algorithms have uncovered results from 1.73% to 47% equal error rate (EER) [32]. One point of interest in such research is the fail enrollment rate(FER), which points to failed collection of data. This can be due to constraints for the subject (user) in an enrollment phase (for example missing eyes for iris scans), or hardware / software failiure. Maio et. al concludes with rejected enrollment rates from 0% to 6.86%. Another measure is the fail aquire rate (FQR), which quantify failiure to capture data of sufficient quality (for example unrecognizeable fingerprints).

Research conducted by Mansfield et. al [33] give the table 2.1 for biometric systems [1].

There is no exact conclusion for this work [33], but the table still give an interesting picture of the reliability and problems such measures encounters. The report is critical to the amount of users they enrolled, and suggests a larger user group for further work.

Attempts to profile user behavior has been mentioned as using file access, shell commands, system calls or application access has been vast and difficult. Most attempts use training and data gathering/sampling as a long learning process, and

[1]Mark that the FAR is read from a graph and may be somewhat inaccurate

different algorithms to calculate intrusions. One of the later attempts was conducted by Kang, Fuller and Honavar (2005)[34] and has provided the *a bag of system calls* solution. They use the more commonly used *bag of words* representation for text classification, assigning simple unordered system calls as data input for a machine learning technique for profile creation. This light complexed approach, proved extremely successful in their test (100% accuracy with 0% false positives in parts of the testing), hence proving that one can achieve almost perfect detection rates even when disregarding the relation between system call and originating process. In real life scenarios however, the accuracy rate achieved was not as high as in the known datasets.

2.3.6 Mouse Movement Profiling

A fairly new field of research within behavioral biometrics is the mouse-movement approach, where user mouse behaviors are sampled every x-thousand times a second to create a representative dataset. Shivani et. al.[35] splits the active mouse movement authentication into three steps:

- (a) enrollment
- (b) training
- (c) verification

For statistical calculations where the components average, mean, standard deviation, minimum and maximum of the readings will be stored for all the vectors, (a) is used.

The training phase (b) consists of a number of repetitions to acquire the variations of the user, define the standard deviation for each user to determine the profile of the user. In other words, repeating (a) for a number of times.

The verification (c) usually take place through a login screen or authentication procedure, and mouse movements are recorded and statistically compared with the users stored profile. If the results are computed within the accepted range of the user, authentication is accepted.

A passive authentication has only two steps, (i) enrollment and (ii) verification. Phase (i) is done over a longer time period. Shivani[35] use the term transition states about dense areas of the screen, meaning the areas where the mouse movements are used the most, (for example near menus and scrollbars) have a higher frequency of samples. This phase ends up with storing a profile computed on the average, mean, minimum, maximum and standard deviation of users speed, angle movement and distance from the center of the transition state. A verification test is ran every two minutes to check if the variation of all the data samples lies within a $\pm$ 1.5 standard deviation range.

Brodley and Pusara [17] have another angle toward the mouse movement authentication and re-authentication. The experiment has mouse events grouped into a hierarchical data set categories, from main node to three children: (1) Mouse wheel points, (2) Non-client area movements (*NC moves*; typically menus and toolbars) and (3) clicks, separating single and double clicks. The vectors saved are distance, angle, time, speed, mean and standard deviation, but also a set of categories which are defined to use a decision tree algorithm.

Basically, a tree of data consisting of means and deviations will be processed, giving a weighted decision tree classifier. Where Shivani et. al compare the samples to the existing profile having 63 different users, the Pardue University group compare each step/group/tree branch with all other users, giving a probability of identity. However the limitation of the latter experiment was a low number of users (11) and only using one specific application (Internet Explorer). Shivanis[35] experiments with all parameters in the active mode had some surprising results. When accepting 3 times the standard deviation, the false rejection rate was 36 %, but adjusting the level of accepted standard deviation value to 2, the false rejection rate went down to 15% on active authentication and a 90% discovery rate on passive authentication.

Brodley and Pusara[17] concludes with a 0.43% false positive rate and 1.75% false negative rate within the one application, after a tweaking process. There was no tests conducted to cover any active authentication in these experiments.

Hashia, Stamp and Pollett published results in 2005[36] with an equal error rate of 15%. Other results have been found in the works of Nisenson et. al [37], with a 92% - 98% accuracy rate. Gamboa and Fred [38] made promise of a 3% - 4% equal error rate in their research.

2.3.7 Keystroke Dynamic Biometric Authentication

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 second in an attempt, to identify users based on habitual typing rhythm patterns[4]. In 1990 Joyce and Gupta [39] showed that keystroke rhythm is a good sign of identity [40], 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[41].

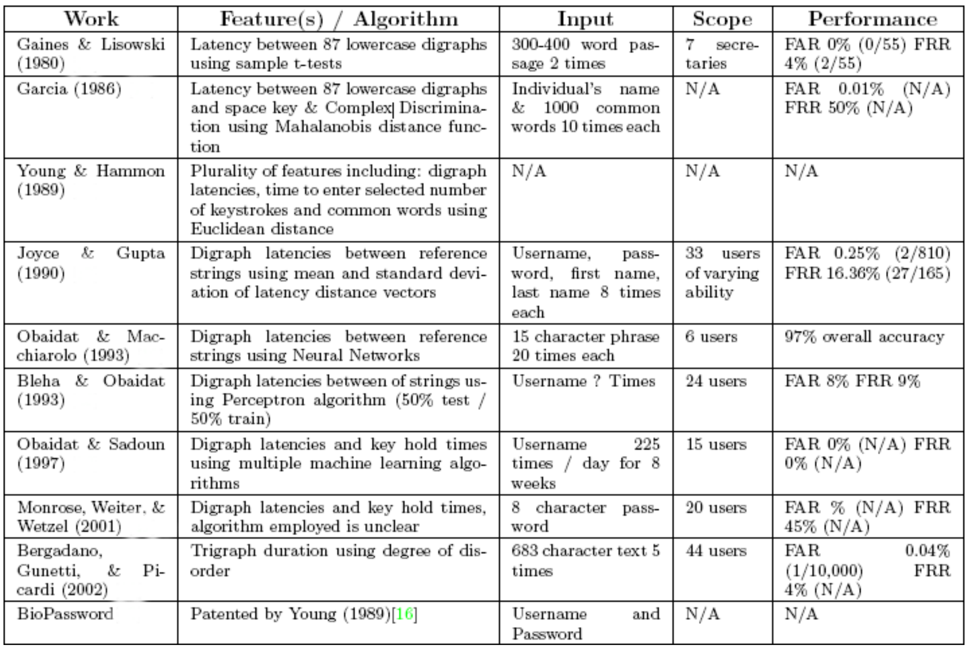
There are two main approaches. Static or active authentication seek to analyze keystroke, verifying characteristics only at specific times (such as during login sequence, forced retyping of password or input boxes etc), however this approach does not provide continuous security, and cannot detect a substituted 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 highjacking.

Recent research [1] made available the following table 2.1 for keystroke dynamic historical results and approaches.

Experiments done by Leggett and Williams[40] 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.[42] address several problems with regards to keystroke timings, and opened up for considerable improvements.

Joyce and Gupta [39] 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.



Work	Feature(s) / Algorithm	Input	Scope	Performance
Gaines & Lisowski (1980)	Latency between 87 lowercase digraphs using sample t-tests	300-400 word passage 2 times	7 secretaries	FAR 0% (0/55) FRR 4% (2/55)
Garcia (1986)	Latency between 87 lowercase digraphs and space key & Complex Discrimination using Mahalanobis distance function	Individual's name & 1000 common words 10 times each	N/A	FAR 0.01% (N/A) FRR 50% (N/A)
Young & Hammon (1989)	Plurality of features including: digraph latencies, time to enter selected number of keystrokes and common words using Euclidean distance	N/A	N/A	N/A
Joyce & Gupta (1990)	Digraph latencies between reference strings using mean and standard deviation of latency distance vectors	Username, password, first name, last name 8 times each	33 users of varying ability	FAR 0.25% (2/810) FRR 16.36% (27/165)
Obaidat & Macchiarolo (1993)	Digraph latencies between reference strings using Neural Networks	15 character phrase 20 times each	6 users	97% overall accuracy
Bleha & Obaidat (1993)	Digraph latencies between of strings using Perceptron algorithm (50% test / 50% train)	Username ? Times	24 users	FAR 8% FRR 9%
Obaidat & Sadoun (1997)	Digraph latencies and key hold times using multiple machine learning algorithms	Username 225 times / day for 8 weeks	15 users	FAR 0% (N/A) FRR 0% (N/A)
Monrose, Weiter, & Wetzel (2001)	Digraph latencies and key hold times, algorithm employed is unclear	8 character password	20 users	FAR % (N/A) FRR 45% (N/A)
Bergadano, Gunetti, & Picardi (2002)	Trigraph duration using degree of disorder	683 character text 5 times	44 users	FAR 0.04% (1/10,000) FRR 4% (N/A)
BioPassword	Patented by Young (1989)[16]	Username and Password	N/A	N/A

Figure 2.1: Previous work and results on Keystroke Dynamics [1]

Further work suggested that median interkey latency of expert typists are approximately 96 ms, while that of a novice is near 825 ms, therefrom making digraph-specific measures would lead to measurable improvements in verification accuracy. Neural network approaches have also been undertaken, but widely found to be expensive and time consuming, and retraining time would prove significant [4].

Joyce and Gupta [39] 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 nonweighted 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. Monrose [4] 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 %.

An analysis of differences in languages has been conducted by Gunetti et. al[43], bringing the conclusion that samples of free text for a combination of small segments of samples is independent of language. Another research by Gunetti and Picardi [41] considers free text and evaluates small parts of the text (for example usage of character combinations like *the*, *tic* and *ica*), hidden inside long passphrases. Their technique for 205 subjects obtained a FRR of less than 5% and a FAR of less than 0.005%. This work has obtained the best result found when researching for this survey, but requires a considerable length for a passphrase.

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) [1]. 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[1] also proved a significant performance difference for long passwords (12 characters), compared to short passwords (8 characters).

2.3.8 Keyboards

Keyboard History

The keyboard technology spun from the teletype machine combined with the properties of the typewriter. The keyboard is normally used as an input device when connected to a computer. The American typewriter Christopher Latham Scholes patented in 1964, the typewriter that we commonly use today in 1868. MIT, Bell Laboratories and General Electric created the first modern keyboard, using video display terminals (VDT), which enabled users to see the actual text on a display screen. This enabled users to create, edit and delete text.

A keyboard consists of a basic structure; The top layer has an arranged set of buttons or keys. These have the ability to send electronic impulses through an interface to the computer when pressed down. Latham Scholes developed the most widespread standard arranging of keys, know as QWERTY. There are other key arranging standards(for example AZERTY and DVORAK). Language adjustments make keyboard design differ for different countries.

Keyboards typically have characters engraved or printed on the keys, which ordinarily corresponds to a symbol. However, some symbols requires pressing several keys sequentially. Other keys does not produce symbols but operates the computer or input itself. A keyboard also contain a microprocessor and a BIOS (Basic Input/Output System) to sense the keys pressed and transmit the corresponding key code to the computer.

Keyboard Communication

When pressing down a key, a conductive contact on the underside touches a pair of conductive lines on the circuit below. This enables a flow of signals. The keyboard control device scans the lines to all keys, and generate a key code (also known as scan code) as the signal pair changes. The code is sent to the computer (for example through a cable or wireless connection). The computer receives the signal bits and decodes it into the appropriate keypress and computes the appropriate action (for example type text to the display device, lit leds or start computations).

The scan codes picked up by the keyboard bios is sent down the serial line in hex format (for example the 'A' key has the scancode of 1C) [44]. If the key is not released within a set time, the keyboard will continue to send the code until another code is picked, up or the key is released. When a key is released, the scan signal 'F0' will be sent followed by the corresponding key scan code for the computer to know which key is released. Each key has exactly one scan code. Different engravings on a single key exists quite commonly, and thus there are control keys (for example shift-key and Caps-Lock key) which are interpreted by the computer to determine the correct keypress.

The keyboard protocol is bi-directional, making it possible to both send and receive data.

2.4 Summary

Using behavioral authentication and re-authentication to detect users pattern anomalies, has been implemented with variable success on different interaction levels (system calls, keystroke dynamics and mouse movement). These are all user behavior based, using non physical biometrics, but has shown themselves valuable for further research even after simple testing. All the tests conducted and mentioned in this section of the paper has by the authors been concluded inadequate without any other authentication method.

Problems such as users drinking coffee while typing, being on the phone or simply switching users or administrators to help conduct work related tasks, are all likely scenarios which can fall through in these applications.

The main problems with keystroke recognition are ranging from:

- The pattern database search time.
- The keystroke improvement of users (profile learning and growth).
- Distractions making users change their writing characteristics. Some of the same problems occur also with mouse detection.

However, in environments and applications where mouse usage is scarce (e.g. some linux environments) mouse-detection tests would prove to have very little data. Also considering different mouse types (e.g. 8 button mice with 2 scroll-weels vs a standard one button Mac mouse), one would need different data enrollment applications. Finally, input profiling systems are complex and add more levels of administration to implementation and maintenance. There are generally fewer standards and less implementation problems using keystroke dynamics.

When it is uncertain whether behavior is anomalous or not, the system may rely on a parameter that is set during initialization to reflect behavior, or gain support from parameters outside the actual event in question. One example of such a system could be a re-authentication system. Adding these ideas to a situation where a statistical uncertainty has occurred and found by a IDS, could greatly increase the decision accuracy, and reducing the numbers of false negatives. Combine a false positive rate of 10% by initial IDS, and 5% by the re-authentication system, the combined false positive rate would come to 0.5%.

Chapter 3

Observations

This chapter include important observations found in the tests conducted, which should be considered before looking at the models and results.

3.1 Terminal Types

One important aspect to consider when analyzing the user profiles, is differences for a user when changing terminal. The terminal is the object or keyboard used to enter the data into the system. This could be a palm pilot, a cell phone or different keyboard types. In this work only different keyboards and computers were used to test. One USB keyboard on computer 1 (type 1), one PS2 keyboard on computer 1 (type 3) and a laptop with an internal laptop keyboard on computer 2 (type 2) were used in the experiments.

Figure 3.1 gives an overview of differences for *user 15* on the computer 2 (laptop) and computer 1 with a USB keyboard. The reason for selecting this specific user is because it is the one user with the least difference between the profiles. As can be seen from the figure the terminal type 1 has lower standard deviations, and on average less KeyDown time than type 2. However the values are quite similar, especially in the last 4 keystrokes. Around 9% of the entries would not accepted as legitimate logins. This test was conducted using full profiles, and it is expected that the FAR could increase somewhat, from the experiences of the other tests. Another fact to consider would be the use of *Alert Levels*, which improved the

Value test rate from 10-30%. This has not been applied in these tests.

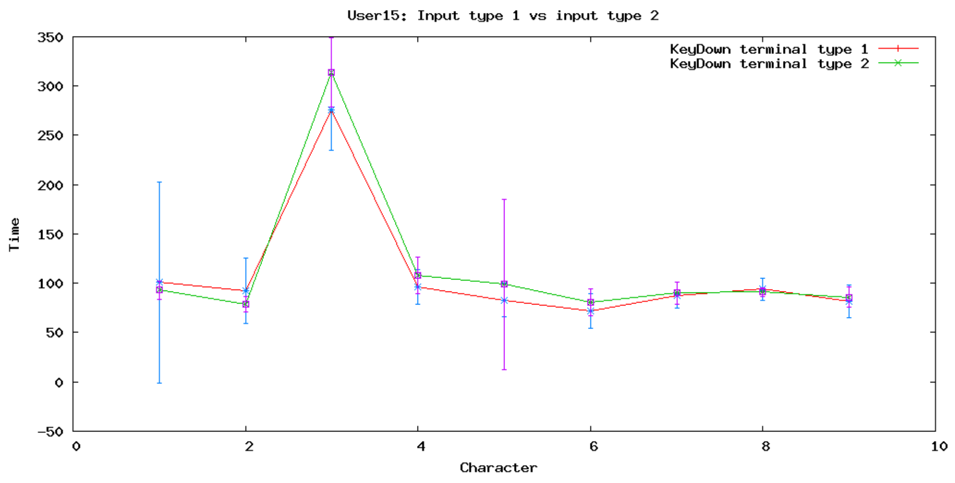


Figure 3.1: Full profile comparison for different terminals on KeyDown parameter.

When examining the figure 3.2 for the KeyWait parameter, the results are quite different. Again the average waiting time is lower for terminal type 1 compared to terminal type 2. The terminal type two have larger deviations, which indicate less consistency. As a matter of fact only 30% of the last 4 keys for type 2 would be within standard deviation limits for input type 1.

Thus 70% of the total attemts to log in with another terminal would fail. There are users where as much as 95% of the logins for all keystrokes would fail. There is a huge difference in how a user types on a laptop as opposed to a regular keyboard. When imagining typing on an unknown laptop (or any other not known keyboard type), it seems likely that users slow down to enter the correct characters. There is a difference of the space between characters on a laptop keyboard and a regular keyboard, which could also cause this behavior.

When comparing the USB keyboard of the same type as a PS2 keyboard (identical in design), there is no recognizable difference in the profiles. There are some small differences, but the profiles match at a rate of 92% for KeyWait and 95% for KeyDown, with standard deviation, on average.

The main parameter that fail in these tests, is the KeyWait. Looking at the KeyDown time from a button is pressed down until released, no difference from a regular keyboard to a laptop keyboard could be found. It is important to point out that only users from proficiency level 3 and 4 (considered experienced users) was

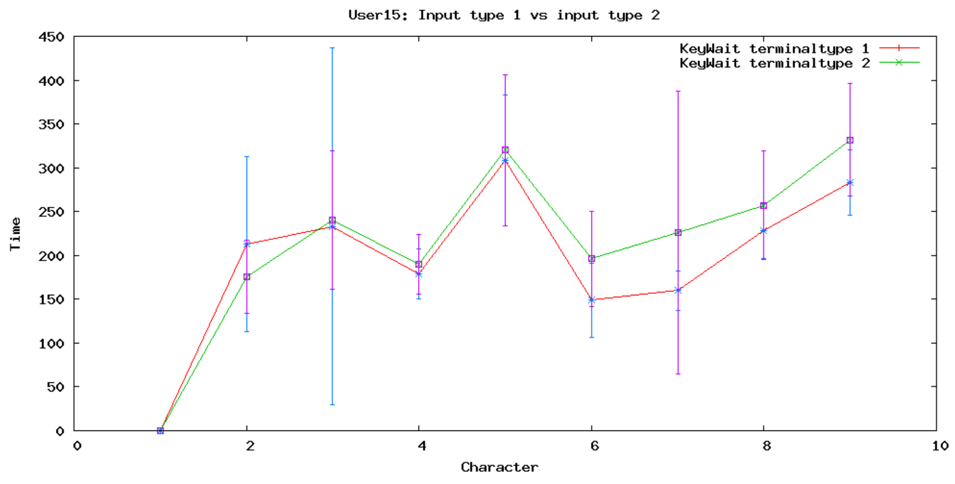


Figure 3.2: Profile comparison for different terminals on KeyWait parameter, presented by samples from user 15.

used in these tests. There could be differences between these users and users on the lower proficiency levels. It was evaluated, and found that the most consistent users only should preform these tests, because they were most likely to have experience with keyboards and have less unknown behavior alteration (less uncertainty factors).

For every input type where there are significant design changes as to form, size or spacing, a legitimate login would probably be considered as an intrusion. There was no time to examine this in further detail with respect to more input terminals, or more detailed proof. When implementing a system this fact has to be taken into consideration. Changing the keyboard could imply a new profile needs to be created. There could be shown no evidence to support that the same would go for different technologies like USB versus PS2.

3.2 The KeyWait parameter

The time between two characters is described in the *KeyWait* parameter. Historically this parameter have been used in different ways, but the most common usage can be seen in figure 3.3. This approach has some disadvantages, which is why another approach was attempted. Section 4.2 discusses and describes how the *KeyDown* and *KeyWait* parameters are used in this work.

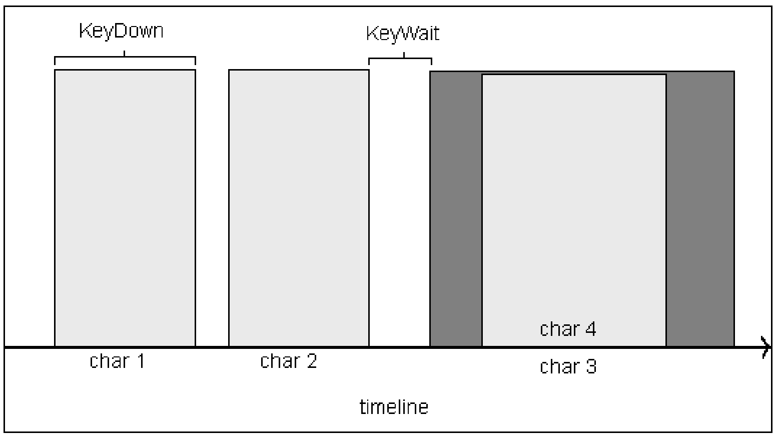


Figure 3.3: The most common usage of the KeyDown and KeyWait parameters. Notice that the KeyWait parameter is used differently in this work.

The *KeyDown* characteristics does not have this flexibility, and is used in the same way other research have been using it.

Chapter 4

Method

This chapter covers the basic concepts of how the test environment was implemented. Using data from multiple logins and multiple users is essential to get variability in the data, reflecting the patterns of each user, and in turn recognizing and differentiating them. This chapter will also cover the data mining and gathering process through profile creation. An overview of the process is illustrated in Figure 4.1.

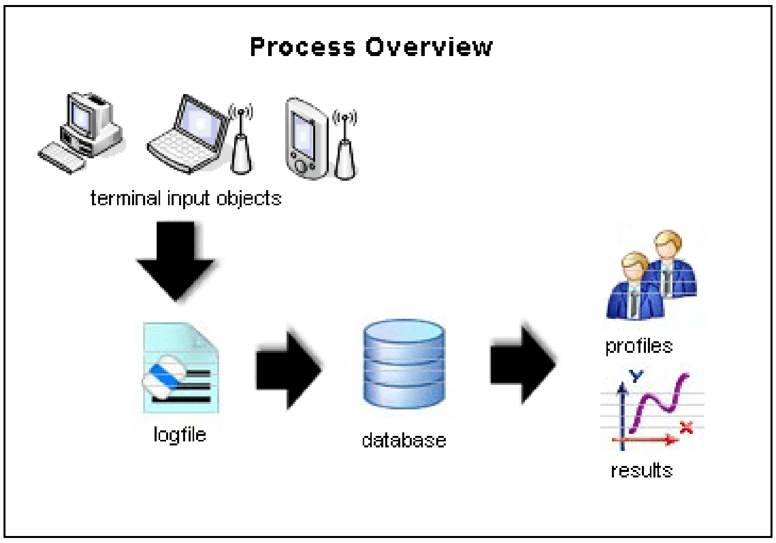


Figure 4.1: Process overview chart. From terminal input through logging and database filtering, to profiles and results. (clip-art: www. yankeedownload.com)

This work has collected input data from multiple sources, including a laptop keyboard, regular USB keyboard and a PS2 keyboard. The result is saved in a log file, which is filtered in an import function and stored in a database structure. The database is used to derive and store user profiles as well as test results. Some results have been calculated directly through using math tools such as *Mathematica* and also using a spreadsheet application.

4.1 Environment

The major operating systems (OS) and applications existing today have a login sequence. This includes OS logins and application logins (both web based and terminal based). Implementation of a *proof of concept* login sequence was done in a separate application. There are no reasonable grounds for implementing such a feature directly into a specific OS login, as opposed to an application on top of the operating system, for these experiments.

Availability is arguably the most common reason for choosing a web application login sequence. It enables users to login at any appropriate time or place. However there are downsides to such an approach. Control of activity is limited, which results in increased management and a high amount of communication to ensure proper data collection. Also there could be uncertainty in the data. Users may have different environments such as keyboards, systems, OS and browsers, to name a few. This may result in invalid or false data.

Based on the assumption that a controlled environment, all users have the same parameters, a stand-alone application was created for all experiments. Users varying in age and technical level was the main reason for choosing a Microsoft Windows like login. This would enable some familiarity to previously used systems and reduces the insecurity for not proficient computer users. Proficient users would have smaller problems in adapting to this kind of technology, look and feel.

4.2 Parameters and Properties

When entering a character on a keyboard, a set of instructions is sent from the keyboard to the computer. In this thesis the parameters KeyDown (also referred to as kd) and KeyWait (also referred to as kw) are the ones used most frequently. The figure 4.2 illustrates the time period these parameters actually span over. The Key-Down 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. For example entering your name or password, which has been entered frequently and has become a habit (there is no thinking when typing). There is not much flexibility in how this parameter is used, and previous research using time in keystroke, use this exact definition.

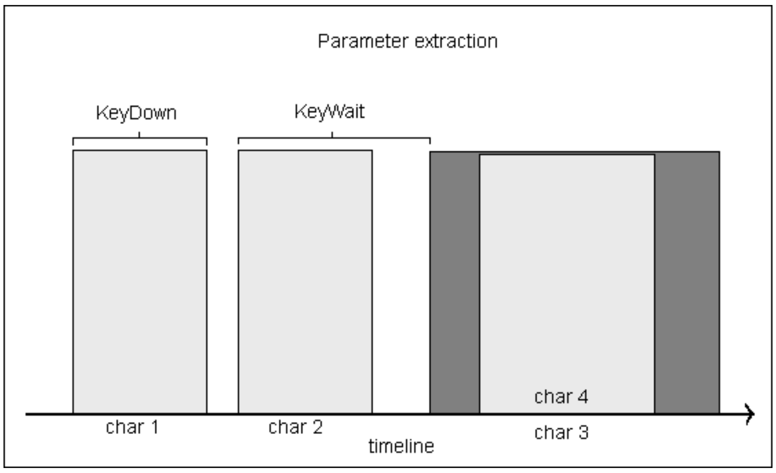


Figure 4.2: Graphical representation of the KeyDown and KeyWait parameters.

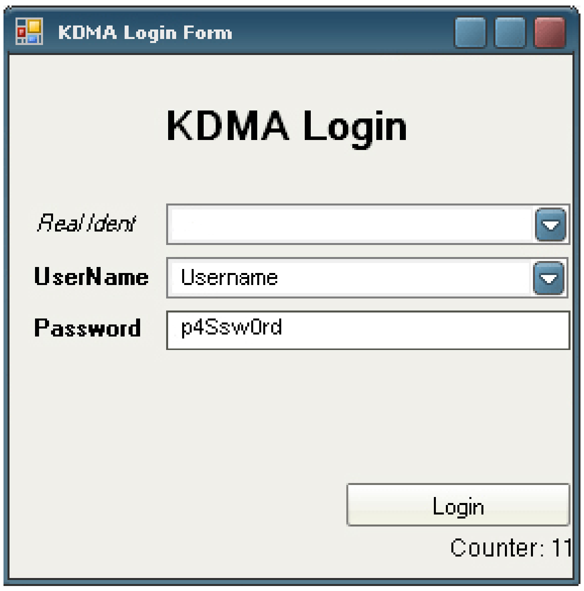
The KeyWait parameter has been used in different manners. Most commonly is the time from one key is released to the next key is pressed. Also research has attempted to use the time from the first key is depressed to the next key is depressed. No research found has used the KeyWait parameter as shown in the figure 4.2. The gray area represent the third key which span over the next key. This indicates a shift-key.

In this thesis the KeyWait time is calculated from one key is pressed, to the next is pressed. The figure shows the KeyWait as the time period from *char 2* (the same as key 2) is pressed down, until the *char 3* is pressed down. There are several reasons to why this approach was selected. One being the fact that no negative numbers can be encountered. The second key would always be pressed down before the third key, or else the password would not match. When considering shift-keys, the key is not actually depressed before the consecutive key is pressed and depressed. To validate and get the correct timestamps, ordering of keys would be needed.

This approach disregards the underlying reason behind the parameter value. It does not matter if it is the time between the characters, or the time a key is depressed, which contribute to the value. But it does depend on the parameter being consistent for each user. The depressed deviation is handled by KeyDown. Ideally the KeyDown is consistent, and the actual waiting time would be accurate.

4.3 The Application

Designing the layout and content of the login was done and tested carefully, making it very much standard without any confusing texts or extra features. The figure 4.3 shows a screenshot of the login window. It consists of one input field with the label *UserName*, one input field with the label *password* and a button (with the label *Login*). One hidden field (shown in italic) was set before each session to contain the username of the tester and the type of terminal used in the session. Users were unaware of this field, and it was used to check the *UserName* field. One element in the login is a counter showing the progress in number of attempts.


KDMA Login Form

KDMA Login

Real Ident

UserName: Username

Password: p4Ssw0rd

Counter: 11

Figure 4.3: The login window used in the login sessions.

Note that the fields *UserName* and *Password* are blank upon opening the login window, and the counter is reset. The password ($p4Ssw0rd$) is just an example which resembles the actual password selected. It is not the password used in these experiments. The *Real Ident* field is hidden for the users, and will contain the real username of the user conducting a login, not the one a user selects. It is used for error checking.

4.4 Technology

Advantages with using existing resources, such as libraries and built in features, made C# the selected option for developing the login application. One advantage of using this programming language is the ability to use the .NET framework for designing windows and visual elements. The .NET C# environment use the `System.Windows.Forms` library with `Control.KeyPress Events`, `Control.KeyDown Events` and `Control.KeyUp Events` classes. These control classes are used to handle the input and output stream of data on the keyboard port. Typically the input stream is used in these experiments as no commands are sent back to the keyboard.

The application can run on any .NET 2.0 supporting OS, including Microsoft Windows 98 and newer. It is compiled into an executable *.exe* file and logs to a CSV (comma separated) text file.

4.4.1 Timestamp

Each time a key is pressed down or released, an event is triggered. This event (type according to key press or key release), will have a timestamp which consists of the number of microseconds (μ) since 01.01.1900. The timestamp value is used for calculating the time from a key to be pressed down to it is released, and for calculating the time between the keys.

4.4.2 Log Data

For each keypress entered into the password field the following attributes are logged:

1. KeyDown time stamp
2. KeyCode value (the character code)
3. KeyUp time stamp
4. KeyWait time value

The *KeyCode* contains the character value associated with the button depressed. For example the character *a* will have KeyCode `A`. Notice that the character *A* will also have the KeyCode `A`. So extracting the actual character is dependent on a signal or beacon from the keyboard, telling the computer that the next character is uppercase or lowercase. This is normally done using the Shift-keys. The KeyCode for the number *6* would be `D6`, short for digit 6. Using the number-keys on the top row of QWERTY keyboards, will give a different KeyCode than using the right-hand side num-pad. None of the users used the num-pad in the login sessions.

The logfile has the following format[2]:

Logfile 1. *testnumber, username, character, timestamp (kd), timestamp (kw), time-wait (kw)*

4.4.3 Extracting Parameters

For future use, an import procedure was created. It read the logfile and stored the samples into a database. Due to mistypes and errors in logins, only valid logins with the correct password and correct KeyCode order are accepted as samples. Entering the correct password is not enough in itself.For example using the backspace key to correct a spelling mistake, is not a valid sample.

There are numerous ways to change the password character ordering in order to pass a simple password comparison on pressing the submit button. However this kind of behavior would severely demolish the data in the tests, as it is depending on a correct character ordering when conducting the *Identification test*.

[2]Timewait is calculated from the KeyDown timestamp of the current character minus the KeyDown timestamp of the previous character. For the first character in the password, the value is always 0.

Another reason for choosing a database structure would also include easy access to the data, as well as great features to visualize the results, using simple SQL queries. SQL support all the mathematical functions needed to build user profiles on the queried data. A web application was created for displaying the results and preliminary results in a easy fashion. More details can be found in the Appendix section C.5.

4.4.4 The Database

The database structure consists of the following tables:

Table 1. tblUser*: fldID, fldUsername, fldPassword, fldAge, fldLevel*

Table 2. tblTest*: fldID, fldUser, fldType, fldStartTime, fldKeyBoard*

Table 3. tblSample*: fldID, fldLoginMother, fldCharNr, fldKeyDown, fldKeyWait*

Table 4. tblProfile*: fldID, fldUser, fldtype,fldKeyBoard, fldKDChar1...n, fldKWChar1...n, fldSKDChar1...n, fldSKWChar1...n*

Table 5. tblTestresult*: fldTestResultID, fldTestResult_testid, fldTestResult_type, fldTestResult_profiletype, fldTestResult_value1, fldTestResult_value2, fldTestResult_value3, fldTestResult_value4*

A database diagram can be found in the Appendix section C.3.

tblUser is used to store user information. Separating users on age (*fldAge*) and computer proficiency (*fldLevel*) is useful in later tests and conclusions. The proficiency level is set in 4 levels (1 through 4) where 1 is lowest and 4 is the highest level of proficiency. The appropriate level selected is based on the expected experience level, computer workings and the general computer knowledge through experiences and communication with the users. The levels are based on the following level description for computer proficiency.

1. Novice User
2. Beginner (with some experience)
3. Advanced User
4. Experienced Expert

For each unique login, a record is inserted into the *tblTest*. The *fldType* is a classifier of the password type. *fldKeyBoard* is used to separate one input terminal and / or keyboard type in order to distinguish pattern differences.

tblSample consists of one entry for each character in any approved login attempt. The *fldLoginMother* relates the character to a specific login attempt. *fldCharNr* is used for mathematical purposes to relate samples to each other and ordering of the samples. *fldKeyDown* is set to be the KeyUp.timestamp - KeyDown.timestamp (the total time the key is pressed down). *fldKeyWait* is the waiting time from the previous entry KeyDown to this entry KeyDown. See figure 4.2 for a graphical representation of these parameters.

The table *tblProfile* has user id to correspond to a specific user (*fldUser*), a field for each kind of user profile type (*fldtype*) and *fldKeyBoard* for separating terminal / keyboard types. The average KeyDown time (kd) and average KeyWait (kw) time is calculated for each character, and the corresponding standard deviation uncertainty of each character on the KeyDown (skd) and KeyWait (skw) is also calculated. As our password will always have the same length (9 characters), this kind of approach can be used. In an environment where the number of characters differs from user to user; a new structure would be needed, where a counter keeps track of the number of characters and the ordering of the keystrokes in the profile.

The table *tblTestResult* contains the result of every test run in order to make comparisons between different tests. *FldTestResult_testid* contains the test id from *tblTest*, while *fldTestResult_type* stores the type of test. Fields *fldTestResult_value1* through *fldTestResult_value4* stores the result of the tests.

4.4.5 Data Collection Results

When making the final analysis there were 25 different users involved. 12 of these conducted tests on multiple terminal types. When verifying the password, 27% of

User distribution		
Gender	Male	15
	Female	10
Age	60+	2
	50+	5
	40+	4
	30+	8
	20+	6
Origin	America	1
	Africa	2
	Asia	2
	Europe	20
Proficiency	1	5
	2	6
	3	9
	4	5

Table 4.1: Subject/user distribution for Gender, Proficiency, Origin and Age

the login attempts failed (passwords were entered incorrectly). This was higher than expected, and unfortunately reduced the number of login samples. One type of password were tested, on three different terminals (keyboards and computers). The total amount of valid logins were 3906.

In order to get a wide range of users, differentiating in gender, age and computer proficiency, users from different environments and professions were chosen (see table 4.1). 10 of the users were female and 15 were male. 2 subjects were over 60 years old. 6 users between 50 and 59, 3 subjects between 40 and 49. The largest group was users between 30 and 39, namely 8, while 6 people were younger than 30 years old. Individuals from Europe, Asia, Africa and America contributed to the tests, although 20 originated from Europe (Norway).

The users were separated into estimated computer proficiency levels 1 through 4, where 4 is the most proficient. At level 1, 5 users were estimated. 6 users were registered as second level proficient. 9 users was found most fit for the third level, and 5 people at the highest level. The proficiency levels are discussed in section 4.4.4.

4.4.6 Choosing Password

An assumption for this work is that the accuracy in how a user types the password will increase as the familiarity of the password is increasing. Studies by Yan et. al [7] shows the memorability of passwords has great influence on the chance of misuse. Verheul conducted research in the security aspect of selecting the password[5]. The conclusion told of a significant improvement in security breaches when selecting a complex password compared to a not complex password. In this study the users have been given the same password without knowing this. This increases the number of samples to use as the login attempts can be considered both as intrusions and valid logins depending on the profile owner.

It is important to use a password consisting of several different elements. It should consist of both lowercase and uppercase letters. This forces the use of key combinations. Uppercase letters are a combination of several keypress, a shift button (left shift, right shift or caps-lock) needs to be used before the consecutive key, and is released after the key is released [3]. It therefore increase the total password length with one character pr. uppercase letter (unless there are multiple consecutive capital letters, which would result in a total increase of one, not more, characters). For example the password: 'iA', is a combination of 3 characters, namely 'i', 'Shift' and 'a'. Recent research [1] on user behavior shows an increased accuracy when separating different shift options to create uppercase letters. This has not been implemented in these experiments, and would increase the accuracy of the results even more.

Dictionary attacks are potential problems when selecting a password, thus the selected password should not be found in any dictionary of any language and also needs to consist of a combination of characters and numerical values. Research also concludes that longer passwords have better accuracy than short passwords [1, 7].

[3]Caps-lock works a little different from the Right and Left Shift buttons, as you actually release the button before entering the next character. After the next character button is released, the Caps-Lock button is again pressed and depressed. Such a behavior would not get approved as a login in the test environment. There is however no reason, other than time constraints, behind the decision of not implementing this and similar features into the scripts used in this thesis.

This in mind, the selected password has the following format: *LDULLLDL*

L: lowercase character

D: digit character

U: uppercase character

The length of the password string is 8 characters, but considering uppercase letters, the total length is 9 KeyCodes. This is often overlooked in many systems, requiring a set number of characters. Using uppercase letters increase the total length may reduce the difficulty of a password, and with increasing the length also increasing the detection accuracy. Users in these experiments subsequently complained about the complexity of the password, which proves that the approach did indeed work.

4.5 Profiling

A *profile* is in this thesis the calculated average of a set of logins from one user. The users underwent several independent login sessions, where 15 logins were attempted in a session (minimum 5 sessions), until 60 successful logins where completed. This provides sufficient amount of data to undergo statistical and mathematical analysis, and a solid data foundation on which to base a profile. The aspect of selecting a complex password proved, as intended, the result of an incrementally growing learning phase as the password became increasingly familiar.

The term *profile growth* is used to describe the way login samples change over time. When the password is entered repeatedly, a noticable trend emerges, due to increased consistency. *Profile growth* is used to describe both the change in pattern for single users, as well as the general trend for all users. The impact *profile growth* has to the test results, is described in detail in section 5.3.

Three different profiles were created for every user.

Profile 1. *First 50% of successful logins attempts*

Profile 2. *Last 50% of successful logins attempts*

Profile 3. *All successful logins*

This setup enables analyzing the learning phase and how the profiles grow for a specific user or set of users. The assumption would be that users have less uncertainty in the second profile, compared to the other two. It also should indicate the proficiency of the users, where users with only basic computer proficiency would grow slower and have higher uncertainty than more proficient users. This would be important in an environment where accuracy in identification is relatively high. An assumption would be to expect users with low uncertainty (standard deviation) profiles to have a lower False Acceptance Rate (FAR). However there is a possibility of an increased False Rejection Rate (FRR) for anomalous login attempts. It could be easier to identify these users in statistical and mathematical comparisons. The Equal Error Rate (EER) is the value where FRR and FAR is exactly the same (if represented as two graphs, this would be where the graphs cross). A Mean Error Rate (MER; mean difference between FRR and FAR) can also be used as a better measure of how the results from FRR and FAR compare. One advantage using MER as opposed to EER is the fact that MER will always have a value between 1% and 100%, EER cannot guarrantee a result.

4.5.1 Profile Parameters

Each login sample consist of a set (X) of characters of n length. Ranging from $x_1, x_2, \ldots, x_n$. With the two selected parameters KeyDown (kd) and KeyWait (kw), each having a corresponding standard deviation value, results are shown in the profile matrix 4.2. The annotation skw and skd will be used throughout the paper, meaning the deviation property of kd and kw. However σ_{kdi} will be the mathematical expression for the deviation of the parameter x_{kdi} of kd.

4.6 The Alphabet

In this work the *alphabet* describes a range of possible values in the data gathered. For keystrokes, time is the dimension we collect data on. A computer has a

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

Table 4.2: The default user profile template with KeyDown, KeyWait, KeyDown deviation (skd) and KeyWait deviation (skw) characteristics.

running process that queries the keyboard port every 1 microsecond (μ). The least amount of time a key can be depressed is 0 (zero) μs, although highly unlikely to occur in a real sample. It will occur in the first KeyWait property, before the first character has been pressed down. When a key has been pressed down for 1500 μs, the computer considers the value to be a new character[4].

This should enable an alphabet of 1500 possibilities. When studying the data samples, there were some values that was repeated too often to be regarded as random hits. Ordering the samples independently and with distinct values (one of each value only), gave a list of numbers with a repeating pattern. There seem to be a 15.5 μs period between all the gathered values. This proved to be the case for different keyboard types (two different USB keyboards and one PS2 keyboard was tested) and different computers. The search for an explanation to this phenomenon was unfortunately not successful. Two hypothesis are suggested, the first being the frequency of which a standard keyboard is sending signals, is every 15.5 μs. Another suggestion could be an unknown computer software process or hardware setting delaying the measurements.

Regardless of the explanation, the data is consistent on this delay, and would affect all users in the same manner. The data is therefore considered valid. This does however reduce the *alphabet* of values for KeyDown and KeyWait from 1500 to 100. A reduction like this would limit any distinction between users, as the granularity of the experiments deteriorate. Still, 15 μs is probably sufficient to separate users on typing behavior.

[4]This max time of 1500 μ will actually be reduced to 1000, 700 and 500 μs in turn, as a feature in order to type repetitive characters faster without releasing (depressing) the button.

4.7 Selecting Parameters from Characteristics

When typing on a keyboard, several characteristics is needed to make a final decision. Attempts of one of these characteristics have been made in related work, however combined results from different types of tests has not been found in any known material. Using a combined result metric for decision making would enable several of the test types to successfully differentiate the user in question. Some of the tests could find itself in a gray area, where other tests would possibly make a successful comparison. There is also a chance of the different analyzing tools to get contradicting result. Where none of the tests can make a positive match, no answer can be found.

The following tests are used to evaluate the data in this thesis:

Value test 1. *The* Value test *is taking the solid values of a sample and comparing them to each keystroke in the profile of the username selected*

Identification test 1. *Using LN methods the* Identification test *is used to determine the total angle in a multi dimensional grid*

Pattern test 1. *Ignoring the values, but looking only at the pattern of values going up or down from the previous character*

4.8 Models and Evaluation Methods

Having defined the profiles of each individual user on different levels, the next step would be to analyze and compare users. In order to select the appropriate approach, knowledge of the data distribution is needed. Distribution models have different advantages and tool for analysis. Several models were tested before an acceptable one could be found. This section describes the steps taken, and why models were accepted or discarded.

This section will also go into detail of the process of verification and identification of a user trying to login. The figure 4.4 shows the process from entering the password on a terminal, through a keystroke evaluator (*Value test*). Not depending

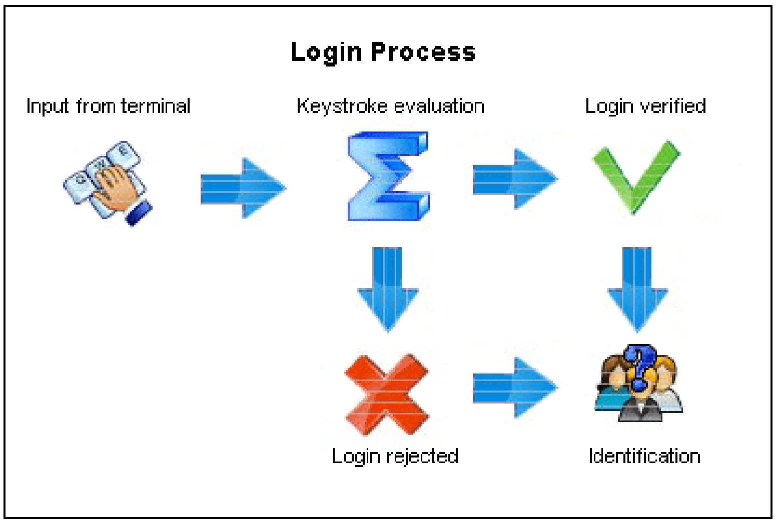


Figure 4.4: A detailed process overview of the entire login sequence, with keystroke evaluation (Value Test) and identification (Identification test). (clip-art: www.yankeedownload.com)

on the success of the evaluation (whether it is verified or rejected), an attempt to identify the user will be conducted. When disregarding the success of the *Value test*, it may be possible to identify both hacking attempts which fail, and also identify the user that succeed in hacking into another user account. The figure 4.4 describe how this is implemented.

4.8.1 Value Test

The *Value test* is conducted to match each single KeyCode, 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 character 1 is always 0. This sample would always have an exact match.

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. This gives the opportunity to decide whether the user should be allowed into the system or not. Experiment tests are based on the mean value for each key, and a calculated standard deviation which gives information of how much the users input vary.

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.

The implementation of the *Alert Levels* is not discussed in this part of the work. It can be looked upon as a strictness level, or the level in which the threat should be regarded. Details on *Alert Levels* can be found in section 5.2.1.

4.8.2 Identification test

There are several motivations to why identification is desirable. When examining a compromised system it is desirable to establish:

1. What has happened ?
2. Who was responsible ?
3. How can it be avoided ?

Identification of the user logged in is usually done by username and login. Password theft disrupts this approach. Especially if a internal intruder have gained privileges through another account. It would be almost impossible to track without direct interrogation. When adding an identification phase to every login, a set of the users most probable to have conducted the login, it would be very useful in this process to locate the actual user behind the false accepted login.

When a password is falsely entered a number of times because of failing a keystroke evaluation process, a identification for each attempt can give insight in which users are trying to obtain privileges through another account. In this manner it can be viewed as a intrusion detection tool, and not only an extra security layer.

The purpose of the *Identification test* is to identify a user attempting to log in. Focus on internal attacks has received more attention as of late. There is no research found on the actual identification of users trying to login to a system. The *Value* *test* (4.8.1) approach to reject logins with behavior outside of the set limits of an ordinary login for one user. When facing an inside threat, normally a intruder would also have a profile on the system based on his/her previous behavior.

Adding an *Identification test* to a login procedure for both accepted and rejected logins could be useful in tracking the actual user. To accomplish this some assumptions has to be made. The first assumption is that in order to identify a user, the user must have a profile on the system. The test will always find the most likely candidate, but if the intruder does not have a profile, the person with the most similar behavior will be selected. This does not prove that this is the user trying to log in, it merely gives an indication.

The *Value test* use all the parameters needed to differentiate the user from the profile. When trying to connect a user to a profile, another approach is suggested. An established profile in this work will have a certain amount of deviation to each keystroke. The *Identification test* makes an attempt to disregard the values in itself, and investigates the difference between the sample login and every profile to find the best match.

To obtain the best possible match, a multi-dimensional vector is used. Each keystroke will be defined as a point in a graph, and the angle difference between the keystroke and the profile will used for comparison.

Calculation of Angle (θ)

Imagine the login sample value as x, and the profile value y. The angle between first keystroke in the login sample (x_1) and the first keystroke in the profile (y_1) describes the variance. This can be repeated for all keystrokes until reaching x_n and y_n. Combining the variance for every keystroke gives a total variance, which describes the difference between a sample and a profile. The angle (θ) between two points through origo is calculated as follows:

$$X \cdot Y = |X| \cdot |Y| \cdot \cos\theta \tag{4.1}$$

$$\cos\theta = \frac{X \cdot Y}{|X| \cdot |Y|} \tag{4.2}$$

$$\theta = \arccos\left(\frac{X \cdot Y}{|X| \cdot |Y|}\right) \tag{4.3}$$

The crossproduct is defined as:

$$X \cdot Y = \sum_{i=1}^{n} x_i y_i \tag{4.4}$$

The scalarproduct is defined as:

$$|y| = \sqrt{y_i^2 + y_i^2 + \ldots + yn^2} \tag{4.5}$$

Entering this into a geometric hyperplane with n dimensions, using the knowledge obtained from (3.1), (3.2), (3.3) and (3.4) as numerator and (3.5) as the denominator gives the following formula (3.6) for calculating the total angle difference:

$$\theta = \arccos \frac{x_1 \cdot y_1 + x_2 \cdot y_2 + \ldots + x_n \cdot y_n}{\sqrt{x_1^2 + x_2^2 + \ldots + x_n^2} \cdot \sqrt{y_1^2 + y_2^2 + \ldots + x_n^2}} \tag{4.6}$$

The comparison will result in a number between 0 and 1, with 0 being a perfect match.

Identifying User

Comparing one sample to all the profiles, will give a numerical value for each profile. Ordering the list of profiles on this value in ascending order enables the test to find the most similar profile. It is important to remember that this test type may not be directly possible to implement into a system. It is improbable that an intruder has a profile based on the same set of characters. However in this proof of concept, imagine the test to test a typing pattern of random words. Even though this is testing one password, one can see the idea of testing parts of a password that is similar for all users. Another angle of usage is through passive behavior sniffing. Looking at words typed outside of a login sequence, like e-mail writing, commands given in a shell, or other suitable situations where input from the keyboard is given.

It can also easily be introduced if the *Value test* only barely accepts the entry, and a pass phrase is presented to a user in addition to a password to identify the user and get a *second opinion* or approach to the verification. Implementation suggestions will be covered in greater detail in the conclusion and discussion chapter.

4.8.3 Result Rating Terminology

These are the definitions of the terms used for explaining and comparing results in the tests and comparing the result with related work.

Terminology 1. *FAR - False Acceptance Rate: Rate the number of impostors falsely accepted to the system. For example, an FAR of 3% indicates that 3 out of every 100 imposters are falsely accepted as valid users*

Terminology 2. *FRR - False Reject Error Rate: Quantifies the number of legitimate users which are falsely rejected and denied access to the system. For example, FRR of 3% indicates that 3 out of every 100 legitimate users are rejected as invalid users.*

Terminology 3. *EER - Equal Error Rate: Rate at which FAR equals FRR. In the above examples the CER is 3%. Other terms are CER - Cross-over Error Rate*

Terminology 4. *MER - Mean Error Rate: Rate of the mean between FAR and FRR. In the above examples the MER is 3%. Other terms are AER - Average Error Rate*

Terminology 5. *FQR - Fail acQuire Rate: Rate of distribution of invalid users for a specific feature. For example a user with a invalid biometric feature when being blind, and in such failing the iris scan. Other terms are FAR - Fail Acquire Rate*

Terminology 6. *FER - Fail Enrollment Rate: Rate of which the user samples are invalid. Examples of this is extremely high deviations (for example user with parkinson for keystroke dynamics), and fingerprints where pattern is unavailable due to for example fireburn or missing fingers.*

Chapter 5

Results

This chapter cover the results found, step by step for each test. It describe samples, comparison of users and creation of profiles based on the observations. It goes into debt on profile growth and error distribution, and give insight on how the *Alert Levels* is used to improve results.

5.1 Data Values and Profiles

One factor for making any assumptions and studying the test result, is knowledge about the data and values at hand. This knowledge would be put into use before creating profiles and determining what should be considered as normal behavior. This section goes into detail on these matters.

5.1.1 Samples

A random sample from a successful login can be found in table 5.1. This particular user sample has several values worth noticing. The KeyDown values range, from 93 μs to 343 μs. The KeyWait from 203 μs (not considering the kw_1 with 0 μs wait time before beginning to type), to 531 μs. This sample provide the basic knowledge to how widely spread the data samples are, and what can be considered as low or high values in the following sections.

User 10									
kd	109	93	343	156	140	125	140	125	93
kw	0	281	328	203	375	250	250	531	343

Table 5.1: One login sample table for user 10, with KeyDown (kd) and KeyWait (kw) values.

Another thing to recognize in these data, is the repetitive behavior of some sample values. The values 93, 140 and 250 can be found several times within this one login sample. With the sample granularity of microseconds, it would be highly unlikely to hit the exact microsecond for multiple values in one sample. This behavior is discussed in greater details in section 4.6.

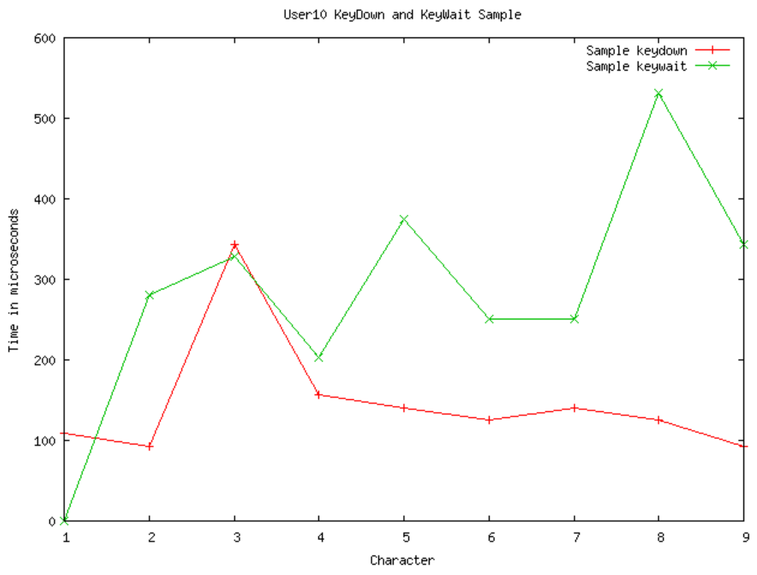


Figure 5.1: A graphical representation of one login sample for user 10, with Key-Down and KeyWait values.

A representation of these sample data is shown in graph 5.1. A pattern can be found in this data sample, and is quite common for most users, namely the fact that the KeyDown values are generally lower than the KeyWait for that same character. This is reasonable if imagining how long one actually holds a key down compared to waiting time between keystrokes. However the third character KeyDown value (right or left Shift button), is considerably higher, as the Shift button needs to be depressed all while the next character is entered, this behavior is expected.

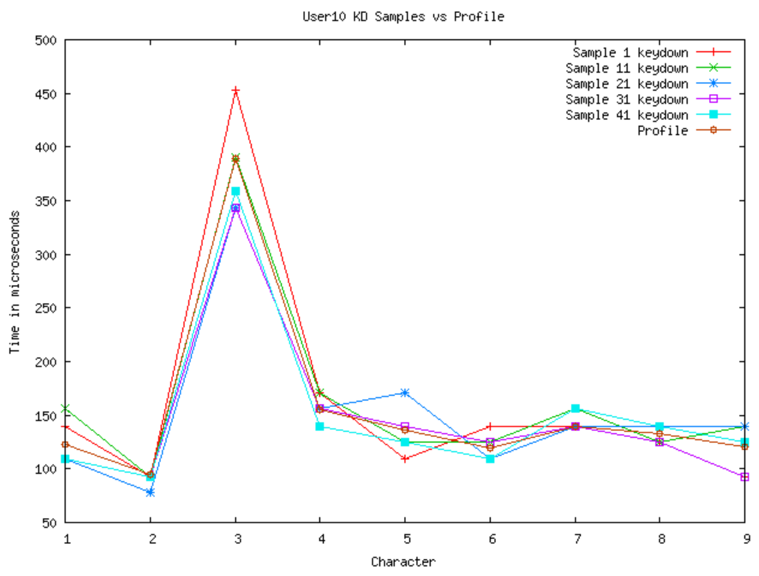


Figure 5.2: Five login samples and profile of user 10, showing the KeyDown trend.

Looking at the trend, a user in figure 5.2 show 5 login KeyDown samples (sample number 1, 11, 21, 31 and 41) and the KeyDown profile. The sample graph show some inconsistency for this user, but a general trend can be found. There are some anomalies character 3 in sample 1, character 5 from sample 21 and 1, character 9 of sample 31. Sample 41 seems to be slightly lower on average than the rest of the samples, and more importantly lower than the profile. Sample 1 has the opposite quality, usually higher values than the other samples, and higher than the profile. This will be discussed in considerable detail in section 5.3 on profile growth.

Having looked at the attributes of one user, there is an indication of a trend that could be used to distinguish one user from the next. More information about comparing users is needed, and more specifically how accurate the calculated profiles are.

5.1.2 Profile Values

The KeyDown and KeyWait values granulates from 0 - 1500 microseconds (μs). This contributes for the average of a set of login samples. The sKeyDown and sKeyWait are derived from the uncertainty (variation) of the samples in the set of logins.

5.1.3 Entropy / Uncertainty

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. It defines the root mean square deviation from the mean, and will directly indicate how wide the spread the values in the dataset are [45, 46].

The mean (average) value of one variable (x) in our dataset can be described as:

$$\overline{x} = \frac{(x_1 + x_2 + \ldots + x_n)}{n} \tag{5.1}$$

The uncertainty can be formulated as:

$$\sigma = \sqrt{\frac{1}{n}\sum (x_i - \overline{x})^2} \tag{5.2}$$

5.1.4 Profile Analysis

Looking at the deduced profiles one can identify several interesting aspects. Table 5.2, 5.3 and 5.4 show three example users, of different computer proficiency. *User 14* is estimated at level 2 computer proficiency, *User 11* at level 3 and *User 1* at level 4.

User 1 Profile

User 1									
kd	112	88	134	120	89	83	107	117	85
kw	0	110	76	67	219	158	109	73	192
skd	20.54	10.91	14.48	12.70	11.33	13.65	19.32	11.17	10.51
skw	0.00	19.76	18.44	14.42	20.43	18.81	28.11	20.07	22.87

Table 5.2: Full user profile, with all logins for user 1.

User 1 (table 5.2 and figure 5.3) show a range from 83.0 μs to 134.0 μs KeyDown. KeyWait on the other hand ranges from 73 to 219 μs. KeyDown standard deviation ranges from 10.51 to 20.54 μs and for KeyWait 14.42 to 22.87 μs. Notice that

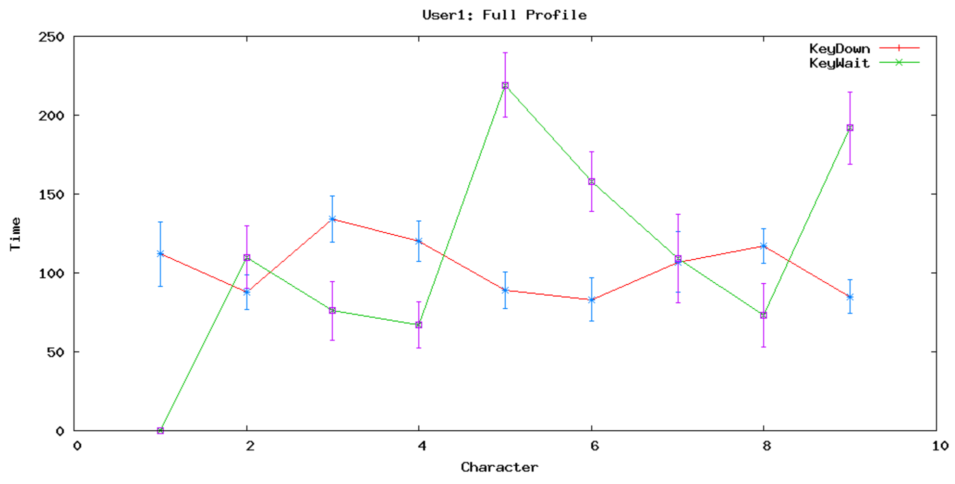


Figure 5.3: A graphical representation of the KeyDown and KeyWait parameters, with error bars showing deviation, for user 1.

KeyWait and sKeyWait will always be 0 as there is no waiting period before typing in the first character.

User 11 Profile

User 11									
kd	95	80	230	140	97	90	97	99	77
kw	0	244	386	151	480	465	364	405	213
skd	9.56	8.62	30.60	20.40	11.57	12.70	9.55	10.36	10.22
skw	0.00	54.27	119.86	21.39	253.85	226.92	268.03	224.16	46.82

Table 5.3: Full user profile, with all logins for user 11.

User 11 (table 5.3 and figure 5.4) KeyDown have values from 77.0 to 230.0 μs, not that different from the first user. Looking at KeyWait however the table offer values from 151.0 to 480.0 μs. This indicates a longer waiting period between characters than *User 1*. The uncertainty of KeyDown ranges from 8.62 to 30.60 μs and KeyWait from 46.82 to 253.85 μs. A higher range of variation indicate less familiarity with the password.

The fact that standard deviation on KeyDown is higher than the two other subjects is not given proper emphasis in the figure 5.4 due to the extreme values of KeyWait and sKeyWait. The graphical representation will therefore not show the

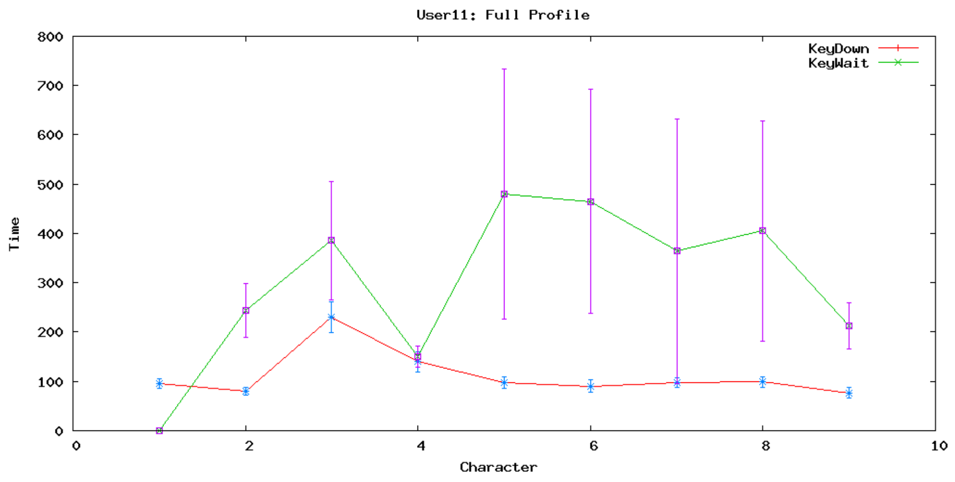


Figure 5.4: A graphical representation of the KeyDown and KeyWait parameters, with error bars showing deviation, for user 11.

differences in these parameters, but it can be read from the profile table 5.3. It will also be given more attention in the comparison sections. It does display the difference between KeyDown and KeyWait when focusing on the time property. Compared to user 1, the results show a huge value difference in KeyWait. This gives reason to hope a difference in profiles which would differentiate the users.

User 14 Profile

User 14									
kd	81	79	120	84	83	88	88	84	83
kw	0	920	467	208	589	493	336	449	489
skd	10.15	7.74	81.42	9.55	9.99	9.90	8.57	8.86	7.41
skw	0.00	660.11	221.29	76.18	416.17	512.98	113.43	237.98	379.22

Table 5.4: Full user profile, with all logins for user 14.

The last user in this example, *User 14* (table 5.4 and figure 5.5), show a KeyDown range of 83.0 to 120.0. Notice the consistency in KeyDown average values. KeyWait on the other hand springs from 208 to 920 μs. Looking at the deviation it proves to be between 7.41 to 81.42. This said, 8 of the 9 keystrokes runs close to 10, and one near 82. The anomaly keystroke spawns from the third character, namely the *shift button*. Standard deviation for the KeyWait period also varies greatly. Almost evenly distributed from 113.43 to 660.11. Considerably higher

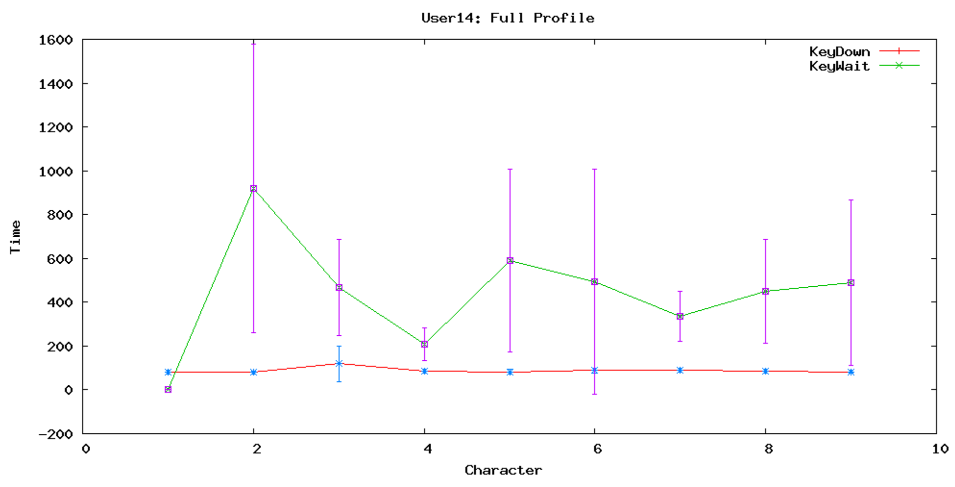


Figure 5.5: A graphical representation of the KeyDown and KeyWait parameters, with error bars showing deviation, for user 14.

than the other two profiles show. This user appears to be a very consistent but slow typist.

User 14 has the highest sKeyWait of all the profiles created. This may indicate that a vast number of users could succeed in having behavior which could falsely but successfully fall under the profile of this user. Large deviations increase the possibility for samples from other users to fall within the allowed limits. But still depend on the original *KeyWait* level, to make a successful comparison.

5.1.5 Sample Remarks

User	**KeyDown time**	**KeyWait time**	**Total Time**
User 1	935	1004	1939
User 11	1005	2708	3713
User 14	790	3951	4741

Table 5.5: Accumulated time values comparison, for users 1, 11 and 14, with respect to KeyDown, KeyWait and Total time in μs.

Looking at table 5.5 the three user profiles are listed in descending order by computer proficiency. An interesting fact to remark is the total time versus the total KeyDown time. Even though user 1 is almost 2800 μs faster than user 14, the

User	KeyDown σ	KeyDown σ avg	KeyWait σ	KeyWait σ avg.	Total σ
User 1	124,61	13,85	162,91	20,36	287.52
User 11	123,58	13,73	1214,30	151,78	1337,88
User 14	153.59	17,06	2617,36	327,17	2770,95

Table 5.6: Deviation comparison for users 1, 11 and 14. Showing total and mean KeyDown deviation, total and mean KeyWait deviation, and total deviation.

KeyDown time of the user 1 is higher than that of user 14. This indicate that the KeyDown characteristic has no direct relation to the proficiency level. The KeyWait time on the other hand, can be related to the given proficiency level for these users. This may also be a case of the users ability to memorize the password.

Table 5.6 gives an overview of the deviation range of the three users. Considering the consistency of user 1 and 11, a low deviation average of 14 pr. character for KeyDown can be identified. User 14 show close to 17 range standard deviation value pr character. Combining this knowledge with the information shown in table 5.5 gives hope to differentiate users. User 14, which has the lowest KeyDown time, has the highest deviation. Users 1 and 11 have similar KeyDown time and KeyDown deviation, however there is a large difference between the KeyWait and sKeyWait of these two users.

User 1 has a sKeyWait of 163 pr character in KeyWait, while user 11 and 14 are at 1214 and 2617 average deviation range. A concern at this point would be to separate the two latter users 11 and 14. But the difference in KeyDown could make this possible. The conclusion of this comparison, is that the data range and deviation range, considering both KeyDown and KeyWait, should be large enough to make an accurate match between the users.

Comparing these initial results with the estimated proficiency level of users (for all users, not only the selected ones in this section), makes way for the following argument: *The lower proficiency level, the higher KeyWait and sKeyWait*. This does not reflect the truth for KeyDown and sKeyDown, where the samples are spanning over less value differences. Anomalies can be found for both KeyDown and KeyWait, and thus the conclusion can only be an indication.

The figures 5.3, 5.4 and 5.5 represent the behavior of users on KeyDown, KeyWait, sKeyDown and sKeyWait. The value differences are so vast, that a conclusion cannot be drawn from this. The uncertainty is very high for many users. To show this, a comparison of all users with focus on the false acceptance rate (FAR) and false rejection rate (FRR) is needed[5].

5.2 Value Test

The two terms *accuracy* and *precision* is used to tell the quantifying difference in a set of data values. Accuracy is the degree of veracity (how close to the mean the values are) whilst the precision is the degree of reproducibility (how the results scatter) [6].

One weakness of using mean and standard deviation, is the lack of ability to recognize outliers or high anomalies which can corrupt the end result. For example the dataset $\{1, 1, 2, 3, 114\}$ will give a mean value of 24.2, while a median would be 2. The standard deviation will describe how spread the dataset is, but would not know the difference of two anomalies and one anomaly with the value of the two combined. So even if the profile is based on an accurate dataset, no knowledge of the precision is presented.

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, 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_i will have to be within the limits of $y_i \pm\sigma\ y_i$. For an acceptance rate of 3 times the standard deviation x_i will have to be within the limits of $y_i \pm(3*\sigma y_i)$.

In a normal *Gaussian distribution*, the standard deviation of a dataset is defined at precision level 1. It can be identified as a bell curve, identical on both sides of a mean. In a large dataset, and by definition, 68% of the data will be within 1 standard deviation (symmetrical with 34% on both sides). This accuracy rate is not accurate enough for a positive match as almost 35% of all logins would be considered anomalies. By doubling the standard deviation, 95% of the samples would be accepted. And 99.7% would be within 3 standard deviation, by the distribution definition known as *the empirical rule* [46, 45].

[5]The FAR would be where an intruder is falsely accepted into the system, while the FRR is where a legitimate users is rejected.

[6]A section on accuracy and precision can be found in the appendix B.2.

The level of success will be described as access (a) or positive match within the set limits for KeyDown and KeyWait. Higher or lower values than the profile value $y_i \pm l\sigma$, will be considered anomalies. Where this is true, akd or akw will be increased by one.

The akd describe the number of alerts or errors detected for the KeyDown parameter. Accordingly the akw value describe the alerts or errors were raised from the KeyWait parameter.

The following algorithm 1 describes the comparison done:

Algorithm 1 Value test: Login sample comparison algorithm

```
1:  for all $logins$ do
2:    $i$ =1
3:    $akd$ =0
4:    $akw$ =0
5:    for $i = 1$ to 9 do
6:      if $xkd \geq ykd + \sigma l$ then
7:        $akd = akd + 1$ {If exceeding kd upper limit}
8:      else if $xkd \leq ykd - \sigma l$ then
9:        $akd = akd + 1$ {If exceeding kd lower limit}
10:     end if
11:     if $xkw \geq ykw + \sigma l$ then
12:       $akw = akw + 1$ {If exceeding kw upper limit}
13:     else if $xkw \leq ykw - \sigma l$ then
14:       $akw = akw + 1$ {If exceeding kw lower limit}
15:     end if
16:   end for
17: end for
```

5.2.1 Alert Levels

When conducting the *Value test* (see Algorithm 1), for each login an *Alert Level* can be selected from the variable a. An additional algorithm 2 is run to determine *Alert Level.* If the sample had no errors for akd and akw would be 0, it 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.

Algorithm 2 Alert level algorithm

```
$al0...al4$ =0
**for all** $logins$ **do**
    $akdtot = akdtot + akd$ {Increase statistics}
    $akwtot = akwtot + akw$
    $atot = akd + akw$ {Total errors for login}
    **if** $atot \leq 0$ **then**
        $al0 = al0 + 1$ {Increase alertgroup 0}
    **else if** $atot \leq 1$ **then**
        $al1 = al1 + 1$ {Increase alertgroup 1}
    **else if** $atot \leq 2$ **then**
        $al2 = al2 + 1$ {Increase alertgroup 2}
    **else if** $atot \leq 3$ **then**
        $al3 = al3 + 1$ {Increase alertgroup 3}
    **else**
        $al4 = al4 + 1$ {Increase alertgroup 4}
    **end if**
**end for**
```

The *Alert Level* can be viewed as the level of strictness. If only *Alert Level* 0 is accepted, there is no tolerance for errors. There are eighteen (one pr character for both KeyDown and KeyWait) tests in total for each login. Failing for every keystroke, $atot$ would contain the value 18. *Alert Level*1 will accept one deviation from the eighteen tests. This gives a flexibility to adjust the strictness of the test.

The distribution of errors can also be seen with the changing of standard deviation level.

5.2.2 Full Profile FAR and FRR

The *Value test* is run for each standard deviation from 1.5 to 4 in order to find the lowest accepted FAR and FRR. The figure 5.6, show the FAR and FRR for *Alert Level* 0, with no mistakes accepted.

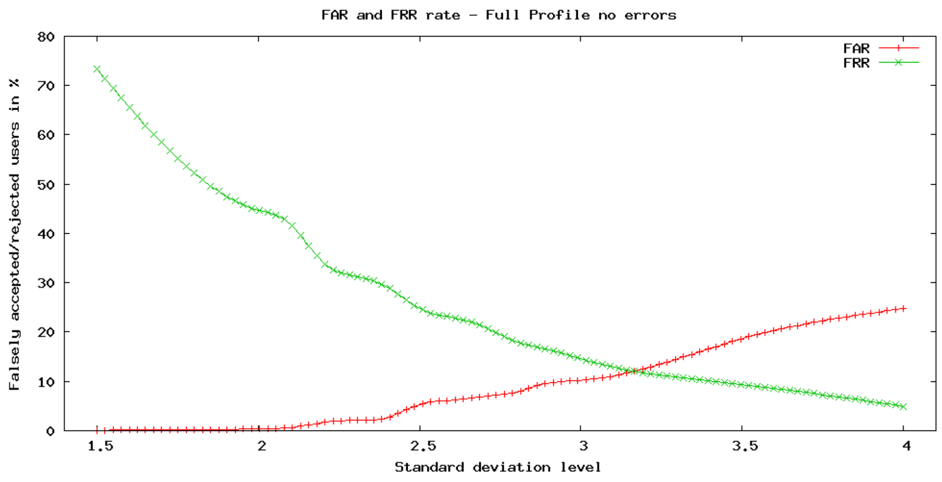


Figure 5.6: Representation of the False Acceptance Rate and False Rejection rate for standard deviation levels from 1.5 to 4, in percent (%).

The point where the FAR and FRR graphs cross, is the best possible outcome of the test. The standard deviation level in this point is at about 3.2, and it will result in a 12% false rejection rate and 12% false acceptance rate. A standard deviation of 3.2 should include more than 99.7% of the data to be within the standard deviation for the dataset of FRR. As only 88% of the samples for FRR is within the limits in the test, it indicates a change in the data pattern.

Increasing the accepted *Alert Level* to include one acceptable error (*Alert Level* 0 and *Alert Level*1), the graph (figure 5.7) shows the result.

Notice the increased steepness of both graphs. This indicates a higher precision with one error is accepted. Observe that both FAR and FRR are now crossing in around 2.7 standard deviation of the mean. By definition this should include more than 95% of the logins for FRR, however there is still only around 88% of the logins which are accepted as correct logins. And also the same amount of FAR where misuse has not been detected. There is no change in the accept or reject rate. Thus the accuracy of the data does not change with this setup. Again this indicates a change in the data collected over time. This is based on the fact that random numbers would not change if the dataset is large enough. This concludes that the numbers are infact not totaly random, but is affected by some (unknown) factor. It is believed that this factor is training.

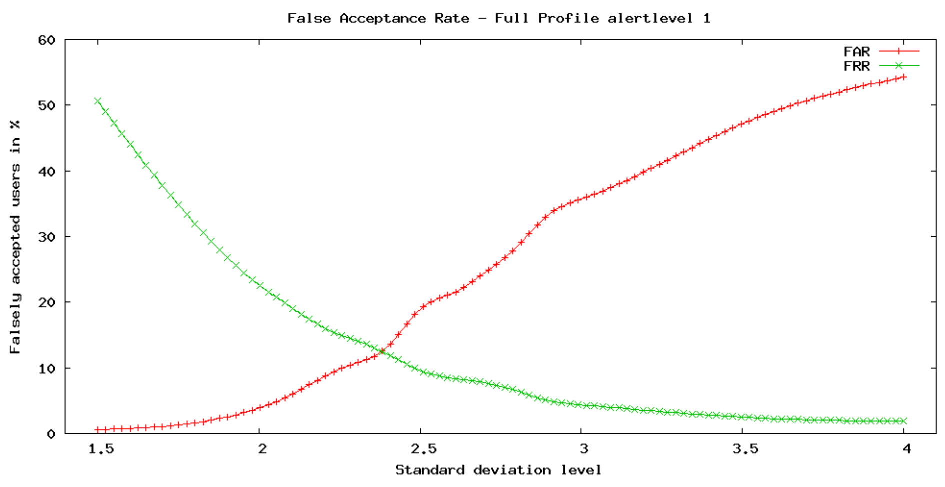


Figure 5.7: Representation of the False Acceptance Rate and False Rejection rate for standard deviation levels from 1.5 to 4, in percent (%). Accepting Alert Level 1 (one deviation from sample to profile).

5.2.3 Error Distribution

To gain more knowledge of where or why data does not match, a histogram of login error rate was created (5.8).

A user ($user19$) was selected for this example to show distribution. The y-axis show the number of errors for sample X, and the type of rejection (KeyDown or KeyWait). The x-axis presents a timeline from the first to the last successful login sample. A noticeable higher rate of errors spawn from the first 20 logins compared to the last 20. Such a trend displays an increased accuracy in typing, and a high rate of variability in the earliest logins.

5.3 Profile Growth

This section will discuss the profile growth and learning phase problems encountered in the results. An unacceptable FRR has been uncovered for the *Value test* conducted. However the majority of rejected logins with a valid user name and password, can be traced to the earliest of login samples. Therefore an investigation into the growth of a profile, meaning how the samples change over time and as the password gets more and more familiar, is needed to give more reliable

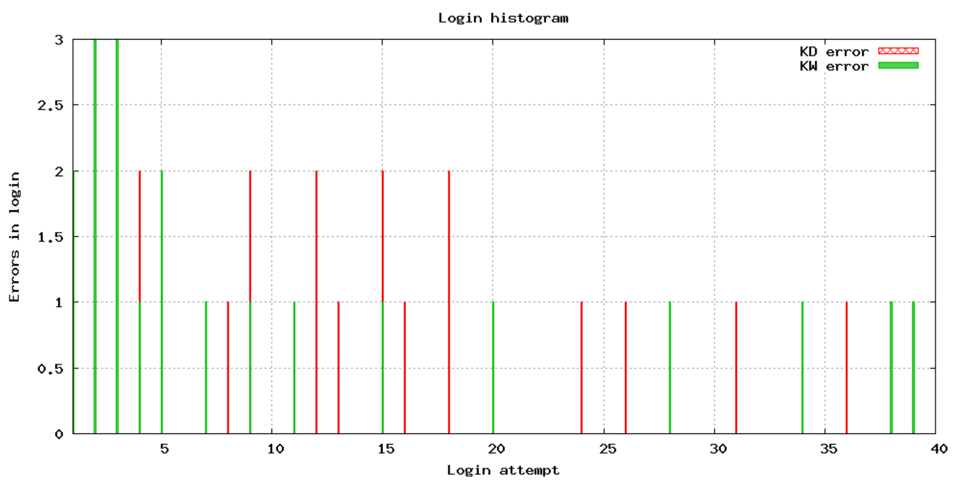


Figure 5.8: Alert distribution histogram showing the first 40 login samples for user 19.

results.

In order to show how profiles grow, the samples of each user is divided into two chunks. The first being the first 50% of the login attempts and the second being the last 50% of the login attempts. By directly comparing these for all users, a growth trend could be detected.

It is important to focus a bit on how the samples are collected, and how passwords are entered into the application. As discussed previously the users have gone through sessions each consisting of 15 logins. After each attempt the password should be easier and easier to remember. One problem when looking at user profile growth is that this is not accurately mirroring how a real world scenario would look like. The learning phase could start with many repeated logins to enable a base for a basic profile. But rarely a repeated 15 login attempt session would be conducted. A problem could arise between login 15 and 16, as there may be (and in this study is) a week between each session. Keystroke Dynamics is a quite sensitive biometric and one of the major issues is removing large anomalies in the learning phase. The data in these experiments is not necessarily absolutely accurate to a real life situation, and could possibly enable outliers and unwanted large differences between two consecutive logins, with a larger real world time gap between them.

Another fact to recognize is the way a repetitive process affect a person. When conducting the exact same action over and over, a clear and consistent profile could emerge from that. But the environment around the person could change. Some disturbance when entering the data could happen. A slip of the finger, or a consecutive run of incorrect login could make a user concentrate harder, and in such alter the behavior of previous attempts. In our data some few samples are very different than the overall behavior. In a real implementation of such a system, these data would be discarded. For gathering data on the other hand, this is important information to collect. The next section will present data, without the outliers, to get a more accurate statistic of the growth of a profile. Four profile attributes (from users 18 and 25) is disregarded from the KeyWait and sKeyWait samples. This is done only for the profile growth, not for any of the other tests.

When conducting these comparisons, two methods are selected for each category. Both the mean value, and the median of samples are calculated. Median is a stronger method in order to detect high anomalies, whereas mean is better at giving the general behavior. In other words, to detect the precision of the sample growth, median is used. In order to gain knowledge about the accuracy of the sample growth, the mean value is calculated.

The first profile from the first 50% logins is labeled p_1, the second profile with the last 50% of the logins is labeled p_2. One assumption is that as the familiarity and training of the passwords, the p_2 would consist of lower values than p_1 in general.

The following formula was used for this comparison:

$$growthrate = \frac{p_2}{p_1} \tag{5.3}$$

KeyDown Growth

The property of KeyDown characteristics has shown to be more consistent with lower standard deviation values, compared to KeyWait. The table 5.7 shows the growth summary for all users in percent (%). If the value is equal to 0, no growth is detected. A value of 5 means the average growth has been reduced by 5% from the newest to the earliest profile. A negative value tells of a increased growth time. This is also the case for the table 5.8.

As table 5.7 indicate, a general low growth rate can be found in most of the characters in the password. The highest average rate can be found in characters 1 and 6, with a 10% time decrease. The best median growth keystrokes are the characters

Char	1	2	3	4	5	6	7	8	9
Avg.	10,7	5,2	1,8	4,2	3,5	10.1	0,2	1,7	3,0
Med.	7,4	3,0	5,5	3,0	3,4	7,4	2,9	2,0	2,3
Avg.σ	14,9	8,9	22,7	2,8	-21,8	23,4	0,6	-6,0	13,0
Med.σ	15,2	8,3	31,3	11,5	-5,6	23,4	11,3	-3,6	9,2

Table 5.7: KeyDown profile growth comparison for each keypress, in percent (%).

1 and 6, with a decrease of 7%.

Looking at the deviation rates, the average for character 3, is 22% growth reduction which is the highest in the dataset. This would be the *shift-key* in the password. The 5th has a negative precision by almost 22%. The median for this character shows an increased deviation of 5%, which indicate a few high anomalies for this keystroke. However, an increased value (rank is above 0) show that the general growth is not as good as the other characters. The best growth rate for deviation can be found in character 3, with a 31% decrease. This, as mentioned, is the *Shift-Key*.

In general most values have decreased, and can be found more consistent in the latter profile. The median show better growth than the mean in most keystrokes. But some few values are also higher. This means that the precision is generally growing more than the accuracy, but both values are decreasing over time.

KeyWait Growth

In previous tests the range of values for KeyWait and sKeyWait have been generally higher than KeyDown and skd. There could be a potential for higher growth in these values. The table 5.8 show the results of the profile comparison.

Char	2	3	4	5	6	7	8	9
Avg.	26,6	3,3	5,7	17,8	4,9	25,5	10,5	20,8
Med.	22,9	6,3	12,3	18,9	22,8	24,7	22,4	17,6
Avg.σ	39,7	2,1	4,5	-12,1	11,8	38,7	47,2	40,5
Med.σ	65,0	24,5	27,9	-3,5	58,0	72,8	49,5	46,9

Table 5.8: KeyWait profile growth comparison for each keypress, in percent (%).

There can be found a higher variation of growth in the KeyWait parameter. Notice that the first character cannot be improved as it is always 0, and has been removed from the table as it has no significance to the results. The average KeyWait shows all character time values have decreased. The best growth can be found in character 2, 7 and 9, with 26%, 25% and 21%. The minimum growth can be found in character 3, which is somewhat unexpected. This can however suggest that there is a defined pattern concerning the *Shift-key* since it is dependent on another keystroke to complete, and its value is not dependent on itself alone as the other keystrokes are.

The median growth rates are always performing better than the average except for character 9. The lowest growth can also here be found in character 3 with only 6%. The best on the other hand are characters 2, 6, 7 and 8. All ranging from 20-22% decrease. Knowing that the profiles roam from generally 100 - 800 μs, an increase of 20% will mean a time decrease with 20-160 μs, which can be described as a high time improvement rate.

When examining the mean deviation values in the table, the average span from 47.2 to -12.1. The fifth character is the one following the release of the shift-button, and has the worst growth rate. It actually increases with 12%. The characters 2, 7, 8 and 9 decreases with almost 40 - 50%. The consistency of typing has on average improved more on the sKeyWait parameter than any other parameter in the tests. This also goes to prove the assumption stated that it had the highest improvement potential from the dataset.

The median values are consistently lower than the average, and quite a bit so. One can find in characters 7 and 2 with a decreased median rate of a stunning 65% and 73%. Character 2 is located close to character 1, and training seem to improve the time between these two. Character 7 on the other hand have no location or pattern wise qualities which could explain this phenomenon. A password possesses the trait of being memorable, but only for a given number of consecutive characters. This character 7 key, could be where the natural flow begins to fail, and a pause to remember the password sequence is needed. After repeated entries the entire password comes quickly to mind. There are no data that implies this is wrong, although no signs that this is a general trend could be found from the earlier evaluation of profiles.

The human *short-time memory* is able to store 7 (plus or minus 2) pieces of information [47, 48]. The limited capacity can be a possible reason to explain the latter growth phenomena. With training, the brain puts two pieces of information (keystrokes) into one piece, which reduces the total number of pieces in the information, and it becomes easier to remember. When the entire information set is put into one piece only, it is transferred into the *long-term memory*. This can explain the unusual growth rate of character 7. The brain process it, and merge several characters into one piece. Thus no pause is needed to think when entering the password. This is an interesting observation, and would need more investigation to be proved.

Comparison Overview

Looking at the bigger picture, a table 5.9 was created to show the total average and median growth for KeyDown, sKeyDown, KeyWait, sKeyWait, calculated from the growth rate of all characters. Again, a high positive number refers to better and more consistent user behavior.

Type	KeyDown	sKeyDown	KeyWait	sKeyWait
Mean	4,5	6,5	14,4	21,5
Median	4,1	11,2	18,5	36,2

Table 5.9: Total growth comparison with mean and median values, for the parameters KeyDown, KeyWait, KeyDown deviation and KeyWait deviation, in percent (%)

One can find a growth rate of about 5% for the KeyDown parameter, and a slightly better improvement rate for the sKeyDown. There is definitely a growth in the KeyDown characteristics behavior. And it becomes more consistent by 6-11% pr. character when looking at mean and median values. This may not be enough to have significant impact on the FAR and FRR, but could cause some differences due to the growth. Because of the difference in mean and median values, some anomalies could be expected, and impact the FRR.

KeyWait on the other hand have an average decreasing growth of 14% and the median shows 18% improvement. This could cause some direct impact on the test results for FRR and FAR. The sKeyWait average and median shows an improvement rate of 22 - 36% pr. character for all users. This implies a greatly improved consistency, and could mean a much lower FAR, if only the last 50% of the logins were used for tests and profiling.

5.3.1 Value Test on Grown Profiles

The growth of KeyWait and sKeyWait has shown indications of being influential. More influential than KeyDown and sKeyDown. In order to see its significance, another *Value test* was performed. The profiles were now based only on the last 50% of the successful logins. For FAR, all logins are tested against all new profiles. For FRR, only the last 50% of the logins are tested against the new profile of that user. This creates huge statistical difference, due to the fact that over 3000 logins will be tested for FAR pr. user, and a maximum of 30 logins can be tested for the FRR, as this approach cuts the number of logins by 50%. A statistical golden rule is: a minimum of 30 data samples is sufficient to get a valid result. More than 30 is often appreciated [45].

The first *Value test* was applied with no errors accepted in any of the 18 tests. Remembering the similar test done on a complete profile (figure 5.6), a standard deviation value of 3.2 gave the best result. As accepted profile ranges have become tighter, the rates should go down and stay generally lower for this test. For the FRR on the other hand, the profiles have also become more consistent, and not as open to errors as earlier. The general behavior of the users has with the growth of the profiles, also become stronger and more consistent. This could also lead to more similar typing.

The figure 5.9 shows the result of the new *Value test*.

Looking at the FAR graph (in figure 5.9) compared to the FAR graph for a full profile (figure 5.6), the accuracy has increased significantly. Even at standard deviation level 3, there are only 1% falsely accepted users out of the 3900 login attempts. This is 1/10 of the rate from a full profile. When increasing the allowed deviation even more, only a low increase in FAR % is found. At a standard deviation level of 4 (which is the max in these tests), the FAR is 6.87%. For a full profile the same allowed deviation level has a 24.76% false acceptance rate. This indicate a significant improvement in user verification, as the consistency has improved, making it easier to differentiate users.

Furthermore, the FRR has not improved. On the contrary it seems to reject more legitimate users on all standard deviation levels. Because of the increased consistency, also the accepted limits for FRR is stricter. Deviations in behavior which was accepted earlier would now be rejected. At a standard deviation level of 4, 7 out of 100 would experience rejected logins.

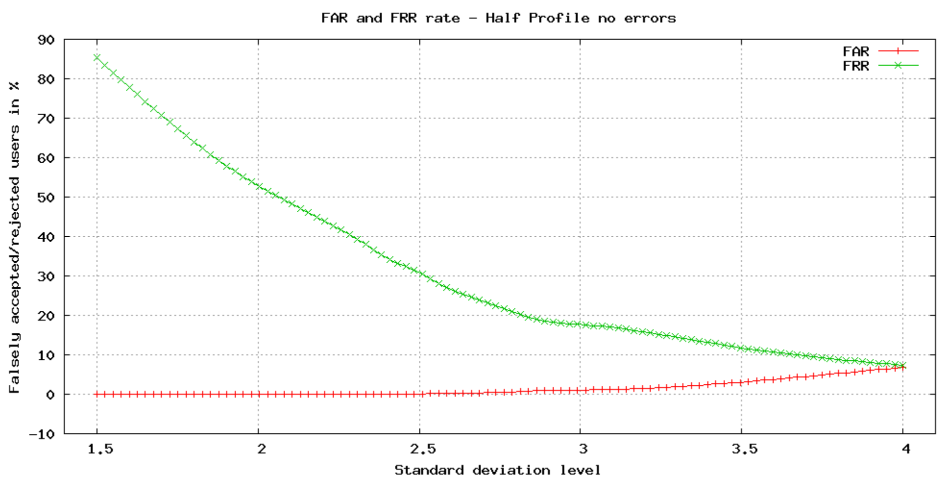


Figure 5.9: False Acceptance Rate and False Rejection Rate results on increasing standard deviation acceptance level from 1.5 to 4. Using type two profiles (last 50% of the logins), accepting no errors.

Because of the extremely low FAR, there could be room for looking at a slightly less conservative acceptance rate. The *Value test* was run again, now with one error (one deviation from the profile in either KeyDown or KeyWait) acceptance. The experience from this same operation for a full profiled proved to reduce the standard deviation level point, but increasing the percent rate for FAR and reduce the FRR. It is expected that this would be the result of this test. The result is shown in figure 5.10.

Somewhat unexpectedly, the FRR greatly improved. At a standard deviation level of 2.5, an FRR of 14% can be achieved. The FAR for this deviation level is 1.41%, which can be considered as very low. The linear pattern which could be found for *Alert Level* 0, now has bends and turns. That is a sign of the variation levels in the typing behavior the test are trying to detect. There are no significant changes in FRR from 2.5 standard deviation to 2.7, where it again begins to drop. From standard deviation level 2.9 to 4 there is a slow descending trend from 5.1% to 2.2%. Standard deviation level of 2.9 proves to be the meeting point of the graphs. The FAR in this point is 5.02% and the FRR is 5.1%. This is a considerable improvement in the FRR, while the FAR has only a slightly reduced rate, even with a lower acceptance limit.

One approach to find the best suited allowed standard deviation level is to calculate the mean error rate (MER), which is the mean value between the false acceptance rate and the false rejection rate. This gives a good indication of where the best

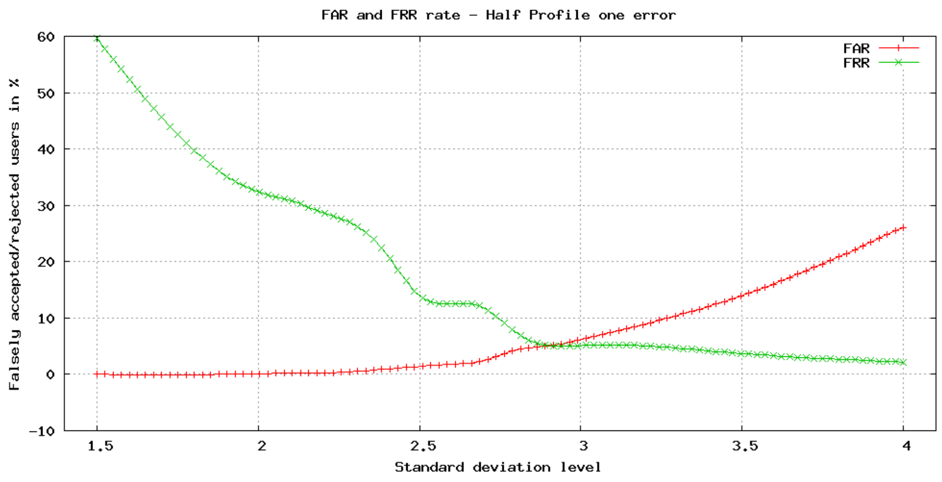


Figure 5.10: False Acceptance Rate and False Rejection Rate results on increasing standard deviation acceptance level from 1.5 to 4. Using type two profile (last 50% of the logins), accepting one login sample error to deviate from the profile.

mean result could be found.

$$MER = \frac{FAR}{FRR} \tag{5.4}$$

Another approach which is useful for studying the results is the Equal Error Rate(EER). This is describing at which point (if any) the FRR and FAR is equal. In table 5.10 the average percent rate between FRR and FAR has been calculated for the standard deviation. The same data can be found with a graphical representation in figure 5.11.

The table 5.10 and figure 5.11 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 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

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	**3.21**
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	**4.73**	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	**5.09**	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	**7.11**	14.10	22.40	22.4

Table 5.10: 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.

rates. 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 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

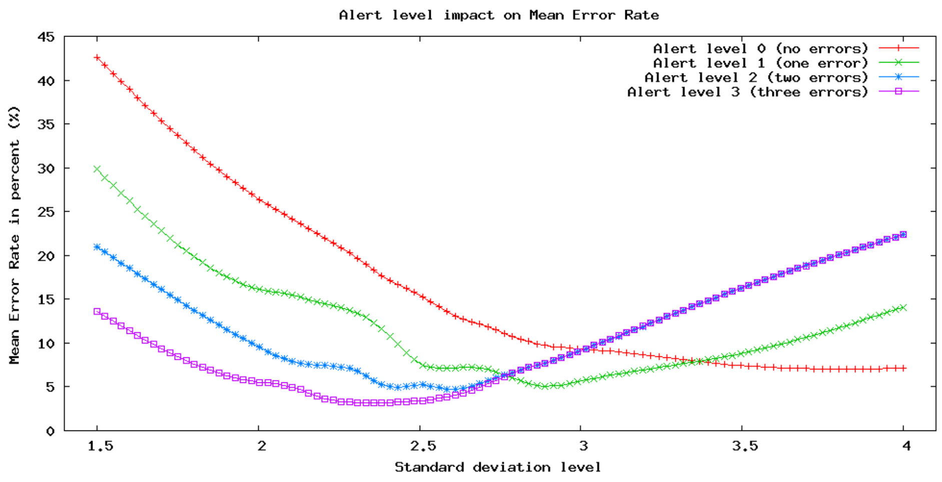


Figure 5.11: Graphical representation of the impact the Alert level, have on the Mean Error Rate with increasing standard deviation from 1.5 to 4.

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. The profile growth explains that the FRR actually can increase after a profile growth period, because the consistency increase leads to smaller accepted deviations. However by allowing some deviations, 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.

Error Distribution in percent (%)

Knowing the factor of FRR, a table 5.11 was created to visualize the distribution of errors. In this example, a standard deviation level (l) of 2.8 is used. For each user the number of successful logins was calculated in percent (%). Doing this enables a closer study of the error distribution. Each range of 5% is given a 'range slot'. The table shows how many percent of users had 100% successful logins, 95-99% successful logins and so on.

Make note that 64% of the users had no errors, which means that all the login attempts were successful. About 9% had over 95% of their logins accepted. No users were in the group where 90-94% of their logins were successful. More interestingly is the fact that 18% were within the range of 85% - 89%, and the rest of the users (9%) had less then 84% successful logins.

This shows that there is of a gap between the users. The largest group (73%) is always within 95% acceptance rate. There are however a group of users (around 27%) that generate a high amount of anomalous login attempts. The number of samples for the last FRR test is weak in a statistical overview. Only 136 login attempts can be tested. Outliers mentioned in the previous profile section, has not been removed. Identified behavior in denied logins, only a few users generate most of the errors.

Range	Distribution
100	63,63
95 - 99	9,09
90 - 94	0
85 - 89	18,18
- 84	9,09
Total	100

Table 5.11: The table describes the error distribution for users, represented by acceptance % error slots.

Though only a few of the users generates most of the errors, these are still valid users in a real scenario of users. What can be learned from the distribution is that not every user would have the same problem with logging in. After studying the rejected logins, it is clear that the behavior is very different on some keystrokes, and the decision to reject the login attempt seems to be correct.

When verifying the collected logins, 27% of the passwords was not matching the original password. These logins had to be discarded to create valid profiles. It however indicates that approximately every 3 out of 10 times a user type the password incorrectly, with respect to the keyCode value ordering. This contributes to a 30% FRR on password verification. Even when knowing the password well, and having typed it often daily for a longer period of time, the password is still entered incorrectly. Thus the human factor will always be a contributor to errors when it comes to behavior, also so when it comes to typing behavior.

Error Type Distribution

For each login sample in the *Value test* a counter was created. Each time a sample character was outside of the allowed limits, the type of error was stored in order to track what kind of errors were most frequent for each level of standard deviation. Because of time restraints, the granularity of this does not include the different *Alert Levels*. Figure 5.12 shows the distributed score for any KeyDown and KeyWait error in the test. One KeyDown is presented both for False Acceptance Rate and False Rejection Rate, as is the KeyWait parameter. As the numbers are presented in percent (%), notice that the graphs are opposite symmetrical. When KeyDown goes up, the KeyWait goes down with the same percent (%) value.

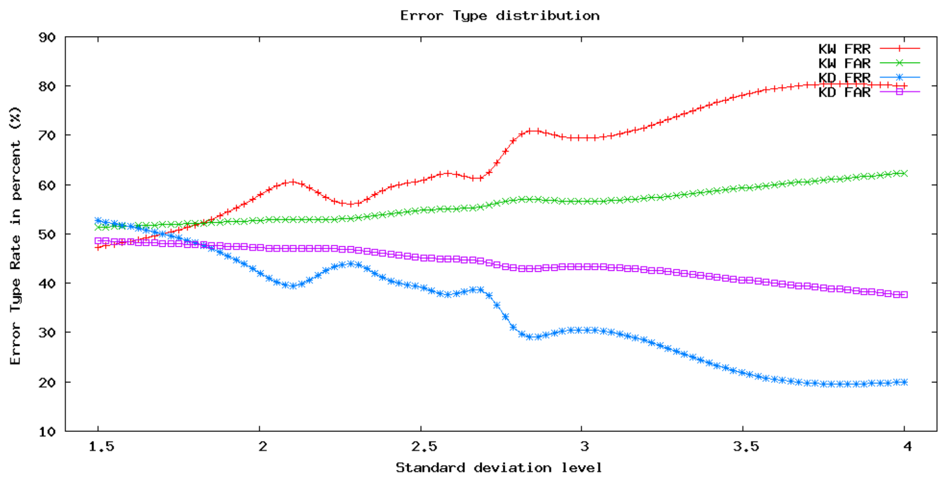


Figure 5.12: Error type distribution for False Acceptance Rate and False Rejection Rate with KeyDown and KeyWait parameters.

An interesting observation for FAR (in the figure 5.12) is the distribution itself. From a accepted 1.5 standard deviation level to 4.5 standard deviation, a less than 10% change is noticed. As the deviation levels increase, the frequency of KeyWait is increasing slowly. It is somewhat easier to separate users on KeyWait errors because of this.

When looking at the FRR, which has proved the most difficult to achieve, low standard deviation values seem to have higher accuracy for KeyDown than KeyWait. This changes with increasing accepted standard deviation. With standard deviation of 4, 80% of the errors detected are with the KeyWait property. The ranges found most usable in the tests (deviation levels from 2.5 - 3.0) have changes from 10-40% between KeyDown and KeyWait.

When considering these results, a suggestion can be made as the KeyDown is more consistent because of the low FRR %. Most falsely rejected logins come from deviations in KeyWait. The inconsistency for some users (see table 5.11) mostly come from KeyWait. However when examining the FAR both KeyDown and KeyWait are almost equally important to separate users.

5.4 Identification

The *Identification test* makes an effort to compare the current login sample to every available profile. It defines each keystroke as a point in a graph and compares the angle between the login sample and the profile keystroke. This makes every keystroke parameter, a dimension in a multidimensional graph of n dimensions.

When attempting to reject intruder logins from the keystroke parameters, as many parameters as possible is used in order to make the correct decision. The section on profile growth and the *Value test* give knowledge of the data and the uncertainty of data. Based on this a hypothesis can be made: *An accurate identification will be easier using the least deviating parameters of a login process.*

It is as important to identify users whose access is denied upon login, as it is to identify users who have accessed a system through another user account, making use of a password that is no longer secret.

5.4.1 General Identification

For a general identification, the test is looping through all logins. The result is a list of the top three closest candidate profiles. Algorithm 3 loops through all login samples, and for each login makes an identification comparison against all available profiles. It stores the top three (angle closest to 0) hits in variables. After looping through all the profiles these values are stored in the db table *tblTestResult* (db details in section 4.4.4).

Algorithm 3 Identification test comparison and sort

```
1: for all $logins$ do
2:     $first = 1$
3:     $second = 1$
4:     $third = 1$
5:     for all $profiles$ do
6:         $\theta = RUN : identAlgorithm$ {details in equation 3.6}
7:         if $\theta \geq first$ then
               $third = second\ second = first\ first = current$
8:         else if $\theta \geq second$ then
               $third = second\ second = current$
9:         else if $\theta \geq third$ then
               $third = current$
10:        end if
11:    end for
12:    RETURN orderedListToDB
13: end for
```

5.4.2 Analyzing Test Results

In order to obtain the best result, the test was run with different settings. The following tests were conducted:

KeyDown Full 1. *KeyDown Full Profile test will compare the sample to a full profile, disregarding the growth rate for KeyDown time.*

KeyDown Half 1. *KeyDown Half Profile test, consists of the last 50% of the samples as the profilebase, but only the KeyDown parameter is used.*

KeyDown and KeyWait Full 1. *KeyDown and KeyWait Full Profile use both the KeyDown and the KeyWait parameters, with a full profile comparison*

KeyDown and KeyWait Half 1. *In the KeyDown and KeyWait Half Profile test, both KeyDown and KeyWait time is used, but only the last 50% samples as profile*

When conducting these tests, some specific login types were not included. Because of the difference found when using different terminal types, only keyboard type 3 (laptop keyboard) was used. The uncertainty (variation) in the data is smaller, and thus will increase accuracy for logins. In a real time scenario one can never know if the intruder uses his/her regular terminal for logging in. In order to ensure quality in the result, only samples with similar conditions could be used.

Through the profile growth analyzing phase, it became apparent that KeyDown has a lower uncertainty (higher consistency) in general than KeyWait. This is why a test with only KeyDown and one with KeyDown and KeyWait was run. It was expected that the success rate would be higher with only the KeyDown parameter.

When saving the top three closest profiles to a sample, it is also possible to determine the distribution of the hit rate. After the verification test, a test to compare the identification position, to the actual login user was done. If the original user was first on the list, the test is classified as first hit. If the actual user was found in the second position of the result list, the test is classified as second hit. And for the third position, the classification would be third hit. If the actual user was not found in any of the top three spots, the test would be classified as unsuccessful for identifying the user. In these tests all the login attempts for all users were used (except the samples mentioned), regardless of failing or succeeding in actually logging in (succeed in the *Value test*). This is interesting as it gives an initial knowledge of how hard it is to identify users in general.

	kd full	**kd half**	**kd and kw full**	**kd and kw half**
First hit	*64.84*	62.24	60.42	56.77
Second hit	6.77	9.90	9.64	8.07
Third hit	3.39	2.86	5.99	4.69
Top three	75.00	75.00	*76.04*	69.53
Unsuccessful	25.00	25.00	23.96	30.47

Table 5.12: Identification success in percent (%), for KeyDown full profiles, KeyDown half profile (last 50% of samples only), KeyDown and KeyWait full profiles, KeyDown and KeyWait half profiles (last 50% of samples only).

The table 5.12 shows the result for the four test types and the hit rate for each test classification. The best success rate can be found with a KeyDown full profile test with 64.84% accurate matches. There is a 75% chance for the user to be amongst the top three users. Only for 10% of the logins, could the user be found among the second or third position. This indicate that the user behavior for most logins are either accurate or inaccurate, as only a few could be found in the list without a *first hit* match.

For KeyDown half profile one should expect a better accuracy rate as there are less uncertainty in the KeyDown parameter with training in entering the password. But this is only true when comparing with the last 50% of the samples. There is no knowledge of how many attempts or how strong an intruder profile is. This is why no tests are run for only the latter half of the login attempts in the *Identification test*. Doing such a test would give a better result, but it is improbable that an intruder has written the password enough time to get a good flow. We do not care about the success or failiure to enter as another user, merely looking at how easily a user is recognizable in a given input from a keyboard.

Classification success was reduces by 2.6%, but increased for second hit classification by attaining a 9.9% hit rate. With a 2.86% third hit result, a total of 75% of the original user in the logins could be identified among the top three users. This nuber is exactly the same as with a KeyDown Full sample test. This is an indication to suggest a low profile growth for KeyDown. The number of successful identifications of first hit decreases somewhat, which probably come from early logins which no longer match the profile when the training gets better. It is however somewhat surprising that the number of unsuccessful identifications did not increase. Even with the profile growth, the KeyDown parameters still lie close to the profile for both profile types.

The test with both parameters for full profiles was expected to be lower compared to only one parameter because of the high standard deviation of KeyWait. With a successful identification of user for first class of 60.42% the expectations were met. The KeyWait can be considered more fragile than KeyDown, but interestingly enough the third hit classification group increased so the total amount where the original user could be found in one of the three top classifications actually increased by 1.04% compared to the first two tests.

The trend found in KeyDown half profile also became apparent for both parameter half profile test. The total identification rate is found as low as 69.53% with a successful first hit rate of 56.77%.

Looking at the entire picture raises argument to continue with only the KeyDown parameter. This is where the identification rate of first hit is the highest, even if a combined KeyDown and KeyWait full profile test gives a higher top three rate, it is considered more essential to attain first hit successful classifications.

Identifying Falsely Accepted Users

One of the goals of the test is to indicate the original user behind a hacked login. With this comes the ability to find the actual user that falsely succeeded to login as another user. For accomplishing this the tests which was tagged as falsely accepted in the *tblTestResult* as tested for genuine original user. Again the three levels of classification is used. The table 5.13 presents the result of the test.

First hit	50.76
Second hit	8.20
Third hit	3.65
Top three	62.61
Unsuccessful	37.39

Table 5.13: Identification of falsely accepted users in the Value test in percent (%)

A decrease in successful first hit identifications can be shown in 50.76%, but also an increase in second hit classifications with 8.20% identification rate. A total of 62.61% of the users could be identified within the three classification groups, and accordingly a 37.39% could not be found among the top three most probable groups.

5.5 Pattern Recognition Test

The *Value test* makes an attempt to set boundaries for accepting or rejecting a login based on an average value and a deviation. The *Identification test* uses a mathematical calculation to find the closest matching profile to a sample. None of the above tests manage to catch user patterns. A user pattern is not based on the values itself, but in the rhythm in which the characters are entered with. This approach tries to catch whether the values in the login attempt increase where it is supposed to increase, or decrease where it usually decreases, not as much how much they go up or down. Depending on the distribution of the data, an approach for multiple models can be selected.

5.5.1 Wilcoxon Test

The Wilcoxon rank-sum test was the first model tested. An advantage with the Wilcoxon test is the fact that it is independent of distribution. This could give an indication to how distinguishable samples are. A description of the test can be found in the appendix B.1.

The first step is to define a range to distribute the keystrokes scale. Our tests will always have 18 parameters, over the interval 0 to 1500 (1500 μs is the max latency for an allowed KeyWait and max KeyDown for a single keystroke). The granularity of this scale would be too large, and not give a sufficient answer in the comparison. The data obtained suggests a 15 μ latency in the input phase, so a maximum effective scale would be 1500/15 = 100. This means every 15 μ will have a scale slot, the first being values from 0 to 14, second being 15-29 etc. This also proved to be too much granularity to be able to distinguish the users. So the scale was set to 10 chunks. The number of samples for the users within one time slot is counted and entered into the new scale.

Secondly, appropriate users must be selected. Looking at the tests *user7* and *user11* had differences in behavior that should be sufficient to separate then in such a test. The average of the first 30 samples for each of the users were used, and the result entered into the correct time slot in the selected scale. Notice that by doing this, the results have been ordered from lowest to highest in the scale.

	User 7	**User 11**
Slot 0	6	5
Slot 1	13	9
Slot 2	4	11
Slot 3	3	3
Slot 4	3	2
Slot 5	0	0
Slot 6	1	0
Slot 7	0	0
Slot 8	0	0
Slot 9	0	0

Table 5.14: Slot distribution for Wilcoxon test

The table 5.14 shows the distribution of values over the selected slots. *User 7* have 6 values in time slot 0, while User 11 has 5. For slot 1 the first user has 13 and the second user only 9 values.

To calculate if there is a significant difference between the samples, different significance levels can be chosen (0.1, 0.05 or 0.01). The higher number, the lower need for difference is needed for a determination of significant difference. The eye can see that the distribution of these data is different, especially for slot 2, where *user7* have 4 values and *user11* 11 values in that range.

The result of this test is that none of the significance levels make any distinctions between the samples. This model cannot be used to distinguish pattern differences between the user samples.

5.5.2 Other Models

Another model which has a flexible shape is the Weibull distribution model (details on the Weibull model can be found here [49]). One can choose the exponential, but the model is bound to a symmetrical pattern on both sides of the mean value. No applicable result came out of using calculation with this model.

The extreme value distribution (details can be found in [50]), is not necessarily symmetrical, but rather focus on an uneven distribution of values. One has the flexibility to adjust the acceptance levels for testing the data. This model also failed to acquire any statistical result which could be accepted.

When considering the profile growth, increasing accuracy and precision over time, this could very well be the reason for the failing of these models. It is reasonable to assume that fully grown profiles does not have as many anomal samples deviating strongly from regular behavior. Also through some control unit when accepting a login could also solve some of these problems to get more evenly distributed data. This would be acceptable for a proof of concept test. In a real-life situation the data are as they are, and the tests should be able to apply even if users have high deviations in the enrollment phase.

Chapter 6

Conclusion

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. The test proved increasingly accurate when considering profile growth, because process repetition leads to increased user behavior consistency.

Changing terminal type (for example from a laptop keyboard to desktop keyboard) seem to sufficiently affect behavior, causing legitimate logins to be rejected in the *Value Test*. It is advisable to regard each user profile dependent on the terminal type, in which the profile entries are built.

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 [4, 39].

Using techniques to evaluate error distribution, the *KeyWait* parameter was found more effective in detecting behavior differences. It became apparent that a small group (27% of the users), were generating most of the errors (almost 80%), which affected the *False Rejection Rate*.

The goal of the *Identification test* was to identify the subject behind a login attempt, when disregarding the username. It was developed to compare a sample to every available profile, determining which profile is closest. This study was able to correctly identify 65% of all login attempts, and approximately identify (sample owner among the top three closest profiles) 75% of the users, based purely on the sample values. Investigating logins which would be falsely accepted in the *Value test*, 51% of the logins matched closest the profile of the hacker. This means that 51% of the hacks can be traced back to the imposter, when succeeding in logging into another user account.

This study was unable to find an appropriate statistical model to accurately correlate typing patterns to a specific user, or differentiate samples from multiple users. The *Pattern Recognition test* made an effort to describe the data distribution model through; the *Wilcoxon* model, the *Weibull* distribution model and the *Extreme Value* distribution.

This thesis made efforts in hardening a login authentication scheme. Furthermore, identify illegitimate user logins, and combine results from the tests using joint decision theory. Applying methods such as in this study can increase the security of a traditional user authentication process by 97%. It can, with a 51% accuracy, identifying the user origin of a login hack, assuming the user has a registered profile. Because of the identification rate and no pattern recognition results, there were not enough conclusive parameters available, to make a combined test authentication scheme.

Establishing user proficiency groups, made it possible to draw the following conclusions. Proficient users have less sample variation and are harder to break into, but are also more vulnerable to rejected logins. Less proficient users increases their typing consistency more and faster. Their inconsistent behavior increase the possibility of being hacked, since more variation is accepted. It is feasible that a malicious user would have highest probability of successfully hacking into a login of a user at the same proficiency level.

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. The *Identification test*, based purely on sample values, can also be considered unique for short pass phrases (like a password). There are few studies which has achieved equal to, or more satisfying, *Mean Error Rate* than this work. Joyce and Gupta [39] 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% [51].

The cost, in terms of implementation and hardware, could be greatly reduced compared to physical biometric methods. The administration cost would probably remain the same for any system dealing with enrollment and profile building, and human anomalies. It is considered relatively cheap in regards to resource demands in a system. Ethically, a keystroke dynamic authentication scheme could be considered less intrusive and less discriminating, as any user of the system would always be able to enroll (*Failed Enrollment Rate* is 0), without giving away any sensitive information.

A keystroke dynamic authentication implementation can be used to establish a higher security level for a login scheme, in order to avoid compromised systems due to password theft or the like. It will also stand very strong against computer generated login attempts, because of the extremely low *KeyDown* and *KeyWait* time. This having been said, a biometric keystroke dynamic feature as described, is alone not sufficient for high security demanding environments.

Chapter 7

Discussion and Future Work

This section will focus on discussion around some of the challenges found in the thesis. It will also shed light on weaknesses and suggest paths of improvement.

7.1 Samples

When analyzing and testing, it is very important to have a proper amount of data. This work would benefit from having more samples and users. The task of acquiring large amounts of data is formidable, especially with the time constraints of a thesis in mind. Although the setup enabled using a sample both as an impostor login and a genuine login for all users, the *False Rejection Rate* is based on samples from one user only. Users with high variance or many anomaly logins, greatly affect the end result. A larger amount of users could possibly even out, improve or impair the test results.

An average profile growth from between 10-30% was observed for the training phase. With only 60 logins for each user, it would be very interesting to see the continuous growth after for example 100, 150 and 200 logins. It is expected that both the *False Acceptance Rate* and the *False Rejection Rate* would improve with even more samples.

In the learning phase, all samples were considered valid (assuming a correctly ordered sequence of *KeyCodes*). This caused some problems due to samples with extreme values. This was discussed with respect to the difference between the mean and the median rates of profile growth. High anomalies would increase the chance of falsely accepted logins, and reduce the chance of falsely rejected logins. It is however undesirable to base profiles on samples with high anomal values for one or more keystrokes. A more strict set of rules could be applied in filtering samples for a profile, to obtain more accurate profiles, which would result in more trustworthy results.

7.2 Experiencing a false login

Unpublished research observes as high as a 30% of regular passwords entered incorrectly. The natural reaction to a blocked login, would for many be to slow down the typing and concentrate, to get the password right. When using keystroke dynamic authentication, there is a high probability that the login would be rejected because of extra "concentration" effort. This could result in a bad circle, ending up with a very annoyed user.

The same goes for a person logging in when talking on the phone, drinking coffee or being generally distracted. The suggested solution is imperfect, but enabling the user to see what actually failed (wrong password or invalid sample), could help solving this problem. On the other hand this would give a possible intruder useful information.

7.3 Scalability

A login authentication process will take place on a client computer more often than not. The client will send the obtained login sample to an authentication service for verification. With a keystroke dynamic authentication module installed, more data would have to be sent to the authentication server than compared to a traditional authentication scheme. The server would also have to perform more calculations for each login. This would increase the server load and network traffic in a system, and increasingly so if there are many users with a high amount of repeated logins. Making use of local copies and local calculations could solve this issue. Implementing a keystroke dynamic authentication module is not expected to make vital impact on system resources, due to the low amount of resources needed for the calculations, and the data transfers.

Another scalability problem could arise with a high amount of users on a system. How this system would function in an environment of several thousand users, is not known. The *Identification test* could positively identify only 64% of all users in this setup. This rate would probably go down if there were a vastly increased number of profiles to compare with. One solution could be to use more parameters (include *KeyWait*). This should be inspected closer as further work.

7.4 Terminal Types

The problem of differentiating behavior for one user on multiple terminal types, force some restraints onto a system like this. One learning phase for each terminal type is a possible solution. Only a few users are frequently changing main terminals, and those who do frequently access many different machines, often remote login to these without physically move. However, there are examples of environments where this is not true, and a keystroke dynamic authentication model could be less desirable as the administration of the system, and the training process, would require a lot of time.

7.5 Proficiency

Three subjects were removed from the sample database, due to a low sample rate. Because of invalid use of the shift-button, very few logins were accepted. A constraint of a maximum of 1500 μs *KeyDown* time is set by the keyboard. When holding down a key longer than that, the keyboard signal will be repeated. This is not visible in the login application, but can be read off the logs. All these samples were disregarded by the *KeyCode* filtering, and not entered as valid samples. In order to accept these samples, a more advanced filtering engine needs to be developed. This behavior is very characteristic for the subjects, and could be very hard to mimic. There is a possibility of these users having a high deviation rate, but because their behavior clearly differs greatly from the rest of the subjects, the *False Acceptance Rate* would be very low. All these subjects were considered to be at the lowest proficiency level, novice users. Deeper knowledge about this could be interesting, as they could prove harder to hack then normal users.

7.6 Biometric Alterations

A profile is defined by a usual behavioral pattern. Some events could possibly alter the biometric behavior of a user. For example, breaking an arm or finger. Chances are, behavior would be so different from the original profile, a new profile with a new learning phase would have to be introduced. This is considered to be a administrative problem.

7.7 Password Samples

Results presented in this work, is based on a setup where the password used is the same for every user. This simulates an environment where all users, except for the profile owner, are considered malicious. They have attained the password, and use this information to access the system. Since they also go through a training phase, the pattern, in which the password is written, will grow over time. One advantage of this is that one can observe attacks throughout the entire growth process. This give insight in how unique a profile is, compared both to a trained and an untrained user.

In a real situation it is not probable that an intruder has spent time to train. In that respect it would be plausible to assume that the behavior from the first login attempts would be more like a real intruder. As this work has indicated, the earlier logins have a smaller chance to succeed than after training, especially when comparing to a fully grown profile. There is reason to believe, the *False Acceptance Rate* would decrease when testing only a few logins attempts with many users, compared to many login attempts from few users, as has been the case in this study.

It is also not probable that all users have selected the same password. If the profiles are based on different passwords for each user, an interesting study would be to see how writing patterns develop for completely different sets of passwords. Another fact that could be investigated is looking at the values itself, disregarding the *KeyCodes*. A study on how different typing behavior for any password string, could give valuable information in how important the actual password phrase is, compared to the *KeyDown* and *KeyWait* values. Especially interesting is the effect this would have on the *Identification test*.

A study of how different behavior grouped on specific keys or characters is also suggested. Examining the differences between all users on keystroke `a`, `b` etc, could also help understanding relationships between short keystroke combinations, or one key only. This would also benefit a later study of free text user identification. Expanding this study to investigate often used keystroke combinations, for example `the`, could enable deeper knowledge to also improve free text comparisons.

Bibliography

[1] Nick Bartlow. Username and password verification through keystroke dynamics. Master thesis, College of Engineering and Mineral Resources West Virginia University, 2005.

[2] GartnerGroup. Gartner security conference. In *Gartner Annual IT Security Summit*, number 12 in 3, 2006.

[3] Dieter Gollman. *Computer Security*. Wiley Publishing, 4 edition, 2004.

[4] Fabian Monrose and Avivel Rubin. Keystroke dynamics as biometric for authentication. *AT&T and New York University*, pages 48–56, 1997.

[5] RSA security conference, editor. *Selecting secure passwords*, Cryptographers Track, 2007.

[6] ISC, editor. *Practical Authenticated Key Agreement using Passwords*, volume 3225 of *p1-12*. Springer-Verlag, 2004.

[7] Yan et al. The memorability and security of passwords. Technical report, Cambridge University Computer Laboratory, 2004.

[8] IEEE Communication Letters, editor. *Security Analysis and Improvement of the Efficient Password-based Authentication Protocol*, volume 9. IEEE, 2005.

[9] Naumann Denning. Requirements and model for ides - a real-time intrusion detection system. *Computer Science Laboratory, SRI International*, 1985.

[10] Brodely Lane. Temporal sequence learning and data reduction for anomaly detection. In *ACM Conference on Computer and Communications Security*, volume 2 of *3*, pages 295–331. ACM Conference on Computer and Communications Security, ACM, 1998.

[11] Jones and Sielken. Computer system intrusion detection. US Defence Research Project Agency, 2000.

[12] Ohwada et al Totsuka. Network-based intrution detection -modeling for a larger picture. *Usenix LISA Paper*, 2002. NTT, Cyber Solutions and Tohoku University.

[13] Nullari and Kar. A web based approach to id. In *A web based approach to ID*. Consortium for Computing Siences, 2001.

[14] Sekar et al. Specification-based anomaly detection: A new aprroach for detecting network intrusions. In *Proceedings at ACM Conference on Computer and communication security*, volume 1, 2002. Stony Brook UNI.

[15] Park Lee and Stolfo. Automated intrusion detection using nfr: Methods and experiences. In *Proceeding in the 1st USENIX Workshop in Intrusion detection network monitoring*, volume 1, 1999. Columbia University.

[16] Upadhyaya Chinchani and Kwiat. Towards the scalable implementation of a user level anomaly detection system. In *MILCOM 2002 Conference Proceedings*, 2002. State University of Buffalo and Air Force research Labs.

[17] Brodley and Pusara. User re-authentication via mouse movements. In *ACM Proceedings of the 2004 October Workshop*, pages 1–8, 2004. Tufts University.

[18] Northcut and Novac. *Network Intrusion Detection*. New Riders, 3 edition, 2002.

[19] Robert Erbacker. Misuse detection in large scale systems. In *CG&A Editorial Calender*, volume 22, pages 38–47, 2002.

[20] Snort: www.snort.org. website, Last visited 15th of Febuary 2007.

[21] Pieter de Boer and Martin Pels. Host-based intrusion detection systems. Amsterdam University, 2005.

[22] Tripwire: http://www.tripwire.org. website, Last visited 15th of Febuary 2007.

[23] Aide: http://sourceforge.net/projects/aide. website, Last visited 15th of Febuary 2007.

[24] Portsentry: http://sourceforge.net/projects/sentrytools. website, Last visited 15th of Febuary 2007.

[25] Leonard Padulla. State of the art in intrusion detection and reaction. Technical report, Mitre Institute / US Air-Force, 1999.

[26] Baik et al Bala. Application of a distributed data mining approach to network intrusion detection. In *Autonomus agents and multiagent setup*, volume 3, pages 1419–1420, 2002.

[27] Anil Somayaji Stephanie Forrest, Steven Hofmeyr and Thomas Longstaff. A sence if self for unix processes. *IEEE Symposium on Security and Pricacy*, 1996. IEEE Computer Press.

[28] Sandeep Kumar. Classification and detection of computer intrusions. Master's thesis, Purdue University, 1995.

[29] Sirohey Chellappa, Wilson. Human and machine recognition of human face images. In *Proceedings of the IEEE*, volume 1, pages 705–741, 1990.

[30] Kersta. Voiceprint identification. *Nature Magazine 1256*, dec 1962. p1253-1257.

[31] Isenor and Zaky. Fingerprint identification using graph matching. In *Pattern Recognition Letters*, volume 19, pages 39–43, 1986.

[32] Maio et al. Fvc2000: fingerprint verification competition. In *Pattern Analysis and Machine Intelligence, on*, pages 402–412. Dipt. di Elettronica, Inf. e Sistemistica, Bologna Univ, Consortium for Computing Siences, 2002.

[33] Mansfield et al. Biometric product testing. Technical report, CESG, 2001.

[34] Fuller Kang and Honavar. Learning classifiers for misue detection using a bag of system calls. In *Proceeding at Intelligence and Security Informatis*, volume 3495, pages 511–516, 2005.

[35] Chris Pollett Shivani Hashia and Mark Stamp. On using mouse movements as biometric. In *Proceeding in the International Conference on Computer Science and its Applications*, volume 1, 2005. San Jose University.

[36] Stamp Hashia and Pollett. On using mouse movements as a biometric. In *Proceedings of 3rd Conference on Computer Science and its Applications*, volume 3, page N/A, 2005.

[37] Nisenson et al. Towards behaviometric security systems: Learning to identify a typist. *European Conference on Principles and Practice of Knowledge Discovery*, 7:363–374, 2003.

[38] Gamboa and Fred. An identity authentication system based on human computer interaction behaviour. In *Proceedings of SPIE*, volume 5404, pages 381–392, 2004.

[39] Rick Joyce and Gopal Gupta. Identity authentication based on keystroke latencies. *Communications of the ACM*, 1990.

[40] 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.

[41] Gunetti and Picardi. Keystroke analysis of free text. In *ACM Transactions on Information and System Security*, volume 8, pages 312–347, 2005.

[42] S.Press R. Gains, W. Lisowski and N. Shapiro. Authentication by keystroke timing. *Rand report*, 2001. R-2526-NSF.

[43] Picardi Gunetti and Ruffo. Keystroke analysis of different languages: A case study. In *Advances in Intelligent Data Analysis VI*, volume 3646, pages 133–144. Springer Berlin, 2005.

[44] Andries Brouwer. The linux keyboard driver. 14:5–15, 1995.

[45] Volden Foosnæs, Halvorsen. *Statistikk en innføring*. Fagbokforlaget, 2003.

[46] Gunnar Løvås. *Statistikk for universiteter og Høgskoler*. Universitetsforlaget, 2004.

[47] Abowd Dix, Finlaym and Beale. *Human Computer Interaction*. Prentice-Hall, 3rd edition, 2004.

[48] George Miller. The magical number seven plus or minus two: Some limits on our capacity for processing information. 63:81–97, 1956.

[49] Xie Murthy and Jiang. *Weibull Models (Wiley Series in Probability and Statistics)*. Wiley-Interscience, 2003.

[50] Stuart Coles. *An Introduction to Statistical Modeling of Extreme Values*. Springer, 2001.

[51] Gunetti Bergadano and Picardi. User authentication through keystroke dynamics. In *Information and System Security*, volume 5, page n/a. ACM Transactions, 2002.

[52] Burr et al. Electronic authentication guideline. Technical report, National Institute of Standards and Technology, http://csrc.nist.gov/publications/nistpubs/800-63/SP800-63V1_0_2.pdf, 2006.

[53] Techtarget: http://whatis.techtarget.com/definition/. website, Last visited 10th of March 2007.

[54] Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1:80–83, 1945.

Appendix A

Terms

This following chapter covers the main terminology used within the fields of computer security, biometrics and intrusion detection. The following list describe the terms in detail, and relate these to one another [52, 53, 3].

Term	Description
Active Attack	An attack on the authentication protocol where the attacker transmits data to the claimant or verifier. Examples of active attacks include a man-in-the-middle, impersonation, and session hijacking.
Active Impostor Acceptance	When an impostor submits a modified, simulated or reproduced biometric sample, intentionally attempting to relate it to another person who is an enrollee, and is incorrectly identified or verified by a biometric system as being that enrollee.
Asymmetric keys	Two related keys, a public key and a private key that are used to perform complementary operations, such as encryption and decryption or signature generation and signature verification.
Application Developer	An individual entrusted with developing and implementing a biometric application.
Attack	An attempt to obtain a subscribers token or to fool a verifier into believing that an unauthorized individual possess a claimants token.
Attempt	The submission of a biometric sample to a biometric system for identification or verification. A biometric system may allow more than one attempt to identify or verify.
Authentication	The process of establishing confidence in user identities. Often comparing a submitted biometric sample against the biometric reference template of a single enrollee whose identity is being claimed, to determine whether it matches the enrollee's template. Contrast with 'Identification.' The preferred biometric term is 'Verification.'
Automatic ID/Auto ID	An umbrella term for any biometric system or other security technology that uses automatic means to check identity. This applies to both one-to-one verification and one-to-many identification.
Behavioral Biometric	A biometric which is characterized by a behavioral trait that is learned and acquired over time rather than a physiological characteristic.

Term	Description
Biometric	A measurable, physical characteristic or personal behavioral trait used to recognize the identity, or verify the claimed identity, of an enrollee. Often a image or template of a physiological attribute (e.g., a fingerprint) that may be used to identify an individual. In this document, biometrics may be used to unlock authentication tokens and prevent repudiation of registration.
Biometric Application	The use to which a biometric system is put. See also 'Application Developer.'
Biometric Data	The extracted information taken from the biometric sample and used either to build a reference template or to compare against a previously created reference template.
Biometric Engine	The software element of the biometric system which processes biometric data during the stages of enrollment and capture, extraction, comparison and matching.
Biometric Identification Device	The preferred term is 'Biometric System.'
Biometric Sample	Data representing a biometric characteristic of an end-user as captured by a biometric system.
Biometric System	An automated system capable of: capturing a biometric sample from an end user; extracting biometric data from that sample; comparing the biometric data with that contained in one or more reference templates; deciding how well they match; and indicating whether or not an identification or verification of identity has been achieved.
Biometric Technology	A classification of a biometric system by the type of biometric.
Capture	The method of taking a biometric sample from the end user.
Certification	The process of testing a biometric system to ensure that it meets certain performance criteria. Systems that meet the testing criteria are said to have passed and are certified by the testing organization.
Comparison	The process of comparing a biometric sample with a previously stored reference template or templates. See also 'One-To-Many' and 'One-To-One.'
Claim of Identity	When a biometric sample is submitted to a biometric system to verify a claimed identity.

Term	Description
Claimant	A person submitting a biometric sample for verification or identification whilst claiming a legitimate or false identity.
Closed-Set Identification	When an unidentified end-user is known to be enrolled in the biometric system. Opposite of 'Open-Set Identification.'
Credential	An object that authoritatively binds an identity (and optionally, additional attributes) to a token possessed and controlled by a person.
Crossover Rate	Synonym for 'Equal Error Rate.'
Cryptographic key	A value used to control cryptographic operations, such as decryption, encryption, signature generation or signature verification. This means that it must be as hard to find an unknown key or decrypt a message, given the information exposed to an eavesdropper by an authentication, as to guess a random number of the selected value. (See also Asymmetric keys, Symmetric key.)
Cryptographic strength	A measure of the expected number of operations required to defeat a cryptographic mechanism.
Data integrity	The property that data has not been altered by an unauthorized entity.
Discriminant Training	A means of refining the extraction algorithm so that biometric data from different individuals are as distinct as possible.
End User	A person who interacts with a biometric system to enroll or have his/her identity checked.
End User Adaptation	The process of adjustment whereby a participant in a test becomes familiar with what is required and alters their responses accordingly.
Enrollee	A person who has a biometric reference template on file.
Enrollment	The process of collecting biometric samples from a person and the subsequent preparation and storage of biometric reference templates representing that person's identity.
Enrollment Time	The time period a person must spend to have his/her biometric reference template successfully created.
Entropy	A measure of the amount of uncertainty that an attacker faces to determine the value of a secret. (Entropy is usually stated in bits.) It is also used for measuring the uncertainty in a data set, meaning the variance of a set of samples.

Term	Description
Equal Error Rate	When the decision threshold of a system is set so that the proportion of false rejections will be approximately equal to the proportion of false acceptances. A synonym is ’Crossover Rate.’
Extraction	The process of converting a captured biometric sample into biometric data so that it can be compared to a reference template.
Failure to Acquire	Failure of a biometric system to capture and extract biometric data.
Failure to Acquire Rate	The frequency of a failure to acquire.
False Acceptance	When a biometric system incorrectly identifies an individual or incorrectly verifies an impostor against a claimed identity. Also known as a Type II error.
False Acceptance Rate/ FAR	The probability that a biometric system will incorrectly identify an individual or will fail to reject an impostor. Also known as the Type II error rate.
False Match Rate	Alternative to ’False Acceptance Rate’. Used to avoid confusion in applications that reject the claimant if their biometric data matches that of an enrollee. In such applications, the concepts of acceptance and rejection are reversed, thus reversing the meaning of ’False Acceptance’ and ’False Rejection.’ See also ’False Non-Match Rate.’
False Non-Match Rate	Alternative to ’False Rejection Rate’. Used to avoid confusion in applications that reject the claimant if their biometric data matches that of an enrollee. In such applications, the concepts of acceptance and rejection are reversed, thus reversing the meaning of ’False Acceptance’ and ’False Rejection.’ See also ’False Match Rate.’
False Rejection	When a biometric system fails to identify an enrollee or fails to verify the legitimate claimed identity of an enrollee. Also known as a Type I error.
False Rejection Rate/FRR	The probability that a biometric system will fail to identify an enrollee, or verify the legitimate claimed identity of an enrollee. Also known as a Type I error rate.
Field Test	A trial of a biometric application in ’real world,’ as opposed to laboratory, conditions.

Term	Description
Goats	Biometric system end users whose pattern of activity when interfacing with the system varies beyond the specified range allowed by the system, and who consequently may be falsely rejected by the system.
Genetic Penetrance	The degree to which characteristics are passed from generation to generation.
Identity	A unique name of an individual person or another entity. Since the legal names of persons are not necessarily unique, the identity of a person must include sufficient additional information (for example an address, username, or some unique identifier such as an employee or account number) to make the complete name unique.
Identification/Identify	The one-to-many process of comparing a submitted biometric sample against all of the biometric reference templates on file to determine whether it matches any of the templates and, if so, the identity of the enrollee whose template was matched. The biometric system using the one-to-many approach is seeking to find an identity amongst a database rather than verify a claimed identity. Contrast with 'Verification.'
Impostor	A person who submits a biometric sample in either an intentional or inadvertent attempt to pass him/herself off as another person who is an enrollee.
In-House Test	A test carried out entirely within the environs of the biometric developer which may or may not involve external user participation.
Live Capture	The process of capturing a biometric sample by an interaction between an end user and a biometric system.
Man-in-the-middle (MitM) attack	An attack on the authentication protocol run in which the attacker positions himself in between the claimant and verifier so that he can intercept and alter data traveling between them.
Match/Matching	The process of comparing a biometric sample against a previously stored template and scoring the level of similarity. An accept or reject decision is then based upon whether this score exceeds the given threshold.

Term	Description
Network	An open communications medium, typically the Internet, that is used to transport messages between the claimant and other parties. Unless otherwise stated no assumptions are made about the security of the network; it is assumed to be open and subject to active (e.g., impersonation, man-in-the-middle, session hijacking) and passive (e.g., eavesdropping) attack at any point between the parties (claimant, verifier, relying party).
Off-line attack	An attack where the attacker obtains some data (typically by eavesdropping on an authentication protocol run, or by penetrating a system and stealing security files) that he/she is able to analyze in a system of his/her own choosing.
On-line attack	An attack against an authentication protocol where the attacker either assumes the role of a claimant with a genuine verifier or actively alters the authentication channel. The goal of the attack may be to gain authenticated access or learn authentication secrets.
One-To-Many	Synonym for 'Identification.'
One-To-One	Synonym for 'Verification.'
Open-Set Identification	Identification, when it is possible that the individual is not enrolled in the biometric system. Opposite of 'Closed-Set Identification.'
Out Of Set	In open-set identification, when the individual is not enrolled in the biometric system.
Passive attack	An attack against an authentication protocol where the attacker intercepts data traveling along the network between the claimant and verifier, but does not alter the data (for example eavesdropping).
Passive Impostor Acceptance	When an impostor submits his/her own biometric sample and claiming the identity of another person (either intentionally or inadvertently) he/she is incorrectly identified or verified by a biometric system. Compare with 'Active Impostor Acceptance.'
Password	A secret that a claimant memorizes and uses to authenticate his or her identity. Passwords are typically character strings.
Performance Criteria	Pre-determined criteria established to evaluate the performance of the biometric system under test.

Term	Description
Personal Identification Number (PIN)	A password consisting only of decimal digits.
Physical/Physiological Biometric	A biometric which is characterized by a physical characteristic rather than a behavioral trait.
Receiver Operating Curves	A graph showing how the false rejection rate and false acceptance rate vary according to the threshold.
Recognition	The preferred term is 'Identification'.
Response Time	The time period required by a biometric system to return a decision on identification or verification of a biometric sample.
Salt	A non-secret value that is used in a cryptographic process, usually to ensure that the results of computations for one instance cannot be reused by an attacker.
Shared secret	A secret used in authentication that is known to the claimant and the verifier, for example a password.
Template/Reference Template	Data which represents the biometric measurement of an enrollee used by a biometric system for comparison against subsequently submitted biometric samples.
Third Party Test	An objective test, independent of a biometric vendor, usually carried out entirely within a test laboratory in controlled environmental conditions.
Threshold/Decision Threshold	The acceptance or rejection of biometric data is dependent on the match score falling above or below the threshold. The threshold is adjustable so that the biometric system can be more or less strict, depending on the requirements of any given biometric application.
Throughput Rate	The number of end users that a biometric system can process within a stated time interval.
Type I Error	See 'False Rejection.'
Type II Error	See 'False Acceptance.'
Token	Something that the claimant possesses and controls (typically a key or password) used to authenticate the claimants identity.

Term	**Description**
User	The client to any biometric vendor. The user must be differentiated from the end user and is responsible for managing and implementing the biometric application rather than actually interacting with the biometric system.
Validation	The process of demonstrating that the system under consideration meets in all respects the specification of that system.
Verifier	An entity, computer or system that verifies the claimants identity by verifying the claimants possession of a token using an authentication protocol. To do this, the verifier may also need to validate credentials that link the token and identity and check their status.
Verification/Verify	The process of comparing a submitted biometric sample against the biometric reference template of a single enrollee whose identity is being claimed, to determine whether it matches the enrollee's template. Contrast with 'Identification.'
Zero Effort Forgery	An arbitrary attack on a specific enrollee identity in which the impostor masquerades as the claimed enrollee using his or her own biometric sample.

Appendix B

Detailed Descriptions

B.1 Wilcoxon signed-rank test

The Wilcoxon signed rank test[54] is named after Frank Wilcoxon. It was originally used as a test to determine if two sample observations origin from the same distribution. A popular use of this test is to evaluate a set of ranked questions and answers. Imagine a set of questions with a multiple choice alternative from 1 through 4 (for example low to high or strongly disagree to strongly agree to question statement). The harshness of each percipient in such a questionnaire could differ. One way to know if two of the subject are agreeing disregarding their harshness, is looking if their values go up or down (not looking at how high or low the values selected are).

Ordering the samples and counting entries within defined value slots (chunks), can reveal if the trend is similar or not for the two samples.

B.2 Accuracy and Precision

The most common analogy used to explain accuracy and precision is comparing the measurements to arrows fired at a target. While accuracy describes the closeness of the arrows to the bullseye center, the closer to the center the arrows are, the higher is the accuracy is. Precision on the other hand goes to describe how close the arrows are to each other, or the size of the cluster in which the arrows are scattered. In the example of the bullseye, many arrows on the lower left hand side of the target would be considered precise. The closer the arrows are to the bullseye, the more accurate it is. And combined a high precision with high accuracy would be the best dataset in both aspects.

One point to keep in mind is that the accuracy is somewhat dependent on precision. All measurements cannot be close to the bullseye, if the precision is low.

Appendix C

Media Content

C.1 Thesis

Folder `thesis` contains this `thesis.pdf` document and the `thesis.bib` file used in the bibliography.

C.2 Spread Sheets

In the `calc` folder, the spreadsheets used for preliminary testing and graphing can be found. Most are in `.xls` (*Microsoft Excel*) format, however some `.ods` (*OpenOffice*) files can also be found.

C.3 Figures

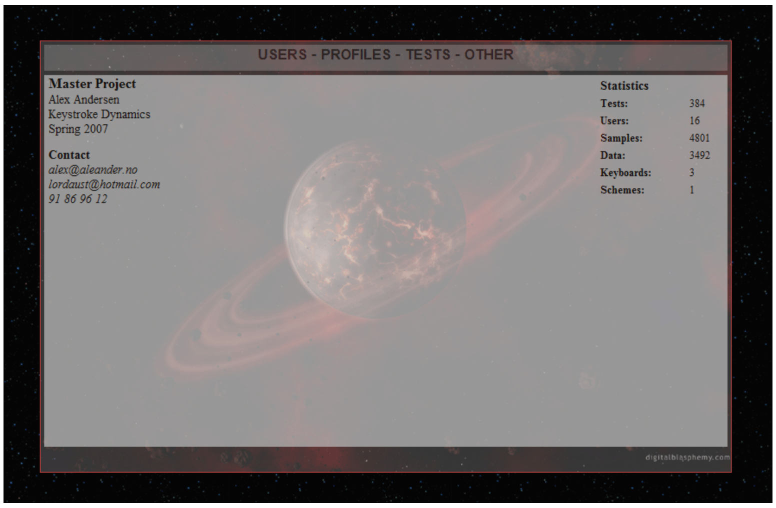
All figures generated or created in the process can be found in the `fig` folder. Included is also the datafiles and *gnuplot* files used to generate the figures found in the thesis.

C.4 Login application

The entire C# login application can be found in the `login_app` folder. It can be run using Visual Studios, and source can be read using any standard text editor.

C.5 Web application

Folder `webapp` contains the scripts used to calculate the profiles. It also contains the test scripts used for all the tests conducted. It is written in ColdFusion, a Adobe/Macromedia product, and needs ColdFusion server to run. It supports Internet Information Services on Windows 98, XP, Vista, 2000 and 2000 Server, and Apache for Windows or supported Linux distributions.



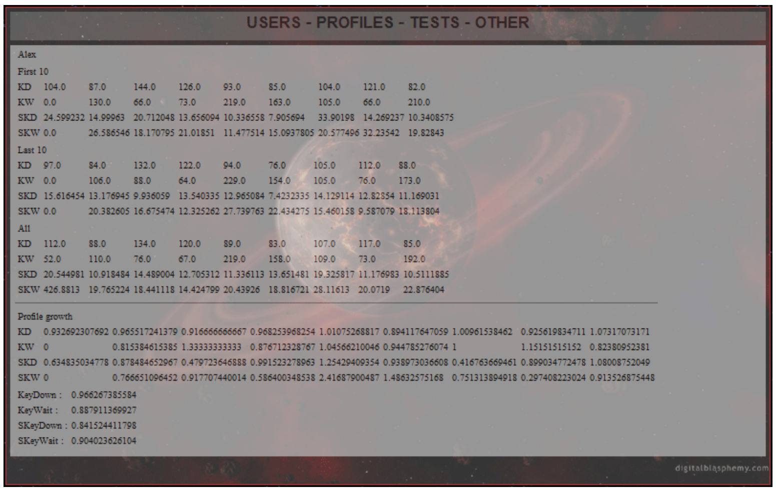
Master Project
Alex Andersen
Keystroke Dynamics
Spring 2007

Contact
alex@aleander.no
lordaust@hotmail.com
91 86 96 12

Statistics	
Tests:	384
Users:	16
Samples:	4801
Data:	3492
Keyboards:	3
Schemes:	1

Figure C.1: Screenshot from the web-application front page

It needs the correct database ODBC connection setting to an available SQL server, defined in the file `Application.cfm`. Though originally developed using Microsoft SQL Server, it also supports MySQL and several other database types.



```
Alex
First 10
KD  104.0     87.0      144.0     126.0     93.0      85.0       104.0     121.0     82.0
KW  0.0       130.0     66.0      73.0      219.0     163.0      105.0     66.0      210.0
SKD 24.599232 14.99963  20.712048 13.656094 10.336558 7.905694   33.90198  14.269237 10.3408575
SKW 0.0       26.586546 18.170795 21.01851  11.477514 15.0937805 20.577496 32.23542  19.82843
Last 10
KD  97.0      84.0      132.0     122.0     94.0      76.0      105.0     112.0     88.0
KW  0.0       106.0     88.0      64.0      229.0     154.0     105.0     76.0      173.0
SKD 15.616454 13.176945 9.936059  13.540335 12.965084 7.4232335 14.129114 12.82854  11.169031
SKW 0.0       20.382605 16.675474 12.325262 27.739763 22.434275 15.460158 9.587079  18.113804
All
KD  112.0     88.0      134.0     120.0     89.0      83.0      107.0     117.0     85.0
KW  52.0      110.0     76.0      67.0      219.0     158.0     109.0     73.0      192.0
SKD 20.544981 10.918484 14.489004 12.705312 11.336113 13.651481 19.325817 11.176983 10.5111885
SKW 426.8813  19.765224 18.441118 14.424799 20.43926  18.816721 28.11613  20.0719   22.876404

Profile growth
KD  0.932692307692 0.965517241379 0.916666666667 0.968253968254 1.01075268817 0.894117647059 1.00961538462  0.925619834711 1.07317073171
KW  0              0.815384615385 1.33333333333  0.876712328767 1.04566210046 0.944785276074 1              1.15151515152  0.82380952381
SKD 0.634835034778 0.878484652967 0.479723646888 0.991523278963 1.25429409354 0.938973036608 0.416763669461 0.89903477247 8 1.08008752049
SKW 0              0.766651096452 0.917707440014 0.586400348538 2.41687900487 1.48632575168  0.751313894918 0.297408223024 0.913526875448
KeyDown :   0.966267385584
KeyWait :   0.887911369927
SKeyDown :  0.841524411798
SKeyWait :  0.904023626104
```

Figure C.2: Screenshot from the web-application example user profile page

C.6 Database

The folder `database` contains the `.mdf` and `.ldf` database files needed to attach in *MSSQLSERVER Enterprice Manager*. The figure C.3 shows the structure of the database. Each field and table is commented under the database information. A section on the database can be found in section 4.4.4. It contains the raw data from the logging scripts, and the samples filtered with the web-application. It also contains all the profiles generated for all the terminal types for all users. There are no sensitive data, like full names or personal information in any of the tables.

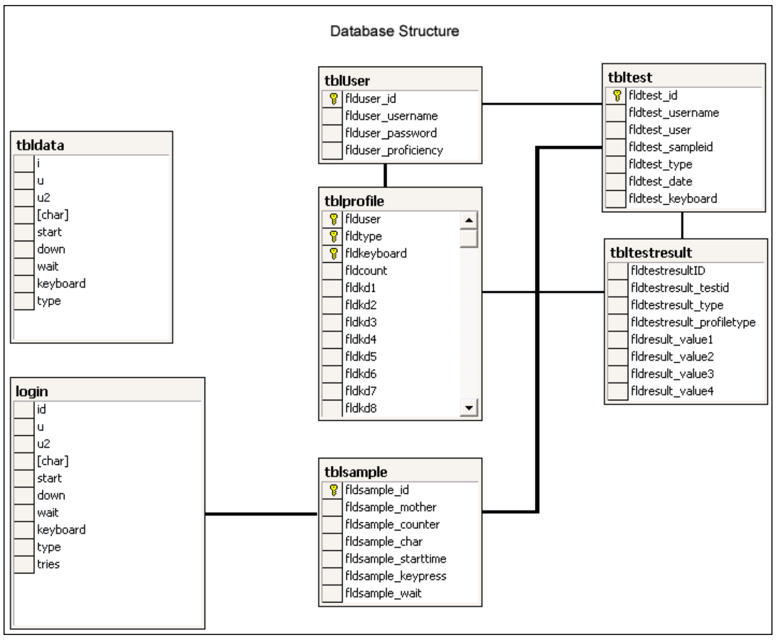


Figure C.3: Database table relationship structure