CHAPTER 3

Continuous Authentication with Keystroke Dynamics

Patrick Bours and Soumik Mondal

In this chapter we will discuss how keystroke dynamics can be used for true continuous authentication.We have collected keystroke dynamics data of 53 participants who used the computer freely and we have analysed the collected data. We will describe a system that decides on the genuineness of the user based on each and every single keystroke action of the current user and we will represent the results in a new manner. The continuous authentication system will lock out a user if the trust in genuineness of the current user is too low. Ideally such a system would never lock out a genuine user and detect an impostor user within as few keystroke actions as possible.

3.1 Introduction

Keystroke dynamics is used for recognition of a person by his or her typing rhythm. Mainly this has been applied to strengthen the username/password log on mechanism. In such a case, not only the value of the password needs to be correct, but also the typing rhythm when entering the password (and maybe also username) needs to be

Patrick Bours and Soumik Mondal
Norwegian Information Security Laboratory (NISlab)
Gjøvik, University College, Gjøvik, Norway
e-mail: {patrick.bours, soumik.mondal}@hig.no

Editors: Y. Zhong and Y. Deng, *Recent Advances in User Authentication Using Keystroke Dynamics Biometrics*
DOI: 10.15579/gcsr.vol2.ch3, GCSR Vol. 2, pp. 41-58, 2015

correct [6]. However, after the initial log on procedure, the computer system does not assume that a user changes during a session. Users do tend to leave computers unattended while they are logged on, for longer or shorter periods of time, without locking either computer or the room where the computer is located. During such a period that the computer is left unattended, an impostor can get access. The impostor then does not need to guess user's password or even mimic the typing rhythm of the genuine user. He has access to the same sources as the genuine user, meaning he can read documents, copy or delete files, access social networks, or send email in the name of the genuine user.

This is obviously an unwanted situation and we need to have a system in place that checks if the user has changed after the log on procedure. Such a system is called a Continuous Authentication (CA) system [14, 15, 16, 10, 1]. Because a user still uses the keyboard after the log on procedure, to type documents, chat with friends and colleagues, navigate the internet and much more, we can use keystroke dynamics also to establish continuous authentication. The main difference between use of keystroke dynamics for the log on procedure (called Static Keystroke Dynamics (SKD) in this chapter) and keystroke dynamics for continuous authentication (called Continuous Keystroke Dynamics (CKD) in this chapter) is that for SKD the typed information is fixed, while for CKD the information is never fixed.

The two main requirements for a true CA system is (1) that a user is not interrupted in his daily activity and (2) that the system uses each single keystroke to determine the genuineness of the user. Requirement (1) is important because the system should be unobtrusive. If a user is interrupted in his daily business then this might lead to the user disabling the CA system. This implies that a CA system does not depend on specific activities of the user, e.g. a CA system cannot be based on providing a fingerprint because that requires the user to present his finger to a fingerprint scanner on a regular basis. Besides keystroke dynamics one can also use mouse dynamics [2, 19] or face recognition [27] for CA. The second requirement ensures that the CA system does in fact continuously check the genuineness of a user. Most existing research in Continuous Authentication is based on checking the genuineness of a user based on a fixed block of N actions. These systems take as input N actions and the identity of the user and provide a binary output if this input belongs to the provided identity or not. Besides the fact that technically the identity is checked periodically and not continuously, does this approach also have as a disadvantage that an impostor can do N keystrokes before his identity is checked for the first time. That is why we need requirement (2), that every single keystroke of the user will contribute in the determination of the genuineness of the user. We need to be able at any point in time, *i.e.* after every single keystroke, to lock the system if it is detected that a user is an impostor. In this chapter we will introduce a CKD system that is capable of deciding after every single keystroke if the user is the correct one or not and will be able to lock the system at any time.

One further disadvantage of current research in CA is that it focuses on specific tasks by the user. For example in [4] users were asked to type three different reports, each of 400-500 words. It is known that typing behaviour changes based on the application that a user is using [11], so in order to check the actual performance of a true CA system it is important to allow the user to perform his daily activities when collecting

Keystroke Dynamics data [14].

Most research on continuous keystroke dynamics reports performance in terms of False Match Rate (FMR) and False Non-Match Rate (FNMR) or even Equal Error Rate (EER) [1]. However, in this chapter we will show that these performance indicators are not the best to use in case of CA. It is in fact not important to know **if** an impostor is detected, but **when** he or she is detected, *i.e.* how many activities he/she has been able to perform before detection. We will use ANIA (Average Number of Impostor Actions) and ANGA (Average Number of Genuine Actions) as our performance indicators. Hence, our performance indicator shows how much an impostor can type before he is locked out and how much can a genuine user type before he is, wrongfully, locked out of the system. These are the equivalents of FMR and FNMR for a CA system.

In this chapter we will give an overview of the research done in CKD and we will introduce our own system for evaluating the performance. Our CKD system is based on trust in the genuine user and we will use a so-called *penalty-and-reward* function to adjust the trust based on each single keystroke by the user [10]. We will further focus on performance evaluation and we will show our current results in comparison to other research in this area. For our analysis we collected keystroke dynamics data of 53 persons while they used their own computer for their daily activities. The data collection took at least one week per person, but due to different use of the computer does the amount of collected data vary a lot. We will finally present conclusions and topics for further research and focus on potential application areas.

3.2 Related Work

In Keystroke Dynamics (KD), users are identified or authenticated based on the way they type on a keyboard. When a password is typed not only the correctness of the password itself is checked, but also if the typing rhythm when entering the password is correct. This process is called password hardening. The use of KD as a method of identification is not new. During the early days, the telegraph operators were able to identify each other by their Morse code typing pattern. This identification method, known as "The Fist of the Sender", was used as a verification or identification method during World War II. Nowadays software is available for password hardening, for example, the software from BioPassword (see `http://www.biopassword.com`).

A KD based authentication or identification system is a low cost and easy to implement security solution because, this system is software based. In such a system, the keystroke timing information has to be captured, and features for authentication or identification are extracted. Sometimes some special keyboard (pressure sensor based keyboard [26] or key press sound information [21]) was used to capture the key pressure information or key sound information for the same.

Research on continuous authentication was started on 1995 when Shepherd [24] showed some impressive result on continuous authentication using keystroke dynamics. These days continuous authentication is getting more popular because of the security requirements in office environments and the DARPA's Active Authentication project announced in 2012.

Dowland et al. [14] used digraph, tri-graph and word latency as a feature and distance based classifier as a classification tool and achieved an FAR of 4.9% and an FRR of 0% over 35 users.

Gunetti et al. [15, 7] have used free text keystroke dynamics for authentication. They have used digraph latency disorder as a feature and distance based classifier as a classification tool. They got an FAR of 3.16% and an FRR of 0.02% for full session over 40 legitimate users and 165 imposter users.

Stewart et al. [25] have introduced burst authentication. The primary motivation for using this concept of burst authentication is to reduce the frequency of independent authentication checks. This has the advantages of reducing the false alarm rate, avoiding the capture of unnecessarily large quantities of data and using excessive computing resources to process the input while still providing sufficient data for continual training of the biometric system. They have used the keystroke time information and stylometry as a feature and KNN with Euclidean distance as a classifier. They have achieved a 0.5% EER for 40 participants.

Messerman et al. [18] presented a non-intrusive verification scheme which continuously verifies the user identity using free text based keystroke dynamics. Their solution is an extension of the scheme of Gunetti et al. [15, 7]. Here the target applications which require decision-making processes in real-time are webmail applications. They have addressed major challenges in these scenarios such as changes in human-behaviours, scalability and response time. They have provided the solution like, an adaptive user model, extend the decision process, and bring in a randomized approach for addressing the scalability issue. They have tested over a data set which has been collected from 55 users over a period of a year. Evaluation of the solution was performed under consideration of external and internal attackers and achieved satisfactory results. They have used n-graph time latency as a feature and distance based classifier as a classification tool. They have achieved an internal FAR of 2.02%, an external FAR of 2.61% and an FRR of 1.84% with the execution time of 0.57 sec.

Ordal et al. [22] have used key press duration and digraph latency as a feature distance based classifier for their work. They have achieved zero false lockout rates and 5% missed lockout rates by using 150 keystroke segments.

Ahmed et al. [1] have used key press duration and digraph latency as a feature and Artificial Neural Network as classifier for their work and achieved a 2.46% EER for 53 users.

3.3 Background Knowledge

In this section, we are going to discuss some background knowledge required to better understand this research. This includes some classification techniques and our analysis methods.

3.3.1 Support Vector Machine

Support Vector Machine (SVM) is a very well-known supervised learning algorithm which can be used for classification problem [12]. This classifier is capable of creating

a linear decision margin that is as wide as possible, depending on the Support Vectors (SV). The SV are those data points from the different classes that are closest to the decision line. In this research, we have used the LibSVM software distribution for the SVM classifier [13]. The main motivation to use the LibSVM is not only because it is a well implemented optimization technique for the cost function of SVM and widely used in the research community, but also because it will provide the classification score (probability) along with the class label. In our research, we are going to use only the classification score for our analysis. Initially we tried SVM with a *Linear Kernel*, but found that the classifier did not perform well due to the small feature set (see Section 3.4.3). We decided to use *Gaussian Kernel* as a similarity measure function in this research.

3.3.2 Artificial Neural Network

Artificial Neural Network (ANN) is a combination of multiple artificial neurons which can be used for classification and regression analysis [9]. In our research, the neurons consist of a *linear* activation function with a 2-layer *Feed-Forward* neural network. In this research, we have used the NETLAB software distribution for the ANN classifier [20]. NETLAB has a well implementation of *Scaled Conjugate Gradient* algorithm which is efficient to optimize the cost function and also it will reduce the ANN training time. We have tested different numbers of hidden nodes, and different regularization parameter values (α) for different users to maintain the trade of between *Bias* and *Variance* and the training time of the classifier model.

3.3.3 Counter-Propagation Artificial Neural Network

Counter-Propagation Artificial Neural Network (CPANN) is a hybrid learning mechanism based on Artificial Neural Network to handle the supervised problems. In CPANN, the output layer is added to the Kohonen layer which is very similar to the Self Organizing Maps and provides both the advantages of supervised and unsupervised learning. It can also guarantee to find the correct network weights, that can be seen as drawbacks of regular back-propagation networks. We have used a free CPANN software distribution for our research [5]. The number of neurons is optimized for different users.

3.3.4 Multi-Classifier Fusion

Multi-Classifier Fusion (MCF) is a technique to combine multiple classifiers on a same biometric modality to improve the performance of that modality [17]. Researchers generally prefer to use multiple classifiers on a same modality when the modality is considered to be a weak modality. The architecture of the MCF technique is very much similar to the multi-modal biometric technique but, with MCF technique only score level and decision level fusion are possible [23].

3.3.5 Trust Model

The concept of *Trust Model* was first introduced by Bours [10]. In this model, the behaviour of the current user is compared to the template of the genuine user. Based on each single action performed by the user, the trust in the genuineness of the user will be adjusted. If the trust of the system in the genuineness of the user is too low, then the user will be locked out of the system. In particular if the trust drops below a pre-defined threshold $T_{lockout}$ then the system locks itself and will require static authentication of the user to continue working.

The basic idea is that the trust of the system in the genuineness of the current user depends on the deviations from the way this user performs various actions on the system. If a specific action is performed in accordance with how the genuine user would perform the task (*i.e.* as it is stored in the template), then the system's trust in the genuineness of this user will increase, which is called a *Reward*. If there is a large deviation between the behaviour of the genuine user and the current user, then the trust of the system in that user will decrease, which is called a *Penalty*. The amount of change of the trust level can be fixed or variable [10]. For example, a small deviation from the behaviour of the user, when compared to the template, could lead to a small decrease in trust, while a large deviation could lead to a larger decrease.

No single person will be able to always behave in exactly the same manner [8]. For the genuine user this means that he will also sometimes deviate from his normal behaviour, which will lead to a decrease in trust. However, the majority of actions that a genuine user will perform will be close to his normal behaviour, *i.e.* will lead to an increase of trust. Overall this would lead to a high level of trust. For an impostor however the opposite holds. In some cases he will behave as the genuine user, increasing his level of trust, but the majority of actions will lead to a decrease in trust due to the deviation from the behaviour of the genuine user. This will then lead to a general decrease of the trust over time for an impostor user. Obviously an ideal system should perform in such a way that the trust in anyone other then the genuine user will decrease fast to a value below the threshold $T_{lockout}$. In such an ideal system, also a genuine user would never reach a trust level that would result in a lockout, *i.e.* the genuine user would not notice the influence of the continuous authentication system in his daily activities.

3.4 Data Description and Processing

In this section we will discuss the data description and data processing techniques used in this research. This also includes data collection and feature extraction process.

3.4.1 Data Description

We have designed a Windows operating system based logging tool, which can capture the Keystroke and Mouse interaction data continuously. Log data is stored locally in a CSV file or can be transmitted to a server over a secure channel. Privacy of the users and confidentiality of the sensitive data were taken into account throughout the development of the tool.

Table 3.1: Data format for keystroke events.

Event ID	Event Type	Action	Value	Time	Relation	Flag	Additional fields
n	'K'	'D' 'U'	String	ms	evt. ID	Int	n/a Count

The data format for keystroke events is shown in Table 3.1. The *event type* is always 'K'. Keystroke events have only two types of *actions*, key press ('D'/down) and key release ('U'/up). The UTF-8-encoded *Value* field states which key was pressed or released; it is either the typed character or a String with a descriptive key name, enclosed by '|', for example "|enter|" or "|space|". An ISO8601-compliant *time-stamp* of when the event occurred is recorded in milliseconds. *Flag* is an Integer indicating which alternate/system key was active. Bit 0 (LSB) is set when the `Alt` key was down, bit 1 is set for `Ctrl`, bit 2 for `Shift`, bit 3 for `Win`, bit 4 for `Caps Lock`, bit 5 for `Num Lock`, bit 6 for `Scroll Lock`. E.g., if `Alt+Shift` was pressed, *Flag* would be set to 5. In case of a key up event, an additional field *Count* indicates if a key was kept down for a longer period of time.

3.4.2 Data Collection

Due to a high degree of privacy concern, we have managed only to get 53 volunteers to participate in our experiment. We have deployed our data logging software among these 53 volunteers and collected the data continuously for 5 to 7 days. All the participants of this data collection process are university students and permanent staff members and they are regular computer users. Our collected data has the following properties:

1. No instructions or any specific task were given to the user;
2. This continuous data collection was done in a complete uncontrolled environment to represent the users natural computer usage behaviour;
3. All of our participants installed the software on their own system to remove the effect on the natural behaviour due to change of hardware.

Table 3.2, shows the quantitative and qualitative comparison between our data and the data collected from previous research on continuous authentication. The collection software collected both keystroke dynamics information as well as mouse related activities. On average a user provided 47600 keystroke related actions and 29200 mouse related actions. In this paper we only consider the keystroke dynamics related data.

We have separated our data to build and train the system (training dataset), parameter adjustment of the algorithms used in this research (validation dataset) and finally to test the system performance (test dataset). The 35% of the total data was used as a training dataset, 10% of the of the total data was used as a validation dataset and the remaining 55% of the data (or approximately 26200 keystroke dynamics data samples) was used as a test dataset. In fact, if a user provided a large amount of keystroke related data then the amount used for training was limited to a maximum of 20000, which means that even more data of that user would be available for testing.

Table 3.2: Data comparison with previous research.

Reference	# User	Time period (per user)	Environment	Task	Applications
[7]	44	5 times	Controlled	Fixed	Predefined
[14]	35	3 months	Uncontrolled	Uncontrolled	MS Windows
[22]	26	several days	Uncontrolled	Uncontrolled	Uncontrolled
[18]	55	12 months	Uncontrolled	Uncontrolled	Web-mail
[25]	30	4 sessions	Controlled	Fixed	Predefined
[1]	53	209 hours	Uncontrolled	Uncontrolled	Uncontrolled
This work	53	5-7 days	Uncontrolled	Uncontrolled	Uncontrolled

3.4.3 Feature Extraction

In our research, we have converted the keystroke events into two different actions.

1. `Single Key Action`, where the feature is the *key hold time*.

2. `Key Digraph Action`, where the features are the *Total Duration*, the time between first key press and second key press (*Down-Down Time*), the time between first key release and second key press (*Up-Down Time*) and the time between first key release and second key release (*Up-Up Time*) of a particular key digraph [3].

Figure 3.1, is the graphical representation of the keystroke dynamics feature extraction process. In our analysis, we have applied a constraint for *Key Digraph Action* that the latency between two consecutive keys should be below 2000 ms. Reason for this is that such a high latency does not represent the normal typing behaviour of a user, but represents a pause in the typing.

3.4.4 Classification

We have done three different verification processes in our analysis, which are following:

1. Verification Process 1 (VP-1): In this protocol, we are not going to use any imposter data during the template building phase and we used 50% of the imposter users (26 biometric subjects) for algorithmic parameters adjustments.

2. Verification Process 2 (VP-2): In this protocol, we have considered all the imposter users during the template building phase and parameters adjustment process for algorithms.

3. Verification Process 3 (VP-3): In this protocol we have only considered 50% of the imposter users (26 biometric subjects) in the template building phase and parameters adjustment process for algorithms.

We have applied separate classification techniques for different modalities along with different Verification Processes. The description of these techniques is given in the below.

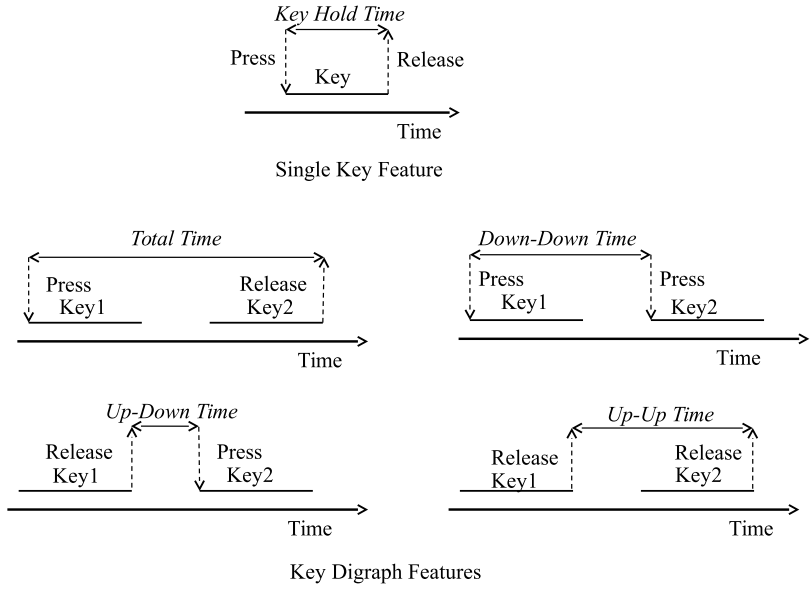

Figure 3.1: Keystroke dynamics features.

Single Key Classification

VP-1 We have used pairwise `Scaled Euclidean Distance` for particular key observations for this verification process. For example, the key a has n observations on the training data, then we will get n scaled Euclidean distances for the new test sample of a. The distance metric vector used is (f_1, f_2, f_3) = (mean, minimum, maximum) of these n distance values. From these 3 attributes we are going to calculate a score that is going to be used in the Trust Model (see Section 3.5.1) in the following way:

$$sc = 1 - \frac{f_1 - f_2}{f_3 - f_2}$$

VP-2 In this protocol, we have used 2 regression models and one prediction model in a multi-classifier architecture. For regression models we have applied ANN and CPANN, and for prediction model we have SVM. The score vector we used here is $(f_1, f_2, f_3) = (Score_{ann}, Score_{svm}, Score_{cpann})$. From these 3 classifier scores we are going to calculate a score that is going to be used in the Trust Model (see Section 3.5.1) in the following way:

$$sc = \frac{w_1 \times f_1 + w_2 \times f_2 + w_3 \times f_3}{w_1 + w_2 + w_3}$$

where, w_1, w_2 and w_3 are the weights for the weighted fusion techniques. The weights are optimized using *Genetic Algorithm* techniques (see Section 3.5.2).

VP-3 We have followed here the same classification approach as described for *VP-2*.

Key Digraph Classification

VP-1 For a digraph action, we have four attributes on the feature vector. Here we have used two distance metrics, *i.e.* pairwise `Scaled Euclidean Distance` (SED) and `Correlation Distance` (CD) for particular key observations. For example, assume the key digraph `ab` has n observations on the training data. Now, we will get n scaled Euclidean distances and n Correlation Distances for the new test sample of digraph `ab`. Then we are going to take distance metric vector as $(f_1, f_2, f_3) =$ (mean of SED, minimum of SED, maximum of CD). From this we are going to calculate the score sc used in the Trust Model of Section 3.5.1 in the following way:

$$sc = \frac{f_1 \times f_3}{f_2}.$$

We have followed the same classification approach as described in the previous section for *VP-2* and *VP-3* verification processes.

3.5 Methodology

This section describes the methodology we have followed to carry out analysis.

3.5.1 Applied Trust Model

In this research, we have implemented a variable Trust Model. The model uses several parameters to calculate the change in trust and to return the system trust in the genuineness of the current user after each separate action performed by the user. All the parameters for this trust model can be user specific. Also, for each user the parameters can be different for different kind of actions. The actual value for these 4 parameters are optimized using `Genetic Algorithm` optimization (see Section 3.5.2).

The change in trust (Δ_{Trust}) is calculated according to Eq.(3.1) and depends on the classification score of the current action performed by the user as well as on 4 parameters. The parameter A represents the threshold value between penalty and reward. If the classification score of the current action (sc_i) is equal to this threshold then $\Delta_{Trust} = 0$. If $sc_i > A$ then $\Delta_{Trust} > 0$, *i.e.* a reward is given and if $sc_i < A$ then $\Delta_{Trust} < 0$, *i.e.* the trust decreases because of a penalty. Furthermore, the parameter B is the width of the sigmoid for this function (see Fig.(3.2)), while the parameters C and D are the upper limit of the reward and the penalty. In Fig.(3.2), we have shown the Δ_{Trust} produced by the Eq.(3.1) based on the classification score of the current action for different parameters.

$$\Delta_{Trust}(sc_i) = \min\{-D + D \times (\frac{1 + \frac{1}{C}}{\frac{1}{C} + \exp(-\frac{sc_i - A}{B})}), C\} \tag{3.1}$$

If the trust value after i actions is denoted by $Trust_i$, then we have the following relation between the trust $Trust_{i-1}$ after $i-1$ actions and the trust $Trust_i$ after i actions when the particular i^{th} action had a classification score sc_i:

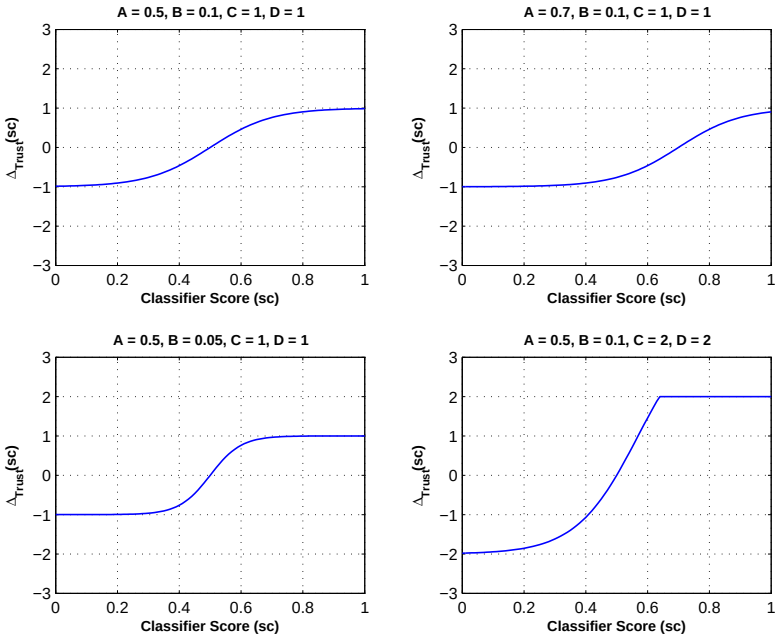

Figure 3.2: Score sc vs. $\Delta_{Trust}\,(sc)$ from Eq.(1) for different parameters.

$$Trust_i = \min\{\max\{Trust_{i-1} + \Delta_{Trust}(sc_i), 0\}, 100\}. \tag{3.2}$$

3.5.2 System Architecture

In this section, we are going to discuss the methodology of our system. The system was divided into two phases (see Fig.(3.3)).

In the training phase, the training data is used to build the classifier models and store the models in a database for use during the testing phase (marked as dotted arrow in Fig.(3.3)). Each genuine user has his/her own classifier models and training features.

In the testing phase, we are going to use test data which was separated from the training data for comparison. In the comparison, we will use the models and training features stored in the database and obtain the classifier score (probability) on each sample of the test data according to the performed action. This score will then be used to update the trust value $Trust$ in the trust model (see Section 3.5.1). Finally, the trust value $Trust$ is used in the decision module, to determine if the user will be locked out or can continue to use the PC. This decision is made based on the current trust value and the lockout threshold ($T_{lockout}$).

For each action done by the current user, the system calculates the score for that action (see for details the subsections in Section 3.4.4) and used that to calculate the change in trust according to Eq.(3.2). The parameters A, B, C, and D in Eq.(3.1)

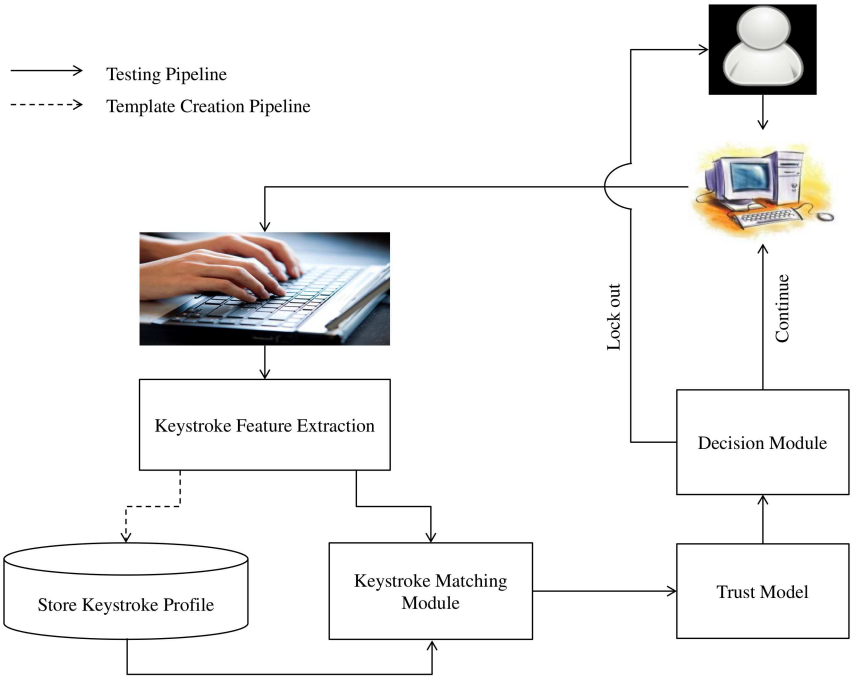

Figure 3.3: Block diagram of the proposed system.

are dependent on the type of action and were optimized by `Genetic Algorithm` optimization technique, where the cost function was to maximize $ANGA - ANIA$ (see Section 3.5.3 for description of ANGA and ANIA).

3.5.3 Performance Measure

In the testing phase the performance of the system will be measured in terms of *Average Number of Genuine Actions (ANGA)* and *Average Number of Impostor Actions (ANIA)* [19]. In this case an action of the user can be anything done with the `Keyboard`. The details explanation of the performed action was given in Section 3.4.3.

In Fig.(3.4), we see how the trust level changes when we compare a model with test data of an impostor user. In this figure is the trust level displayed on the vertical axis, while the horizontal axis displays the number of typed keys. The trust will drop (in this example) 5 times below the lockout threshold ($T_{lockout}$ marked with red color) within 500 user actions. The value of ANIA in this example equals $(N_1 + N_2 + N_3 + N_4 + N_5)/5$. We can calculate ANGA in the same way, if the genuine user is locked out based on his own test data. In our analysis, whenever a user is locked out, we reset the trust value to 100 to simulate the start of a new session after a log on to the system. In Fig.(3.4) can clearly be seen that the trust is set back 5 times to 100.

The maximum value for trust is set to 100, even if a reward would give rise to a

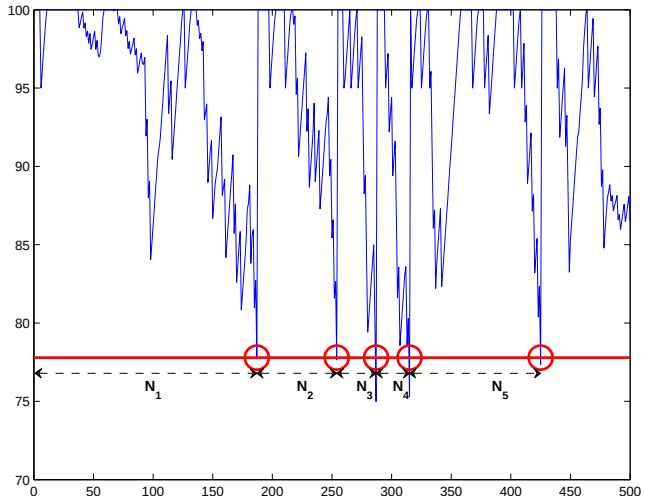

Figure 3.4: Trust for impostor test set. X-axis represents action number and Y-axis represents system trust.

a higher trust value. The reason for this is that otherwise the genuine user could be working for a longer time and build up a high trust level, say for example a trust of 1000. This means that if an impostor would take over the computer at that point of time, he could first profit from the high trust value that was build up by the genuine user, allowing him more time for malicious activities before the trust level would drop below the lockout threshold.

The goal is obviously to have ANGA as high as possible, while at the same time the ANIA value must be as low as possible. The last is obviously to assure that an impostor user can do as little as possible, hence he/she is detected as quickly as possible.

3.6 Result Analysis

In this section, we are going to analyse the results that we got by applying the analysis discussed in Section 3.5. In this research, we have performed leave-one-out testing. Therefore, we have test data of 1 genuine user and 52 impostor users.

The total number of data sets of genuine users is 53 and the total number of data sets of impostor users is $53 \times 52 = 2756$. We are going to report the results in terms of ANIA and ANGA along with the total number of impostors not detected for person based lockout threshold ($T_{lockout}$) which was optimized by using *Genetic Algorithm*.

Interpretation of the tables: When we present the results from our analysis, the results are shown for 4 possible categories. The categories are divided based on genuine user lockout and non detection of impostor users. Each of the 53 biometric subjects can be classified in 4 categories:

- **Very Good :** This is the best case category. In this category, the genuine user is never locked out, and all the 52 impostors are in fact detected as impostors.

- **Good :** This category is divided into two parts based on the trade off between security and the user friendliness.
  - The genuine user is not locked out, but some impostors are not detected.
  - The genuine user is locked out by the system but all the impostors are detected.
- **Bad :** In this category, the genuine user is locked out by the system, while some of the impostors are not detected.
- **Ugly :** In this case will the ratio between ANGA and ANIA also is low or more precisely ANIA is higher than the ANGA.

In our analysis, the system trust was calculated based on the actions described in Section 3.4.3. The column '# Users' in Tables 3.3, 3.4, and 3.5, will show how many models will fall within each of the above categories (*i.e.* the values sum up to 53). In the column 'ANGA' a value will indicate the Average Number of Genuine Actions in case indeed genuine users are locked out by the system. If the genuine users are not locked out, then we actually cannot calculate ANGA. The column 'ANIA' will display the Average Number of Impostor Actions, and is based on all impostors that are detected. The actions of the impostors that are not detected are not used in this calculation, but the number of impostors that is not detected is given in the column '# Imp. ND'. This number should be seen in relation to the number of biometric subjects in that particular category. For example in the second line of Table 3.3 we see that for 22 impostor users are not detected for a total of 10 biometric subjects. Given that for each biometric subject there are 52 impostor users, this means that 22 out of $10 \times 52 = 520$ impostor users are not detected, *i.e.* approximately 4% of the impostors for these three biometric subjects.

Table 3.3, shows the result we got from our analysis for *VP-1*. We can observe from the table that 31 participants qualify for the *Very Good* category, where the average ANIA is 304 actions. In this category none of the genuine users is locked out and all of the impostor users are detected. In the category *Good* we find 19 participants. Of these there are 10 genuine users that are not locked out but the average ANIA related to these 10 user is relatively high (1317 actions) and a total of 22 imposters were not detected (*i.e.* 4%). The remaining 9 genuine users in this category were locked out at least once by the system. For these users we have an average ANGA and ANIA of 1772 and 411 actions, respectively. There are only two participants that fall into the *Bad* category where, ANGA and ANIA are 8942 and 771 actions respectively and three imposters are not detected (*i.e.* 3% of the impostors is not detected). Finally we found that one participant falls into the *Ugly* category where ANGA and ANIA are 105 and 187 respectively. Clearly this represents a user that is not sufficiently protected by the system because he/she is locked out even faster than the impostor users.

Tables 3.4 and 3.5 represent similar findings for the analysis for *VP-2* and *VP-3* respectively. Results are very similar to the ones we reported above. *VP-2* seems to give better results than *VP-1* where the number of genuine and impostor actions for the first two categories are concerned. On the other hand we see more genuine users in the last two categories and for those categories we can also see that impostors can

Table 3.3: Results obtained from our analysis for *VP-1*.

Categories	#User	ANGA	ANIA	# Imp.	ND
Very Good	31		304		
Good	10		1317		22
	9	1772	411		
Bad	2	8942	771		3
Ugly	1	105	187		

Table 3.4: Results obtained from our analysis for *VP*-2.

Categories	#User	ANGA	ANIA	# Imp.	ND
Very Good	29		275		
Good	11		723		27
	8	3364	218		
Bad	2	2382	852		5
Ugly	3	261	335		

Table 3.5: Results obtained from our analysis for *VP-3*.

Categories	#User	ANGA	ANIA	# Imp.	ND
Very Good	30		155		
Good	14		620		29
	5	2299	220		
Bad	3	17405	573		7
Ugly	1	475	591		2

do more actions compared to *VP-1*. Results for *VP-3* seem to be better than for *VP-1* because less genuine users are locked out from the system, and if they are locked out then they can perform more actions on average. Only for the genuine user in the Ugly category we can clearly see that impostors can perform many actions and even 2 of them are not detected by the system.

3.7 Conclusion

In this chapter we have discussed a real continuous authentication system that acts on each and every separate keystroke of a user. We described the trust model where the trust is updated based on correct (reward) and deviating (penalty) typing behaviour. We have collected data of 53 participants over a period of 5 to 7 days and we have analysed that data in 3 different settings. We have not found major differences in the results found for any of these 3 analysis settings. From the results in the previous sections we can see that the described Continuous Keystroke Dynamics Authentication system gives good or very good results for the majority of genuine users. However, we also clearly see that a number of genuine users does not perform very well and that both the number of undetected impostors, as well as the average number of actions that can be performed by an impostor, preferably should be significantly lower.

In the future we will apply different classification techniques as well as different trust change functions to improve the results we have found so far. We will also investigate in detail why specific genuine users do not perform very well and use this information to create a better system.

References

[1] A.A. Ahmed and I. Traore. Biometric recognition based on free-text keystroke dynamics. *IEEE Transactions on Cybernetics*, 44(4):458–472, 2014.

[2] A.A.E. Ahmed and I. Traore. A new biometric technology based on mouse dynamics. *IEEE Transactions on Dependable and Secure Computing*, 4(3):165–179, 2007.

[3] L.C.F. Araujo, L.H.R. Jr. Sucupira, M.G. Lizarraga, L.L. Ling, and J.B.T. Yabu-Uti. User authentication through typing biometrics features. *IEEE Transactions on Signal Processing*, 53(2):851–855, 2005.

[4] K.O. Bailey, J.S. Okolica, and G.L. Peterson. User identification and authentication using multi-modal behavioral biometrics. *Computers & Security*, 43:77 – 89, 2014.

[5] D. Ballabio and M. Vasighi. A matlab toolbox for self organizing maps and supervised neural network learning strategies. *Chemometrics and Intelligent Laboratory Systems*, 118:24–32, 2012.

[6] S.P. Banerjee and D.L. Woodard. Biometric authentication and identification using keystroke dynamics: A survey. *Journal of Pattern Recognition Research*, 7(1):116–139, 2012.

[7] F. Bergadano, D. Gunetti, and C. Picardi. User authentication through keystroke

dynamics. *ACM Transactions on Information and System Security*, 5(4):367–397, 2002.

[8] R.M. Bergner. What is behavior? and so what? *New Ideas in Psychology*, 29(2):147–155, 2011.

[9] C.M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Inc., New York, NY, USA, 1995.

[10] P. Bours. Continuous keystroke dynamics: A different perspective towards biometric evaluation. *Information Security Technical Report*, 17:36–43, 2012.

[11] P. Bours and H. Barghouthi. Continuous authentication using biometric keystroke dynamics. In *Norwegian Information Securiry Conference (NISK)*, pages 1–7, 2009.

[12] C.J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[13] C.C. Chang and C.J. Lin. Libsvm: A library for support vector machines. *ACM Trans. on Intelligent Systems and Technology*, 2(3):27:1–27:27, 2011.

[14] P.S. Dowland and S.M. Furnell. A long-term trial of keystroke profiling using digraph, trigraph and keyword latencies. In *Security and Protection in Information Processing Systems*, volume 147 of *IFIP The International Federation for Information Processing*, pages 275–289. Springer US, 2004.

[15] D. Gunetti and C. Picardi. Keystroke analysis of free text. *ACM Transactions on Information and System Security*, 8(3):312–347, 2005.

[16] K. Hempstalk. *Continuous typist verification using machine learning*. PhD thesis, The University of Waikato, 2009.

[17] J. Kittler, M. Hatef, R.P.W Duin, and J. Matas. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):226–239, 1998.

[18] A. Messerman, T. Mustafic, S.A. Camtepe, and S. Albayrak. Continuous and non-intrusive identity verification in real-time environments based on free-text keystroke dynamics. In *International Joint Confonference on Biometrics (IJCB)*, pages 1–8. IEEE, 2011.

[19] S. Mondal and P. Bours. Continuous authentication using mouse dynamics. In *International Conference of the Biometrics Special Interest Group (BIOSIG)*, pages 1–12, 2013.

[20] I.T. Nabney. *Netlab: Algorithms for Pattern Recognition*, volume XVIII of *Advances in Computer Vision and Pattern Recognition*. Springer, 2002.

[21] T.T. Nguyen, T.H. Le, and B.H. Le. Keystroke dynamics extraction by independent component analysis and bio-matrix for user authentication. In *PRICAI 2010: Trends in Artificial Intelligence*, volume 6230 of *Lecture Notes in Computer Science*, pages 477–486. Springer Berlin Heidelberg, 2010.

[22] P. Ordal, D. Ganzhorn, D.V. Lu, W. Fong, and J.B. Norwood. Continuous identity verification through keyboard biometrics. *Journal of Undergraduate Research*, 4(1):20–24, 2005.

[23] A. Ross and A. Jain. Information fusion in biometrics. *Pattern Recognition Letters*, 24(13):2115–2125, 2003.

[24] S.J. Shepherd. Continuous authentication by analysis of keyboard typing characteristics. In *European Convention on Security and Detection*, pages 111–114, 1995.

[25] J.C. Stewart, J.V. Monaco, S.H. Cha, and C.C. Tappert. An investigation of keystroke and stylometry traits for authenticating online test takers. In *International Joint Conference on Biometrics (IJCB)*, pages 1–7, 2011.

[26] L. Tang, C. Yang, and Z. Wu. User authentication based on force-sensing keypad by fuzzy c-means clustering method. *Journal of Computational Information Systems*, 6(11):3659–3667, 2010.

[27] R.H.C. Yap, T. Sim, G.X.Y. Kwang, and R. Ramnath. Physical access protection using continuous authentication. In *IEEE Conference on Technologies for Homeland Security*, pages 510–512. IEEE, 2008.