INFORMATION SECURITY TECHNICAL REPORT XXX (2012) 1–8

Available online at www.sciencedirect.com

SciVerse ScienceDirect

www.compseconline.com/publications/prodinf.htm

Information Security Technical Report

Continuous keystroke dynamics: A different perspective towards biometric evaluation

Patrick Bours

Gjøvik University College, Norwegian Information Security Lab, Norway

ABSTRACT

In this paper we will describe a way to evaluate a biometric continuous keystroke dynamics system. Such a system will continuously monitor the typing behaviour of a user and will determine if the current user is still the genuine one or not, so that the system can be locked if a different user is detected. The main focus of this paper will be the way to evaluate the performance of such a biometric authentication system. The purpose of a performance evaluation for a static and for a continuous biometric authentication system differ greatly. For a static biometric system it is important to know how often a wrong decision is made. On the other hand, the purpose of a performance evaluation for a continuous biometric authentication system is not to see *if* an impostor is detected, but *how fast* he is detected. The performance of a continuous keystroke dynamic system will be tested based on this new evaluation method.

1. Introduction

When talking about biometric systems, we most generally refer to so-called static authentication systems. In a static authentication system, the identity of the person is verified at the start of that session, for example using a fingerprint to get access to a computer or using an iris scan to get access to a room. In addition to static authentication we can also consider continuous authentication, in which case the genuineness of the identity of a user is checked during a full logon period, for example checking the genuineness of the identity of a user of a computer from the moment he logs on (using static authentication) to the moment he logs off or locks the computer.

Static biometric systems are generally evaluated in terms of False Match Rate (FMR) and False Non-Match Rate (FNMR), and the overall performance is often only reported with a single value: Equal Error Rate (EER). This kind of performance evaluation works well for static biometric authentication systems, but is no longer useful when reporting the performance of continuous biometric authentication systems. For a continuous biometric system, it is not relevant to know *if* an impostor is detected, but *when* he is detected. In particular in Continuous Keystroke Dynamics (CKD) we are interested in knowing after how many keystrokes an impostor is detected.

In this paper we will we will describe a method for evaluating a CKD system by measuring the trust of the system in the genuineness of the current user. Although this paper focuses on Keystroke Dynamics (KD) can the techniques described here be applied to other biometric modalities.

The remainder of this paper is organized as follows. In Section 2 we will give an introduction into biometrics in general and will go into details of static and continuous keystroke dynamics. Furthermore we will describe the general way to evaluate a static biometric system in Section 2.2. Section 3 will be the core part of the paper where we will come up with a novel way to evaluate the performance of a continuous keystroke dynamics system. Finally we will apply the techniques from Section 3 to experimental data and present the results in Section 4. Section 5 will conclude the paper and present topics for future research.

E-mail address: patrick.bours@hig.no.

1363-4127/$ – see front matter © 2012 Elsevier Ltd. All rights reserved.
doi:10.1016/j.istr.2012.02.001

2. Background

In this section we will give an introduction into biometrics in general and KD in particular. We will also go into more details on how a performance evaluation of a biometric system in general is done.

2.1. Biometrics and keystroke dynamics

It is well known that there are three ways to verify the identity of a person. Most used method is by something we know, like passwords or PIN codes. The second method is by something we have, for example a token or smartcard. The third method of identity verification is by something we are, which is also referred to as biometrics. In fact biometrics can be split into two different categories, being physiological biometrics and behavioural biometrics. In physiological biometrics the identity of a person is verified by physical characteristics of his body, for example face, fingerprint, vein or iris. In behavioural biometrics we consider learned behaviour. In this case we authenticate a person because he learnt to perform a specific task in a personal, standardized manner. Examples of behavioural biometrics are gait, keystroke or mouse dynamics or signature.

A biometric system should comply with 7 different characteristics (Jain et al., 1998): universality, uniqueness, permanence, collectability, performance, acceptance, and circumvention. The two factors that are important for performance evaluation are uniqueness and permanence. Uniqueness refers to need to be able to distinguish between two different persons, while permanence refers to the need to be able to recognize a person over a longer period of time. Uniqueness is measured in terms of FMR, i.e. the probability that another person is recognized wrongly as the genuine user. Permanence is measured in terms of FNMR, i.e. the probability that a genuine person is not recognized as himself. We will go into more details on how to measure FMR and FNMR for a biometric system in Section 2.2.

Keystroke Dynamics is a behavioural biometric that measures how a user types on a keyboard. It is mostly used for typing on a computer keyboard, but it can also be applied to keyboards of ATMs or mobile phones too. As mentioned, in KD we recognize a person by the way he uses the keyboard. For this we can use various sources of information. The most used is timing information, where we measure when a particular key is pressed down and when it is released (Giot et al., 2009b). The timing information can be measured used a small software program. It is possible to also use other information like pressure (Saevanee and Bhattarakosol, 2009) applied to a key and the relation between a key and the finger used to type that key. In order to measure this kind of information special or additional hardware is needed. In the remainder of this paper we will restrict to timing information.

In static keystroke dynamics, the user types his password and the system does not only check if the correct password is typed, but also if the password is typed in the correct manner. Assume the password is $k_1k_2\ldots k_N$, then for each key k_i we can measure the time that key is pressed down (t_i^{down}) and when it is released (t_i^{up}). This will give us $2N$ timing values for an N letter password. Note that we ignore special keys like Shift, Alt and Ctrl for the moment. If the password for example contains a capital A, then we can regard this as 2 keys that need to be pressed: the Shift-key and the a-key. The actual timing values are not relevant, but the difference between consecutive timing values is interesting in keystroke dynamics. We define the *duration* of a key as the time it was pressed down, meaning the difference between the time it was pressed down and the time it was released:

$$d_i = t_i^{up} - t_i^{down}$$

We also define the *latency* between two consecutive keys as the "time that the keyboard is not used", or:

$$l_i = t_{i+1}^{down} - t_i^{up}$$

Note that duration is always positive, while latency can be negative, which occurs if the next key is pressed down before the current key is released. This can be either due to fluency in typing or due to the use of the special keys (Shift, Alt or Ctrl). Alternative definitions of latency exist, which ensure that the values of the latency are positive (Giot et al., 2009a). Note that for an N key password we have N durations but only $N-1$ latencies.

As in normal biometric systems a keystroke dynamics system has an enrolment phase and an authentication phase. Enrolment is slightly different for keystroke dynamics than for "ordinary" biometrics as fingerprints or iris. During enrolment the user of the biometric system types a password not just once but multiple times, say K times. The value of K varies greatly from 10 (Revett et al., 2007) to 200 (Killourhy and Maxion, 2009). An overview can be found in (Giot et al., in press), where the authors propose a method where users only need to type the password 5 times for enrolment. There are again various ways to create the template from these K typings of the password, but an often used method is to calculate the mean and standard deviations of the durations and latencies. Hence the template contains $2N-1$ pairs (μ_i, σ_i), N related to durations the $N-1$ related to latencies. A small value of σ_i means that the user types the related duration or latency in a stable manner, while a large value means less stability. During the authentication phase the user types the password again, now only once. In this case the durations and latencies are represented by the $2N-1$ values $t_1, t_2, \ldots t_{2N-1}$. As in normal biometrics we need to determine the distance between the template T and the new input I and an example distance metric is (Araújo et al., 2004):

$$d(T,I) = \sum_{i=1\ldots 2N-1} |t_i - \mu_i|/\sigma_i$$

This distance metric is referred to as Scaled Manhattan distance in Killourhy and Maxion (2009). In Killourhy and Maxion (2009) a list of 14 distance metrics is applied to one database in order to compare the performance results. In terms of Equal Error Rate (EER, see also Section 2.2) the Scaled Manhattan distance performed best.

Besides Static Keystroke Dynamics, which is applied at the beginning of a session, we can also use Keystroke Dynamics to continuously monitor if a user has changed. Little research

Please cite this article in press as: Bours P, Continuous keystroke dynamics: A different perspective towards biometric evaluation, Information Security Technical Report (2012), doi:10.1016/j.istr.2012.02.001

has been done in this area (Sim and Janakiraman, 2007; Sheperd, 1995; Gunetti and Picardi, 2005; Bergadano et al., 2002; Dowland et al., 2002; Furnell and Dowland, 2000; Dowland and Furnell, 2004). In case of CKD we cannot use the same kind of template as for Static KD, as we cannot predict what a user is going to type. The enrolment phase and template creation have to change for CKD. In CKD we will monitor the user for a period of time, e.g. 1 or 2 days, and determine from all the typing done within that period what a typical way of typing for this user is. This will then be reflected in the template. In CKD we will no longer consider a fixed text to be typed, but look at typing characteristics of single letters or combinations of letters, i.e. combinations of two letters (called digraphs) or three letters (called trigraphs) or even longer combinations. The context in which a letter is typed determines often the way it is typed. For example in the combination "*and*" the letter *n* most likely has a different duration than in a combination "*yna*" (which is a part of the word "dynamics"). A requirement for including a letter or combination of letters in the template is that it is typed often enough during the enrolment phase, so that we can get statistically sound mean and standard deviation on the timing information. In case of a single letter, the information included in the template is only the duration of that letter. In case of an *N*-letter combination the template will contain the statistics on *N* durations and $N-1$ latencies. Additionally the template might contain information on how often a particular letter or letter combination is typed during the enrolment phase.

The template does not need to include all letters or letter combinations that are typed often during the enrolment phase. We can restrict the template to a particular maximal number of features or include only those features that are "stable" in some manner, e.g. only those features that have a small standard deviation compared to the mean value.

Obviously, during template creation, statistics on all of the typing needs to be stored, but during the authentication phase only statistics needs to be collected on the typing of those features that are included in the template. When using a CKD system, we still need to use a distance metric to determine the distance between the current way of typing and the information stored in the template. This distance metric will then used to update the trust of the system in the genuineness of the current user and will not directly be used to determine if a user gets continued access to the system or if he gets locked out. More on this procedure will be described in Section 3.

2.2. *Biometric evaluation*

In this section we will describe how generally a biometric evaluation for a static authentication system is performed. There are variations possible on the method described in this section, but the general settings will be the same. Most readers will be already familiar with the information described in this section, but it is added for the sake of completeness and for a good comparison with the way a continuous authentication should be evaluated. The description will be generic for any biometric system, so not specific for keystroke dynamics.

Before we start the description we will make some assumptions. We assume we have collected data from a number of participants on an experiment. Let the number of participants be equal to *N*. Furthermore let us assume that each participant provided $K+1$ data samples, meaning we have a total of $N^*(K+1)$ data samples. The reason we assume $K+1$ data samples is because another assumption is that 1 data sample per person will be used to create a template and the remaining *K* data samples will be used for testing. In behavioural biometrics generally more data samples are used for template creation, but the most important here is the *K* data samples used for testing. Finally we do assume that we have a distance metric that will be used to determine the distance between a template and a test input. If the purpose of the evaluation is to compare performance of the system for various distance metrics, then the procedure below can be repeated for each distance metric.

We will denote the template of user *i* by T_i and the *K* test inputs from user *i* are denoted by $I_{i,j}$ where *j* runs from 1 to *K*. The distance metric has as input a template T_i and a test input $I_{k,l}$. When template and test input are from the same person, i.e. when *k* equals *i*, then this is called a genuine attempt and the score from the distance metric is labelled *genuine score*. On the other hand if template and test input are from two different persons, i.e. if *k* is not equal to *i*, then it is called an impostor attempt and the score is labelled *impostor score*. In the assumed settings it is easy to see that we will have N^*K genuine scores and $N^*(N-1)^*K$ impostor scores when comparing each template with each test input.

The genuine scores are used to determine the FNMR. When given a particular threshold, then the FNMR for that threshold can be determined as:

FNMR(T) = # of genuine scores above the threshold/total # of genuine scores.

In a similar way the FMR depends on the threshold and can be determined from the impostor scores in the following way:

FMR(T) = # of impostor scores below the threshold/total # of genuine scores.

Using these two formulae we can determine the FMR and FNMR for all different threshold values. This can then be used to derive a Decision Error Trade-off (DET) Curve. A DET-curve displays the FMR against the FNMR (Phillips et al., 2000; Jain et al., 2002). Furthermore, the most interesting point on a DET curve is the intersection of the DET curve with the $x=y$ line. For this point it holds that the FMR equals the FNMR and this point is referred to as the EER point. Performance of a static biometric authentication system is often reported with only the EER value.

3. Evaluation of continuous keystroke dynamics

In static biometrics it is important to know the probability that a genuine user is not recognized by the system (False Non-Match Rate) and what the probability is that a different user is recognized as the genuine one (False Match Rate). If a genuine user is not recognized by the system then this will cause annoyance for that user because he has to repeat the authentication procedure. At least in such a case no sensitive information has been exposed to an unauthorized person. In case of a False Match an impostor will have gained access to

Please cite this article in press as: Bours P, Continuous keystroke dynamics: A different perspective towards biometric evaluation, Information Security Technical Report (2012), doi:10.1016/j.istr.2012.02.001

the system under false pretences, i.e. using the username of the genuine user. Depending on the application one can argue which is more important: a lower FMR or a lower FNMR.

As mentioned before, for a continuous biometric authentication system it is not so important *if* an impostor is detected, but *when* he is detected. For a CKD we will assume for the moment that we do not want to lock out genuine users, but on the other hand we want to detect an impostor as fast as possible, i.e. within as few keystrokes as possible. Obviously, the later an impostor is detected, the more harm he can do while still having access to the system.

For the evaluation of a CKD we will introduce the concept of trust in a user, which can be applied to any continuous biometric authentication system. This concept will be described in Section 3.1. The way we will implement this concept of trust is through a so-called *penalty-and-reward* function, which will be described in Section 3.2. Finally in Section 3.3 will we give an overview of how a CKD system should be evaluated. In Section 4.2 we will apply the concepts introduced in this section on the data collected in an experiment with 25 participants.

3.1. Concept of trust

In a static biometric authentication system the user enters all the information before this information is matched against the stored template. After the distance function has determined the distance between the new input and the stored template, then this distance value is used to determine if the user is accepted or not to the system. This is done by comparing the distance value d to a pre-determined threshold value T, where the threshold value can be either a global or a personal value. In case the distance value is less than the threshold, then the user is accepted by the system, otherwise the system will reject the user.

In a CKD system the decision of acceptance or rejection of a user is re-evaluated after every single keystroke. A user will not be locked out of the system based on a single wrongly typed key, but if the user types in an incorrect manner over a longer period of time, then the system might lock him out. A CKD system must determine a level of trust of the genuineness of the user based on the typing behaviour. A CKD system will only lock out a user if the trust level has dropped below a pre-determined global or personal threshold T_{lockout}. If a user gets locked out by the system, then a static authentication mechanism must be used to gain access again.

The level of trust will be denoted by C and will range from 0 (absolutely no trust) to 100 (complete trust). The trust level C will be initially set to $C = 100$ after a user has successfully completed a static authentication procedure. In Section 3.2 we will discuss how the level of trust will go up or down, but it is important to notice here that the value of C will never rise above 100, even if the current user types according to the template. Once the trust level would drop below the global or personal threshold, then the system would not allow access any longer to the current user.

For a static biometric authentication method it is important that the distance between a genuine template and a genuine input is low, i.e. a genuine user is accepted. In terms of trust for a CKD system this means that the system must be designed in such a way that the typing rhythm of a genuine user will lead to a high trust value, preferably always above the threshold T_{lockout}. Similarly, for a static biometric authentication method it is important that the distance between a genuine template and an impostor input is high, i.e. an impostor is rejected by the system. Translating this again to the trust level for a CKD system, this means that the trust level of an impostor should preferably drop as fast as possible and reach below the threshold T_{lockout} as soon as possible.

In a static biometric authentication system we evaluate the performance by looking at the probability of an error, which can be either a False Match or a False Non-Match. For a CKD system these probabilities do not make sense anymore. Obviously it is important to know if an impostor gets locked out by the system, but it is more important to know how many keys he could have typed before being locked out. It is clear that an impostor who can type 1000 keys before being locked out can do more harm or can have access to more sensitive data then an impostor who is locked out after typing only 100 keys. In particular, for a continuous keystroke dynamics system, the performance could be reported in the number of keys an impostor can use before being detected. For other continuous biometric methods other criteria might be used to report performance.

3.2. Penalty-and-reward function

We have come up with a way to implement the levels of trust presented in Section 3.1, through the use of a so-called *penalty-and-reward* function. If a person types as he should, i.e. in compliance with the template, then he will get a reward in the form of an increased trust level. Similarly, if the typing does not comply with the template, he will get a penalty in the form of a decreased trust level. This means that the penalty-and-reward consists of two stages, where in the first stage it is determined if the typing complies with the template or not and in the second stage, the trust level is changed based upon the findings in the first stage.

The same techniques as in static keystroke dynamics can be used in the first stage. In this case we determine the distance between the typed text and the template. As there are no restrictions on the typed text in CKD, it could happen that the user types a key or key-combination that is not represented in the template. We will discuss this situation separately later, but for the moment we will assume that the typed key or key-combination is represented in the template. In case of a single key k, the template will contain the mean $\mu_{\text{dur},k}$ and the standard deviation $\sigma_{\text{dur},k}$ of the duration of this key. If the duration of typing this key k during the authentication phase is given by $t_{\text{dur},k}$, then we can define the distance D between the typed key and the template as:

$$D = d_{\text{dur},k} = \left|\mu_{\text{dur},k} - t_{\text{dur},k}\right| / \sigma_{\text{dur},k}$$

In case the user types a key-combination of 2 or more letters, then the template contains the mean and standard deviation of 2 or more durations and 1 or more latencies. For simplicity sake we will concentrate in the description on 2-key combinations, but this can easily be extended to 3 or more key combinations. Assume the user typed the 2-key combination p q, then the template contains for this 2-key

Please cite this article in press as: Bours P, Continuous keystroke dynamics: A different perspective towards biometric evaluation, Information Security Technical Report (2012), doi:10.1016/j.istr.2012.02.001

combination the following 3 pairs: ($\mu_{dur,p}$, $\sigma_{dur,p}$), ($\mu_{dur,q}$, $\sigma_{dur,q}$), and ($\mu_{lat,pq}$, $\sigma_{lat,pq}$). Also assume that the user current typing resulted in $t_{dur,p}$, $t_{dur,q}$, and $t_{lat,pq}$, then we can calculate the following distances:

$$d_{dur,p} = \left|\mu_{dur,p} - t_{dur,p}\right| \Big/ \sigma_{dur,p}$$
$$d_{dur,q} = \left|\mu_{dur,q} - t_{dur,q}\right| \Big/ \sigma_{dur,q}$$
$$d_{lat,pq} = \left|\mu_{lat,pq} - t_{lat,pq}\right| \Big/ \sigma_{lat,pq}$$

From these 3 distance values we need now to create a single distance value D between the current typing of the 2-key combination p q and the information on that 2-key combination in the template. There are different ways to combine the three, for example taking (1) the average of the three; (2) the minimum, median, or maximum of the three; (3) taking the average of the duration distances only; (4) taking a weighted average of the three where distances related to durations get a different weight compared to distances related to latencies. In case the average of the three is taken, then the value of D will become equal to:

$$D = \left(d_{dur,p} + d_{dur,q} + d_{lat,pq}\right)/3$$

Once the system has determined the distance D between the current typing and the template, then this value can be used in the second stage to update the trust level. The first decision to be made now is whether the trust level will go up or down, or in other words, if the distance value D will result in a penalty or a reward. This can simply be done by comparing the value D to a threshold $T_{distance}$ and in case the distance value is below the threshold, then the user gets rewarded, otherwise he gets a penalty. Now that the decision for penalty or reward has been made it remains to determine how large the penalty or reward should be, i.e. how much the trust level C should decrease or increase. As with the distance metrics, there are different possibilities here, which fall into two main categories: (1) a fixed change or (2) a variable change. In case of a fixed change than the trust level will change by a predetermined value. The simplest implementation of this would be to increase or decrease the value C by 1 in the case of a reward or a penalty. Variations of this are possible, where for example the reward for correct typing is less than the penalty for incorrect typing. In such a case we could choose to decrease the value of C by 1 for a penalty and increase it by 0.5 in case of a reward. Another variation could be to have different levels of penalty or reward. This could be a system where a reward is always an increase of C by 1, while the penalty would be a decrease of C by 0.5 in case the distance D is more than the threshold $T_{distance}$ but less than twice that threshold. In case the distance is more than $2*T_{distance}$ then the penalty could be a decrease of the trust value C by 1.

Besides the fixed changes in the trust value we can also make them more variable, depending on the distance D and the threshold $T_{distance}$. Let $\Delta = (T_{distance} - D)$, which is a positive value in case of a reward and a negative value in case of a penalty, then we can adjust the trust value by adding Δ to C to get the new trust value. In this case the better the user types (i.e. the lower the value of D), the higher the reward is. Also the opposite is true, the worse the typing is (i.e. the higher the value of D), the higher the penalty will be. Obviously a variable change in the trust level can be combined with minimal and/or maximal changes, so no unlimited penalties or rewards.

In case a typed combination is not represented in the template, then it is clear that the trust level will not increase. However, it can be debated if the trust level should not be changed or if it should be decreased. In case the trust level would not change, one can argue that an attacker could use knowledge of the template to attack the system by not typing the key-combinations that are presented in this template. A way to avoid such an attack would thus be to decrease the value C slightly when a typed key or key-combination is not represented in the template. This would ensure that attackers will get locked out sooner. We can assume that it will not work against the genuine user, as the template contains those keys and key-combinations that he types frequently and thus his correct way of typing will increase the trust level value to compensate for the decreases due to the fact that some typing patterns are not described in the template. This does imply also that the decrease in such a case should only be small. As we cannot compare a current typing that is not represented in the template to anything, the penalty in such case should be a fixed decrease of the trust level C.

Finally it is also possible to combine fixed and variable changes in the trust level, e.g. by having fixed penalties, but variable rewards. When testing a CKD system, possible options, both for distance metrics as for the penalty-and-reward function need to be tested to see which performs best.

3.3. *Performance evaluation of a CKD system*

In this section we will give a short summary of how the performance evaluation of a Continuous Keystroke Dynamic System should be done.

1. The first step is to create a template where the typing behaviour of the user is monitored for a particular period (generally one or a few days) and information is collected on how often particular keys or key-combinations are typed. If this number is above a threshold, then statistical data on the typical way of typing that key or key-combination can be included in the template.
2. Once the template exists a distance metric and a penalty-and-reward function need to be set, as well as the thresholds $T_{distance}$ and $T_{lockout}$.
3. When a user logs on to the system the trust level value C is set to 100 and the typing of the user will be monitored continuously. In case the currently typed key or key-combination
   a. is *not* in the template then the value C is decreased by a fixed value. If the value C drops below the threshold $T_{lockout}$ then the current user is locked out from the system;
   b. is in the template then the distance D between the template and current typing is calculated using the distance metric. If the distance D is
      i. below the threshold $T_{distance}$ then the penalty-and-reward function is used to increase the trust level value C;
      ii. below the threshold $T_{distance}$ then the penalty-and-reward function is used to decrease the

Please cite this article in press as: Bours P, Continuous keystroke dynamics: A different perspective towards biometric evaluation, Information Security Technical Report (2012), doi:10.1016/j.istr.2012.02.001

trust level value C. In case the value of C now drops below the threshold T_{lockout} then the current user is locked out from the system.

If in step 3 the user is locked out of the system because the trust level dropped below the threshold, then he needs to use the system's static authentication mechanism to log in again.

4. Experimental results

In this section we briefly describe the results of an experiment we have conducted to see how well a continuous keystroke dynamic system performs. We will first describe the experiment setup and the data collected in this experiment before we describe the analysis of this data. This entails amongst others the description of the distance metric and the penalty-and-reward function. Finally we will present the performance results of our experiment in Section 4.3. The experiment and the data analysis were a part of a final Master Thesis project (Bours and Barghouthi, 2009).

4.1. *Experiment design and data description*

The experiment was conducted at Gjøvik University College (GUC) in Norway and 35 persons participated in this experiment. The participants were all staff or students at GUC and of those 35 persons only 25 (20 male and 5 female) provided enough data for further analysis. The participants were asked to run a program that collected all information on their way of typing and provide the collected data back for analysis. The program should run for at least 6 days and was designed such that the participant could stop the data collection in cases where sensitive information (e.g. passwords) had to be typed. All the data was collected in a single file. After completion of the experiment the collected data was subjected to a first quick analysis. Files with a size of less than 1 MB (representing approximately 9000 keystrokes) were discarded. Only those participants who provided sufficient data were considered for further analysis, hence the drop from 35 to 25 participants.

The data collected by the program was stored in a text file, so that the participants could see what information was sent back for analysis. The file stored the key-down and key-up timing information of each typed key. Each single line in the data text file contained an identifier for the key-down or the key-up event, the "name" of the key that was pressed, the time it was pressed or released and the application in which the key was pressed. Furthermore did the data file include information to verify on which day the key was typed.

4.2. *Description of analysis*

The analysis of the data included pre-processing of the raw data file. During this pre-processing the representation of the data was changed and a single data file from a participant was split into multiple data files, each file related to a single day of data collection. In the new representation of the data it was possible to see the name of a key, the duration of each key, the name of the next key, and the latency between the key and the next key. As a method to remove noise from the data set extreme duration or latency values (either very short or very long) were removed. The data from the first day would then be used to create a template for a user. The template does contain information on single keys and on 2-key combinations. We decided to use a fixed set of single keys and 2-key combinations that are typed frequently in English or other languages, and that were typed often by all participants in the experiment. The single keys used in the template creation are E, A, T, I, N, O, S, L, Space, and Backspace. The 2-key combinations included in the template are AT, TH, HE, ME, AN, IC, IS, OF, TE, BE, OC, OR, and BY. For each of the 10 single keys, the template contained both the mean and the standard deviation of the durations of typing those single keys. For the 13 2-key combinations, the template contained the mean and standard deviation of the duration of the first and the duration of the second key, as well as the mean and standard deviation of the latencies. From each of these 23 characteristics also the number of occurrences in the template creation data was stored in the template, but these numbers were not used in the performance analysis.

Once the templates were created the remainder of the data was used to test the performance of the system. The distance metric used was the same as described in Section 3.2. The penalty-and-reward function used was the following:

$C = 100$ at start-up

$C = \min(C + R, 100)$ if $D < T_{\text{distance}}$

$C = C - (D - T_{\text{distance}})$ if $D \geq T_{\text{distance}}$

$C = C - \alpha$ if not in template

From the definition we can see that we have chosen to use a fixed reward, a variable penalty and a fixed penalty in case the typed text is not represented in the template. Testing the system with various values showed that good results were obtained then using $T_{\text{distance}} = 0.5$, $R = 1.3$, and $\alpha = 0.01$. All these values were chosen to be global, i.e. they hold for all users. The value T_{lockout} will be user specific and it is decided to choose such values that genuine users will never be locked out of the system. This means that the personal value T_{lockout} should be slightly less than the minimal trust value C that we observe when we test the remaining user data against his own template. This choice obviously influences the performance of the system, but it is clear that the average amount of keystrokes an attacker can use in this setting is highest, as in any other setting the value of T_{lockout} would be higher. Results of the analysis are given in the next subsection.

4.3. *Results*

First the personal thresholds T_{lockout} are determined for all users by checking the user's own test data against the user's template. In the second step the data of a person was checked against the template of another person. In this test we recorded how fast an impostor was detected, i.e. how many keys he could use before the trust level C dropped below the threshold

Please cite this article in press as: Bours P, Continuous keystroke dynamics: A different perspective towards biometric evaluation, Information Security Technical Report (2012), doi:10.1016/j.istr.2012.02.001

$T_{lockout}$. The average number of used keys that an attacker could type before being detected ranged from 79 (against the template of user number 5) to 348 (against the template of user number 12). These averages are taken over all 24 attackers of the template of a genuine user. The average number of keys that any attacker could type against any genuine template is 182, which is comparable to the number of keystrokes in the current sentence (which equals 187). The previous sentence gives an indication of the average amount of information that an attacker can type before being detected.

As mentioned in Section 4.1, we also collected information on the application in which the keys were typed. We have defined 3 groups of applications and created for each user 3 different templates, where each template was related to one of the groups. When performing the analysis the application in which the user was typing was taking into account again and for the distance calculation between the current typing and the template, the template for the particular application was used. In this case the performance improved significantly. The average number of keys that an impostor could type before being detected ranged from 46 (for user 5) to 181 (for user 12) and the average over all users goes down to 98, which is an improvement of more than 46% compared to the first result. Actually this improvement does not only hold for the average of all users, but can also be observed on a per user basis. The improvement per user ranges from 41.8% to 48.0%.

5. Conclusions and future research

An ordinary (static) biometric system is always evaluated in terms of errors that are made by the system. One of these errors is falsely matching the biometric input of an impostor with the biometric template of the genuine user. On the other hand the system can also falsely reject a new biometric input of the genuine user. These errors are expressed in terms of False Match Rate and False Non-Match Rate and these values depend on a personal or global threshold. When changing the threshold, FMR will increase while FNMR will decrease or vice versa. For a specific value of the threshold both values will be equal and the Equal Error Rate is defined as the value of the FMR and FNMR for this threshold. The quality of different biometric system is then compared by comparing the EER values: the lower the EER value, the better the system performs.

In a continuous biometric system we can no longer express the quality of a system based on errors made. In case of a biometric continuous keystroke dynamics system the genuine user will make errors, meaning that his/her typing will not conform to what is stored in the template of that user. This does not mean that the user should be locked out of the system immediately, only that the system's confidence in the genuineness of the user decreases. The system's confidence in the genuineness of the user will increase again when his/her typing is conforming to the stored information in the template. The confidence (or trust) level in the genuineness of the user will therefore increase and decrease during a session and once the trust level drops below a specified level, the user will be locked out of the system.

In contrast to the static biometric systems we cannot evaluate the quality of a continuous biometric system anymore in terms of EER. A continuous biometric keystroke dynamic system needs to be evaluated in terms of how fast an impostor user is detected, i.e. how many keystrokes he/she can use before the trust of the system drops below the specified level. The specified level should be chosen in such a way that the genuine user will never be locked out of the system. Thus, the lower the number of keystrokes that an impostor can use, the better the quality of a continuous biometric keystroke dynamics system is.

In this paper we have described how a continuous keystroke dynamics system needs to be evaluated. We have focussed on the differences between a static and a continuous keystroke dynamics system in terms of creating the template and determining the system thresholds. The key factor in the evaluation of a CKD system is a so-called "penalty-and-reward" function that is designed to adapt the trust level of the system. This penalty-and-reward function is an addition to the normal distance function that is used in both static and continuous biometrics. We have also provided performance results of an experiment on 25 persons to show how the penalty-and-reward function works.

Future research will be on the one hand directed towards more extensive experiments, both in number of participants and in the length of the experiment, i.e. the amount of data collected. Furthermore we will also concentrate on differences in performance for different penalty-and-reward functions and use of additional information that could influence the typing behaviour, for example type of keyboard, time of day, language or application.

REFERENCES

Araújo LCF, Sucupira LHR, Lizárrage MG, Ling LL, Yabu-uti JBT. User authentication through typing biometrics features. In: Proceedings of the 1st International Conference on Biometrics Authentication (ICBA'04); 2004. p. 694–700.

Bergadano F, Gunetti D, Picardi C. User authentication through keystroke dynamics. ACM Transactions on Information System Security; 2002:367–97.

Bours P, Barghouthi H. Continuous authentication using keystroke dynamics. In: Proceedings of the Norsk Informasjonssikkerhetskonferanse (NISK'09). p. 1–11, www.tapironline.no/fil/vis/208; 2009. Available from:.

Dowland P, Furnell S. A long-term trial of keystroke profiling using digraph, trigraph and keyword latencies. In: Security and protection in information processing systems. Springer; 2004. p. 275–89.

Dowland P, Furnell S, Papadaki M. Keystroke analysis as a method of advanced user authentication and response. In: Proceedings of the IFIP TC11 17th International Conference on Information Security: Visions and Perspectives; 2002. p. 215–26.

Furnell S, Dowland P. A conceptual architecture for real-time intrusion monitoring. Information Management & Computer Security 2000;8(2):66–76.

Giot R, El-Abed M, Rosenberger C. GREYC keystroke: a benchmark for keystroke dynamics biometric systems. In: Proceedings of the IEEE 3rd International Conference on Biometrics: Theory, Applications, and Systems (BTAS'09); 2009a.

Giot R, El-Abed M, Rosenberger C. Keystroke dynamics with low constraints SVM based passphrase enrollment. In: Proceedings of the IEEE 3rd International Conference on

Please cite this article in press as: Bours P, Continuous keystroke dynamics: A different perspective towards biometric evaluation, Information Security Technical Report (2012), doi:10.1016/j.istr.2012.02.001

Biometrics: Theory, Applications, and Systems (BTAS'09); 2009b. p. 1–6.
Giot R, El-Abed M, Hemery B, Rosenberger C. Unconstrained keystroke dynamics authentication with shared secret. Computers & Security, Elsevier Ltd, in press.
Gunetti D, Picardi C. Keystroke analysis of free text. ACM Transactions on Information System Security; 2005:312–47.
Jain A, Bolle R, Pankanti S, editors. Biometrics: personal identification in a networked society. Kluwer Academic Publishers; 1998.
Jain A, Bolle R, Pankanti S. Biometrics: personal identification in a networked society. Kluwer Academic Publishers; 2002.
Killourhy K, Maxion R. Comparing anomaly detection algorithms for keystroke dynamics. In: Proceedings of the IEEE/IFIP International Conference on Dependable System Networks (DSN'09); 2009. p. 125–34.
Phillips PJ, Martin A, Wilson CL, Przybocki M. An introduction evaluating biometric systems. Computer 2000;33(2):56–63.
Revett K, de Magalhaes S, Santos H. On the use of rough sets for user authentication via keystroke dynamics. In: Proceedings of the 13th Portuguese Conference on Progress in Artificial Intelligence (EIPA'07); 2007.
Saevanee H, Bhattarakosol P. Authenticating user using keystroke dynamics and finger pressure. In: Proceedings of the 6th IEEE Consumer Communications and Networking Conference; 2009. p. 1–2.
Sheperd SJ. Continuous authentication by analysis of keyboard typing characteristics. European Convention on Security and Detection; 1995:111–4.
Sim T, Janakiraman R. Are digraphs good for free-text keystroke dynamics?. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'07); 2007.

Please cite this article in press as: Bours P, Continuous keystroke dynamics: A different perspective towards biometric evaluation, Information Security Technical Report (2012), doi:10.1016/j.istr.2012.02.001