Unconstrained Keystroke Dynamics Authentication with Shared Secret

Romain Giot, Mohamad El-Abed, Baptiste Hemery, Christophe Rosenberger

Unconstrained Keystroke Dynamics Authentication with Shared Secret

Romain Giot*, Mohamad El-Abed, Baptiste Hemery, Christophe Rosenberger

GREYC Laboratory
ENSICAEN - University of Caen - CNRS
6 Boulevard Maréchal Juin 14000 Caen Cedex - France

Abstract

Among all the existing biometric modalities, authentication systems based on keystroke dynamics present interesting advantages. These solutions are well accepted by users and cheap as no additional sensor is required for authenticating the user before accessing to an application. In the last thirty years, many researchers have proposed, different algorithms aimed at increasing the performance of this approach. Their main drawback lies on the large number of data required for the enrollment step. As a consequence, the verification system is barely usable, because the enrollment is too restrictive. In this work, we propose a new method based on the Support Vector Machine (SVM) learning satisfying industrial conditions (*i.e.*, few samples per user are needed during the enrollment phase to create its template). In this method, users are authenticated through the keystroke dynamics of a shared secret (chosen by the system administrator). We use the GREYC keystroke database that is composed of a large number of users (100) for validation purposes. We compared the proposed method with six methods from the literature (selected based on their ability to work with few enrollment samples). Experimental results show that, eventhough the computation time to build the template can be longer with our method (54 seconds

*Corresponding author
Email addresses: `romain.giot@greyc.ensicaen.fr` (Romain Giot), `mohamad.elabed@greyc.ensicaen.fr` (Mohamad El-Abed), `baptiste.hemery@greyc.ensicaen.fr` (Baptiste Hemery), `christophe.rosenberger@greyc.ensicaen.fr` (Christophe Rosenberger)

against 3 seconds for most of the others), its performance outperforms the other methods in an industrial context (Equal Error Rate of 15.28% against 16.79% and 17.02% for the two best methods of the state-of-the-art, on our dataset and five samples to create the template, with a better computation time than the second best method).

Key words: Biometrics, Authentication, Keystroke dynamics, Support Vector Machine learning.

---

1. Introduction

Authentication systems allow entities to be recognized before using resources; these resources can be physical, like a building, or logical, like a computer application. Traditionally, individuals authenticate themselves on computers by using the classical couple of *username* and *password*. This scheme, which is based only on one factor: the knowledge of the username and the password, suffers from various security holes [1]. *Strong authentication* uses multiple authentication factors to improve security. In this case, individuals are authenticated with the help of at least two authentication methods using one or several different factors among: (1) something *we know*; (2) something *we have*; (3) something *we are*.

Biometric systems can take part in the strong authentication scheme by providing the factor what *we are* when used with one of the two other factors. We can provide strong authentication in the password authentication scheme (what *we know*) by combining it with *keystroke dynamics* [2], which is a behavioral biometric modality monitoring the way individuals type on the keyboard (what *we are*). Its main interest lies the fact that it is considered as *unobtrusive*, because users already use passwords for authentication on computers and keystroke timing captures do not affect the user's habit. Several types of keystroke dynamics systems exist in the literature and are generally based on *very long texts* [3], *passwords* [4] or *shared secrets* [5] although several studies used a shared secret without referring to this term. The biometric sample can be captured *statically* (*i.e.*, at login phase) or *continuously* (*i.e.*, during the computer session). In this study, we focus on static authentication with shared secrets. Using a shared secret means that all users use the same password. The system always acts as an authentication system, because only a certain group of people is aware of this secret (what *we know*) while all the members of the group type it differently (what *we are*). This kind of authentication is interesting and can be used in different contexts: (i) several users use the same account, but it can be useful to track which user is really using the account (in this case, we talk about *identification* if the user does not specify his own username. This case will not treated in this paper), (ii) in analogy with password-protected buildings, an application requires the same password for all users and this password is changed at regular intervals, etc.

During the verification, the system checks if the password is the required one, if it differs from what is expected, the user is rejected, otherwise, the system checks if the keystroke dynamics match. If the keystroke dynamics correspond to the claimant's, the user is accepted, otherwise he is rejected. We argue on the fact that most of the results presented in studies in the literature cannot be compared easily due to various reasons which will be presented in this paper. In order to help solve this problem, we propose a dataset whose aim is to be used as a reference database in further keystroke dynamics studies. We also propose a new method based on Support Vector Machine (SVM) [6] for unconstrained shared secret keystroke dynamics.

The paper is organized as follows: this first section has presented the objective of this work. In the second section, we present the state-of-the-art of keystroke dynamics. In the third section, we detail the proposed method. In the fourth section, we present an experimental study for the validation of the proposed method. These results are discussed in the fifth section. The sixth section discusses the results. We conclude and present some perspectives in the last section.

2. Background

In this section, biometric systems are first presented. An overview of their evaluation aspects is then provided. Finally various discussions on the differences of keystroke dynamics studies are presented.

2.1. General biometric systems

2.1.1. Presentation

The aim of biometric systems is to verify the identity of an entity which access to a resource. In the case of *physical access*, this resource can be a building or a room, whereas in the case of *logical access*, this resource can be an application on a computer.

Different biometric modalities can be classified among three main families (even though we can find slightly different characteristics in the literature like the biological one that is often forgotten):

- *Biological*: recognition based on the analysis of biological data linked to an individual (e.g., DNA, EEG analysis, ...).
- *Behavioral*: based on the analysis of an individual behavior while performing a specific task (e.g., keystroke dynamics, signature dynamics, gait, ...).
- *Morphological*: based on the recognition of different physical patterns, which are, in general, permanent and unique (e.g., fingerprint, face recognition, ...).

In this work, we are interested in a behavioral biometric modality: the *keystroke dynamics* for managing *logical access* (*i.e*, access to a computer application).

Biometric authentication systems are generally composed of two main modules: (a) the *enrollment module* which consists in creating a template (or reference) for the user with the help of one or several biometric captures (or samples), and (b) the *verification module* which consists in verifying if the provided sample belongs to the claimed user by comparing it with its template. After verification, a decision is taken to decide to accept or to reject the user depending on the result of the comparison. We can also use an optional (c) adaptive module which updates the template of a user after a successful authentication in order to reduce the intra-class variability (the biometric data are not stable which implies that different captures of the same user may be quite different).

2.1.2. Evaluation Methodologies

Many works have already been done on the evaluation of biometric systems [7, 8, 9]. This evaluation may be realized within three different aspects:

- *performance*: the objective is to measure various statistical criteria on the performance of the system (*Capacity* [10], *Equal Error Rate (EER)*, *Failure To Enroll (FTE)*, *Failure To Acquire (FTA)*, *computation time*, *Receiver Operating Characteristic (ROC) curves*, *False Acceptance Rate (FAR)*, *False Rejection Rate (FRR)* etc [8]);
- *acceptability and user satisfaction*: this gives some information on the individuals' *perception*, *opinions* and *acceptance* with regard to the system [7, 11];
- *security*: this quantifies how well a biometric system (algorithms and devices) can resist several types of logical and physical attacks such as Denial of Service (DoS) attack or spoofing or mimicking attacks [12].

In this work, we are mainly interested in performance evaluation, as our work deals with authentication algorithms and not a whole system and its working environment. The used metrics are the following ones:

FAR *False Acceptance Rate* which represents the ratio of impostors accepted by the system;

FRR *False Rejection Rate* which represents the ratio of genuine users rejected by the system;

EER *Equal Error Rate* which is the error rate of the system when it is configured in order to obtain a *FAR* value equal to the *FRR* one. We used this error rate as a measurement of performance to compare the proposed method with six existing methods from the state-of-the-art.

We believe that the three aspects (performance security, acceptability and user satisfaction) should be taken into account simultaneously when comparing different biometric systems: we cannot say that a system is good if it provides very low error rates (*i.e.*, very good performance) but has a very low user acceptance (*i.e.*, a high probability to be refused by users) [13].

2.2. Keystroke dynamics principles

In this section, we present the general principles of keystroke dynamics. The aim of keystroke dynamics systems (when used in a static authentication model) is to provide more security for password-based authentication systems which suffer of many drawbacks [14]: (i) passwords can be shared between users, (ii) passwords can be stolen (written on a piece of paper, from the database where it is stored, through network sniffing, ...), (iii) passwords can be guessed (social engineering [15]). Keystroke dynamics introduces an additional parameter to the password authentication process: something that qualifies the user or his behavior (*i.e.*, the way of typing passwords). Using this additional parameter strengthens the password authentication. The capture process is presented in Figure 1. It consists in capturing several features when the keys are pressed and released (timestamp of the event, code of the key, . . .).

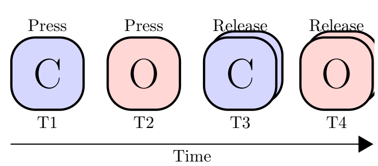

Figure 1: Information capture in a keystroke dynamics system when pressing C and O keys

The features extraction consists mainly in computing different latencies and duration times between each key. Figure 1 shows an a example where the user presses two keys of the keyboard. The user presses "C" at T1, "O" at T2 and releases "C" at T3 and "O" at T4. Note that the following relation is always respected: $T3 > T1$ and $T4 > T2$ (we always release a key after pressing it), while the following condition may not always be respected: $T2 > T3$ (because, as in our example, a user may press another key before releasing the previous one). We can extract three different types of latencies (T2-T1, T4-T3, T2-T3) which we will call PP (latency between two pressures), RR (latency between two releases), RP (latency between one release and one pressure) respectively and one type of duration (T3-T1 or T4-T2) which we will call PR (duration of a key press). In our example, T2-T3 is negative because the user presses O before releasing C (this happens frequently when a user types fast). This is not always true, but it is quite discriminating. The described process is repeated for all the keys.

While it is possible to capture these four types of extracted features, the selected features are not the same in all the studies. The PR and RP timings seem to be the most used in the literature, but sometimes, authors only speak about latency without defining which one is being used. In this paper, we use the four types of timing values (even though they are linearly dependant).

2.3. State of the art

Keystroke dynamics were experimented for the first time in 1980 in a study where seven secretaries were asked to type three different texts [16]. The results were promising, but lacked a sufficient number of users involved in the database.

The first patent on keystroke dynamics was registered in 1986 [17]. Other methods have been defined during the last twenty years, and, one of the latest were proposed recently and uses *Hidden Markov Models* [18].

In 1990, Bleha *et al.* [19] proposed an authentication method based on keystroke dynamics of the *user name* combined with a *static phrase*. They used a *Bayesian classification* and *distance measures*. The authors argued that the longer the password is, the less the error rate becomes; the error rate decreases when the number of enrolled patterns increases and results are better when using the person's name instead of a password (due to their habit of typing it). Studies using neural networks have appeared since 1993 [20]. Brown and Rogers showed the possility to use neural networks in static keystroke dynamics verification[1]. A template is created for each user by using approximately 30 user samples and 45 impostors samples where the samples are the timing information extracted from the typing of the name of the user. Obaidat and Sadoun [5] used latency and duration times of digraphs as features. They obtained an error rate of 0% but used a large number of enrolled patterns (112). Monrose and Rubin worked on keystroke dynamics for free texts [21]. They also used statistical methods, and proposed to split users in different groups in order to speed up the computation time (this is one of the first appearance of soft biometrics).

Cho and Hwang [22] allowed individuals to use *pause* helped by *cues* (which act as metronomes) to improve *unicity*, *consistency* and *discriminability* of their password and make the forgery of typing dynamics more difficult. Rodrigues *et al.* [23] used *Hidden Markov Model* in their authentication method. By using passwords only composed of numbers, they obtained an *EER* of 3.6%. This study was interesting since it demonstrated the use of keystroke dynamics for pin code authentication based environment (*i.e.*, ATM or cell phone). Sang *et al.* [24] tested the efficiency of SVM for keystroke dynamics verification. They used a one-class and a two-class SVM (in this case, simulated impostors' data are generated). The performance tradeoff and time computation were better and faster than with neural networks, but the experiment was done with only 10 individuals, a number too few to be representative (in our study, we used SVM in a different way by using pre-processing (the discretization) and post-processing (the score computing) and validating on a much bigger database). SVM has also been used in a one-class way [25]. This work uses SVM as a novelty detector to detect impostor's pattern (a novel pattern). The presented

[1] even if they use the word *identification* in this paper.

framework includes feature selection through genetic algorithm which greatly improves the recognition rate, but the number of user pattern necessary to create the template is 50.

In 2007, Hocquet *et al.* [4] automatically classied the individuals in different classes depending on various parameters and assigned different thresholds configuration for each class. They obtained an *EER* of about 2%. The classes definition and thresholds configuration are realized using a validation database. Another study [26] used various digraph information and time of typing for both username and password and discretized them into an alphabet of twenty discrete elements. The classification was done by using the rough set paradigm. They obtained an *EER* value lower than 1%. Gaussian mixture modeling of keystroke patterns was used in [27]. Hosseinzadeh and Krishnan also gave valuable informations on how to create good keystroke dynamics databases, and how to present the results. The obtained *EER* was around 4.5%. They argued that if passwords have more than 8 letters, the number of typing mistakes increases (also interpreted in the *Failure to Acquire Rate*) even though the number of recognition errors decreases.

Some studies [28, 29, 30] took into account the typing evolution of the user in order to adapt his template after each authentication. The aim of these approaches is to improve the system performance: as being a behavioral modality, keystroke patterns are subject to evolve through the life time of the keystroke dynamics authentication system. If it does not take into account this variability, we can get a high number of false rejects. It seems that in the majority of papers, the update of the biometric template is realized only with captures from the genuine users (whereas in reality, if an impostor succeeded in authenticate himself on the system, his fake pattern would be added to the template). For more information, readers can access a recent review on keystroke dynamics available in [31].

2.4. Discussion

In this section, we point out why it is really difficult to compare keystroke dynamics methods presented in the state-of-the-art.

2.4.1. Differences in Acquisition Protocols

Most of the studies in the literature use different protocols for their data acquisition [32, 33]. This is totally understandable due to the existence of different kinds of keystroke dynamics systems (static, continuous, dynamic) that require different acquisition protocols. It is known that the performance of each algorithm can vary depending on the used database [27]. In the keystroke dynamics research field, various protocols are used to collect the data. They differ on the *number of individuals* taking part in the study, the *acknowledgement* of the password (the user chooses the password, or the password is an imposed one). This impacts the typing speed and affects the *FTA* measure. They differ on the use of *different computers* (which can impact the timing accuracy depending on the operating system), *different keyboards* (which may impact on the way of typing), the *quantity* of collected data, the *duration* of the collection of the whole database, the *control* of the acquisition process (*i.e.*, acquisition done without knowledge of the researcher who can verify if it is done with respect to the protocol or made at home where no verification is possible), the use of *different* or *identical* passwords (which impacts on the quality of impostors' data). Table 1 illustrates the differences in the protocol used in existing studies in keystroke dynamics.

2.4.2. Differences on the Objective Analysis

Many performance metrics can be used to qualify a biometric system. Nevertheless, two important issues should be considered carefully during the comparison of authentication algorithms:

1. the benchmark database used, most of time, is private, and
2. the number of samples required during the enrollment phase.

Table 1: Summary of the protocols used in different studies in the state-of-the-art (A: Duration of the database acquisition, B: Number of individuals in the database, C: Number of samples required to create the template, D: Is the acquisition procedure controlled?, E: Is the threshold global?). "??" indicates that no information is provided in the article.

Paper	A	B	C	D	E	*FAR*	*FRR*
[5]	8 weeks	15	112	no	no	0%	0%
[19]	8 weeks	36	30	yes	yes	2.8%	8.1%
[23]	4 sessions	20	30	??	no	3.6%	3.6%
[4]	??	38	??	??	no	1.7%	2.1%
[26]	14 days	30	10	??	no	0.15%	0.2%
[27]	??	41	30	no	no	4.3%	4.8%
[21]	7 weeks	42	??	no	no	??	20%
[34]	4 weeks	8	12	??	??	5.58%	5.58%
[33]	8 sessions	51	200	yes	no	9.6%	9.6%

More generally speaking, it is impossible to compare a study using twenty vectors for the enrollment process with another using only five vectors (obviously, the more samples used during the enrollment phase, the better the created templates). In addition to the enrollment size, the degree of expertise of the volunteers has an impact on the illustrated performance results [35]. The same argument also holds when comparing research works using a global threshold, with those using per-user threshold. Table 1 presents the number of vectors used for creating the enrolled template and the use of a global or individual threshold for some protocols in the literature.

2.4.3. Laboratory Environment

The problem of the laboratory environment is inherent for most of keystroke dynamics studies. For this reason, most of the passwords are *artificial* ones generated differently in each study (*i.e.*, dictionary words, random password: combination of letters, numbers and symbols, etc.) and the individuals are not at ease when typing these passwords (because they do not use them daily, and they do not choose the password). In some controlled environments, individuals are in a quiet room without any interference. This does not reflect the reality

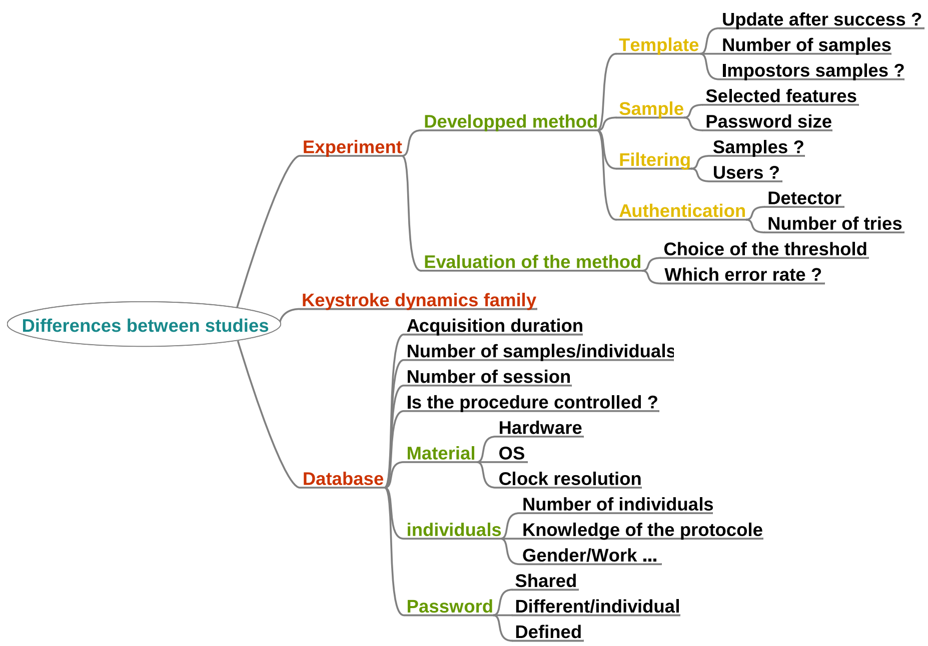

Figure 2: Summary of the differences which can be observed in keystroke dynamics studies

where we can authenticate on our machines while talking with other people or in a noisy environment. In an uncontrolled environment, nothing guarantees that all the typing patterns of a user have been done by the same user with respect to the protocol.

Figure 2 presents a mindmap of the differences between the keystroke dynamics studies referenced in this paper. These differences were also presented in [33].

2.5. Conclusion

Most systems in the literature do not propose viable solutions for a daily use at work owing to the high quantity of captures required to create the template. The diversity of the protocols implies the difficulty to compare them. The comparison between all the keystroke dynamics studies is impossible due to the use of different protocols, and especially, the lack of a public database. Another problem concerns the “configuration” of the algorithms by using different numbers of captures to create the template, or by using template adaptation methods or not. Moreover, very few works use incremental learning which is fundamental for behavioral biometric systems.

The aim of the following section is to present a solution to these problems. We compared our algorithm with six others following a rigorous protocol with a database [36] we created which contains more than 100 users and is acquired from 5 sessions separated each by, at least, one week.

3. Proposed Method

The goal of the developed method is to limit the number of captures required during the enrollment step (for obvious usability reasons) while maintaining good performance. Its originality is due to:

- the use of discretization as pre-processing,
- the computation of a decision score from the response of the SVM (in order to correct some errors of the SVM classification),
- the use of different supervised incremental learning schemes to update the biometric template of an individual after each genuine verification.

The template structure is explained in Section 4.2.2. We present some details on the above points in the following subsections. Figure 3 summarizes the global process.

3.1. Enrollment

Users are asked to type the passphrase set by the administrator five times. The feature vector is discretized in an alphabet of five values with equal size bins (we did not try other scheme of discretization) during the preprocessing steps. The bin computation method is presented in the next section. Then, a support vector machine is used for the learning step (see Figure 4). The template contains two informations: the information on the bins (in order to be able to correctly discretize test patterns) and the trained SVM.

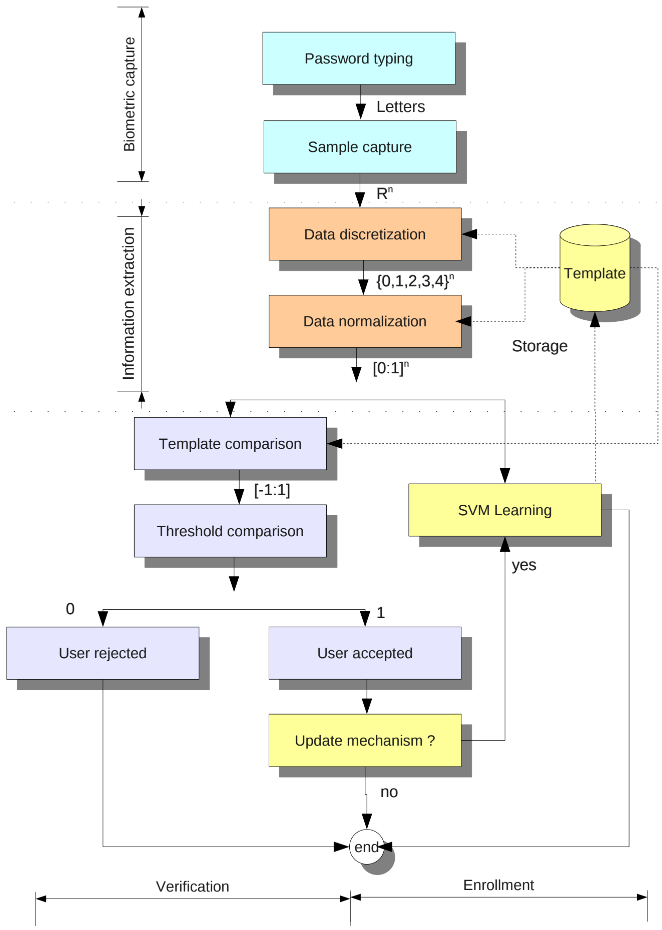

Figure 3: Global view of the system.

For the enrollment, the machine learning method is a two-classe SVM. Supposing that we have a training set $\{\mathbf{x}_i, \mathbf{y}_i\}$ where $\mathbf{x}_i$ is an enrolled vector and $\mathbf{y}_i$ the class of the associated individual (genuine/impostor). For problems with two classes, with the classes $y_i \in \{-1, 1\}$, a support vector machine [6, 37] implements the following algorithm. First, the training points $\{\mathbf{x}_i\}$ are projected into a space $\mathcal{H}$ (of possibly infinite dimension) by means of a function $\Phi(\cdot)$. The second step is to find an optimal decision hyperplane in this space. The criterion for optimality will be defined shortly. Note that for the same training set, different transformations $\Phi(\cdot)$ may lead to different decision functions.

A transformation is achieved in an implicit manner using a kernel $K(\cdot, \cdot)$ and consequently the decision function can be defined as:

$$f(\mathbf{x}) = \langle w, \Phi(\mathbf{x}) \rangle + b = \sum_{i=1}^{\ell} \alpha_i^* y_i K(\mathbf{x}_i, \mathbf{x}) + b \tag{1}$$

with $\alpha_i^* \in \mathbb{R}$. The values w and b are the parameters defining the linear decision hyperplane. In the proposed system, we use a linear function as the kernel function.

In SVMs, the optimality criterion to maximize is the margin, that is to say, the distance between the hyperplane and the nearest point $\Phi(\mathbf{x}_i)$ of the training set. The α_i^* which optimize this criterion are obtained by solving the following problem:

$$\begin{cases} \max_{\alpha_i} \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i K(\mathbf{x}_i, \mathbf{x}_j y_j) \\ \text{with constraints,} \\ 0 \ \leq \ \alpha_i \ \leq \ C \ , \\ \sum_{i=1}^{\ell} \alpha_i y_i \ = \ 0 \ . \end{cases} \tag{2}$$

where C is a penalization coefficient for data points located in or beyond the margin and provides a compromise between their numbers and the width of the margin. The biometric reference of one user is given by the $\alpha_i^*, \ i = 1 : \ell$ coefficients.

In addition to obtaining the guessed label, it is possible to calculate an estimate of its probability:

$$p(y = i|x) \approx P_{A,B}(f) = \frac{1}{1 + e^{A\hat{f}+B}} \tag{3}$$

where A and B are estimated by minimizing the negative log-likelihood function using known training data and their decision values $\hat{f}$.

As we are using impostors patterns (patterns from other users of the system, which do not mimic the genuine user behavior) during the enrollment step, the definition of a biometric reference requires the use of all existing references in the database.

When the data of all users (m is the number of users) are taken into account (this is the scenario we have chosen in the experiment), there are $5 * m$ training vectors (5 belonging to the user and $5 * (m - 1)$ belonging to the impostors). If a new user is added to the system later, different scenarios can be applied:

- We compute the template of all users. Thus, there are $m + 1$ templates to compute using $5 * (m + 1)$ samples each. This method can be very long if there are many users.

  Moreover, the performance of the method for a user might not increase by adding the 5 training vectors of the new user as impostor data. These new impostor data could be insignificant regarding to the existing 5*(m-1) impostor training vectors. Thus, the ratio between the time consumption and the performance's evolution may not lead to a good trade-off.

- We compute the template of the new user. This is more efficient because only one template has to be generated.

We have not explored which of these scenarios is the best, because we have not tested inclusion of users during the life of the system.

3.2. Verification

The verification step consists in realizing a recognition procedure with the SVM algorithm for a given biometric capture. We define a score and we use a

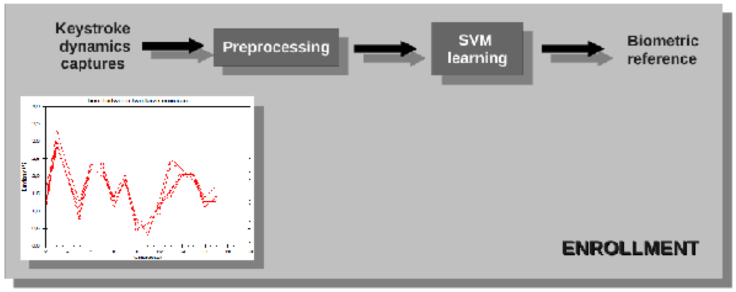

Figure 4: Enrollment scheme

threshold to decide if the user is the genuine one or an impostor. We propose different solutions to set this threshold. If the verification is successful, we use this new capture to update the biometric reference of the user in order to take into account the evolutions of keystroke dynamics (see Figure 5). The test

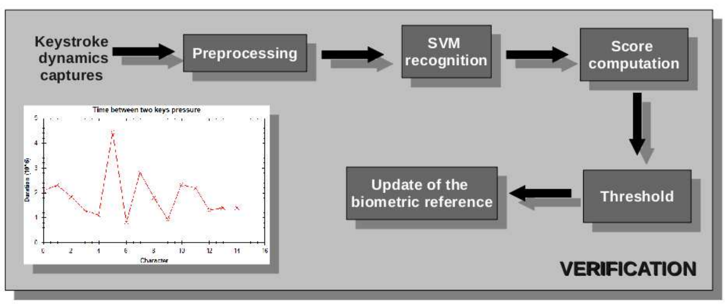

Figure 5: Verification scheme

patterns are discretized according to the information available in the template. We then classify the test pattern using the trained SVM and also estimate its probability. It is then necessary to compute a score in order to obtain a ROC curve with several points allowing a better configuration of the system. The score value for the verification test is computed as follows:

$$Score = -prb * prd \tag{4}$$

where prb stands for the probability accorded to the SVM result and prd corresponds to the class of the result which is -1 for an impostor and 1 for a genuine user.

The decision threshold can be set by following two different approaches:

- by using the same threshold for all the users
- by using a user specific threshold for each user.

The statical performance of biometric systems is different given this setting [27, 38]. It is not the aim of this work to present different ways of selecting these thresholds. Their configuration depend on the targeted security level of the system and could be defined empirically or automatically (by computing it based on the enrolled samples). Both approaches are compared in Section 4.3.5.

3.3. Updates of the biometric reference

As keystroke dynamics is a behavioral modality, it is useful to update the biometric reference when the user authenticates himself on the system. Among the different algorithms proposed in the literature, the following four methods were implemented:

- *adaptive*: a method replacing the oldest enrolled sample by the new one [29, 30]. This method is called “sliding window” in [28];
- *progressive*: a method adding the new sample in the list of enrolled vectors. This method is called “growing window” in [28];
- *average*: while the number of required enrolled vectors is not reached (set at 15), the progressive method is used, whereas when the total number is reach, the adaptive one is used. Samples are added only when they are not far too different from the enrolled samples (by comparing the difference between the test vector and the mean considering the standard deviation). This method is almost similar to [39];
- *correlation*: in this mode, the new sample is added to the database only if it is well-correlated with the enrolled samples. To test this correlation,

we use the absolute value of the Pearson correlation factor between the test vector and the average enrolled vectors. If the score is higher than 0.5, we add the vector to the template in the same way as the "average" procedure".

4. Validation

In this section, attempts are made at answering several questions about keystroke dynamics: Which are the parameters value of the verification method ? What is the performance of our method compared to the ones in the literature ? Is there any keyboard dependency ? What is the impact of the number of captures during the enrollment on the performance ? What is the best template update strategy ? Is there any computation timing difference between each method ?

4.1. Authentication Method Configuration

In this section, we present the process involved during the development of our method. A development benchmark (a private database) was used while creating the method. This is the same as the benchmark used in [40] which was created with the same software. Sixteen users provided fifteen samples in three sessions, each of each was separated by a one week period. Each session consists of five captures of the password "laboratoire greyc".

4.1.1. Choice of the kernel

When using SVM, it is necessary to choose the appropriate kernel. For this experiment we chose to compute the feature V (presented in the following paragraph) and used a multi-classe SVM (each user has his own label). Five samples per user are used to create his template, while the other samples are used for the test. The samples were chosen randomly and the experiment was launched 10 times (averaged results are presented). The SVM error rate when using different kernels is presented in Figure 6. We operated a grid search and selected the parameters giving the best results in order to reduce error rate. Having obtained these results, we chose the linear kernel as it works well and does not require a lot of parameters. This can be explained by the high dimension of our patterns and because the data is almost linearly separable [41]. In the implemented method, we use the default parameters of the *libsvm* (*i.e.*, $C = 1$). The method could be improved by selecting the best C parameter using the data in the enrollment step.

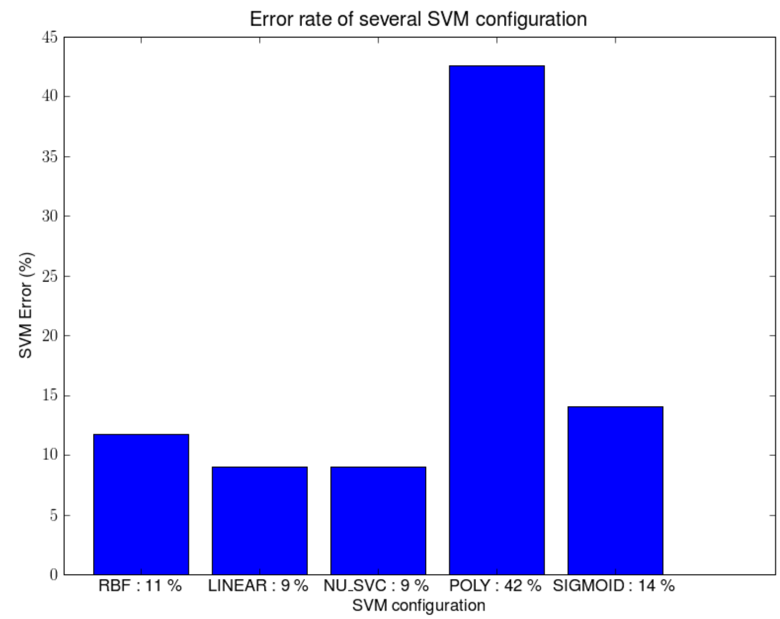

Figure 6: Performance of SVM using different kernels

After having chosen the kernel, several additional experiments allowed to conclude that using a two-class SVM provides better results.

4.1.2. Choice of the extracted features

It was previously seen that different kinds of extracted features can be used. Different configurations of extracted features were tested in order to choose the best one:

- RR or RP or PR or PP times only;

- V which is the concatenation of the four previous timing vectors. It is a feature fusion commonly used in keystroke dynamics by using the duration and one type of latency;
- *ex1* which is the timing vector V with the total typing time in addition;
- *VN* which is vector V divided by the total typing time;

Table 2 presents the *EER* value obtained from the proposed method (without using any discretization) depending on the extracted features used. It can be seen that the extracted vector V gives the best results. This is why it will be used throughout the experiments. These results are better than those in Figure 6 because a two class SVM is used instead of a multi-classe one.

Table 2: Performance of the proposed system depending on the extracted features

Input	RR	RP	PR	PP	**V**	ex1	VN
EER	07.36%	08.95%	09.00%	12.81%	**03.81%**	04.36%	04.31%

4.1.3. Numbers of bins for the discretization

As mentioned before, the proposed method uses a discretization process. Therefore, a parameter of this method is the number of bins to use. We present here the methodology adopted in order to set this number. We computed the *EER* value of the method with various numbers of bins for the discretization. Supposing we have an n-dimension template and p samples per template (*i.e.*, p enrolled captures). For each dimension, we detect the maximum and minimum value through the p templates, which gives us n different ranges. Each of these ranges is split into i bins of equal width (expect of the boundaries where they are of infinite size) (*i.e.*, $(max - min)/i)$). To discretize a template, we replace each value by the number of the bin containing it. For example, if the minimum and maximum value of the selected dimension[2] are 0 and 99 respectively and we decide to use 3 bins. The width of the bin is $(99 - 0)/3 = 33$, the first bin

[2]The process is repeated for each dimension

embeds the range $[-\infty; 33[$, the second one the range $[33; 66[$ and the last one $[66; +\infty[$. The values 40 and 120 are thus replaced by 1 and 2. Table 3 presents an example of range computation with a four-dimension pattern.

Table 3: Bin computation

	Dim 1	**Dim 2**	**Dim 3**	**Dim 4**
Sample 1	100	5	200	-8
Sample 2	110	7	300	-7
Sample 3	140	-1	250	-10
Sample 4	80	3	320	2
Min	80	-1	200	-10
Max	140	7	320	2
Width	12	1,6	24	2,4

Table 4 presents the *EER* values of different discretization methods and the differences without using such procedure. By using 5 bins, as in [38], the best results are obtained. Depending on the database, five bins could not be the best choice, but it is estimated that a number of bins between five and ten can be chosen without any problem.

Table 4: EER for different bins size during the discretization

Nb bins	2	3	4	**5**	6	7	8	9	10	20
EER	49.77%	40.05%	10.04%	**2.77%**	3.86%	3.59%	3.64%	4.55%	3.64%	3.64%
Difference	-45.96%	-36.24%	-6.23%	**1.04%**	-0.05%	0.22%	0.17%	-0.74%	0.17%	0.17%

After having configured the parameters of the authentication method with a development benchmark, it is necessary to validate the method with another one.

4.2. Experimental protocol

In this section, we define the biometric database used for testing the proposed method. Six methods were selected from the literature. Their results would be compared with the proposed method. The *EER* value is used as an objective information on the performance.

4.2.1. Definition of a validation database

Different databases of different qualities have been used in the literature, but they are rarely shared with the community (although another huge database has been constructed at the same time as ours [33]). It is known that the results can be highly dependent on the used database. The main benefit of using a common database is to help researchers avoid having to spend too much time creating a database, and to easily compare the performance of different algorithms with the same input data.

Hosseinzadeh and Krishnan [27] presented some very interesting informations on a possible method to create a good keystroke dynamics database supposed to be used with specific confidence intervals. They applied their method to create a database used in their work, but unfortunately did not make it available. In [13], we argue that to create a good behavioral biometric database, the number of required sessions have to be higher than or equal to three ; these sessions must be spaced in time, the population must be large and diversified. These requirements were not always followed in previous works.

GREYC-Keystroke is a software allowing the creation of keystroke dynamics databases. It is available for download at the following address: `http://www.ecole.ensicaen.fr/~rosenber/keystroke.html`. A screenshot of the application is shown in Figure 7. We developed this application in order to create our own keystroke dynamics database, to share it with the biometric community and to allow other researchers to create their own databases. The data are stored in an sqlite file which allows quick and easy extraction of specific information, thanks to SQL (Structured Query Language) queries.

We created a meaningful keystroke dynamics database with the help of the

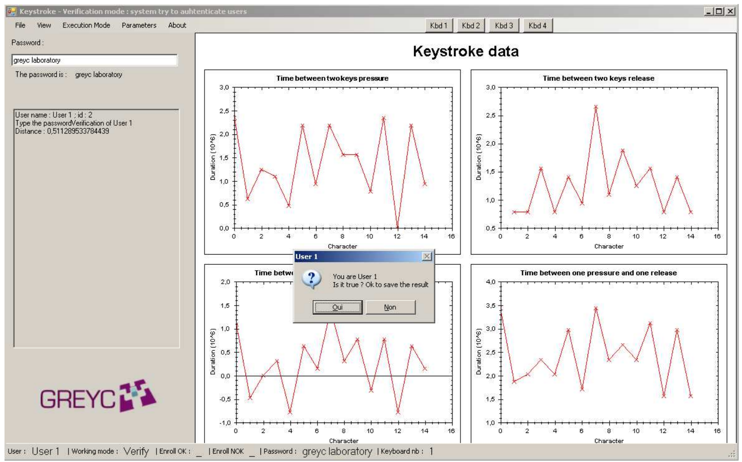

Figure 7: Screenshot of the database collecting tool

GREYC-Keystroke software by respecting various constraints presented in [13] as a guideline for creating a good behavioral biometric database (in terms of number of sessions, duration between each session, number of individuals, etc.) Most of the population in the database is composed of researchers in computer science, secretaries, students in computer science and chemistry. There are different kinds of typists: fast, slow, two fingers, all fingers, etc., but we did not tracked this information.

A total of 133 individuals in the capture process by typing the passphrase "greyc laboratory" between 5 and 107 times, between 03/18/2009 and 07/05/2009. There are 7555 available captures, and the average number of acquisitions per user is 51, with 100 of them having more than 60 captures. Most of the individuals participated in at least 5 sessions. We choose this password for two main reasons (i) this is the name of our laboratory, and using it could help the laboratory become better known, and (ii) it is a long enough password, with a good distribution of the keys on the keyboard which can help improve discriminability [34]. To type this password on an *AZERTY* keyboard, users would likely need both hands to type with as the keys are positioned across the keyboard. The position of the letters in the password on the keyboard are represented in Figure 8. We have not tested other passwords due to the amount of time required to create another database. The software is available freely in

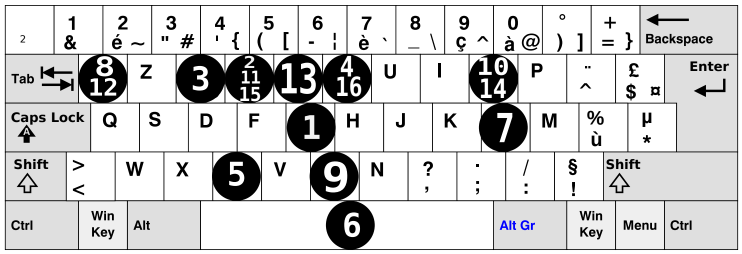

Figure 8: Position of the keys on a French AZERTY keyboard. Marked keys belong to the password. Numbers indicate the order of the character in the password. The original layout is taken from http://fr.wikipedia.org/wiki/Fichier:KB_France.svg.

the hope that fellow researchers will use it to create other databases in order to do other kind of experiments. More information about this software is available in [36].

In comparison with the databases presented in Table 1, ours is a rather large database, collected over a reasonably long period. The participants were asked to participate in one session every week (a few of them did two sessions within a week due to time constraints). Each session consists in typing the password correctly twelve times. Except for the first session during which the participants have the possibility to practice at typing the password over a short period. It came to our notice that very few of them actually participated in all the sessions by considering the number of available samples. Two keyboards (the original laptop's keyboard, and a USB keyboard plugged onto the laptop) were used to verify if the template is only dependent on a user or if it is dependent on both the user and the keyboard used. That is why during each session, individuals were asked to type the password six times on each keyboard and to alternate the keyboard each time. As the participants have to change the keyboard after typing the password each time, their is a small break before typing the next password which can help avoid the problem of users typing mechanically too similar patterns (without removing hands from the keyboard or a break between each input, the intra-class variability is too weak and not representative enough of a real keyboard usage where passwords are not typed so frequently in such a short period of time). Figure 9 shows the two different keyboards used for the experiment. It can be seen that their shapes are quite different. The key pressure is also different and the presence of the cursor ball (in red) in the middle of the laptop's keyboards is disturbing for most users.

(a) Laptop (b) USB

Figure 9: Differences of the two keyboards used during the experiment.

At the beginning of the first session, the participants were able to practice at typing of the password on the two keyboards as long as they wanted (we did not keep a track of the number of tries per user, but, most of them did it not more than five times). This training is necessary because as it is an imposed password, users are not used to typing it (especially when written in a foreign language). This is a necessary step because intra-class variability would be too significant without this test. So even if five samples are used to create the template, some user may have provided up to ten samples (where only the last five were saved and used). The participants were aware of the fact of being in a training or capturing mode. In another context where it would be up to the user to choose his own password, this training phase may not be so useful. The training step was not allowed during the other sessions.

A summary of the subset of the used database for our study is presented in Table 5. It belongs to the family of database with one unique password as in [33, 42].

Table 5: Summary of the information provided in the subset of the database used in the experiment. The users providing answers to our questionnaire are not necessarily the ones who participated in this study.

Information	Description
Users	100 users
Database sample size	6000 passphrases (60 samples per user)
Data sample length	16 characters ('greyc laboratory')
Typing error	not allowed
Controlled acquisition	yes
Age range	between 19 and 56 (repartition presented in Table 6)
Gender	approximately 73% of males and 27% of females
PC usage frequency	unknown
User profession	students, researchers, secretaries, labourers (unknown repartition)
Keyboard	2 AZERTY keyboards (1 laptop, 1 USB)
Acquisition platform	Windows XP/Greyc keystroke software

4.2.2. Biometric Sample

As mentioned earlier, different kinds of information can be extracted from the keystroke dynamics captures. In this work, we decided to use all the different latencies and durations timings. In the rest of the paper, we call v the biometric sample. This sample is created with the help of one capture of the password. The vector v is built as followed:

$$v = V = \{RR_0, PP_0, RP_0, PR_0, RR_1, PP_1, RP_1, PR_1 ...\} \tag{5}$$

where RR, PP, PR, RP stands for timing between two key releases, two key presses, one press then one release (the duration of pressure of a key), one release then one press respectively. The size of the feature vector depends on the size of the password (this is not a problem because vectors are compared to the template only if the right password has been typed). For a password of n characters, v has a dimension of $3 * (n - 1) + n$.

4.2.3. Selected methods for the comparative study

In this section, we present the methods in the literature that were selected for the comparative study. We denote v as the test vector (extracted from the test sample) and i as the size of this vector (and of the other vectors embedded in the template). As this study is done only with one password, the generated scores were not normalized.

4.2.4. Statistic-based Algorithm

Three different statistical methods are tested. They differ in the content of their computed template and the complexity of their score computing method. In the first method, the template embeds μ which is the mean of the samples [19]:

$$STAT1 = \frac{(v-\mu)^t\,(v-\mu)}{||v||.||\mu||} \tag{6}$$

For the second method, the template embeds both μ, the mean of the samples and σ, its standard deviation [4]:

$$STAT2 = 1 - \frac{1}{n}\sum_{i=1}^{n} e^{-\frac{|v_i-\mu_i|}{\sigma_i}} \tag{7}$$

The third method uses μ, the mean, σ, the standard deviation and m, the median [34] of the samples of the enrollment. We name this method $STAT3$. While the two previous methods could be represented by a simple equation, the third one is more complex because it requires several stages of calculations. First, we check if the test vector satisfies the condition specified in (8) which is vectorial (the test is done in all the dimensions of v).

$$boolres = min(\mu, m) * (0.95 - \frac{\sigma}{\mu}) \leq v \leq min(\mu, m) * (1.05 + \frac{\sigma}{\mu}) \tag{8}$$

The result of this comparison is a boolean array containing true when the criterion is verified for the required dimension of the vector, and false otherwise. In the second step, all the occurrences of false are replaced by a 0, each occurrence of true preceded by false is replaced by 1.5, while the other true values are replaced by 1. We now have now an array of numbers. The third step consists in summing all the elements of the array ; this sum is the score of this biometric method.

4.2.5. Distance Based Algorithm

We consider a simple metric based on an Euclidean distance [21]. In this method, the template is simply the list of enrolled samples. This distance is computed between the test vector and each of the enrolled samples. The score is then the minimum computed distance, as described in (9).

$$DIST = \min\left(\forall_{u \subset enrol}, \sqrt{\sum_{i=1}^{n}(u_i - v_i)^2}\right) \tag{9}$$

4.2.6. Rhythm Based Algorithm

This method consists in discretizing keystroke values along five different classes and computing a classical Hamming distance [4]. The template embeds the bin definition (in order to discretize test sample in the same way as the enrolled mean sample) and μ the discretized version of the mean enrolled samples. The score computation is described in (10).

$$RHYTHM = \frac{1}{n}\sum_{i=1}^{n} abs(class(v_i) - class(\mu_i)) \tag{10}$$

where $class(i)$ is a function returning the class of i (*i.e.*, we operate a discretization of the time) along five different classes. To compute the classes, we divide the space in five clusters of the same size between the minimal and the maximal value of the learning database (Equation 11). The assigned classes of the whole dimension of each vector is the number of the cluster.

$$cluster\_width = \frac{max(train\_data) - min(train\_data)}{5} \tag{11}$$

4.2.7. Neural Networks

Neural networks have been used in various keystroke dynamics studies [20, 5, 43, 44]. They usually require a huge number of samples in order to create the template. Nevertheless, we chose to present it here because it seems to be the closest method to our proposal (*i.e.*, use of impostors samples). In our experiment, we use a feed forward multi layer perceptron, with one hidden layer containing 45% of number of input nodes, and one output node giving a score. We empirically chose this number of hidden nodes in order to limit the computation time of the learning. The cost function is the sum of the squared difference. The constrained truncated Newton algorithm (TNC) is used as the learning method.

The learning data are arranged in the same way as our SVM-based method. No other neural network configuration has been tested. Thus, in this method, the template embeds the trained network which has been computed with clients' and impostors' enrolled samples (whereas the other methods from the literature only use clients' enrolled samples).

4.3. Experimental results

In this section, we present different experimental results on the database. In this part of the text, CONTRIB refers to our keystroke verification method.

4.3.1. Acquisition

100 volunteers were asked to fill in a questionnaire. Their age and gender are presented in the Table 6. In keystroke dynamics authentication, there is

Table 6: Diversity of the population in the database (in term of gender and age).

	Male	**Female**	**Total**
18-25	46	13	59
26-35	19	6	25
36-49	8	6	14
50+	0	2	2
Total	73	27	100

quite a large number of failures during acquisitions. These failures are due to the fact that no mistake is allowed while typing the password: a typing mistake obliges the user to type the password again from scratch. It would thus be useful to analyze the causes of these mistakes. Figure 10 presents the quantity of captures done by each user (sorted by amount of provided samples) by dividing the correct samples (in gray) by the erroneous samples (in black). The number of mistakes made is quite huge for most of the volunteers. The average mistake rate is about 20%: one input out of five is incorrect due to typing mistakes.

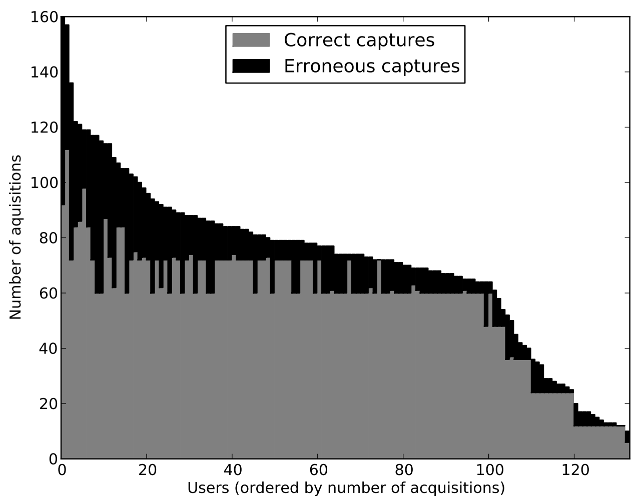

Figure 10: Number of acquisitions for each user. Correct and erroneous acquisitions are both represented

These mistakes are due to several reasons: (i) the password is quite long to type (16 characters) and it is known that typing mistakes increase if more than height characters are used [27], (ii) users may be not used to the keyboard and may hesitate a lot while typing (some participants do not often use computers and are not very familiar with keyboard usage, while others are perturbed by the use of a passphrase in English), (iii) users want to type faster than they are able to do, (iv) users forget the password (sessions were separated by one or more weeks and some users also participated in the creation of other benchmarks with different passwords), (v) users are disturbed by the environment (*e.g*, discussion with a colleague, noisy background, ...), (vi) users have to type a predefined password, (vii) knowing that their typing times are saved disturbed some users.

Usually, we type our own passwords faster than an imposed one. We have tested if this error rate of acquiring process is dependant on the user's typing speed, but it seems that there is no significant correlation (The Pearson correlation factor between typing speed and acquisition error rate is -0.28). Figure 11 represents the acquisition error rates (during the acquisition of the database) depending on the mean typing speed of users. As can be seen, the experiment reveals no dependency between these factors. In all intervals, we have high error rates.

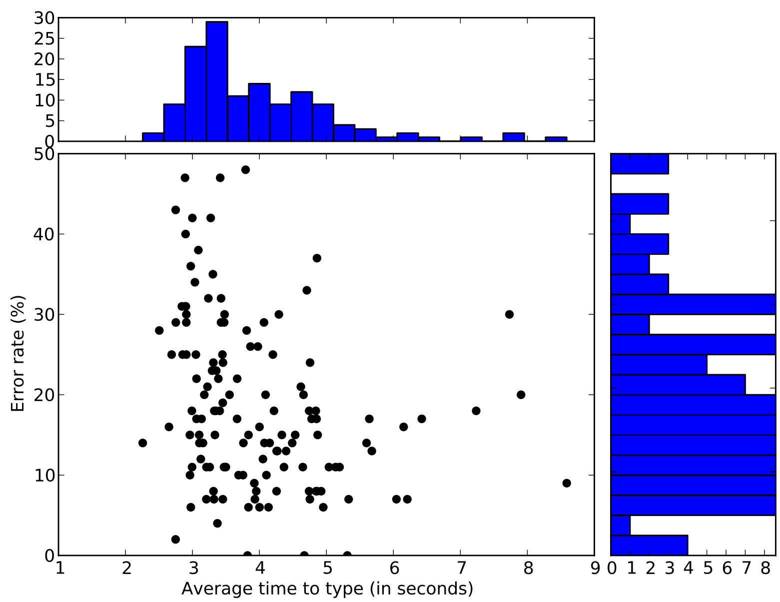

Figure 11: Acquisition error rate depending on the mean typing speed of each user.

4.3.2. Independence of the keyboard

Table 7 represents the *EER* values depending on the keyboard used for enrollment and verification using the proposed method against six from the literature? The *EER* value of each method was computed by keeping the first ten samples for enrollment, and the others for the verification process. We did not use any update mechanism in this experiment and the decision threshold is the same for all the individuals. When the keyboards used for enrollment and verification are different, the computation is done several times by selecting enrolled vectors randomly and averaging the results.

Table 7: Error(%) rates of methods depending on the keyboard configuration. "EERnm" means captures from keyboard "n" for enrollment and captures from keyboard "m" for verification, where "1", "2", "a" stands for keyboard 1, keyboard 2 and no distinction of keyboard respectively. The best *EER* value of each method is presented in italic, while the best *EER* of each configuration is presented in bold.

Method	EER11	EER22	EER12	EER21	EERaa
STAT1	24.91%	23.96%	24.73%	*23.51%*	25.50%
STAT2	17.68%	*16.55%*	17.10%	16.65%	17.58%
STAT3	15.10%	13.81%	14.68%	*13.22%*	15.43%
DIST	27.01%	26.00%	26.46%	*25.07%*	27.56%
RHYTHM	19.40%	20.09%	*19.25%*	19.50%	19.78%
NEURAL	12.65%	12.03%	12.15%	***11.21%***	13.62%
CONTRIB	**10.68%**	**10.37%**	***10.30%***	11.76%	**11.96%**
Mean	18.20%	17.54%	17.81%	17.27%	18.77%

Columns *EER11* and *EER22* represent *EER* values when enrolled and tested samples belong only to keyboard 1 and keyboard 2 respectively. Column *EER12* represents the *EER* value computed by using keyboard 1 for enrollment and keyboard 2 for verification (and vice versa for the column *EER21*). In the column *EERaa*, samples were used without distinguishing of their origin for enrollment and verification.

We can see that results are rather different, depending on keyboard configuration. Curiously, six times out of seven, the best results are obtained when the test and enrolled keyboards are different, whereas the best performances were expected when using the same keyboard for enrollment and verification. Five times out of seven, the best results are obtained when the enrolled keyboard is keyboard 2 (the USB keyboard). This can be interpreted as the fact that templates are more precise when using a classical keyboard instead of a laptop keyboard. The worst results are obtained when using both keyboards for enrollment and verification. The proposed method outperforms all the other ones for most of the configurations.

Using Kruskal-Wallis test on the 5 vectors (*EER11*, *EER22*, *EER12*, *EER21* and *EERaa*), we have a p-value equal to 0.9673. This p-value shows that there is no significant difference between the two keyboard during the enrollment and verification phases.

We have also tested if it was possible to recognize the keyboard which was used to type the password. By using an SVM with a 10-fold-cross-validation and repeating the process 50 times, we obtain a keyboard recognition accuracy of 61.48% with a standard deviation of 0.17. These results are not sufficiently different from a random choice to argue that we are able to recognize the keyboard and explains the differences.

4.3.3. Number of Enrolled Templates

An interesting point is that the trend of *EER* values depends on the number of captures used to create the biometric reference for an individual because in most studies, this number differs. The performance of the algorithms varies depending on the number of samples used to create templates. Most studies used more than twenty captures in order to create the template, whereas we think five samples per user is really the maximum for usability reasons (especially when considering that users can practice at typing the password before saving these samples). Figure 12 represents the *EER* value of different tested algorithms depending on the number of enrolled patterns. It is clear that the performance increases with the number of enrolled samples in the template. For all the methods, less than ten captures give very bad results. In order to obtain the best results, the required number of enrolled samples seems to be around forty captures (but, in this case, the number of patterns used to test the performance is very small and the results are less significant). For some methods, the performance decreases when using more than fifty captures, but it can be due to the fact that not enough samples are provided for the comparison and

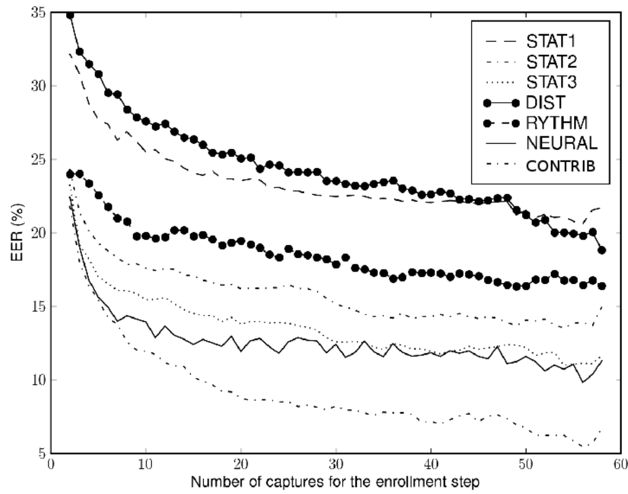

Figure 12: Evolution of the *EER* of the tested algorithms by considering the number of patterns used to create the template.

these results are not statistically relevant. Once again, the proposed method gives the best results when using more than ten captures. Even if with less than ten captures, the performances are degraded, our method outperforms most of the others and can be used in a non critical environment. Therefore, it is not absurd to use only five captures (see Section 5.1 for more information on the statistical analysis). It is up to the authentication system to be able to update the reference on the fly in order to improve the recognition rate.

4.3.4. Update of the biometric reference

Keystroke dynamics is a behavioral modality and is subject to high intraclass variability. That is why an *adaptation* (or *update*) mechanism can be applied in order to improve its performance [28]. This way, the biometric reference evolves depending on the evolution of user's manner of typing. We compared the performance of different algorithms detailed in Section 3.3 with the classic one (no incremental mechanism). Table 8 presents the *EER* values of the described methods obtained by using different incremental mechanisms. These values were computed using five enrolled vectors, the captured data from both keyboards without any distinction and using a global threshold. It is also important to note that the aim of this test is to show if there is an evolution in the way the user types. It is not our objective to decide which is the best method to use to update the template (adaptation is done after the verification of the test vector against all the templates, only with the template of the owner of the test pattern, even if the verification test fails). The presentation of the test vector is as followed: for each user, we test the first test vector against all the templates (we then update the templates). Then, we test the second test vector and so on until having tested all the test patterns. A more operational and realistic method would be to try to adapt the template of the user, if the verification is successful, with all the test vectors (even if this success is an error). This implies setting a decision threshold to obtain the FAR and FRR values, which requires a lot of computations[3].

Table 8: EER(%) values for different incremental mechanisms with five captures for the enrollment step on both keyboards. The best EER value for each adaptation method is presented in bold. The templates cannot be adapted with impostors patterns.

Method	Classic	Progressive	Adaptive	Average	Correlation
STAT1	27.7%	21.24%	23%	*20.94%*	21.67%
STAT2	19.29%	15.09%	11.71%	10.75%	*10.39%*
STAT3	17.02%	12.57%	9.78%	*8.64%*	9.21%
DIST	30.81%	23.75%	25.7%	*24.65%*	24.99%
RHYTHM	22.56%	15.49%	14.36%	13.21%	*13.19%*
NEURAL	15.79%	*8.43%*	10.03%	8.75%	10.39%
CONTRIB	**15.28%**	***6.69%***	**9.21%**	**6.96%**	**7.88%**

We can see in Table 8 that using a template update mechanism improves the performance of the system. For most algorithms, the best update mechanism is the *average* one, even if it can use fewer samples to create the template than the progressive one (where overfitting can occur). Therefore, filtering the captures

[3]For each of our cases, for each interval of threshold of each method, compute the confusion matrix and get its FAR and FRR.

before adding them to the template improves the performance by reducing the *EER* by approximately 8%. Our method gives once again the best results and provides the minimal *EER* value of less than 7% for the progressive mode.

4.3.5. Independence of the Threshold

Using individual thresholds instead of global ones is supposed to improve the performance of algorithms. Table 9 presents the improvements in term of *EER* when using individual thresholds. The *EER* were computed using: five captures to compute the template, the average update method and data from both keyboards.

Table 9: EER(%) value for each method when using global and individual thresholds, by using data of both keyboards and an incremental mechanism. The best *EER* value of each method is presented in bold.

Method	EER(global)	EER(individual)	Gains
STAT1	20.94%	*19.54%*	1.4%
STAT2	10.75%	*9.22%*	1.53%
STAT3	9.78%	*8.64%*	1.14%
DIST	24.65%	*21.53%*	3.12%
RHYTHM	13.21%	*10.02%*	**3.18%**
NEURAL	10.3%	*8.75%*	1.55%
CONTRIB	**6.96%**	***6.95%***	0.01%
Mean	13.8%	**12.1%**	1.7%

Automatically configuring the individual threshold with a system using a shared secret is possible, but it cannot be applied in the case of using a different password for each user (nobody would agree to give his own password to impostors in order to get their samples as attack). A solution to this problem is presented in [38] where users are classified in different groups depending on various parameters. These groups are created thanks to a training database, and each group shares the same parameters of the method computed with the training databases. Using the Kruskal-Wallis test, we obtain a *p-value* of 0.2774, which indicates that the gain of using individual thresholds is acceptable but not significant in comparison to the global threshold approach. Nevertheless, in general, the individual thresholds approach leads to better results (which is also clearly shown in Table 9).

4.3.6. Computation Time

The computation time taken to verify a pattern against the template is quite similar for all the methods, but, it is not the case for the template creation. Computation times for template creation of all the users (including database reading) are presented in Table 10. The timings were computed when using 5 and 10 captures to create the template with a python script on a Linux PC Desktop with an Intel Pentium IV processor with a speed of 3GHz and 1Gb of RAM.

We can see that template creation is quite fast for STAT1, STAT2, STAT3, DIST, RHYTHM and the timings are not really dependant on the number of patterns used to create the template. Computation time is higher for CONTRIB and NEURAL, but CONTRIB remains much faster than NEURAL (almost seven times faster). All the scripts were written in the Python language (both the algorithms and evaluation scripts) using the psyco [45] module which speeds up the execution of Python code (by using mechanisms similar to JIT compiler). We used the ffnet [46] library for the neural network and libsvm [47] for the SVM.

Table 10: Computation time involved in biometric reference of all the users creation for each method, when 5 and 10 captures are required to create the template.

Nb	STAT1	STAT2	STAT3	DIST	RHYTHM	NEURAL	CONTRIB
5	3s	3s	3s	3s	4s	5m 55s	54s
10	3s	3s	3s	3s	4s	30m 4s	4m 24s

5. Discussion

5.1. Confidence intervals

The performance difference between each methods could be very small which implies that these methods are not statistically different. In order to compare

Table 11: Confidence intervals of the EER when using a confidence of 95%.

Method	EER min	EER max	Interval width
STAT1	27.09%	28.42%	1.33%
STAT2	18.69%	19.85%	1.16%
STAT3	16.64%	17.71%	1.07%
DISTANCE	30.12%	31.49%	1.37%
RHYTHME	21.70%	22.95%	1.25%
NEURAL	15.32%	16.40%	1.08%
CONTRIB	14.62%	15.69%	1.07%

the algorithms more easily, we can use hypothesis tests or confidence intervals.

We computed the confidence interval of the EER when using five samples for the enrollment, no adaptation scheme and a global threshold. We applied the method presented in [48][4] and obtained the results presented in Table 11.

When using no adaptation and only five samples to build the template, our contribution performs better 90% of the time with the *STAT1*, *STAT2*, *STAT3*, *DISTANCE* and *RHYTHME* methods (we have 5% of EER outside of the confidence interval for both methods). There is a small overlap between the *NEURAL* and *CONTRIB* methods. The proposed method is slightly better in term of error rate than the *NEURAL* method, but this is not statistically significant. The method remains more interesting because template computation takes less time.

5.2. Detector Variability

Another new consequent database is available in the keystroke dynamics research area [33] ; table 12 summarises this database. This database and ours were constructed with the same objectives, but we some differences are present:

- we have twice as many users and can obtain results on a higher population of individuals or can split it in two datasets: one for configuration and one for validation;

[4]by using a confidence of 95% instead of 90%

- we obtain more intraclass variability because:
  - our sessions are more spaced (one week instead of one day) for the majority of users[5];
  - their is a break before the user retypes the password. The user does not type a password many times in one shot.
  - we use two keyboards
- nevertheless we have less sessions.

The authors at [33] tested 14 different anomaly detectors on this database (refer to this work for more information). They presented their results differently from us: a ROC curve is computed for each user and its EER is extracted, then the mean and standard deviation of the EER computed for each user is presented.

We used the same protocol in order to observe the behavior of our method on this database (which contains a lot more scores per user). With a global threshold, we obtain an EER of 10.63%, while with individual threshold, we obtain an averaged EER of 9.39% (with a standard deviation of 6.72). This would place our method at the first place of their Table 2 (which presents methods ordered by performance) because their best method (Manhattan scaled) gives an EER value of 9.6% (with a standard deviation of 6.9). Our results are also better than their one-class SVM application. These better results can be explained by the fact that we use impostor samples in our method, instead of the anomaly detector which only uses genuine samples in its template.

6. Conclusion and Perspectives

The keystroke dynamics authentication is an interesting biometric modality as it does not require any additional sensor and is well-accepted by users [11]. The performance of such systems for authentication purposes is sufficiently high.

[5] the timestamp of each capture is saved in the database

Table 12: Summary of the information provided in the database presented in [33].

Information	Description
Users	51 users
Database sample size	20400 passphrases (50*8 samples for each user)
Data sample length	10 characters ('.tie5Roanl')
Typing error	not allowed
Controlled acquisition	yes
Age range	between 18 and 70
Gender	30 males & 21 females
PC usage frequency	unknown
User profession	unknown
Keyboard	QWERTY keyboards (laptop)
Acquisition platform	Windows
Timing accuracy	200 microseconds with an external clock

The proposed method in this work outperforms all methods in the literature in deployment conditions (*i.e.*, if the number of captures for the enrollment is limited to 5) even though the computation time for enrollment remains higher. We can argue that this method is efficient when users type the same shared secret to authenticate themselves, and even if the template creation can take more time, the authentication process is as fast as in the other methods.

In order to compare the performance of the proposed method with that of the other ones, we have created a large database [36] with more than 100 users with at least 5 sessions for the acquisition phase. This database is available for the research community (some databases were used in several works [49, 25] but not used by other researchers or were not made publicly available) and has allowed us to answer multiple questions.

We saw that using individual thresholds could improve the performance of the system. One of our future works will involve identifying a method allowing a quick and easy configuration of individual thresholds without impostors' data. Good robustness was shown for these algorithms for different keyboards. The benefit of supervised template update mechanisms of the biometric reference was also demonstrated.

Several factors have to be tested in the keystroke dynamics domain. This often implies creating a new database especially designed for the corresponding tests (*i.e.*, dependency on the keyboard, computer operating systems, knowledge of the password, size of the password, content of the password). These databases can be created by merging different databases from different researchers or by creating new ones with the help of *GREYC-Keystroke* software.

A security analysis of keystroke dynamics will also be an interesting point to explore in the future (*i.e.*, analysis of security problems inherent to the modality or its implementations).

7. Acknowledgments

The authors would like to thank the "Région Basse-Normandie" and the French Research Ministry for their financial support for this work. We would also like to thank all the individuals who have participated to the definition of the keystroke dynamics database, as well as the authors of libsvm [47] and ffnet [46], the two libraries used during this project. Finally, we would like to thank the reviewers whose suggestions have helped improve the content of this paper significantly.

References

[1] A. Conklin, G. Dietrich, D. Walz, Password-based authentication: A system perspective, in: Proceedings of the 37th Hawaii International Conference on System Sciences, Hawaii, 2004, p. 10.

[2] A. Peacock, X. Ke, M. Wilkerson, Typing patterns: A key to user identification, IEEE Security & Privacy (2004) 40–47.

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

[4] S. Hocquet, J.-Y. Ramel, H. Cardot, User classification for keystroke dynamics authentication, in: The Sixth International Conference on Biometrics (ICB2007), 2007, pp. 531–539.

[5] M. Obaidat, B. Sadoun, Verification of computer users using keystroke dynamics, Systems, Man and Cybernetics, Part B, IEEE Transactions on 27 (1997) 261–269.

[6] V. Vapnik, Statistical learning theory, NY Wiley, 1998.

[7] M. Theofanos, B. Stanton, C. A. Wolfson, Usability & Biometrics: Ensuring Successful Biometric Systems, National Institute of Standards and Technology (NIST), 2008.

[8] ISO, Biometric performance testing and reporting, Tech. rep., ISO/IEC 19795-1:2006(E) (2006).

[9] A. Mansfield, J. Wayman, Best practices in testing and reporting performance of biometric devices, NPL Report CMSC 14 (02).

[10] J. Bhatnagar, A. Kumar, On estimating performance indices for biometric identification, Pattern Recognition 42 (2009) 1803 – 1815.

[11] M. El-Abed, R. Giot, B. Hemery, C. Rosenberger, A study of users' acceptance and satisfaction of biometric systems, in: 44th IEEE International Carnahan Conference on Security Technology (ICCST'10), San Jose, California, USA, 2010, pp. 1–10.

[12] ISO, Information technology - security techniques - security evaluation of biometrics, Tech. rep., ISO/IEC 19792-1:2008(E) (2008).

[13] F. Cherifi, B. Hemery, R. Giot, M. Pasquet, C. Rosenberger, Behavioral Biometrics for Human Identification: Intelligent Applications, IGI Global, 2009, Ch. Performance Evaluation Of Behavioral Biometric Systems, pp. 57–74.

[14] M. Sasse, S. Brostoff, D. Weirich, Transforming the 'weakest link' – human/computer interaction approach to usable and effective security, BT Technology Journal 19 (2001) 122–131.

[15] I. Winkler, B. Dealy, Information Security Technology?... Don't Rely on It A Case Study in Social Engineering, in: Proceedings of the Fifth USENIX UNIX Security Symposium, 1995, p. 6.

[16] R. Gaines, W. Lisowski, S. Press, N. Shapiro, Authentication by keystroke timing: some preliminary results, Tech. rep., Rand Corporation (1980).

[17] J. D. Garcia, Personal identification apparatus, patent (November 1986).

[18] V. V. Phoaha, S. Phoha, A. Ray, S. S. Joshi, S. K. Vuyyuru, Hidden markov model (hmm)-based user authentication using keystroke dynamics, patent (2009).

[19] S. Bleha, C. Slivinsky, B. Hussien, Computer-access security systems using keystroke dynamics, IEEE Transactions On Pattern Analysis And Machine Intelligence 12 (1990) 1216–1222.

[20] M. Brown, S. J. Rogers, User identification via keystroke characteristics of typed names using neural networks, Int. J. Man-Mach. Stud. 39 (1993) 994–1014.

[21] F. Monrose, Rubin, Authentication via keystroke dynamics, in: Proceedings of the 4th ACM conference on Computer and communications security, 1997, pp. 48–56.

[22] S. Cho, S. Hwang, Artificial rhythms and cues for keystroke dynamics based authentication, in: IAPR International Conference on Biometrics, Vol. 5, 2006, pp. 626–632.

[23] R. Rodrigues, G. Yared, C. do NCosta, J. Yabu-Uti, F. Violaro, L. Ling, Biometric access control through numerical keyboards based on keystroke dynamics, Lecture notes in computer science 3832 (2006) 640.

[24] Y. Sang, H. Shen, P. Fan, Parallel and Distributed Computing: Applications and Technologies, Springer, 2005, Ch. Novel impostors detection in keystroke dynamics by support vector machine, pp. 666–669.

[25] E. Yu, S. Cho, Keystroke dynamics identity verification,its problems and practical solutions, Computers & Security 23 (2004) 428–440.

[26] K. Revett, S. de Magalhaes, H. Santos, On the use of rough sets for user authentication via keystroke dynamics, Lecture notes in computer science 4874 (2007) 145.

[27] D. Hosseinzadeh, S. Krishnan, Gaussian mixture modeling of keystroke patterns for biometric applications, Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on 38 (2008) 816–826.

[28] P. Kang, S.-s. Hwang, S. Cho, Continual retraining of keystroke dynamics based authenticator, in: S.-W. Lee, S. Li (Eds.), Proceedings of ICB 2007, Vol. 4642 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2007, pp. 1203–1211.

[29] J. Lee, S. Choi, B. Moon, An evolutionary keystroke authentication based on ellipsoidal hypothesis space, in: Proceedings of the 9th annual conference on Genetic and evolutionary computation, 2007, pp. 2090–2097.

[30] K. Revett, A bioinformatics based approach to user authentication via keystroke dynamics, International Journal of Control, Automation and Systems 7 (1) (2009) 7–15.

[31] M. Karnan, M. Akila, N. Krishnaraj, Biometric personal authentication using keystroke dynamics: A review, Applied Soft Computing In Press, Corrected Proof (2010) –. doi:DOI: 10.1016/j.asoc.2010.08.003.

[32] R. Giot, M. El-Abed, C. Rosenberger, Keystroke dynamics with low constraints svm based passphrase enrollment, in: IEEE International Conference on Biometrics: Theory, Applications and Systems (BTAS 2009), 2009, p. 9.

[33] K. S. Killourhy, R. A. Maxion, Comparing anomaly-detection algorithms for keystroke dynamics, in: 39th Annual International Conference on Dependable Systems and Networks (DSN-2009), 2009, pp. 125–134.

[34] K. Revett, S. de Magalhães, H. Santos, Enhancing login security through the use of keystroke input dynamics, Lecture notes in computer science 3832 (2006) 661.

[35] E. P. Kukula, C. R. Blomeke, S. K. Modi, S. J. Elliott, Effect of human-biometric sensor interaction on fingerprint matching performance, image quality and minutiae count, IJCAT 34 (4) (2009) 270–277.

[36] R. Giot, M. El-Abed, C. Rosenberger, Greyc keystroke: a benchmark for keystroke dynamics biometric systems, in: IEEE International Conference on Biometrics: Theory, Applications and Systems (BTAS 2009), 2009, p. 9.

[37] B. Scholkopf, A. Smola, Learning with kernels: Support vector machines, regularization, Optimization, and Beyond. MIT Press 1 (2002) 2.

[38] S. Hocquet, J.-Y. Ramel, H. Cardot, Estimation of user specific parameters in one-class problems, in: ICPR '06: Proceedings of the 18th International Conference on Pattern Recognition, 2006, pp. 449–452.

[39] N. Grabham, N. White, Use of a novel keypad biometric for enhanced user identity verification, in: Instrumentation and Measurement Technology Conference Proceedings, 2008. IMTC 2008. IEEE, 2008, pp. 12–16.

[40] R. Giot, M. El-Abed, C. Rosenberger, Keystroke dynamics authentication for collaborative systems, in: The IEEE International Symposium on Collaborative Technologies and Systems (CTS), IEEE Computer Society, Baltimore, Maryland, USA, 2009, pp. 172–179. doi:10.1109/CTS.2009.5067478.

[41] C.-W. Hsu, C.-C. Chang, C.-J. Lin, A practical guide to support vector classification (2010).
URL http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf

[42] Y. Sheng, V. Phoha, S. Rovnyak, A parallel decision tree-based method for user authentication based on keystroke patterns, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 35 (4) (2005) 826–833.

[43] A. Anagun, Development of committee neural network for computer access security system, Lecture Notes in Computer Science 3982 (2006) 11.

[44] S. Cho, D. H. Han, Chigeun Han, H.-I. Kim, Web-based keystroke dynamics identity verification using neural network, Journal of organizational computing and electronic commerce 10 (2000) 295–307.

[45] A. Rigo, Psyco - home page (2010).
URL `http://psyco.sourceforge.net/`

[46] M. Wojciechowski, Feed-Forward neural network for python, Technical University of Lodz (Poland), Departement of Civil Engineering, Architecture and Environmental Engineering, http://ffnet.sourceforge.net/ (2007).

[47] C.-C. Chang, C.-J. Lin, LIBSVM: a library for support vector machines, software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm` (2001).

[48] A. Mayoue, Biosecure tool - performance evaluation of a biometric verification system (2007).
URL `http://svnext.it-sudparis.eu/svnview2-eph/ref_syst/Tools/PerformanceEvaluation/doc/howTo.pdf`

[49] J. R. M. Filho, E. O. Freire, On the equalization of keystroke timing histograms, Pattern Recognition Letters 27 (2006) 1440–1446.