User Identity Verification via Mouse Dynamics

Clint Feher[1], Yuval Elovici[1, 2], Robert Moskovitch[1], Lior Rokach[1, 2], Alon Schclar[1]

[1] Deutsche Telekom Laboratories at Ben-Gurion University,
Ben-Gurion University of the Negev, Beer-Sheva, 84105, Israel;
{clint, elovici, robertmo, liorrk, schclar} @ bgu.ac.il

[2] Department of Information Systems Engineering,
Ben-Gurion University of the Negev,
Beer-Sheva, 84105, Israel;

ABSTRACT

Computers and services such as eBanks and WebMails that identify users only at login via credentials are vulnerable to *Identity Theft.* Hackers perpetrate fraudulent activity under stolen identities by using credentials, such as passwords and smartcards, unlawfully obtained from legitimate users or by using logged-on computers that are left unattended. *User verification* methods provide an additional security layer by continuously confirming the identity of logged-on users based on their *physiological* and *behavioral* characteristics.

We introduce a novel method that continuously verifies users according to characteristics of their interaction with the pointing device of the computer e.g. mouse, touch pad and stylus.

The contribution of this work is three-fold: first, user verification is derived by combining the classification results of *each individual mouse action*, in contrast to the histogram approach in [2] in which verification is based on aggregations of mouse actions. Second, we propose a hierarchy of mouse actions from which the features are extracted. Third, we introduce new features to characterize the mouse activity which are used in conjunction with features proposed in previous work.

The proposed algorithm outperforms current state-of-the-art methods by achieving higher verification accuracy while reducing the response time of the system.

1. INTRODUCTION

Currently, most computer systems and on-line websites identify users solely by means of credentials such as passwords and PINs (personal identification numbers). These systems expose their users to *Identity Thefts* – a crime in which hackers impersonate legitimate users in order to commit fraudulent activity. Hackers exploit other identities by stealing credentials or by using logged-on computers that are left unattended.

According to the non-profit Identity Theft Resource Center (ITRC), identity theft from a consumer perspective is divided into four categories: (a) *Financial identity theft* in which stolen identity is used to obtain goods and services, for example a bank fraud; (b) *Criminal identity theft* in which a criminal impersonate a legitimate user when apprehended for a crime; (c) *Identity cloning* - using the information of another person to assume his or hers identity in daily life; and (d) *Business/commercial identity theft* - using a stolen business name to obtain credit.

A major threat to organizations is identity thefts that are committed by internal users who belong to the organization. Usually, the hacker gains access to sensitive information which can be exploited for industrial espionage, extortion, etc.

The drawbacks of identification methods that only rely on credentials lead to the introduction of *user verification* techniques which are used in conjunction with credential-based user identification. Verification methods confirm the identity of the users according to *behavioral* and *physiological biometrics* which are assumed to be relatively constant to each user, and harder to steal. The verification may be performed once during login or *continuously* throughout the session. In the latter case, biometric measurements of the user are taken at regular intervals while the user is logged-on and are compared with measurements that were collected in advance. Common behavioral biometrics include characteristics of the interaction between the user and input devices such as the mouse and keyboard. Physiological biometrics, on the other hand, use fingerprints, iris patterns and other physiological features that are unique to each individual. Thus, systems utilizing biometric user verification require a hacker who wants to infiltrate the system not only to steal the credentials of the user but also to mimic the user's *behavioral* and *physiological biometrics* making identity thefts much harder.

A major drawback of user verification methods that are based on physiological biometrics is that they require dedicated hardware devices such as fingerprint sensors and retina scanners which are expensive and are not always available. Although fingerprint verification is becoming widespread in laptops, it is still not popular enough and it cannot be used in web applications. Furthermore, fingerprints can be copied. Behavioral biometrics [26[28], on the other hand, do not require special designated devices since they use common hardware such as the mouse and keyboard.

Another major difference between physiological and behavioral biometrics is the temporal aspect - behavioral biometrics may differ depending on the time of day in which they are captured. This makes them harder to intercept and imitate but also harder to utilize. Furthermore, several challenges [26], which will be elaborated in Sections 2 and 6, still need to be overcome in order to make this approach fully operational. Consequently, behavioral biometrics was largely ignored for user verification in the past. In this paper we propose a novel user continuous verification technique based on behavioral biometrics of mouse activity.

The rest of the paper is organized as follows: in Section 2 we describe various aspects of behavioral biometrics verification systems such as general architecture and challenges inherent in their construction. We also survey currently available state-of-the-art techniques and give an in-depth description of mouse behavioral biometrics. The proposed algorithm is described in Section 3. Experimental results are presented in Section 4. Finally, we conclude in Section 5 and describe the various challenges and open problems that need further investigation in order to make this approach fully operational.

2 BEHAVIORAL BIOMETRICS SYSTEMS FOR USER VERIFICATION

A biometric-based user verification system **Error! Reference source not found.** is essentially a pattern recognition system that acquires biometric data from an individual, extracts a feature set to form a unique user signature and constructs a verification model by training it on the set of signatures. User verification is achieved by application of the model to on-line acquired signatures of the inspected user that are constructed using a process identical to the one used during the model construction.

2.1 General architecture

Figure 1 depicts the typical architecture of a behavioral biometrics user verification system. Such systems include the following components:

- *Feature acquisition* – captures the events generated by the various input devices used for the interaction (e.g. keyboard, mouse)

- *Feature extraction* – constructs a signature which characterizes the behavioral biometrics of the user.
- *Classifier* – Consists of an *inducer* (e.g. Support Vector Machines, Artificial Neural Networks, etc) that is used to build the user verification model by training on past behavior, often given by samples. During verification, the induced model is used to classify new samples acquired from the user.
- *Signature database* – A database of behavioral signatures that were used to train the model. Upon entry of a username, the signature of the user is retrieved for the verification process.

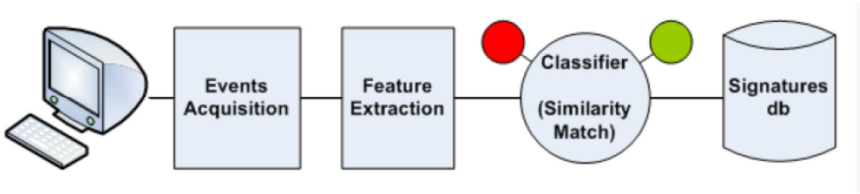


Figure 1: A typical framework of a behavioral biometric identification system.

2.2 Related work

According to [6], most common behavioral biometrics verification techniques are based on: (a) *mouse dynamics*, which are derived from the user-mouse interaction and are the focus of this paper; (b) *keystroke dynamics*, which are derived from the keyboard activity; and (c) *software interaction*, which include, for example, how features of a specific software tool are utilized.

Behavioral methods can also be characterized according to the learning approach that they employ. *Explicit learning* methods monitor user activity while performing a *predefined* task such as playing a memory game [20]. *Implicit learning techniques*, on the other hand, monitor the user during his usual activity rather than while performing a specific task. Implicit learning is more challenging due to high inconsistency owed to the variety of the performed tasks, mood changes and other influencing factors. Nevertheless, it is the best way to learn unique user behavior characteristics such as frequently performed actions.

In the following, we list current available user verification systems along with their performance evaluations. Biometric systems are usually evaluated according to *False*

Acceptance Rate (FAR), False Rejection Rate (FRR) and *Equal Error Rate (ERR)* which are described in Section 4.2.

Mouse-based user verification methods

Gamboa et al [20] proposed to verify a user based on his interaction with a memory game. The user was required to identify matching tiles and was verified based on characteristics of the mouse-strokes performed in order to reveal the tiles. A mouse-stroke was defined to be *the set of traversed points from one click to the next* and a set of one or more strokes was used in order to verify a user. Features such as curvature and velocity, were used to characterize each mouse-stroke. The learning procedure employed maximum likelihood with various distributions such as the Weibull [27] and Parzan distribution[27]s. Evaluation was performed using 50 users with a varying number of mouse-strokes having an average duration of *1* second. Equal error rates (ERRs) of 0.007 and 0.002 were achieved for 100 and 200 mouse-strokes, respectively.

Ahmed et al[1] monitored the mouse activity of users while they performed their daily tasks within their own chosen operating conditions and applications. Features were extracted and aggregated into histograms that were used to characterize each user. Four action types were defined:

- **Mouse-Move (MM)** – General movement between two points.
- **Drag-and-drop (DD)** – An action composed of the following sequence: a mouse-button down event, a movement and then a mouse-button up.
- **Point and Click (PC)** – Mouse-movement between two points followed by a click.
- **Silence** – No movement.

Every action is described by properties such as the duration, traveled distance and the direction of the movement (the travelling properties are excluded for silence actions). The general movement angle is fitted into 8 equal size sectors of the circle - each covering 45 degrees of the angle space as illustrated in **Error! Reference source not found.**.

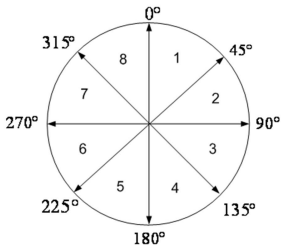


Figure 2: Angle space of movement direction: 8 equal-sized sectors of the circle. Direction 2 represents angles between 45˚ and 90˚. Direction 5 represents angles between 180˚ and 225˚.

Examples of collected actions are illustrated in Table 1.

Type of action	Distance(pixels)	Time(Seconds)	Direction
MM	50	1	3
PC	237	3	4
PC	80	2	2
Silence	-	2	-

Table 1 – Raw mouse activity data. The first action was *Mouse-move* which took 1 second, travelled in direction 3 to a distance of 50 pixels. The second action was a *Point and Click* which took 3 seconds and was to a distance of 237 pixels.

A session is defined as a sequence of mouse activities performed by a user. The sequence is limited to a predefined number of actions and a period of time. The user is characterized by a set of 7 histograms that are constructed from the raw user session data. In order to form the histograms, the data are averaged across the session and discretisized in a manner similar to the fitting of movement angle into 8 directions.

1. **Traveled Distance Histogram (TDH)** – The distribution of the travelled distance for every action type which is illustrated in **Error! Reference source not found.**(a). Only the first two features (distances 0-100 and 100-200 pixels) are used to represent the user.
2. **Action Type Histogram (ATH)** – The relative frequency of the MM, DD and PC actions within a session - illustrated in Figure 3(b).
3. **Movement Direction Histogram (MDH)** – The ratio of actions performed in each one of the eight directions. This feature is represented by 8 values and illustrated in **Error! Reference source not found.**(c).
4. **Average Movement speed per movement Direction (MDA)** – The average speed over all the actions performed in each one of the eight directions. This feature is represented by 8 values and is illustrated in **Error! Reference source not found.**(d).

5. **Average movement speed per Types of Actions (ATA)** – The average speed of performing the MM, DD and PC actions. This feature is represented by 3 features and illustrated in **Error! Reference source not found.**(e).
6. **Movement Speed compared to traveled Distance (MSD)** – Approximation of the average traveling speed for a given traveling distance (derived via a Neural Network). This feature is represented by 12 values sampled from the curve. This is illustrated in **Error! Reference source not found.**(f).
7. **Movement elapsed Time Histogram (MTH)** – The time distribution for performing an action. Represented by 2 features and illustrated in **Error! Reference source not found.**(g).

The histograms are used to construct a feature vector composed of 39 features which characterize each session of every user. **Error! Reference source not found.** summaries the extracted features.

A binary neural network model was built for every user based on the feature vectors drawn from the different histograms. The Neural Network was trained via the back propagation algorithm. Training consisted of 5 sessions - each of which contained 2000 actions (~13.55 minutes). This experiment achieved FAR of 2.4614% and FRR of 2.4649%. Shorter times (about 4 minutes) produced results of less than 24% FRR and 4.6% FAR. Thus, in order to construct accurate histograms, it requires a significant amount of mouse activities, monitored over a relatively long duration of time.

Factors	MSD	MDA	MDH	ATA	ATH	TDH	MTH
Features	12	8	8	3	3	2	3

Table 2: 39 Features used in Ahmed et al [1] to characterize mouse behavior biometrics.

Pusara and Bordley [19] proposed a user verification scheme based on mouse movements while participants browsed a predefined set of web pages using a web browser. Features such as the mean, standard deviation, third moment of distance, angle and speed were extracted from a sequence of *N* events. Three main evaluations were performed: the goal of the first was to check the behavior difference between each pair of users. Results showed that a relatively large number of users can be discriminated from one another. In the second evaluation, the discrimination of each user *x* from the *set* of the remaining

users was tested. A binary model was created for each user *x*. An FAR of 27.5% and FRR of 3.06% was achieved on the average. The third evaluation was similar to the second but used only 11 (out of the 18 that participated) users and also applied a smoothing filter to the data. An FAR 0.43% and an FRR of 1.75% were achieved.

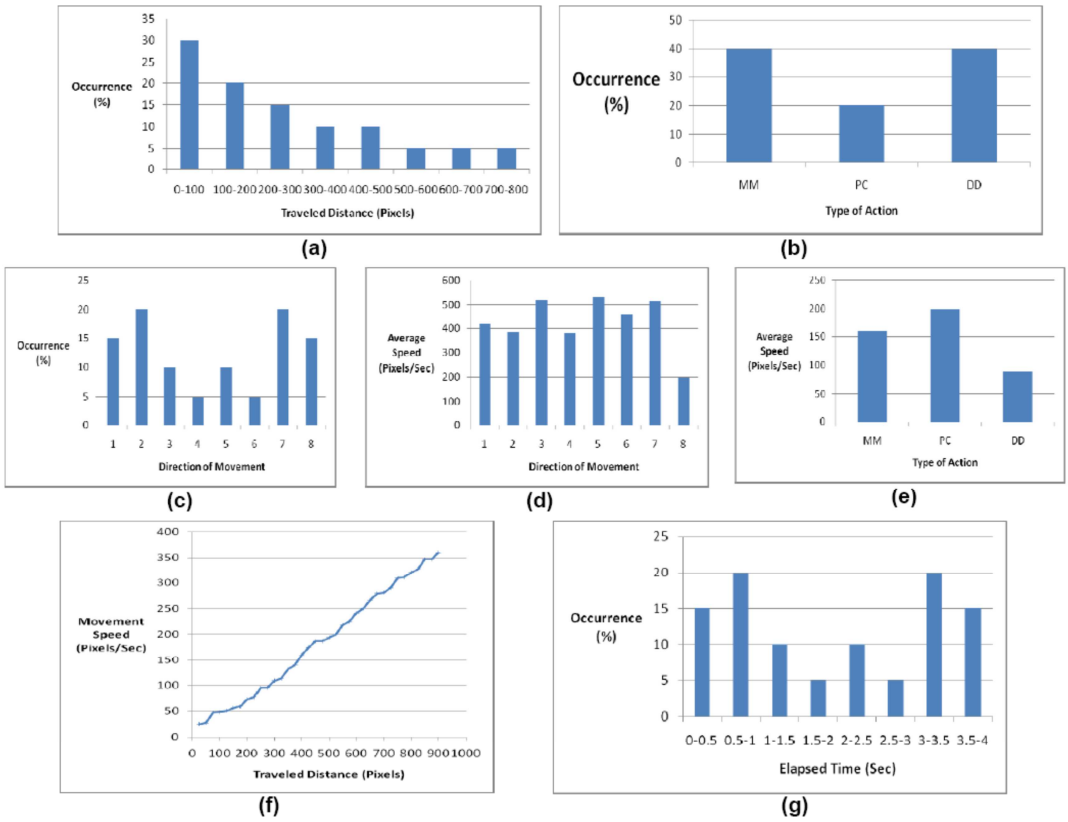


Figure 3 – Constructed histograms from user activity session in [2]. (a) Traveled Distance Histogram (TDH), (b) Action Type Histogram (ATH), (c) Movement Direction Histogram (MDH), (d) Average Movement speed per movement Direction (MDA), (e) Average movement speed per Types of Actions (ATA), (f) Movement Speed compared to traveled Distance (MSD), (g) Movement elapsed Time Histogram (MTH).

Other user verification approaches

Alternative approaches to user verification utilize *keyboard dynamics* and *software interaction* characteristics. Keyboard dynamics features include, for example, latency between consecutive keystrokes, flight time, dwell time - all based on the key down/press/up events. Keyboard-based methods are divided into methods that analyze the user behavior during an initial login attempt and methods that continuously verify the user throughout the session. The former typically construct classification model according to feature vectors that are extracted while the users type *a predefined text* (usually short)

[3,21,22,29,30,31]. Bergadano et al [3], extracted the typing durations of two (di-graph) and three (tri-graph) consecutive characters from a sample and used to associate it to a user. The extracted graphs were ordered by their duration and their relative ordering was compared to the relative order of the training samples of other users.

Keyboard-based methods for continuous verification of users extract feature vectors while the user types free text. Gunetti et al. [24] extended the approach of [3] to also handle free text. Furthermore, they proposed another distance measure based on absolute times. Curtin et al [23] constructed a nearest neighbor classifier that was trained according to the duration of common characters, transition times of common di-graphs and the occurrence frequency of special keys

Although being effective, keyboard-based verification is less suitable for web browsers since they are mostly interacted with via the mouse.

Several types of software are suggested in the literature to characterize behavioral biometrics of users. These include board games [13][14], email clients [7][8][9], programming development tools [10][11][12], command line shells [17][18] and drawing applications [15][16]. These biometric features may be partially incorporated in user verification systems.

3 THE PROPOSED METHOD

We propose a novel verification method which verifies a user based on each *individual* mouse action. This is in contrast to the histogram-based method in [2] which requires the aggregation of dozens of activities before accurate verification can be performed. Verification of each individual mouse action increases the accuracy while reducing the time that is needed to verify the identity of the user since fewer actions are required to achieve a specific accuracy level, compared to the histogram-based approach. In order to effectively characterize the mouse actions, we construct a hierarchy of features whose lowest level consists of fundamental mouse events while features at higher levels are composed of lower level ones. In general, high-level features characterize the mouse activity better than low-level ones since they convey more information regarding the task intended by the user. The verification algorithm constructs a classifier using vectors

composed of high level features, which will be described below. Some of the proposed features are new while others bare some resemblance to the ones used in [2] and [20].

3.1 A hierarchy of mouse actions

All mouse activities are formed from five *atomic mouse* events which constitute the lowest level (level 0) of the proposed hierarchy:

(i) Mouse-move Event (***m***) – occurs when the user moves the mouse from one location to another. Many events of this type occur during the entire movement – their quantity depends on the mouse resolution/sensitivity, mouse driver and operating system settings.

(ii) Mouse Left Button Down Event (***ld***) - occurs when the left mouse button is pressed,

(iii) Mouse Right Button Down Event (***rd***) - occurs when the right mouse button is pressed,

(iv) Mouse Left Button Up Event (***lu***) - occurs after the left mouse button is released,

(v) Mouse Right Button Up Event (***ru***) - occurs after the right mouse button is released

Data describing each event is typically collected by a piece of hardware or software which may dispatch it to an event handler for further processing. Mouse events are characterized by (a) their type; (b) the location of the mouse (*x* and *y* coordinates); (c) the time *t* when the event took place. Thus a mouse event is formally described by *event-type*<*x*,*y*,*t*>.

In general, higher-level actions are formed from sequences of lower-level ones. Two consecutive mouse events are considered part of a sequence if the time duration between their occurrences is below a given threshold. We refer to these thresholds as *concatenation time-thresholds* (CTT).

Basic mouse actions (level 1)

This set of basic mouse actions is constructed based on a sequence of the atomic mouse events – *m, ld, rd, lu* and *ru*. In order to concatenate two consecutive mouse events we define the following CTTs:

- **Moving CTT**: Time threshold for concatenation of two consecutive mouse move events which is denoted by $\boldsymbol{\tau_{MM}}$.

- **Mouse move to left click CTT:** The time between a mouse-move (***m***) event and a left mouse-down (***ld***) event to be concatenated into an action. The Mouse-move to Left Click concatenation time is denoted by τ_{MLM}.
- **Mouse-move to right click CTT:** The time between a mouse-move (***m***) event and a right mouse-down (***rd***) event to be concatenated into an action. The Mouse-move to Right Click concatenation Time is denoted by τ_{MRM}.
- **Mouse-down to mouse-up CTT**. The *minimal* time duration between a mouse-down event (***rd*** or ***ld***) and a mouse-up event (***ru*** or ***lu***) event to be concatenated into an action. Optional mouse-move events (***m***) may take place between the mouse-down and mouse-up events. The mouse-down to mouse-up concatenation time is denoted by τ_{DD}.

Given the above thresholds, we define the following basic (level 1) mouse actions:

Silence interval – is defined as a time interval that separates between two consecutive mouse events in which no action took place. Formally, the following silence interval are defined: (a) two consecutive mouse-move events separated by a period of time that is greater than τ_{MM} seconds; (b) a mouse-move followed by a left mouse-down event after more than τ_{MLM} seconds; and (c) a mouse-move followed by a right mouse-down event separated by more than τ_{MRM} seconds. We denote a silence interval by σ.

Left Click (LC) – refers to the action of clicking on the left mouse button. This action consists of a left button down event followed by a left button up event taking place within τ_{LC} seconds. Formally,

$$LC_{t_1}^{t_n} = \left\langle ld_{t_1}, [m_{t_2}, m_{t_3}, ..., m_{t_{n-1}}], lu_{t_n} \mid t_n - t_1 \le \tau_{LC} \right\rangle$$

t_1 and t_n denote the time points at which the left button down and left button up events took place, respectively. The $[m_{t_2}, m_{t_3}, ..., m_{t_{n-1}}]$ refer to optional mouse move events taking place between the mouse down and mouse up events.

Right Click (RC) – denoted the action of clicking on the right mouse button which is composed of a right button up event taking place after a right button down event within τ_{RC} seconds. Formally,

$$RC_{t_1}^{t_n} = \left\langle rd_{t_1}, [m_{t_2}, m_{t_3}, ..., m_{t_{n-1}}], ru_{t_n} \mid t_n - t_1 \le \tau_{RC} \right\rangle$$

Mouse-move Sequence (MMS) – refers to action of moving the mouse from one position to another. This action is defined as a sequence of mouse-move events in which the time gap between every consecutive pair of events is less than τ_{MM}. Formally,

$$MMS_{t_1}^{t_n} = \left\langle m_{t_1}, m_{t_2}, ..., m_{t_n} \mid \forall 1 \le k \le n-1 : (t_{k+1} - t_k \le \tau_{MM}) \right\rangle$$

Drag-and-Drop (DD) – denotes the action in which the user presses one of the mouse buttons, moves the mouse while the button is being pressed and releases the button at the end of the movement. Using atomic events, this action begins with a left or right mouse-down event followed by a sequence of mouse-move events and terminates with a left or right mouse-up event, respectively. The minimal time between the left down event and left up event exceeds τ_{DD}. Formally:

$$DD = \left\langle d_{t_1}, m_{t_2}, m_{t_3}, ..., m_{t_{n-1}}, u_{t_n} \mid t_n - t_1 > \tau_{DD} \right\rangle$$

where the duration of the action has to be greater than the click time, i.e. $\tau_{DD} > \tau_{LC}$ and $\tau_{DD} > \tau_{RC}$, for left button and right button usage, respectively.

The level 1 mouse actions – LC, RC, MMS and DD – are illustrated in Figs. 4(a)-(d), respectively.

Level 2 mouse actions

The next level of mouse actions is composed of level 1 actions and level 0 (atomic) events:

Mouse-move Action (MM) – A sequence of mouse-move events followed by silence time **σ**. Formally:

$$MM = MMS, \sigma$$

Double Click Action (DC) – is composed of a two consecutive left clicks in which the mouse-up of the first click and the mouse-down of the second one occur within an interval of τ_I. Formally:

$$DC = \left\langle LC_{ct_1}^{ct_2} \cdot LC_{ct_3} \mid ct_3 - ct_2 \le \tau_I \right\rangle$$

The level 2 mouse actions – DC and MM – are illustrated in Figs. 4(e) and 4(f), respectively.

Level 3 mouse actions

This is the highest level of mouse actions. The actions in this level are composed of level 1 and level 2 actions as follows:

Mouse-move and Left Click Action (MM_LC) – is composed of a sequence of mouse-move events followed by a left click taking place at most $\boldsymbol{\tau_{MLM}}$ seconds after the last mouse-move event. Formally:

$$MM\ \_LC = \left\langle MMS_{t_1}^{t_{n-1}} \cdot LC_{t_n} \mid t_n - t_{n-1} \le \tau_{MLM} \right\rangle$$

Mouse-move and Right Click Action (MM_RC) – consists of a sequence of mouse-move events and a right click taking place at most $\boldsymbol{\tau_{MRM}}$ seconds after the last mouse move event. Formally:

$$MM\ \_RC = \left\langle MMS_{t_1}^{t_{n-1}} \cdot RC_{t_n} \mid t_n - t_{n-1} \le \tau_{MRM} \right\rangle$$

Mouse-move and Double Click Action (MM_DC) – is defined as a sequence of mouse-move events which are followed by a double left click. Formally:

$$MM\ \_DC = \left\langle MMS_{t_1}^{t_n} \cdot LC_{ct_1}^{ct_2} \cdot LC_{ct_3} \mid ct_1 - t_n \le \tau_{MLM}, ct_3 - ct_2 \le \tau_I \right\rangle$$

Mouse-move and Drag-and-drop Action (MM_DD) – is composed of a sequence of mouse-move events, a left/right mouse-down event, another sequence of mouse-move events and a left/right mouse-up event, respectively. Formally,

$$MM\ \_DD = \left\langle MMS_{t_1}^{t_n} \cdot d_{t_{m+1}}, m_{t_{m+2}}, m_{t_{m+3}}, ..., m_{t_{m+k}}, u_{t_{m+k+1}} \mid t_{m+1} - t_{\mathrm{n}} \le \tau_M, t_{m+k+1} - t_{m+1} > \tau_C \right\rangle$$

where $d_{t_{m+1}}$ denotes when the mouse down event took place, $u_{t_{m+k+1}}$ is when the mouse-up event occurred and

$$d_t = ld_t, \tau_C > \tau_{LC}, \tau_M > \tau_{MLM} \text{ (for left button)}$$
$$d_t = rd_t, \tau_C > \tau_{RC}, \tau_M > \tau_{MRM} \text{ (for right button)}$$

The level 3 mouse actions – MM_LC, MM_RC, MM_DC and MM_DD – are illustrated in Figs. 4(g)-(j), respectively. An overall view of the feature hierarchy is depicted in Fig. 5.

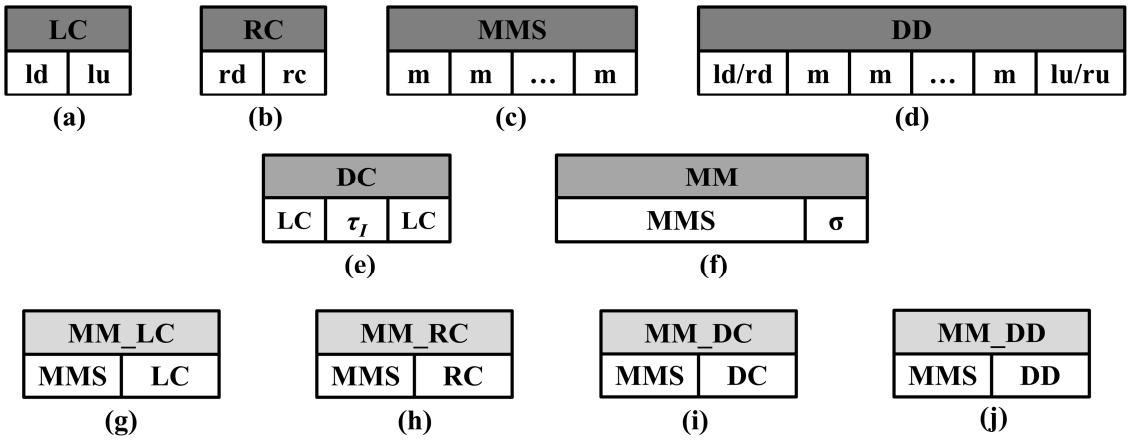


Figure 4: Schematic description of the various mouse actions: (a) Left click. (b) Right click. (c) Mouse-move sequence. (d) Drag-and-drop action. (e) Double click. (f) Mouse-move. (g) Mouse-move followed by a left click. (h) Mouse-move followed by a right click. (i) Mouse-move followed by a double click. (j) Mouse-move followed by a drag-and-drop.

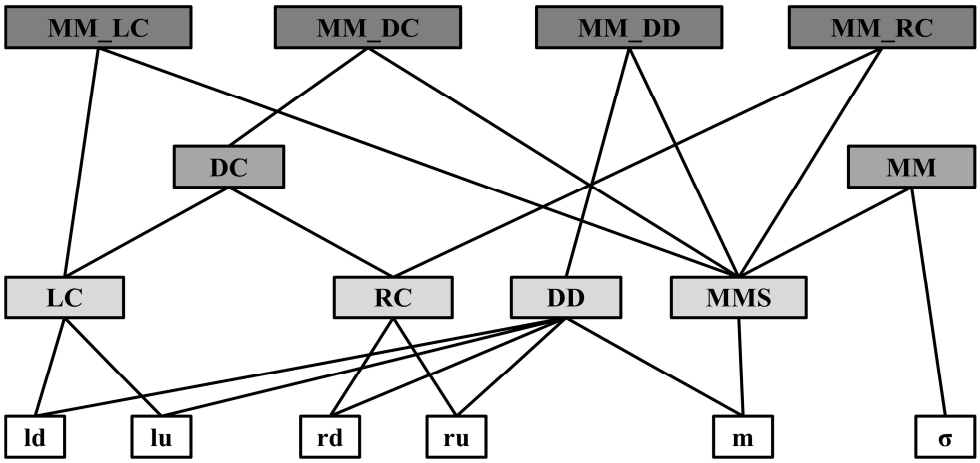


Figure 5: The hierarchy of mouse actions that are used to characterize the mouse activity.3.2 Actions features

All actions, except for LC, RC and DC, contain one or more sequences of mouse-move events together with lower level actions. In the following we describe the features that we use in order to characterize mouse movement. We then describe the features that we associate with each mouse action.

3.2.1 Movement Features (MF)

We adopt a similar approach to the one proposed by Gamboa et al [20] in order to describe a mouse movement action. Formally, each mouse movement is associated with the following three vectors:

$\mathbf{t} = \{t_i\}_{i=1}^{n}$ - The sampling time

$\mathbf{x} = \{x_i\}_{i=1}^{n}$ - The horizontal coordinate sampled at time t_i.

$\mathbf{y} = \{y_i\}_{i=1}^{n}$ - The vertical coordinate sampled at time t_i.

The length of the path produced by the sequence of points until the *i*-th point is defined as:

$$S_i = \sum_{k=1}^{i} \sqrt{\delta x_k^2 + \delta y_k^2}$$

where $\delta x_i = x_{i+1} - x_i$ and $\delta y_i = y_{i+1} - y_i$.

A set of basic features, which are described in Table 3, was extracted in [20] from the vectors $\mathbf{x}, \mathbf{y}$ and $\mathbf{t}$.

	Feature name	**Description**	**Formal definition**
1	Angle of movement	Angle of the path tangent with the x-axis	$\theta_i = \arctan^*\left(\frac{\delta y_1}{\delta x_1}\right) + \sum_{j=1}^{i} \delta\theta_j$ $\delta\theta_i = \min\left\{\delta \arctan^*\left(\frac{\delta y_i}{\delta x_i}\right) + 2k\pi\right\}$
2	Curvature	The relative angle change to the traveled distance	$c = \frac{\delta\theta}{\delta s}$
3	Curvature change rate		$\Delta c = \frac{\delta c}{\delta s}$
4	Horizontal Velocity	Velocity with respect to the x-axis	$V_x = \frac{\delta x}{\delta t}$
5	Vertical Velocity	Velocity with respect to the y-axis	$V_y = \frac{\delta y}{\delta t}$
6	Velocity		$V = \sqrt{\delta V_{x_x}^2 + \delta V_{y_x}^2}$
7	Acceleration		$\dot{V} = \frac{\delta V}{\delta t}$
8	Jerk		$\ddot{V} = \frac{\delta \dot{V}}{\delta t}$
9	Angular Velocity		$w = \frac{\delta\theta_t}{\delta t}$

Table 3: Basic mouse movement features which were proposed in [20] and are used by the proposed approach in this paper.

Based on the features in Table 3, Gamboa et al [20] construct a set of higher-level features. In order to calculate some of these features, the vectors $\mathbf{x}, \mathbf{y}$ are first interpolated and the interpolated results are denoted by $\mathbf{x}', \mathbf{y}'$, respectively. The result is used to obtain the interpolated traveled distance which is denoted by $\mathbf{s}'$.

A subset of the higher-level features proposed in [20] which is utilized by the algorithm proposed in this paper, is given in Table 4.

	Feature name	Description	Number of features	Formal definition
1	minimum, maximum, mean, standard deviation and (maximum-minimum)	The specified statistic of $x', y', \theta, c, \Delta c, V_x, V_y, V, \dot{V}, \ddot{V}$ and w	55	
2	Duration of movement		1	t_n
3	Traveled distance		1	S_{n-1}
4	Straightness(S)		1	$\frac{\sqrt{(x_1 - x_n)^2 + (y_1 - y_n)^2}}{S_{n-1}}$
5	Critical Points (CP)		1	$Critical\ \text{Points(CP)} = \sum_{i=1}^{n} z_i$ where $z_i = \left\{ \begin{matrix} 1 \text{ if } \Delta c_i = 0 \wedge \lvert c_i \rvert > \alpha \\ 0 \text{ otherwise} \end{matrix} \right\}$ for $\alpha > \frac{\pi}{10} \frac{rad}{pixel^2}$
6	Jitter(J)		1	$\frac{S'}{S_{n-1}}$

Table 4: Additional extracted features based on x',y',s' and the basic features.

We introduce a set of new features that are used in conjunction with the features in Table 4. These features include:

1. **Trajectory Center of Mass (TCM) –** a single feature that measures the average time for performing the movement where the weights are defined by the traveled distance:

$$TCM = \frac{1}{s_{n-1}} \sum_{i=1}^{n-1} t_{i+1} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}$$

2. **Scattering Coefficient (SC) –** measures the extent to which the movement deviates from the movement center of mass:

$$SC = \frac{1}{S_{n-1}} \sum_{i=1}^{n-1} t_{i+1}^2 \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} - TCM^2$$

3. **Third and Fourth Moment (M_3, M_4)** –

$$M_k = \frac{1}{S_{n-1}} \sum_{i=1}^{n-1} t_{i+1}{}^k \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \text{ where } k=3,4.$$

4. **Trajectory Curvature (TCrv) -** The average of the following quantity is taken over all the sampled points:

$$TCrv = \frac{\dot{x}\ddot{y} - \dot{y}\ddot{x}}{(\dot{x}^2 + \dot{y}^2)^{\frac{3}{2}}}$$

5. **Velocity Curvature (VCrv).** The average is taken as the feature.

$$VCrv = \frac{\ddot{v}}{(1+\dot{v}^2)^{\frac{3}{2}}}$$

Table summarizes the features which are used by the proposed algorithm in order to characterize mouse movement actions.

Factors	x'	y'	θ	c	Δc	$\overline{V_x}$	$\overline{V_y}$	$\overline{V}$	$\overline{\dot{V}}$	$\overline{\ddot{V}}$	w	t_n	S_{n-1}	S	CP	J	TCM	SC	M_k	TCrv	VCrv
Features	5	5	5	5	5	5	5	5	5	5	5	1	1	1	1	1	1	1	2	1	1

Table 5: 66 features used to represent a movement sequence.

3.2.2 Mouse action features

In order to describe the LC, RC, DC, DD, MM_LC, MM_RC and MM_DD mouse actions, additional features are extracted depending on the action type at hand. Table 6 provides a detailed description of the features that are used to characterize each of the actions.

Action	**Features**	**Number of features**
Left Click (LC)	• **Click Time (CT)** – The time between the mouse down event and the mouse up event, which must be less than τ_{LC}. • **Traveled Distance during Click (TDC)** – The distance traveled between the mouse down event and the mouse up event.	2
Right Click (RC)	• **Click Time (CT)** – The time between the mouse down event and the mouse up event which is less than τ_{RC}. • **Traveled Distance during Click (TDC)** – The distance traveled between the mouse down event and the mouse up event.	2
Drag and Drop (DD)	• The features of the movement between the mouse-down and mouse-up events which are summarized in Table 6.	66
Double Click (DC)	• **First Click Time (FCT)** – The time between the mouse- down and mouse-up events, which is less than τ_{LC}. • **First Click Distance (FCD)** – The distance traveled between the mouse-down and mouse-up events of the first click. • **Interval Time (IT)** – The time interval between the first click and the second one, which is less than τ_I. • **Interval Distance (ID)** – The distance traveled between the first click and the second one. • **Second Click Time (SCT)** – The time between the mouse-down and mouse-up events, which is less than τ_{LC}. • **Second Click Distance (SCD)** – The distance traveled between the mouse-down and mouse-up events of the second click.	6
Mouse Move and Left or Right Click Action (MM_LC)	• **Mouse movement features** from the beginning of the action until the mouse down event (Table 6). • **Time to click (TC)** – The time between the mouse-move event immediately preceding the mouse-down event and the mouse-down event itself. • **Distance to click (DC) –** The distance between the mouse-move event immediately preceding the mouse-down event and the mouse-down event itself. • **Click Time (CT)** – The time between the mouse-down and mouse-up events, which is less than τ_{LC}. • **Traveled Distance during Click (TDC)** – The distance traveled between the mouse-down and the mouse-up events.	70
Mouse Move and Double Click Action	• **Mouse movement features** from the beginning of the action until the mouse down event (Table 6). • **Time to click (TC)** – The time between the mouse-move event immediately preceding the	74

(MM_DC)	mouse-down event and the mouse-down event itself. • **Distance to click (DC) –** The distance between the mouse-move event immediately preceding the mouse-down event and the mouse-down event itself. • **First Click Time (FCT)** – The time between the mouse-down and the mouse-up events, which is less than τ_{LC}. • **First Click Distance (FCD)** – The distance traveled between the mouse-down and the mouse-up events of the first click. • **Interval Time (IT)** – The time interval between the first click and the second, which is less than τ_I. • **Second Click Time (SCT)** – The time between the mouse- down and the mouse-up events, which is less than τ_{LC}. • **Second Click Distance (SCD)** – The distance traveled between the mouse-down and the mouse-up events of the second click.	
Mouse Move and Drag and Drop Action (MM_DD)	• **Mouse movement features** from the beginning of the action until the mouse down event (Table 6). • **Time to click (TC)** – The time between the mouse-move event immediately preceding the mouse-down event and the mouse-down event itself. • **Distance to click (DC) –** The distance between the mouse-move event immediately preceding the mouse-down event and the mouse-down event itself. • **Mouse movement features** describing the movement between the mouse-down and mouse-up events of the drag-and-drop action (Table 6).	134

Table 6: Features of the mouse actions that are used to describe the mouse activity.

3.3 The Proposed Verification Framework

The framework is divided into 3 parts: (a) Acquisition, (b) Learning, and (c) Verification. A detailed description of these parts is given in the next sections.

3.3.1 Acquisition

The acquisition part captures the mouse events that constitute the users' mouse activity and is illustrated in Figure 6. This part is composed of three modules and an *Actions database*:

- **A feature acquisition module** - responsible for acquiring the events that are produced by the mouse. Each event is described as a quartet *<event type, x coordinate, y coordinate, timestamp>*. For example, the quartet *<MM,220,320,63355951016724>* represents a mouse-move event, at location X=220, Y=320 at time 63355951016724 milliseconds after the year 1970.

- **An action extractor module** - transforms the acquired events into the mouse actions defined in section 3.1. Each action is extracted and associated with its events in order to facilitate the extraction of the different features proposed in Section 3.2.

- **A feature extractor module** - derives features from the given action. It is illustrated by multiple instances in Fig. 7 since different feature extractors are required for

different types of actions. The extracted features are summarized in Table 7**Error! Reference source not found.**.

- **An actions DB** - stores the actions and their associated features of each user. This information is used to construct the profiles of each user in the Learning process.

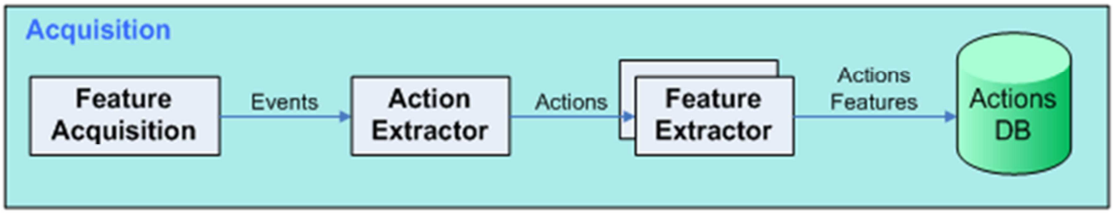


Figure 6: The acquisition process of mouse activity.

3.3.2 Learning

In this part, classifiers are constructed *for each action type*. Training sets in the form of matrices are constructed using the actions of the users that are stored in the actions DB. Each matrix holds the features that belong to a specific action type. Specifically, each action instance forms a row whose columns contain the features that are associated with the action and its label is given by the id of the user who performed the action.

A classifier is trained using the rows of one matrix and the produced model is stored in a database (one model for each action type).

We use the Random Forest [25] classifier which is a multi-class classifier, constructed from an ensemble of decision trees. Given a training set consisting of *N* instances, bootstrap samples of size *N* are drawn from it. Each sample is used to construct a decision tree. The classification of a pattern is obtained by a majority voting scheme applied to the results of the constructed trees. Figure illustrates the training process.

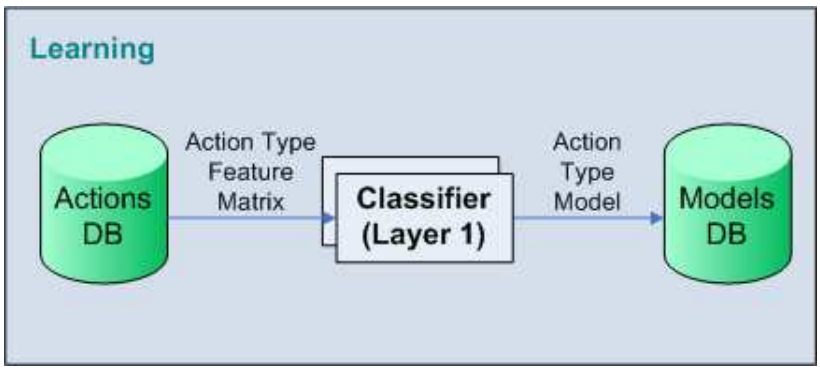


Figure 7: The training process for each of the action types.

3.3.3 Verification

The verification process is composed of the following steps:

1. Features are extracted from the acquired actions via a process that is similar to the one employed by the acquisition part.
2. The extracted features are stored in an *Action Collector DB.*
3. Once a sufficient number of (consecutive) actions are collected (according to a predefined threshold *m*) they are sent to the appropriate classifier according to the action type.
4. The Classifier (Layer 1) predicts for each of the trained users, the probability that each of them performed each of the *m* actions.
5. A layer 2 decision module combines the probabilities to derive a final result.

The process and its components are illustrated in Fig. 8.

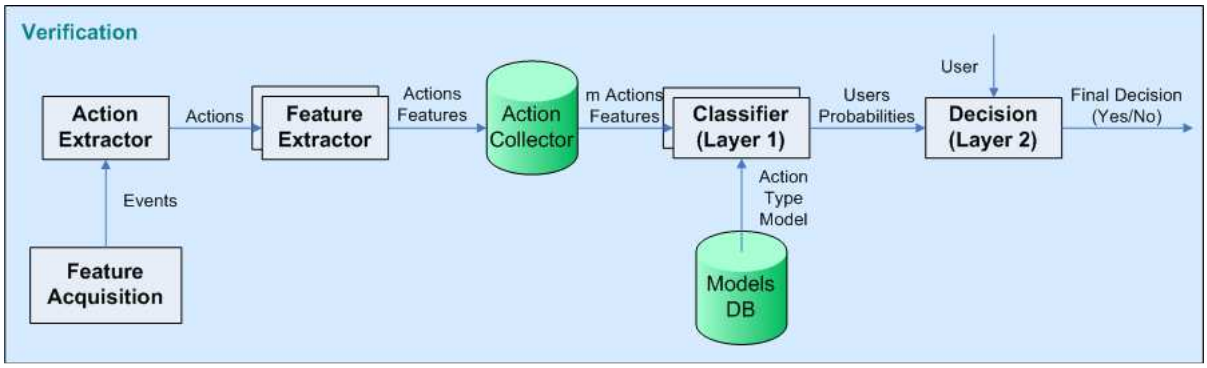


Figure 8: User verification process.

In the following, we give a formal description of the layer 1 classifier and the layer 2 decision module.

Classifier (Layer 1)

As previously mentioned, the classifier used to construct the model for each action type is the Random Forest [25]. Each of the actions collected by the Action Collector is passed to the appropriate classifier according to the type of action. Let $U = \{u_1, \dots, u_n\}$ be the set of trained users and let $A = \{a_1, \dots, a_m\}$ be a set of performed actions.

Each classifier (associated with action a_j) estimates for each trained user u_i the probability he performed action a_j. This probability is denoted by $\hat{P}(u_i|a_j)$.

Let $T_{ik} = \{t_{ik}^1, t_{ik}^2, \dots, t_{ik}^{m_{ik}}\}$ be the set of m_{ik} training instances of action type a_k performed by user i. In many cases m_{ik} may vary between the users for each type of action. This may result in a biased decision by the classifier. In order to overcome this problem, normalization is applied to the probabilities. Specifically, the probability $P^{post}(u_i|a_j)$ that an action a_j was performed by user u_i is given by:

$$P^{post}(u_i|a_j) = \frac{P^{norm}(u_i|a_j)}{\sum_{t=1}^{n} P^{norm}(u_t|a_j)}$$

where

$$P^{norm}(u_i|a_j) = \frac{\frac{\hat{P}(u_i|a_j)}{n \cdot P^{apr}(u_i|a_j)}}{\sum_{t=1}^{n} \frac{\hat{P}(u_t|a_j)}{n \cdot P^{apr}(u_t|a_j)}} = \frac{\hat{P}(u_i|a_j)}{P^{apr}(u_i|a_j)} \sum_{t=1}^{n} \frac{P^{apr}(u_t|a_j)}{\hat{P}(u_t|a_j)}$$

and $P^{apr}(u_i|a_j)$ denotes the a-priori probability derived by the training step.

Decision (Layer 2)

The decision module provides a final decision regarding the performed actions. It combines the probabilities given by the layer-1 classifiers and produces a final probability $P^{post}(u_i|a_1, \dots, a_m)$.

The probability that the set of actions $\{a_1, \dots, a_m\}$ belongs to user u_i is given by the following formula[1]:

$$P^{post}(u_i|a_1, \dots, a_m) = \frac{\sum_{j=1}^{m} P^{post}(u_i|a_j)}{\sum_{i=1}^{n} \sum_{j=1}^{m} P^{post}(u_i|a_j)}$$

The set of actions $a_1, \dots, a_m$ is associated to user u_i if the resulting probability is above a threshold λ i.e.

$$Final\ Decision(\{a_1, \dots, a_m\} \in u_i) = \begin{cases} Yes & P^{post}(u_i|a_1, \dots, a_m) \geq \lambda \\ No & Otherwise \end{cases}$$

[1] Probability multiplication equivalent to Naïve Bayes with Bayes formula was also tested, however due to poor results the experiments were performed using probability summation.

4 EXPERIMENTAL RESULTS

In order to evaluate the proposed approach, we first collected an extensive and diverse data from a wide variety of users and computer configurations. Given the data, the proposed approach was evaluated by performing the following experiments:

1. Comparison between the proposed *action-based multi-class* approach to the *histogram-based binary-class* approach proposed by Ahmed et al [2].
2. Comparison between the proposed multi-class verification and a binary-class model utilizing the proposed approach in order to examine the effectiveness of using a multi-class model.
3. We tested the contribution of the new features introduced in Section 3.2 to the verification accuracy.

4.1 Data Collection

The feature acquisition described in section **Error! Reference source not found.** was performed in 25 computers which were used by 21 males and 4 females. The computers were chosen from a wide variety of brands and hardware configurations. Specifically, the computers included 13 desktops, 12 laptops. The CPU speeds ranged from 1.86Ghz to 3.2Ghz and the pointing devices included optical mice, touch pads and styli.

4.1.1 User groups definition

In general, different users may interact with one or more computer system. These users may be associated with the institution or company to which the computer systems belong or alternatively, they may be external. Accordingly, the following two groups of users were defined:

(a) **Internal Users –** correspond to users that belong to the institution or company.

(b) **External Users –** users that are external to the institution or company.

One or more internal users may be authorized to interact with a particular computer system while the rest of the users (internal and external) are not. We refer to the former interaction type as an *authorized interaction*. It is assumed that the number of authorized interactions performed by an internal user is higher than the number of unauthorized ones since most of the time the legal users interact with their computer systems. Moreover, the

number of unauthorized interactions by external users is even smaller since they are not supposed to have access to any of the computers within the company. This assumption is manifested by the number of legal verification attempts, internal attacks and external attacks that are chosen in the evaluation.

4.1.2 Experiment configuration

The thresholds $\tau_{MM}, \tau_{MLM}, \tau_{MRM}, \tau_{LC}, \tau_{RC}, \tau_{I}$ that were used in order to construct the actions defined in section **Error! Reference source not found.** were empirically set to *500* milliseconds. The action extraction incorporated filtration similarly to the one used in [20]. Namely, calculation of the movement features associated with the different actions such as speed, acceleration and jerk, was only done if a minimal amount of events was at hand. Only movements that contained at least 4 different points were considered. Events whose type and position were equal to those of the event which preceded them were ignored.

Two-fold cross validation was used in the experiments i.e. the data collected for each of the users was split into 2 equal partitions: training and testing. The profile of each user was constructed from the training partition and the testing partition was used to generate legal verifications and illegal attacks. On the average, the training set consisted of 15.494 hours of activity per user and the average action duration was approximately 1.4 seconds.

The set of all available users $U = \{u_1, \dots, u_n\}$ was randomly divided in each fold into a set of k internal users $IU = \{iu_{j_1}, \dots, iu_{j_k} | iu_{j_l} \in U, 1 \leq j_l \leq n, l = 1, \dots, k\}$ and a set of external users $EU = U - IU$. Profiles were constructed for each of the internal users in *IU* according to the training activity that belonged to all users in *IU*. Each of the users in *IU* was tested for authorized and unauthorized access based on a varying number of consecutive actions. In each of the experiments the number of internal users was set to $|IU| = 12$ and the number of actions varied between 1 and 100 actions. All the experiments were conducted using the same testing instances to allow credible comparisons.

Attacks by internal and external users were simulated and are referred to as internal and external attacks, respectively. An internal attack was simulated by changing the user id of an activity that belongs to an internal user to an id of another internal user. An external attack was simulated by associating actions of an external user with an id of an internal

user. Specifically, 24 internal attacks were simulated for each user in each of the two folds, producing 48 internal attacks per user and a total of 48 * 25 = 1200 internal attacks. Six external attacks were simulated for each user in each of the two folds, producing 12 external attacks per user and a total of 12 * 25 = 300 external attacks.

In addition to the attacks, 72 authorized interactions were checked for each user in each of the two folds, simulating a legitimate user working on a computer system. This produced 144 legal verification attempts per user and 144 * 25 = 3600 verification attempts in total.

The training and testing were performed on computer with 16GB RAM and an Intel(R) Xeon(R) CPU running at 2.5Ghz which achieved all the execution times that are specified below.

4.2 Evaluation measures

Since biometric-based verification systems are a special case of classifiers [1], their performance is evaluated using similar measurements. Specifically, the following measurements were used:

- **False Acceptance Rate (FAR)** – measures the ratio between the number of attacks that were erroneously labeled as authentic interactions and the total number of attacks.
- **False Rejection Rate (FRR) –** measures the ratio between the number of legitimate interactions that were erroneously labeled as attacks and the total number of legitimate interactions.
- **ROC Curve** – An ROC curve is a graphical representation of the tradeoff between the FAR and the FRR for every threshold [4][5]. The point (0,0) represents perfect verification while the point (1, 1) represents wrong verification for every instance.
- **Area Under Curve (AUC)** – measures the area under the ROC curve. A lower AUC is sought after since it corresponds to better performance.
- **Equal Error Rate (EER)** – The rate at which both acceptance and rejection error rates are equal.

Based on the above measurements, additional measurements were defined.

- The *INTERNAL_FAR* was attained from the attacks performed by internal users.
- The *EXTERNAL_FAR* was derived from the attacks performed by external users.

4.3 Comparison with a histogram-based approach

The approach introduced in [2] uses histograms in order to *aggregate* multiple actions and utilizes a *binary model* in order to represent each user. The first experiment compares this approach with the two layer approach proposed in this work. In order to construct histograms from the features that are used to characterize the mouse actions (Section 3), discretization is first employed to continuous features. Specifically, one of the following methods was applied to each feature:

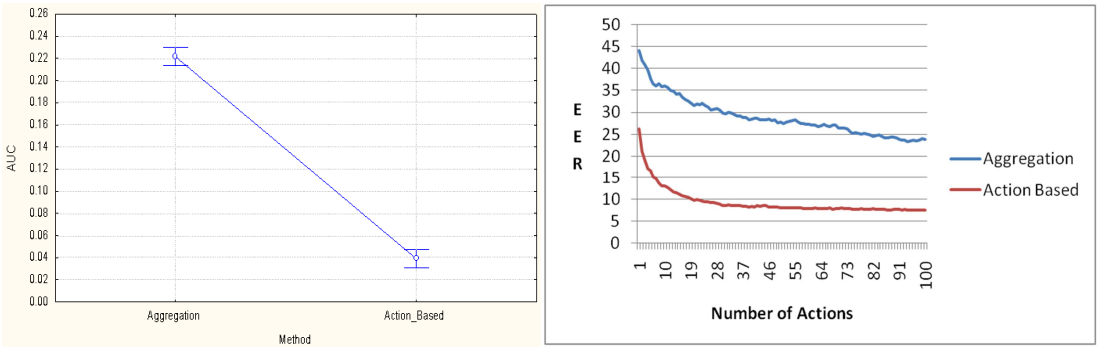
1. **Distance discretization** – In most cases, during click/double click no distance is traveled. Thus, in this case discretization was performed via two *binary* features. The first is set to 1 if no distance was traveled; otherwise the second feature is set to 1. This discretization was applied to the DC, FCD, ID, SCD and TDC features.
2. **Critical Points discretization** – The values observed for the CP feature were 0, 1, 2 and 3. Therefore, the discretization produced *five* binary features. A critical point value of 0 would set the first feature to 1 and the rest to 0, a critical point value of 1 would set the second feature to 1 and the rest to zero and so on. The last feature would be set to 1 if the number of critical points is greater than 3. This discretization was applied to the CP feature.
3. **Equal Frequency (EQF)** – The values of each feature were separated into 5 equally-spaced intervals. This discretization was applied to the remaining features.

The discretisized features were used by both the proposed approach and the histogram-based one. By performing aggregation of the discretisized features of each action, occurrence histograms as in [1] were created. The feature average histograms were created by averaging the remaining features. The features that were used were described in Table 6.

A verification attempt based on *N* actions was performed in the following manner: Each of the eight types of actions was extracted from the *N* actions and was individually aggregated. The aggregated values were concatenated to form a feature vector that characterizes the user's activity. In addition, the relative occurrence of each action was added to the feature vector.

In order to train the model, the training set data was split into 5 equal partitions and each training partition was used to produce a single aggregated vector. Thus, each user was represented by 5 vectors.

Error! Reference source not found.(a)-(b) present the comparison results between the aggregation and the action-based approaches. **Error! Reference source not found.**(a) depicts the comparison between the two methods in terms of the AUC measure incorporating the ANOVA test with 95% confidence intervals. It is evident that the action-based method outperforms the histogram-based approach.



(a) (b)

Figure 9: Comparison between the action-based method and the histogram-based approach. (a) AUC measure comparison: the action-based method clearly outperforms the histogram-based one. (b) EER evaluation of the proposed method: the action-based method is superior for any number of actions and produces EER of 8.53% and 7.5% for 30 and 100 actions, respectively, while the histogram-based method produced EER of 29.78% and 23.77%. There is a sharp decrease in the EER in the action-based method until 30 actions are performed which becomes more moderate for a number of actions higher than 30.

Error! Reference source not found. shows the EER of the two methods for different quantities of actions. The action-based method is superior for any quantity of actions. Furthermore, a sharp decrease in the EER is observed in the action-based method when the number of actions that is used for verification ranges from 1 (26.25% EER) to 30 actions (~8.53% EER). When the number of actions is between 30 and 100, the decrease becomes more moderate and for 100 actions the EER is equal to 7.5%. The aggregation approach produces 29.78% and 23.77% EER for 30 and 100 actions, respectively.

As mentioned above, the average duration of an action was less than 1.4 seconds. The construction of the verification vector and testing time per action was approximately 3*ms*. Thus, the required time for verification based on 30 and 100 actions is approximately 42 $seconds$ and 2.33 $minutes$, respectively. Consequently, the approach proposed in this

paper provides a method for verifying the user in less than 2 minutes with a maximal equal error rate of 10%.

Error! Reference source not found. presents an ROC curve obtained from verification based on 30 actions. The optimal point on the ROC curve in which the acceptance and rejection errors are equal is obtained for an internal EER of 8.53% and a relatively high external FAR of 17.66%. The choice of the optimal point may be altered according to security level that is sought after. For instance, a point where the FAR is low and the FRR is high suits users that have highly confidential information on their computer system while a point with relatively low FRR and higher FAR may reduce the rate false alarms of legitimate access.

It should be mentioned that while in [1] a set of actions performed within a session produced a single instance in the training and test sets, in our proposed method, every action produces an instance. Consequently, the number of instances is higher and thus requires a larger amount of memory. Nevertheless, this requirement only affects the training phase.

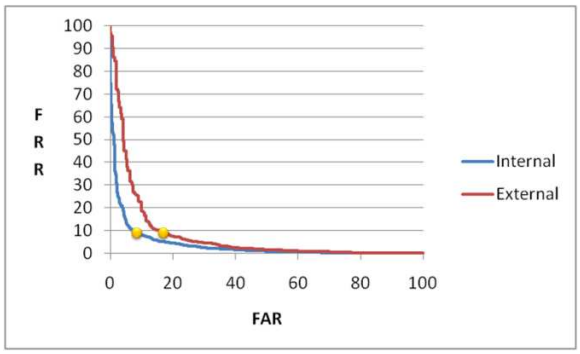
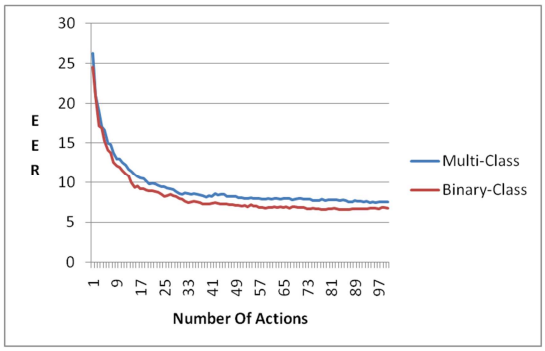


Figure 10: ROC curve for verification based on 30 actions. An internal EER of 8.53% corresponds to an external FAR of approximately 17.66%.

4.4 Comparison between binary and multi-class models

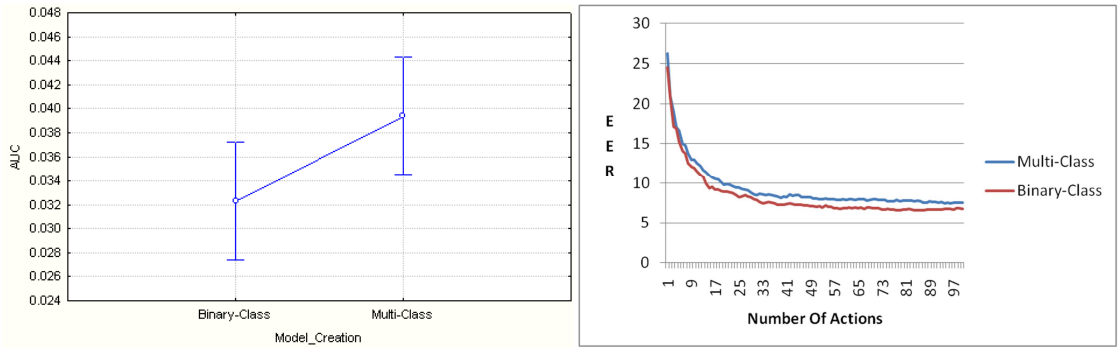
The purpose of the second experiment was to determine whether modeling users by a multi-class approach is superior to modeling the users by binary class models. In the latter, a binary model was constructed for every action and user pair in the training set in order to derive the probability $\hat{P}(u_i|a_j)$.



(b)

Figure 11: Comparison between the binary-class models and the multi-class model approaches. (a) The binary-class approach outperforms the multi-class in terms of AUC with statistical significance. (b) The binary-class approach is superior to the multi-class approach in terms of EER for almost any number of actions between 1 and 100.

(a) presents a comparison between the two modeling approaches in terms of the AUC using the ANOVA test with 95% significance intervals. Results show statistically significant superiority of the binary modeling approach over the multi-class modeling approach. Figure 11: Comparison between the binary-class models and the multi-class model approaches. (a) The binary-class approach outperforms the multi-class in terms of AUC with statistical significance. compares between the equal error rates of the multi-class and binary-class approaches for a number of actions ranging from 1 to 100. The binary approach outperforms the multi-class approach in terms of EER by 1.01% on the average for almost every number of actions.



(a) (b)

Figure 11: Comparison between the binary-class models and the multi-class model approaches. (a) The binary-class approach outperforms the multi-class in terms of AUC with statistical significance. (b) The binary-class approach is superior to the multi-class approach in terms of EER for almost any number of actions between 1 and 100.

A major drawback of the binary class modeling approach is its time and space complexities which are approximately $|U|$ times greater than those of the multi-class model approach where $|U|$ denotes the number of users which take part in the training. Specifically, $|U|$ binary models are constructed for every action instead of a single multi-class model. For example, training each multi class model required $8.1896\ minutes$ on the average while testing required $2.7746ms$. However, since training in the binary-model approach requires the construction of an individual binary model for every user, the training time took $7.031\ minutes * 12\ (users) = 84.372$ minutes and the testing time took $2.7135ms * 12\ (users) = 32.562ms$.

Thus, although the binary-class approach exhibits statistically significant performance superiority over the multi-class approach, considering the time and space complexities that are required for training and testing may render it as unsuitable in time-critical settings. Consequently, choosing one of the approaches depends on the verification time and accuracy which is required. The multi-class approach is suitable when relatively fast verification (at the expense of lower accuracy) is required while the binary class provides a better choice in cases when higher accuracy is required at the expense of slower verification.

4.5 Contribution of the new features

The proposed approach introduces new features to characterize mouse activity. These features are used in conjunction with features that were adopted from [20]. In order to determine the contribution of the newly introduced features two experiments were conducted: the first verified users based only on the features that were adopted from [20] and the second experiment used the new features together with the ones from [20]. Figure (a) and 12(b) present a comparison between the results of the two experiments in terms of the AUC. It is evident that the new features contribute to the accuracy of the model. Figure 12(a) shows that using the additional new features achieves a better result for any number of actions that are used for the verification and the ANOVA test using 95% confidence intervals achieves similar findings which are illustrated in Figure 12(b).

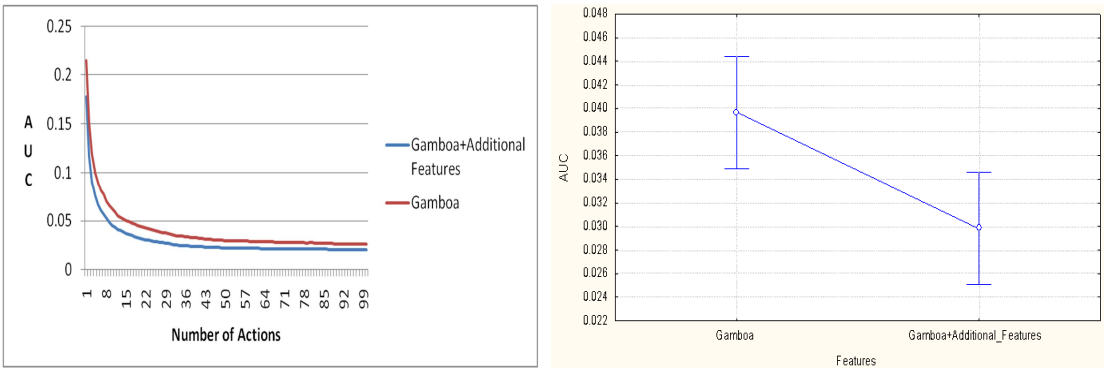


Figure 12: Contribution of the new additional features that were introduced in Section 3.2. (a) The additional features contribute to the verification accuracy for any number of actions ranging from 1 to 100. (b) Contribution in terms of the ANOVA test using 95% confidence intervals.

5 Conclusions and future work

A novel method for user verification based on mouse activity was introduced in this paper. Common mouse events performed in a GUI environment by the user were collected and a hierarchy of mouse actions was defined based on the raw events. In order to characterize each action, features were extracted. New features were introduced in addition to features that were adopted from [20]. A two-layer verification system was proposed. The system employs a multi-class classifier in its first layer and a decision module in the second one in order to verify the identity of a user.

The proposed method was evaluated using a dataset that was collected from a variety of users and hardware configurations. Results showed superiority of the action-based method proposed in this paper over the histogram-based method proposed in [1]. Furthermore, evaluation showed a significant improvement in the verification accuracy when using the newly introduced features.

In the following we describe several issues that need further investigation in mouse-based verification methods.

The original actions intended by the user are logged neither by software nor by observing the user while performing the actions. Accordingly, they are heuristically reconstructed from the raw events which may produce some non-credible actions. Additionally, the obtained actions may vary between different hardware configurations (e.g. optical mouse, touch pad). In order to obtain a higher percentage of credible actions, the parameters that define them should be determined by a more rigorous method.

Furthermore, the data collected from mouse devices may be partially unreliable due to noise. Specifically, lint clogging the moving parts of mechanical mice may affect the functionality of the mouse. However, this type of mice is becoming rare. Optical mice may introduce noise due to their inability to track movement on glossy or transparent surfaces. In some mice, fast movements may be poorly captured.

A significant drawback of mouse-based verification in comparison to keyboard-based verification is the variety of mice, mouse pads and software configurations which may influence the performance of the verification. For example, a person using a laptop in two different places may use the touch pad in one place and an external mouse in the other - thus affecting the events produced and, consequently, the performance of any mouse-based verification method. This problem does not exist in keyboard-based verification techniques since the keyboard is an integral part of the laptop.

In order to establish well structured research and evaluation of methods in the area of behavioral biometric systems, benchmark data sets must be available. In their absence, it is impossible to compare the existing methods (since each uses a different dataset, having unique characteristics). Moreover, each study has to start by putting new efforts in the construction of new datasets. Generally, there are two types of datasets: (a) General activities of a user in an operating system of a local computer, in which all the events are hooked at the operating system level; or (b) Activities generated from interaction with a web application, in which all the events that are related to the web browser are monitored at the client and sent to the server. The technological aspect of such collection tools is not an issue, but rather the ways to collect large-scale authentic data, in which many users perform their daily tasks. The problem here is mainly to convince users to expose their biometric data and to put the time and the efforts for the data collection.

Creating a dataset for *continuous verification* is more challenging, since the dataset should be diverse and reflect the daily tasks of the users. Furthermore, the dataset should reflect the different physiological states of the user during the day which might influence their behavioral biometrics and consequently the verification accuracy. For example, some users are faster in the morning, while slower at night, or after lunch. Moreover, user postures, such as sitting (common), standing or talking on the phone while interacting with the computer, are expected to influence the verification accuracy as well.

References

[1]. P. Grother, and E. Tabassi, "Performance of Biometric Sample Quality Measures", *National Institute of Standards and Technology*, 2006.

[2]. A. A. E. Ahmed, and I. Traore, "A New Biometric Technology Based on Mouse Dynamics", *IEEE Transactions on Dependable and Secure Computing*, Vol. 4, No. 3, pp. 165–179, July-September 2007.

[3]. F. Bergadano, D. Gunetti, and C. Picardi, "User Authentication through Keystroke Dynamics", ACM Transacions on Information and System Security, Vol. 5, no. 4, pp. 367–397, 2002.

[4]. M. H. Zweig, and G. Campbell, "Receiver-operating characteristic (roc) plots: A fundamental evaluation tool in clinical medicine". Clin. Chem., Vol. 39, no. 4, pp. 561–577, 1993.

[5]. T. Fawcett, "An Introduction to ROC Analysis", Pattern Recognition, Letters, , doi:10.1016/j.patrec.2005.10.010, 2006.

[6]. R. V. Yampolskiy, "Human Computer Interaction Based Intrusion Detection", *IEEE Computer Society, In Proc. of the International Conference on Information Technology*, pp. 837–842, 2007.

[7]. S. J. Stolfo, S. Hershkop, K. Wang, O. Nimeskern, and C. W. Hu, "A behavior-based approach to Securing Email Systems", *Mathematical Methods, Models and Architectures for Computer Networks Security*, Springer Verlag, 2003.

[8]. S. J.Stolfo, C. W. Hu, W. J. Li, S. Hershkop, K. Wang, and O. Nimeskern, "Combining Behavior Models to Secure Email Systems", *Technical Report*, Columbia University, Available at: *www1.cs.columbia.edu/ids/publications/EMT-weijen.pdf*, 2003.

[9]. O. D. Vel, A. Anderson, M. Corney, and G. Mohay, "Mining Email Content for Author Identification Forensics", *SIGMOD: Special Section on Data Mining for Intrusion Detection and Threat Analysis*, 2001.

[10]. G. Frantzeskou, S. Gritzalis, and S. MacDonell, "Source Code Authorship Analysis for Supporting the Cybercrime Investigation Process", *1st International Conference on eBusiness and Telecommunication Networks – Security and Reliability in Information Systems and Networks Track*, pp. 85–92, 2004.

[11]. A. Gray, P. Sallis, and S. Macdonell, "Software Forensics: Extending Authorship Analysis Techniques to Computer Programs", *In Proc. 3rd Biannual Conf. Int. Assoc. of Forensic Linguists (IAFL'97)*, 1997.

[12]. E. H. Spafford, and S. A. Weeber, "Software Forensics: Can We Track Code to its Authors?" *15th National Computer Security Conference*, pp. 641–650, 1992.

[13]. J. Ramon, and N. Jacobs, "Opponent modeling by analyzing play", *Proceedings of the Computers and Games workshop on Agents in Computer Games*, 2002.

[14]. A. R. Jansen, D. L. Dowe, and G.E. Farr, "Inductive Inference of Chess Player Strategy", *Proceedings of the 6th Pacific Rim International Conference on Artificial Intelligence (PRICAI'2000)*, pp. 61–71, 2000.

[15]. A. Bromme, and S. Al-Zubi, "Multifactor Biometric Sketch Authentication", *Proceedings of the BIOSIG 2003*, pp. 81–90, 2003.

[16]. S. Al-Zubi, A. Bromme, and K. Tonnies, "Using an Active Shape Structural Model for Biometric Sktech Recognition", *In Proceedings of DAGM*, pp. 187–195, 2003.

[17]. M. Schonlau, W. DuMouchel, H. Ju, A. F. Karr, M. Theus, and Y. Vardi, "Computer Intrusion: Detecting Masquerades", *Statistical Science*, 16, pp 1–17, 2001.

[18]. R. A. Maxion and T. N. Townsend, "Masquerade detection using truncated command lines", In *International Conference on Dependable Systems and Networks (DNS-02)*, IEEE Computer Society Press, 2002.

[19]. M. Pusara, and C. E. Brodley, "User Re-Authentication via Mouse-movements", *Proc. of ACM Workshop Visualization and Data Mining for Computer Security*, 2004.

[20]. H. Gamboa, and A. Fred, "Behavioural Biometric System Based on Human Computer Interaction", *Proc. of SPIE*, vol. 5404, pp. 381–392, 2004.

[21]. S. Bleha, C. Slivinsky, and B. Hussein, "Computer-access security systems using keystroke dynamics", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 12, pp. 1217–1222, 1999.

[22]. S. Cho, C. Han, D.H. Han, and H.I. Kim. "Web-Based Keystroke Dynamics Identity Verification Using Neural Network", *Journal of Organizational Computing and Electronic Commerce*, 10(4):295–307, 2000.

[23]. M. Curtin, C. C. Tappert, M. Villani, G. Ngo, J. Simone, H. S. Fort, and S. Cha, "Keystroke Biometric Recognition on Long Text Input: A Feasibility Study", *Proc. Int. Workshop Sci Comp/Comp Stat*, 2006.

[24]. D. Gunetti, and C. Picardi, "Keystroke analysis of free text", *ACM Trans. Inf. Syst. Secur.*, 8(3):312–347, 2005.

[25]. L. Breiman, “Random Forests”, *Machine Learning*, 45(1), pp. 5–32, 2001.

[26]. A. K. Jain, S. Pankanti, S. Prabhakar, L. Hong and A. Ross, “Biometrics: A grand challenge”, *Proc. Int. Conf. on Pattern Recognition*, vol. 2, pp. 935–942, 2004.

[27]. R. B. Abernethy, “The New Weibull Handbook”, 2001.

[28]. R. V. Yampolskiy, V. Govindaraju, “Behavioral Biometrics: a Survey and Classification”, *Int. J. Biometrics*, Vol. 1, No. 1, 2008.

[29]. K. Revett, P. S. Magalhães, and H. D. Santos, “Developing a keystroke dynamics based agent using rough sets”, In *IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*. University of Technology of Compiègne, 2005.

[30]. D.–T. Lin, “Computer-access authentication with neural network based keystroke identity verification”, *Neural Networks*,1997.

[31]. E. Lau, X. Liu, C. Xiao, and X. Yu, “Enhanced User Authentication Through Keystroke Biometrics”, *Technical report*, *Massachusetts Institute of Technology*, 2004.