Information Sciences 201 (2012) 19–36

Contents lists available at SciVerse ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

User identity verification via mouse dynamics

Clint Feher [a,*], Yuval Elovici [a], Robert Moskovitch [a], Lior Rokach [b,1], Alon Schclar [c]

[a] Telekom Innovation Laboratories at Ben-Gurion University and Department of Information Systems Engineering, Ben-Gurion University, POB 653, Beer Sheva 84105, Israel
[b] College of Information Sciences and Technology, Pennsylvania State University, University Park, PA 16802, USA
[c] School of Computer Science, Academic College of Tel-Aviv, POB 8401, Tel Aviv 61083, Israel

ARTICLE INFO

Article history:
Received 12 May 2010
Received in revised form 16 February 2012
Accepted 29 February 2012
Available online 15 March 2012

Keywords:
Mouse dynamics
Behavioral biometrics
Security monitoring
Verification
Mouse
Pointing devices

ABSTRACT

Identity theft is a crime in which 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 a security layer in addition to the username and password by continuously validating 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 mouse.

The contribution of this work is threefold: first, user verification is derived based on the classification results of *each individual mouse action*, in contrast to methods which aggregate 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 Theft* – a crime in which hackers impersonate legitimate users in order to commit fraudulent activity. Hackers exploit such identities by stealing credentials or by using logged-on computers that are left unattended and unlocked. The unlawful attempts to intercept passwords are commonly known as *phishing* and many phishing methods use *social engineering*. For example, a common way to inject a *keylogger* – a spying software which records every keystroke including passwords – into a victim's computer uses an innocent-looking email such as a greeting card in which the user is required to follow a link. Once the link is followed, the keylogger is injected.

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*

* Corresponding author. Tel.: +972 8 642 8017; fax: +972 8 642 8016.
E-mail addresses: clint@bgu.ac.il (C. Feher), elovici@bgu.ac.il (Y. Elovici), robertmo@bgu.ac.il (R. Moskovitch), liorrk@bgu.ac.il (L. Rokach), alonschc@mta.ac.il (A. Schclar).
[1] On Leave from Ben-Gurion University of the Negev.

0020-0255/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.ins.2012.02.066

cloning – using the information of another person to assume his or hers identity in daily routine; 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 internal 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 *authentication* and *verification* techniques that are based on behavioral and physiological biometrics which are assumed to be relatively unique to each user, and harder to steal. Authentication is performed once during login while verification is performed *continuously* throughout the session. Verification methods typically take biometric measurements of the user at regular intervals while the user is logged-on and these measurements are compared with measurements that were collected beforehand. Common behavioral biometric features include characteristics of the interaction between the user and input devices such as the mouse and keyboard.

Physiological biometrics methods, on the other hand, use human features that are unique to each individual. Examples include: fingerprints, iris patterns, face [12], blinking patterns [48,49], lip movement [10], gait/stride [32], voice/speech [11,14], car driving style [15,16], signature/handwriting [4,28], to name a few.

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/or *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 [29,51], on the other hand, do not require special designated hardware since they use common devices 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 to produce results at a given accuracy. Furthermore, several challenges [29], which will be elaborated in Sections 2 and 5, still need to be overcome in order to make this approach fully operational. Consequently, behavioral biometrics was ignored to a certain extent for user verification in the past. In this paper we propose a novel user continuous verification technique based on behavioral biometrics of the user's mouse activity when she is performing her daily computer 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 [24] is essentially a pattern recognition system that acquires biometric data from an individual, extracts a feature set to establish 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 that is identical to the one used during the model construction.

2.1. General architecture

Fig. 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) via their drivers.
- *Feature extraction* – constructs a signature which characterizes the behavioral biometrics of the user.

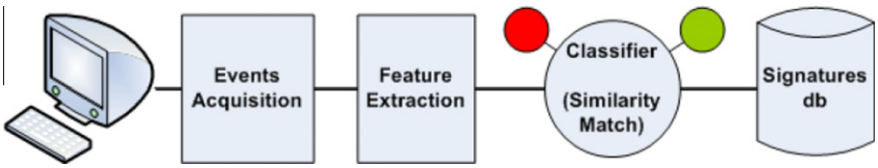

Fig. 1. A typical framework of a behavioral biometric identification system.

- *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.

2.2. Related work

According to [50], 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 rely on features extracted from the interaction of a user with a specific software tool.

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 [19,20]. *Implicit learning* techniques, on the other hand, monitor the user during general day-to-day computer activity rather than during the performance of a specific task – the method described in this paper falls into this category. Explicit and implicit methods are also referred to as *static and dynamic methods*, respectively in [31], where the former is mainly used for authentication while the latter is used for verification. Implicit learning is considered more challenging due to high inconsistency owed to the variety of the performed tasks, mood changes and other influential factors. Nevertheless, it is the best way to learn unique user behavior characteristics such as frequently performed actions.

Since biometric-based verification systems are a special case of classifiers [24], their performance is evaluated using similar measurements. Specifically, the following measurements are 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* – A ROC curve is a graphical representation of the tradeoff between the FAR and the FRR for various threshold values [17].
- *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. Low EER values indicate an accurate authentication system.

In the following, we list current available user verification systems along with their performance evaluations.

2.2.1. Mouse-based methods

Authentication methods identify users at login based on a predetermined sequence of mouse operations that the user needs to follows. During training, the sequence is repeated several times by every user. Features are extracted from each sequence and are used to characterize the user. During authentication, the user is required to follow the same sequence. Continuous verification, on the other hand, repeatedly reconfirms the user's identity throughout the entire session using all mouse activity rather than a predetermined sequence. In this paper we propose a continuous verification method. However, for the sake of completeness, we briefly describe a few authentication methods.

2.2.1.1. Explicit learning methods. Hashia et al. [27] used a sequence composed of pairs of points. Each user was required to move the mouse between the first and second point in each pair where features were extracted from each movement. The method was evaluated using 15 students and produced an EER of 15%.

The method proposed by Gamboa et al. [21] required the users to enter a username and a pin number using only the mouse via an on-screen virtual keyboard. Authentication combined the credentials and the mouse dynamics of their entry. The system was tested on 50 subjects producing an EER of 6.2% for 15-digit pin numbers.

Bours and Fullu [7] characterized the user by tracing the mouse activity while the user navigated through an on-screen maze. The features consisted of the velocity vectors that were extracted from each movement segment and the edit distance was used to measure the similarity between two feature vectors. Testing the technique on 28 subjects yielded an EER of approximately 27%.

Revett et al. [40] used a circular GUI interface which was designed as a combination lock. The users were given a combination which they entered by manipulating the GUI. The users were authenticated according to the timing characteristics of the GUI manipulation. The system was tested on six subjects and produced FAR and FRR of approximately 3.5% and 4%, respectively.

2.2.1.2. Implicit learning methods. Pusara and Bordley [37] proposed a user verification scheme based on mouse movements while participants browsed a set of web pages. Features such as the mean, standard deviation, third moment of distance,

angle and speed were extracted from a sequence of N events. The C5.0 decision tree algorithm [38] was used to classify the users. Three main evaluations were performed: the first checked the difference of behavior between every pair of users. Results showed that a relatively large number of users could be discriminated from one another. In the second evaluation, the discrimination of each user from the set of the remaining users was tested. A binary model was created for each user. An FAR of 27.5% and FRR of 3.06% was achieved. The third evaluation was similar to the second but used only 11 users (out of the 17 who participated) and also applied a smoothing filter to the data. An FAR 0.43% and an FRR of 1.75% were reported. The detection time of the users varied and ranged between 1 min and 14.5 min due to the different parameters chosen for each user.

Schulz segmented mouse strokes into curves which were characterized by length, curvature and inflection-based features [43]. A user was characterized according to histograms that were constructed from the features of several curves. The Euclidean distance was used to measure differences between histograms. Evaluation of the system using mouse data from 72 users yielded an average EER of 24.3% when groups of 60 curves were used and EER of 11.2% for groups of 3600 curves.

Ahmed et al. [2] monitored the mouse activity of users while they performed their daily tasks within their own 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 was 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 was fitted into 8 equal size sectors of the circle – each covering 45° of the angle space as illustrated in Fig. 2.

Examples of collected actions are illustrated in Table 1.

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

1. *Traveled Distance Histogram (TDH)* – The distribution of the travelled distance for every action type where only the first two features (distances 0–100 and 100–200 pixels) were used to represent the user.
2. *Action Type Histogram (ATH)* – The relative frequency of the MM, DD and PC actions within a session.
3. *Movement Direction Histogram (MDH)* – The ratio of actions performed in each one of the eight directions. This feature is represented by 8 values.
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 was represented by 8 values.
5. *Average movement speed per Types of Actions (ATA)* – The average speed of performing the MM, DD and PC actions. This feature was represented by 3 features.
6. *Movement Speed compared to the travelled Distance (MSD)* – Approximation of the average traveling speed for a given traveling distance (derived via a Neural Network). This feature was represented by 12 values sampled from the curve.
7. *Movement elapsed-Time Histogram (MTH)* – The time distribution for performing an action. Represented by 3 features.

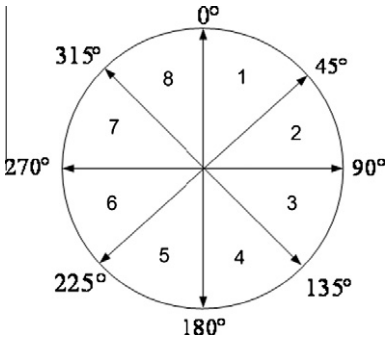

Fig. 2. Angle space of movement direction: 8 equal-sized sectors of the circle. Direction 2 represents angles between 45° and 90°. Direction 5 represents angles between 180° and 225°.

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

Type of action	Distance (s)	Time (s)	Direction
MM	50	1	3
PC	237	3	4
PC	80	2	2
Silence	–	2	–

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

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

The histograms were used to construct a feature vector composed of 39 features which characterize each session of every user. Table 2 summaries the extracted features.

A binary neural network model was constructed for every user based on the feature vectors drawn from the different histograms. Training consisted of 5 sessions having a total length of 13.55 min. The method was evaluated using the personal computers of 22 users. This method achieved FAR of 2.46% and FRR of 2.46%. Shorter times (about 4 min) produced results of less than 24% FRR and 4.6% FAR. Thus, construction of accurate histograms requires a significant amount of mouse activity, monitored over a relatively long duration of time.

Gamboa and Fred [19,20] proposed to verify a user based on her interaction with a memory game. The user was required to identify matching tiles and was verified based on characteristics of the mouse strokes that were performed to reveal of the tiles. A mouse-stroke was defined as *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. Spatial, temporal and statistical features were extracted to characterize each mouse-stroke. Specifically, each mouse movement was 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 was defined as:

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

where $\delta x_k = x_{k+1} - x_k$, $\delta y_k = y_{k+1} - y_k$ and $S_1 = 0$.

The vectors $\mathbf{x}$, $\mathbf{y}$ were interpolated using a cubic spline producing $\mathbf{x}'$, $\mathbf{y}'$, respectively. The result was used to obtain the interpolated traveled distance $\mathbf{s}'$. Given the original and interpolated vectors, a set of features, which is described in Table 3, was extracted. These features along with $\mathbf{x}'$ and $\mathbf{y}'$ were statistically analyzed and the minimum, maximum, mean, standard deviation and (maximum–minimum) difference were calculated for each one of them. A set of global features which is based on the entire mouse movement was also calculated. This set along with the features described above are summarized in Table 4.[2] The algorithm proposed in this paper uses these features together with new features that are introduced in Section 3.1.

Greedy feature selection was employed to find the best subset of features for each user. The learning procedure employed maximum likelihood with various distributions such as the Weibull [1] and Parzan distributions. Evaluating the system on 50 users with a varying number of mouse strokes having an average duration of 1 s produced an Equal error rates (ERRs) of 0.7% and 0.2% for 100 and 200 mouse-strokes, respectively.

2.2.2. Other behavioral biometric 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 models according to feature vectors that are extracted while the users type *a predefined text* (such as a password) [5,6,10,34,41] while the latter extract feature vectors from free text that the users type [12,25]. In a recent paper

[2] Four additional features were introduced in [20]. However, they are not relevant to this paper.

Table 3
Basic mouse movement features which were proposed by Gamboa and Fred [20].

	Feature name	Description	Formal definition
1	Angle of movement	Angle of the path tangent with the x-axis	$\theta_i = \arctan^*\left(\frac{\delta y_1}{\delta x_1}\right) + \sum_{j=1}^{i} \delta\theta_j$ $\delta\theta_j = \delta\arctan^*\left(\frac{\delta y_j}{\delta x_j}\right)$ is the minimal angle change given in $[-\pi, \pi]$ between the jth and $(j+1)$th points
2	Curvature	The relative angle change to the traveled distance	$c = \delta\theta/\delta s$
3	Curvature change rate	The rate of the curvature change	$\Delta c = \delta c/\delta s$
4	Horizontal velocity	Velocity with respect to the x-axis	$v_x = \delta x/\delta t$
5	Vertical velocity	Velocity with respect to the y-axis	$v_y = \delta y/\delta t$
6	Velocity	First displacement moment	$v = \sqrt{v_x^2 + v_y^2}$
7	Acceleration	Second displacement moment	$\dot{v} = \delta v/\delta t$
8	Jerk	Third displacement moment	$\ddot{v} = \delta \dot{v}/\delta t$
9	Angular velocity	Angle change rate	$w = \delta\theta_t/\delta t$

Table 4
The set of features that were extracted by Gamboa and Fred [20].

	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;$S_1 = 0$
4	Straightness (S)		1	$\frac{\sqrt{(x_1 - x_n)^2 + (y_1 - y_n)^2}}{S_n}$
5	Critical points (CP)		1	$\sum_{i=1}^{n} z_i$ where $z_i = \begin{cases} 1, & \Delta c_i = 0 \text{ and } \lvert c_i \rvert > \alpha \\ 0, & \text{otherwise} \end{cases}$ for $\alpha > \frac{\pi}{10}\frac{rad}{pixel^2}$
6	Jitter (J)		1	S'/S_n

Stefan et al. [13] evaluated the security of keystroke-dynamics authentication against synthetic forgery attacks. The results showed that keystroke dynamics are robust against the two specific types of synthetic forgery attacks that were used. 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 have been suggested in the literature to characterize behavioral biometrics of users for authentication and verification purposes. These include board games [30,39], memory games [19,20], web browsers [37], email clients [45–47], programming development tools [18,23,44], command line shells [36,42] and drawing applications [3,9]. These biometric features may be partially incorporated in user verification systems.

Recently, due to the limitations of user authentication systems that employ a single user characteristic such as mouse dynamics or iris patterns, a multi-modal approach has been proposed in various papers. De Marsico et al. [35] proposed a unified single-response framework for combining various biometric features. They demonstrated its effectiveness using face, ear, and fingerprint as test biometries. Kumar and Shekhar [33] proposed a nonlinear rank-level fusion approach for multi-biometrics fusion. They demonstrated their approach using palm print representations and showed a significant accuracy improvement when multiple representations are used compared to using individual palm print representations. Hamdy and Traoré [26] proposed a novel biometric system for static user authentication that homogeneously combines mouse dynamics, visual search capability and short-term memory effect. The mouse was used for its dynamics, and as an input sensor for the other two biometrics features – thus, alleviating the need for hardware other than the standard mouse.

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 histogram-based methods as in [2] which require the aggregation of dozens of mouse 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 such as mouse-down, while features at higher levels are composed of lower level ones. Higher level features incorporate dependencies between lower-level ones which help to characterize more accurately every user. For example, a high-level feature which is composed of mouse-move followed by a double-click incorporates the time between these actions – a feature that cannot be conveyed by neither of the individual actions.

The verification algorithm constructs a classifier using vectors composed of high level features, whose details will be described below.

3.1. A hierarchy of mouse actions

All mouse activities can be decomposed into 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. Each mouse event is characterized by (a) its type; (b) the location of the mouse where the event took place (x and y screen coordinates); and (c) the time t when the event took place. Thus, a mouse event is formally described by *event-type* $\langle x,y,t\rangle$.

In general, higher-level actions are formed from sequences of lower-level ones. For example, a double-click is composed of a mouse-down event followed by mouse-up event which takes place within a predefined time frame. This endows a natural hierarchy in which a double-click event is higher than mouse-up and mouse-down. Generally, in order to decide whether two consecutive events are part of a sequence belonging to higher-level feature, the time between their occurrences must fall below (or be above) a *concatenation time-threshold* (CTT). Different thresholds are defined for different event combinations.

3.1.1. Basic mouse actions (level 1)

This level of basic mouse actions is constructed from a sequence of the atomic mouse events – *m*, *ld*, *rd*, *lu* and *ru*. In order to link two consecutive level-0 mouse events into a level-1 event, we define the following CTTs:

- *Moving CTT*: Time threshold for concatenation of two consecutive mouse move events which is denoted by τ_{MM}.
- *Mouse move to left click CTT*: The time between a mouse-move (***m***) event and a left mouse-down (***ld***) event to be linked 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 linked 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 linked 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 intervals 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 from the button down event. Formally,

$$LC_{t_1}^{t_n} = \langle ld_{t_1}, [m_{t_2}, m_{t_3}, \ldots, m_{t_{n-1}}], lu_{t_n} | t_n - t_1 \leqslant \tau_{LC} \rangle$$

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

Right Click (RC) – denotes 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,

$$RC_{t_1}^{t_n} = \langle rd_{t_1}, [m_{t_2}, m_{t_3}, \ldots, m_{t_{n-1}}], ru_{t_n} | t_n - t_1 \leqslant \tau_{RC} \rangle$$

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,

$$MMS_{t_1}^{t_n} = \langle m_{t_1}, m_{t_2}, \ldots, m_{t_n} | \forall k : 1 \leqslant k \leqslant n-1, t_{k+1} - t_k \leqslant \tau_{MM} \rangle$$

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 must exceed τ_{DD}. Formally:

$$DD_{t_1}^{t_n} = \langle d_{t_1}, [m_{t_2}, m_{t_3}, \ldots, m_{t_{n-1}}], u_{t_n} | t_n - t_1 > \tau_{DD} \rangle$$

where d_{t_1} and u_{t_n} denote either a left mouse button down and button up events or a right mouse button down and button up events. The duration of the action has to be greater than the click time, i.e. $\tau_{DD} > \tau_{LC}$ and $\tau_{DD} > \tau_{RC}$, respectively. Note that although the drag-and-drop event is composed of the same lower level events as the left and right click events, in the former a *minimal* time must pass between the mouse down and mouse up events while in the latter, the mouse down and mouse up events must occur *within* a given time frame.

The level 1 mouse actions – LC, RC, MMS and DD – are illustrated in Fig. 3a–d, respectively.

3.1.2. 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 $\boldsymbol{\sigma}$. 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 seconds. Formally:

$$DC_{t_1,t_2}^{t_3,t_4} = \left\langle LC_{t_1}^{t_2} \cdot LC_{t_3}^{t_4} | t_3 - t_2 \leqslant \tau_I \right\rangle$$

The level 2 mouse actions – DC and MM – are illustrated in Fig. 3e and f, respectively.

3.1.3. Level 3 mouse actions

This is the highest level of mouse actions defined in this paper. 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 τ_{MLM} seconds after the last mouse-move event. Formally:

$$MM\_LC_{t_1}^{t_n} = \left\langle MMS_{t_1}^{t_{n-2}} \cdot LC_{t_{n-1}}^{t_n} | t_{n-1} - t_{n-2} \leqslant \tau_{MLM} \right\rangle$$

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

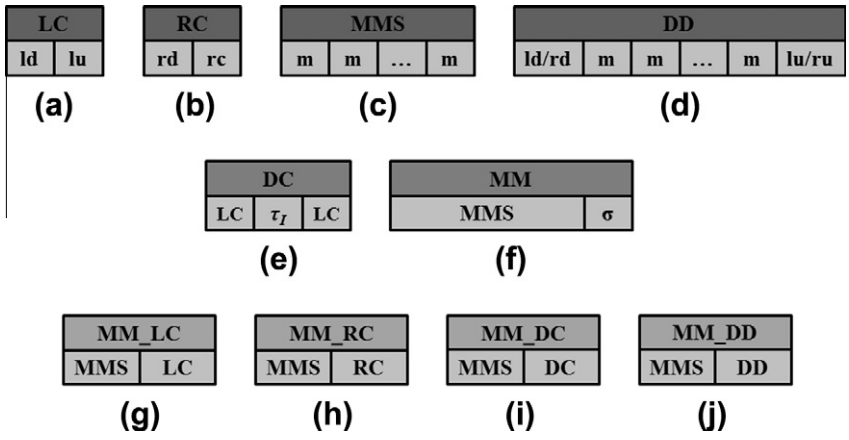

Fig. 3. Schematic description of the various mouse actions: (a) Left click. (b) Right click. (c) Mouse-move sequence. (d) Drag-and-drop action. (e) Double click. (f) Mouse-move. (g) Mouse-move followed by a left click. (h) Mouse-move followed by a right click. (i) Mouse-move followed by a double click. (j) Mouse-move followed by a drag-and-drop.

$$MM\_RC_{t_1}^{t_n} = \left\langle MMS_{t_1}^{t_{n-2}} \cdot RC_{t_{n-1}}^{t_n} | t_{n-1} - t_{n-2} \leqslant \tau_{MRM} \right\rangle$$

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:

$$MM\_DC_{t_1}^{t_n} = \left\langle MMS_{t_1}^{t_{n-4}} \cdot DC_{t_{n-3},t_{n-2}}^{t_{n-1},t_n} | t_{n-3} - t_{n-4} \leqslant \tau_{MLM} \right\rangle$$

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,

$$MM\_DD_{t_1}^{t_n} = \left\langle MMS_{t_1}^{t_{n-2}} \cdot DD_{t_{n-1}}^{t_n} | t_{n-1} - t_{n-2} \leqslant \tau_M \right\rangle$$

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

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

The level 3 mouse actions – MM_LC, MM_RC, MM_DC and MM_DD – are illustrated in Fig. 3g–j, respectively.

3.2. Mouse action 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 extract in order to characterize the mouse movements. We then describe the features that we associate with each mouse action.

3.2.1. Movement Features (MFs)

Histogram-based methods achieve their accuracy by aggregating actions along time. Contrary to the histogram approach, the approach proposed in this paper verifies a user according to individual mouse actions. Thus, in order to compensate for the lack of aggregation while improving the accuracy and verification time, we introduce a set of *new* features that are used in conjunction with the features in Table 4. These additional features alleviate the need to aggregate actions while accurately characterizing a user. We describe them using the notations introduced in Section 2.2.1:

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:

$$TCM = \frac{1}{S_n} \sum_{i=1}^{n-1} t_{i+1} \sqrt{\delta x_i^2 + \delta y_i^2}$$

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

$$SC = \frac{1}{S_n} \sum_{i=1}^{n-1} t_{i+1}^2 \sqrt{\delta x_i^2 + \delta y_i^2} - TCM^2$$

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

$$M_k = \frac{1}{S_n} \sum_{i=1}^{n-1} t_{i+1}^k \sqrt{\delta x_i^2 + \delta y_i^2} \text{ where } k = 3, 4$$

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

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

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

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

Table 5
66 features used to represent a movement sequence.

Factors	$\mathbf{x}'$	$\mathbf{y}'$	θ	c	Δc	$\overline{v_x}$	$\overline{v_y}$	$\bar{v}$	$\bar{\dot{v}}$	$\bar{\ddot{v}}$	w	t_n	S_{n-1}	S	CP	J	TCM	SC	$\boldsymbol{M_k}$	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 summarizes the number of features which are used by the proposed algorithm in order to characterize mouse movement actions. The features in the five rightmost columns correspond to the new features which were introduced above while the rest of the features were described in Table 4 and already used in [20].

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.

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

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 5	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 5) • *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 (MM_DC)	• Mouse movement features from the beginning of the action until the mouse down event (Table 5) • *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 • *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	74
Mouse Move and Drag and Drop Action (MM_DD)	• Mouse movement features from the beginning of the action until the mouse down event (Table 5) • *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 5)	134

3.3. The proposed verification framework

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

3.3.1. Acquisition

The acquisition part captures the mouse events that constitute the users' mouse activity. This part is composed of three modules and an *Actions database* which are illustrated in Fig. 4:

- *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, 63 355 951 016 724⟩ represents a mouse-move event, at location $X = 220$, $Y = 320$ which took place 63 355 951 016 724 ms after the year 1970. This is illustrated by the leftmost module in Fig. 4.
- *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. This module is located second to left in Fig. 4.
- *A feature extractor module* – derives features from the given action. It is illustrated by multiple instances in Fig. 4 (second module to the right) since different feature extractors are required for different types of actions. The extracted features are summarized in Table 6.
- *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. The database is the rightmost component in Fig. 4.

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 [8] classifier which is a multi-class classifier, constructed from an ensemble of decision trees. Ensembles of classifiers have proven to be a powerful tool for classification tasks [22]. 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. Fig. 5 illustrates the training process.

3.3.3. Verification

The verification process is composed of the following components which are illustrated in Fig. 6:

1. Features are extracted from the acquired actions via a process that is similar to the one employed during the acquisition stage. This is performed by the three leftmost modules in Fig. 6 which correspond to the three leftmost modules in Fig. 4.
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.

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

3.3.4. Classifier (Layer 1)

As previously mentioned, the classifier used to construct the model for each action type is the Random Forest [8]. Each of the actions collected by the Action Collector is passed to the appropriate classifier according to the type of action. Let $U = \{u_1, \ldots, u_n\}$ be the set of trained users and let $A = \{a_1, \ldots, a_m\}$ be a set of performed actions.

Each classifier (associated with action a_j) estimates for every trained user u_i the probability she performed action a_j. This probability is denoted by $\widehat{P}(u_i|a_j)$ and it is calculated by the Random Forest classifier during the verification.

Let $T_{ik} = \{t_{ik}^1, t_{ik}^2, \ldots, t_{ik}^{m_{ik}}\}$ be the set of m_{ik} training instances of action type a_k performed by user i. We denote by $P^{apr}(u_i|a_j)$ the *a priori* probability, which is derived from the training data, that action a_j was performed by user u_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 normalized probability $P^{norm}(u_i|a_j)$ is calculated as

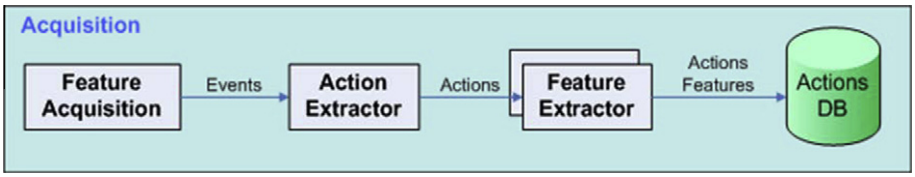

Fig. 4. The acquisition process of mouse activity.

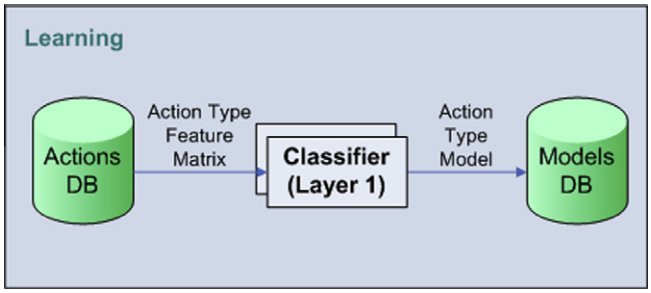

Fig. 5. The training process for each of the action types.

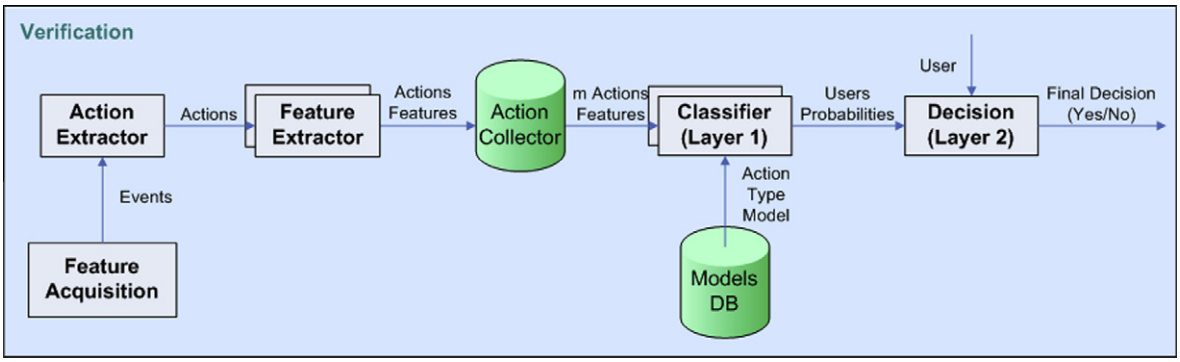

Fig. 6. The user verification process.

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

Finally, we denote by $P^{post}(u_i|a_j)$ the *unbiased* probability that an action a_j was performed by user u_i. This probability incorporates the normalized probabilities and it is given by:

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

3.3.5. 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 $P^{post}(u_i|a_1,\ldots,a_m)$.

The probability that the set of actions $\{a_1,\ldots,a_m\}$ belongs to user u_i is given by the following formula[3]:

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

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

The probabilities are added in the nominator since we assume they are statistically independent. The set of actions $a_1,\ldots,a_m$ is associated to user u_i if the resulting probability is above a threshold λ i.e.

$$\textit{Final Decision } (\{a_1,\ldots,a_m\} \in u_i) = \begin{cases} \text{Yes} & P^{post}(u_i|a_1,\ldots,a_m) \geqslant \lambda \\ \text{No} & \text{Otherwise} \end{cases}.$$

In order to intercept internal attacks the value of λ must be chosen such that the final decision is unique and accurate to a given confidence level. In Section 4.2 we describe a how to choose λ.

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. Contribution of the new features introduced in Section 3.2 to the verification accuracy.

4.1. Data Collection

The feature acquisition described in Section 3 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 and 12 laptops. The CPU speeds ranged from 1.86 GHz to 3.2 GHz 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. These users can be incorporated in the training of the classifiers.
(b) *External Users* – users that are external to the institution or company. No data is available for such users and all access attempts performed by them should be classified as (external) attacks.

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 she performs since usually legal users interact with their computer systems most of the time. 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 was taken into account when the number of legal verification attempts, internal attacks and external attacks were determined for the evaluation.

4.1.2. Experiment configuration

The thresholds τ_{MM}, τ_{MLM}, τ_{MRM}, τ_{LC}, τ_{RC}, τ_I that were used in order to construct the actions defined in Section 3.1 were empirically set to 500 ms. The action extraction incorporated filtering 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 performed if a minimal amount of events was at hand. Only movements that contained at least four different points were considered. Events whose type and position were equal to those of the event which preceded them were ignored.

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

The set of all available users $U = \{u_1,\ldots,u_n\}$ was randomly divided into a set of k internal users $IU = \{iu_{j_1},\ldots,iu_{j_k} | iu_{j_l} \in U, 1 \leqslant j_l \leqslant n, l = 1,\ldots,k\}$ and a set of external users $EU = U - IU$. Profiles were constructed for each of the internal users in IU according to their training activity (the training fold). Each user $u \in U$ was tested for authorized access using her corresponding test fold. Unauthorized access to u's computer system was tested by using test folds of users other than u's. We refer to unauthorized access as *internal attacks* when the test folds belong to users for whom profiles were constructed i.e.

users in *IU*. External attacks are simulated by using mouse activity of users that belong to *EU*. Both access types were tested using 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. Specifically, 24 internal attacks were simulated for each user in each of the two folds. Six external attacks were simulated for each user in each of the two folds.

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 $144 \times 25 = 3600$ verification attempts in total.

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

4.2. Evaluation measures

We evaluated the proposed according to the FAR, FRR, EER and AUC which are formally described is Section 2.2. In order to create the ROC curve we have examined the FAR and FRR for various values of $\lambda \in [0,1]$. We found that decreasing the value of λ produced a lower FRR and a higher FAR. Accordingly, we first calculated $P^{post}(u_i|a_1,\ldots,a_m)$ for all test instances where u_i is the verified user. Then, we sorted the test instances in descending order according to P^{post} and processed the instances one at a time from beginning to end setting $\lambda = P^{post}$. After obtaining the ROC graph, the actual value of λ should be set according to the desired FAR/FRR.

We also defined the following measurements:

- 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] used histograms in order to *aggregate* multiple actions and utilized 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 discretized features were used by both the proposed approach and the histogram-based one. By performing aggregation of the discretized features of each action, occurrence histograms as in [2] 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.

Fig. 7a and b present the comparison results between the aggregation and the action-based approaches. Fig. 7a 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.

Fig. 7b 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 ($\sim$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 s. 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

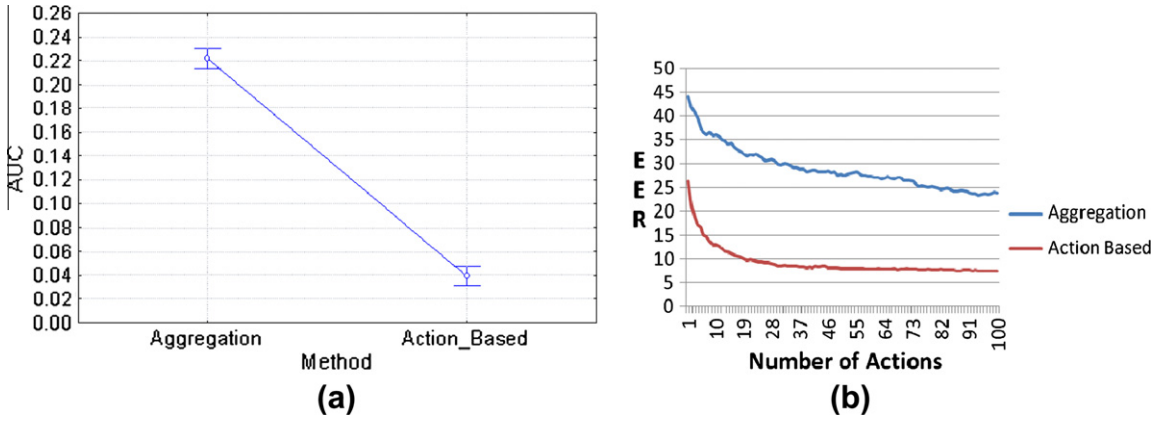

Fig. 7. Comparison between the action-based method and the histogram-based approach. (a) AUC comparison – ANOVA test with 95% confidence interval. (b) EER comparison according to the number of actions that are used for the verification.

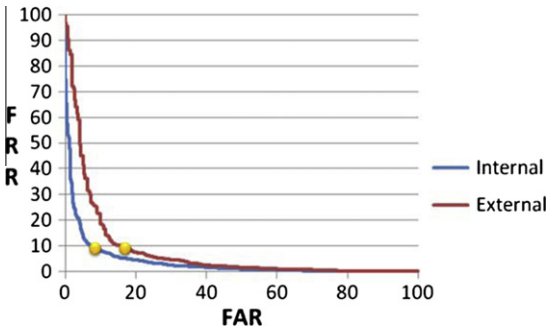

Fig. 8. ROC curve comparing the verification accuracy between internal and external users based on 30 actions.

approximately 42 s and 2.33 min, respectively. Consequently, the approach proposed in this paper provides a method for verifying the user in less than 2 min with a maximal equal error rate of 10%.

Fig. 8 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 [2] 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.

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 $\widehat{P}(u_i|a_j)$.

Fig. 9a 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. Fig. 9b 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 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 min on the average while testing required 2.7746 ms. However, since

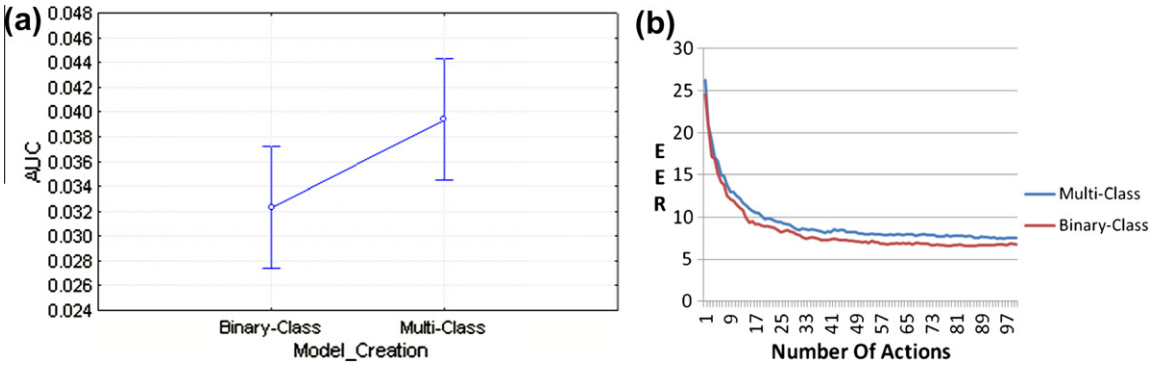

Fig. 9. Comparison between the binary-class models and the multi-class model approaches. (a) AUC comparison – ANOVA test with 95% confidence interval. (b) EER comparison according to the number of actions that are used for the verification.

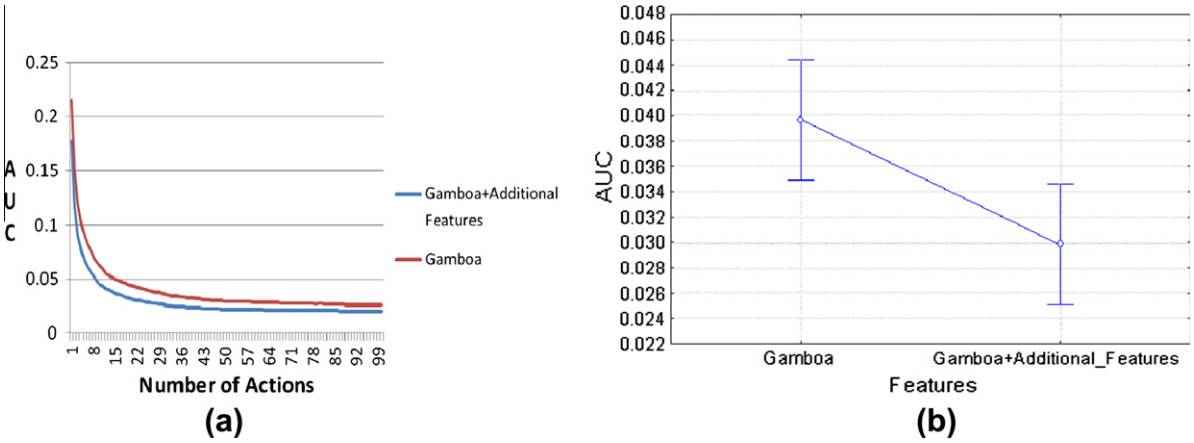

Fig. 10. Contribution of the new additional features that were introduced in Section 3.2. (a) AUC according to the number of actions that are used for the verification. (b) ANOVA test in terms of AUC based on a 95% confidence interval.

training in the binary-model approach requires the construction of an individual binary model for every user, the training time took 7.031 min $*$ 12 (*users*) = 84.372 min and the testing time took 2.7135 ms $*$ 12 (*users*) = 32.562 ms.

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]. Fig. 10a and 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. Fig. 10a 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 Fig. 10b.

5. Conclusion 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 [2]. 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] R.B. Abernethy, The New Weibull Handbook, ISBN-13: 978-0965306232, 2006.
[2] A.A.E. Ahmed, I. Traore, A new biometric technology based on mouse dynamics, IEEE Transactions on Dependable and Secure Computing 4 (3) (2007) 165–179.
[3] S. Al-Zubi, A. Bromme, K. Tonnies, Using an active shape structural model for biometric sketch recognition, in: Proceedings of DAGM, 2003, pp. 187–195.
[4] L. Ballard, D. Lopresti, F. Monrose, Evaluating the security of handwriting biometrics, in: The 10th International Workshop on Frontiers in Handwriting Recognition (IWFHR '06), La Baule, France, 2006.
[5] F. Bergadano, D. Gunetti, C. Picardi, User authentication through keystroke dynamics, ACM Transactions on Information and System Security 5 (4) (2002) 367–397.
[6] S. Bleha, C. Slivinsky, B. Hussein, Computer-access security systems using keystroke dynamics, IEEE Transactions on Pattern Analysis and Machine Intelligence 12 (12) (1999) 1217–1222.
[7] P. Bours, C.J. Fullu, A login system using mouse dynamics, in: Fifth International Conference on Intelligent Information Hiding and Multimedia Signal Processing, 2009, pp. 1072–1077.
[8] L. Breiman, Random forests, Machine Learning 45 (1) (2001) 5–32.
[9] A. Bromme, S. Al-Zubi, Multifactor biometric sketch authentication, in: Proceedings of the BIOSIG 2003, 2003, pp. 81–90
[10] S. Cho, C. Han, D.H. Han, H.I. Kim, Web-based keystroke dynamics identity verification using neural network, Journal of Organizational Computing and Electronic Commerce 10 (4) (2000) 295–307.
[11] Z. Ciota, Speaker verification for multimedia application, IEEE International Conference on Systems, Man and Cybernetics 3 (10) (2004) 2752–2756.
[12] J.-F. Connolly, E. Granger, R. Sabourin, An adaptive classification system for video-based face recognition, Information Sciences, in press (Corrected Proof, Available online 6 March 2010).
[13] D. Stefan, X. Shu, D. Yao, Robustness of keystroke-dynamics based biometrics against synthetic forgeries, Computers & Security 31 (1) (2012) 109–121.
[14] S. Deshpande, S. Chikkerur, V. Govindaraju, Accent classification in speech, Fourth IEEE Workshop on Automatic Identification Advanced Technologies, 17–18 October, 2005, pp. 139–143.
[15] H. Erdogan, A. Ercil, H.K. Ekenel, S.Y. Bilgin, I. Eden, M. Kirisci, H. Abut, Multi-modal person recognition for vehicular applications, in: N.C. Oza et al. (Eds.), MCS, LNCS, vol. 351, Springer, Heidelberg, 2005, pp. 366–375.
[16] E. Erzin, Y. Yemez, A.M. Tekalp, A. Erçil, H. Erdogan, H. Abut, Multimodal person recognition for human–vehicle interaction, IEEE MultiMedia 13 (2) (2006) 18–31.
[17] T. Fawcett, An introduction to ROC analysis, Pattern Recognition Letters – Special Issue: ROC Analysis in Pattern Recognition 27 (8) (2006) 861–874.

[18] G. Frantzeskou, S. Gritzalis, S. MacDonell, Source code authorship analysis for supporting the cybercrime investigation process, in: 1st International Conference on eBusiness and Telecommunication Networks – Security and Reliability in Information Systems and Networks Track (ICETE04), 2004, pp. 85–92.
[19] H. Gamboa, A. Fred, An identity authentication system based on human computer interaction behavior, in: 3rd International Workshop on Pattern Recognition on Information Systems, 2003, pp. 46–55.
[20] H. Gamboa, A. Fred, Behavioral biometric system based on human computer interaction, Proceedings of SPIE 5404 (2004) 381–392.
[21] H. Gamboa, A.L.N. Fred, A.K. Jain, Webbiometrics: user verification via web interaction, in: Proceedings of Biometrics Symposium, 2007, pp. 1–6.
[22] N. García-Pedrajas, J. Maudes-Raedo, C. García-Osorio, J.J. Rodríguez-Díez, Supervised subspace projections for constructing ensembles of classifiers, Information Sciences, in press (Corrected Proof, Available online 8 July 2011).
[23] A. Gray, P. Sallis, S. Macdonell, Software forensics: extending authorship analysis techniques to computer programs, in: Proceeding of the 3rd Biannual Conference of the International Association of Forensic Linguists (IAFL'97), 1997.
[24] P. Grother, E. Tabassi, Performance of biometric quality measures, IEEE Transactions on Pattern Analysis and Machine Intelligence 29 (4) (2007) 531–543.
[25] D. Gunetti, C. Picardi, Keystroke analysis of free text, ACM Transactions on Information Systems Security 8 (3) (2005) 312–347.
[26] O. Hamdy, I. Traoré, Homogeneous physio-behavioral visual and mouse-based biometric, ACM Transactions on Computer–Human Interaction 18 (3) (2011) 1–30 (Article 12).
[27] S. Hashia, C. Pollett, M. Stamp, On using mouse movements as a biometric, in: Proceeding in the International Conference on Computer Science and its Applications, vol. 1, 2005.
[28] A. Jain, F. Griess, S. Connell, Online signature verification, Pattern Recognition 35 (12) (2002) 2963–2972.
[29] A.K. Jain, S. Pankanti, S. Prabhakar, L. Hong, A. Ross, Biometrics: A grand challenge, Proceedings of International Conference on Pattern Recognition, vol. 2, 2004, pp. 935–942.
[30] A.R. Jansen, D.L. Dowe, G.E. Farr, Inductive inference of chess player strategy, in: Proceedings of the 6th Pacific Rim International Conference on Artificial Intelligence (PRICAI'2000), 2000, pp. 61–71.
[31] Z. Jorgensen, T. Yu, On mouse dynamics as a behavioral biometric for authentication, in: Proceedings of the Sixth ACM Symposium on Information, Computer, and Communications Security (AsiaCCS), March 2011.
[32] A. Kale, A. Sundaresan, A.N. Rajagopalan, N. Cuntoor, A. Roychowdhury, V. Kruger, R. Chellappa, Identification of humans using gait, IEEE Transactions on Image Processing 13 (9) (2004) 1163–1173.
[33] A. Kumar, S. Shekhar, Personal identification using multibiometrics rank-level fusion, IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 41 (5) (2011) 743–752.
[34] D.-T. Lin, Computer-access authentication with neural network based keystroke identity verification, in: Proceedings of IEEE International Conference on Neural Networks, Houston, Texas, June 09–12, vol. 1, 1997, pp. 174–178.
[35] M. De Marsico, M. Nappi, D. Riccio, G. Tortora, NABS: novel approaches for biometric systems, IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 41 (4) (2011) 481–493.
[36] R.A. Maxion, T.N. Townsend, Masquerade detection using truncated command lines, in: International Conference on Dependable Systems and Networks (DNS-02), IEEE Computer Society Press, 2002.
[37] M. Pusara, C.E. Brodley, User re-authentication via mouse-movements, in: Proceedings of the ACM Workshop Visualization and Data Mining for Computer Security, (VizSEC/DMSEC 2004), ACM, Washington, DC, USA, 2004, pp. 1–8.
[38] R.J. Quinlan, C4*5: Programs for Machine Learning, Morgan Kaufman, 1993.
[39] J. Ramon, N. Jacobs, Opponent modeling by analyzing play, in: Proceedings of the Computers and Games workshop on Agents in Computer Games (CG'02), 2002.
[40] K. Revett, H. Jahankhani, S.T. de Magalhes, H.M.D. Santos, A survey of user authentication based on mouse dynamics, in: Proceedings of 4th International Conference on Global E-Security, Communications in Computer and Information Science, vol. 12, London, UK, June 2008, pp. 210–219.
[41] K. Revett, P.S. Magalhães, 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, September 19–22, University of Technology of Compiègne, France, 2005, pp. 56–61.
[42] M. Schonlau, W. DuMouchel, H. Ju, A.F. Karr, M. Theus, Y. Vardi, Computer intrusion: detecting masquerades, Statistical Science 16 (2001) 1–17.
[43] D.A. Schulz, Mouse curve biometrics, in: Biometric Consortium Conference, 2006 Biometrics Symposium, 2006, pp. 1–6.
[44] E.H. Spafford, S.A. Weeber, Software forensics: can we track code to its authors?, in: 15th National Computer Security Conference, 1992, pp. 641–650.
[45] S.J. Stolfo, S. Hershkop, K. Wang, O. Nimeskern, C.W. Hu, W. Hu, A Behavior-based Approach to Securing Email Systems, Mathematical Methods, Models and Architectures for Computer Networks Security, Springer-Verlag, 2003.
[46] S.J. Stolfo, C.W. Hu, W.J. Li, S. Hershkop, K. Wang, O. Nimeskern, Combining Behavior Models to Secure Email Systems, Technical Report, Columbia University, 2003. <www1.cs.columbia.edu/ids/publications/EMT-weijen.pdf>.
[47] O.D. Vel, A. Anderson, M. Corney, G. Mohay, Mining email content for author identification forensics, ACM SIGMOD Record: Special Section on Data Mining for Intrusion Detection and Threat Analysis 30 (4) (2001) 55–64.
[48] T. Westeyn, P. Pesti, K. Park, T. Starner, Biometric identification using song-based eye blink patterns, Human Computer Interaction International (HCII), Las Vegas, NV, 2005.
[49] T. Westeyn, T. Starner, Recognizing song-based blink patterns: applications for restricted and universal access, in: 6th IEEE International Conference on Automatic Face and Gesture Recognition, 2004.
[50] R.V. Yampolskiy, V. Govindaraju, Behavioral biometrics: a survey and classification, International Journal of Biometrics 1 (1) (2008) 81–113.
[51] M.H. Zweig, G. Campbell, Receiver-operating characteristic (roc) plots: a fundamental evaluation tool in clinical medicine, Clinical Chemistry 39 (4) (1993) 561–577.