Dynamic User Authentication Based on Mouse Movements Curves

Zaher Hinbarji, Rami Albatal, and Cathal Gurrin

Insight Centre for Data Analytics,
Dublin City University
zaher.hinbarji@insight-centre.org
https://www.insight-centre.org

Abstract. In this paper we describe a behavioural biometric approach to authenticate users dynamically based on mouse movements only and using regular mouse devices. Unlike most of the previous approaches in this domain, we focus here on the properties of the curves generated from the consecutive mouse positions during typical mouse movements. Our underlying hypothesis is that these curves have enough discriminative information to recognize users. We conducted an experiment to test and validate our model in which ten participants are involved. Back propagation neural network is used as a classifier. Our experimental results show that behavioural information with discriminating features is revealed during normal mouse usage, which can be employed for user modeling for various reasons, such as information assets protection.

Keywords: multimedia security, user modeling, human-computer interaction, mouse dynamics

1 Introduction

We take care to encrypt or password protect our computers so that we can restrict unauthorised access. However, we don't typically consider the situation of 'friendly' unauthorised access to our private personal archives. Consider the people that have access to your physical computing devices and can theoretically access your content. This work addresses this aspect of multimedia content storage and protection by proposing a user authentication method based on mouse movements alone. One could easily imagine the computer immediately locking out a user who accesses the private multimedia archive of the data owner. It is common to have three security procedures working together to protect information assets and to ensure the controlled access. These procedures are: authentication, authorization and auditing [1].

We focus here on the first component of security systems, personal identification and authentication, which can be done by something the user knows (e.g., password, PIN code, pattern), something the user has (e.g., card, access token, wrist band) or something the user is or does (e.g., a fingerprint, signature, face, voice, which are known as biometrics) [2]. The need for special hardware devices

for data capture is a big limitation of most biometric systems. The advantage of a mouse-based authentication system is that it can be implemented using a regular mouse [2], [3], [4], [5]. User authentication can be achieved statically or dynamically. In the static approach, the system checks the identity of the user once, usually at the beginning of the session so any change of user after that will be unnoticeable to the system. In contrast, dynamic verification checks the user continuously over the session which can effectively prevent session hijacking, however that should be done passively without interrupting the user [6]. In this work, we present a dynamic user authentication based on mouse movements only and using normal mouse devices.

2 Related Work

Several mouse dynamics approaches for dynamic authentication have been proposed in the literature, presenting different types of features. Hayashi et al. [7] presented one of the earliest research in this domain; users were requested to use the mouse for drawing circles or other figures, and then analytics algorithms were applied on features based on the distances between the mouse coordinates and the centre of the shapes. In [2], Pusara and Brodley used the distance, angle and speed between pairs of data points as raw features which then used to produce their mean, standard deviation and the third moment values (distance, angle and speed) over a window of N data points. In Ahmed and Traore's work [3], raw mouse events are aggregated and then classified by action type. Consecutive actions are grouped into sessions, from which features related to movement speed, movement direction, traveled distance are computed producing user signature. Schulz in [5] presented a model in which raw data are broken into mouse curves; length, curvature and inflection points of the curve are used as main features, and a reference signature is built by generating histograms from the curve characteristics of multiple curves. The Verification is implemented then by computing the Euclidean distance between the reference signature and the mouse activity observed during authentication time. Jorgensen and Yu [8] evaluated the approaches done by [2] and [3] and discussed some of the limitations in their works. According to [8] it is not clear whether the system detects the differences in mouse behavior or differences among the working environment including the performed task.

To overcome the above mentioned drawback of the state-of-the-art methods, our main challenge is to extract features that can reflect the user behavior patterns regardless of both the task that she/he is performing and the environment she/he is working in. Our underlying hypothesis is that mouse curves have such user-specific information that can be used to model users behavior. Our approach is similar to the one followed by Schulz [5] in using histograms of features extracted form the mouse curves. However, the two approaches differ in two major aspects. First, we use different features and emphasise on using ones that satisfy certain mathematical properties related to task-independence. Second, [5] uses a single 'reference' signature per user. The Euclidean distance

between the evaluated signature and reference signature is then used to validate the user. Our method, on the other hand, uses instead a neural network per user which is trained using multiple signatures. The neural network has the potential of better recognition by generalising from different signature training samples.

3 Behavior Modeling

In order to make our approach practical enough to be used in typical working environment, our model uses raw mouse coordinates collected from users during their normal workday activities without any kind of restrictions. The consecutive mouse coordinates are grouped into curves that correspond to the typical performed mouse actions (point-and-click, move, drag-drop). A single curve does not have enough information by itself to refer to its user, that is why we group the curves into sessions in order to study the statistical behavior characteristics observed during each session. A session is a number of consecutive curves belong to the same user. To show how results are affected by the length of the session we will present the accuracy of the system in terms of 3 different values of session length: 100, 200 and 300 curves.

The objective of our method is to verify the identity of the user based on features of the curves followed when moving the mouse from one point to another. The exact values of those features may vary considerably even for curves belonging to the same user. However, we assume that each feature follows a probability distribution that is unique to each user and can serve as a signature of his/her mouse movements. The probability distribution of each feature is approximated by a normalized histogram computed using a large number of curves belonging to a certain user (session). The histograms of different features belonging to a certain user form the signature of that user, which is the input of our detection algorithm.

4 Mouse Curve Features

In this section, we describe the nine different features used to characterize a single mouse curve. Each feature gives a single value as a descriptor of the curve, except for inflection profile, sharpness profile and central moments that generate five, five and three values accordingly. As a result, we have in total 19 values describing each curve. To the best of our knowledge we are the first who introduced these features in this domain except for straightness and inflection profile which are previously used before by [5]. However, we introduce here our own implementation for these features.

A mouse curve is defined as a tuple (ordered list) of two or more 2D points:

$$C = (p_1, p_2, ..., p_n) : n \geq 2,\ p_i = (x_i, y_i) \in \mathbb{R}^2\ , \tag{1}$$

where n is the number of points.
A feature of this curve is simply a function of the coordinates of its points:

$$F(C) : \mathbb{R}^{2n} \rightarrow \mathbb{S} \subseteq \mathbb{R}\ , \tag{2}$$

where $\mathbb{S}$ is a subset of the real numbers.

Since those features will be used to characterize a user, they should be task independent, which implies that any feature should be independent of the position, size and orientation of the curve. This means that feature functions should satisfy the following mathematical properties:

- **Translational invariance**

$$F(p_1 + q, p_2 + q, ..., p_n + q) = F(p_1, p_2, ..., p_n) \ , \tag{3}$$

  where $q \in \mathbb{R}^2$ is a 2D translation vector.
- **Scale invariance**

$$F(\alpha p_1, \alpha p_2, ..., \alpha p_n) = F(p_1, p_2, ..., p_n) \ , \tag{4}$$

  where $\alpha \in \mathbb{R}$ is a scaling factor.
- **Rotational invariance**

$$F(\mathbf{M}\ p_1, \mathbf{M}\ p_2, ..., \mathbf{M}\ p_n) = F(p_1, p_2, ..., p_n) \ , \tag{5}$$

  where $\mathbf{M} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$ is a 2D rotation matrix that rotates points counter-clockwise through an angle θ about the origin.

4.1 Efficiency

The goal of a single mouse movement is to go from the first point p_1 to the last one p_n. The shortest path between those two points is a straight line while any other curve will be longer. Efficiency is defined as the ratio of the length of the shortest path over the length of the curve

$$E = \frac{\sqrt{(x_n - x_1)^2 + (y_n - y_1)^2}}{\sum_{i=1}^{n-1} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}} \in [0, 1] \ . \tag{6}$$

This feature measures how *efficient* a curve is, in achieving its goal. The value of efficiency rages between 0 and 1. A curve with value 1 is the shortest path and the most efficient (It should be a straight line but the converse is not necessarily true), while a curve with value near 0 is a one that moves a lot without going too far. A user whose curves have a high efficiency value tends to move the mouse directly between target positions without making many unnecessary movements.

4.2 Straightness

This feature measure how much a curve resembles a straight line. This is done by studying the correlation between the curve points. First, we compute the covariance matrix of the points coordinates:

$$\Sigma = \begin{pmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{yx} & \sigma_y^2 \end{pmatrix} \tag{7}$$

σ_x^2 and σ_y^2 are the variances along the x and y axes, respectively. σ_{xy} is the covariance of x and y.

Then we compute the eigenvalue decomposition of the covaraince matrix. The largest eigenvalue (let it be λ_1) represents the largest variance of the points along any direction while the second eigenvalue (let it be λ_2) represents the variance along the direction orthogonal to the previous one. As a straightness measure, we use the following relation:

$$S = \frac{\lambda_1 - \lambda_2}{\lambda_1} \in [0, 1] \, . \tag{8}$$

Straightness value range between 0 and 1. When the relation between x and y is almost linear, λ_1 is much larger than λ_2 and the straightness is near 1. When the relation is non-linear, λ_1 and λ_2 are close and the straightness is near 0 (see Fig. 1). Our measure of straightness is meant to replace the one given by [5]. Their measure is a yes/no measure that does not account for that fact that some curves appear more straight than others while our measure gives a zero value when the curve is exactly a straight line and a positive value proportional to how well the curve resembles a straight line.

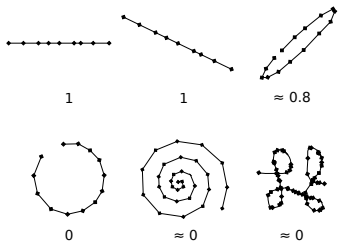


Fig. 1: Different sets of points with their straightness values.

4.3 Regularity

This feature measures how regular a curve is by looking at the distances of its points to its geometrical centre.

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i, \quad \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$$
$$d_i = \sqrt{(x_i - \bar{x})^2 + (y_i - \bar{y})^2}$$

Regularity is then defined in terms of the mean and standard deviation of those distances as

$$R = \frac{\mu_d}{\mu_d + \sigma_d} \in [0, 1] \, , \tag{9}$$
$$\text{where} \quad \mu_d = \frac{1}{n}\sum_{i=1}^{n} d_i, \;\; \sigma_d^2 = \frac{1}{n}\sum_{i=1}^{n} (d_i - \mu_d)^2 \, .$$

Note that curves forming regular polygons, like equilateral triangle or squares, have all their corners at the same distance from their center. This implies that the variance of the distances is zero and thus regularity is one.

4.4 Self-Intersection

This feature counts the number of times a curve intersects with itself. To find this number, we test all pairs of line segments for intersection. This approach takes $\mathcal{O}(n^2)$ which is fine because n is small in our case.

Two line segments p_1p_2 and p_3p_4 intersect, if and only if both:

- p_1 and p_2 are on different sides of p_3p_4,
- p_3 and p_4 are on different sides of p_1p_2.

These are respectively true, if and only if :

- Angles $\angle p_1p_3p_4$ and $\angle p_2p_3p_4$ are of opposite signs.
- Angles $\angle p_3p_1p_2$ and $\angle p_4p_1p_2$ are of opposite signs.

These are respectively true, if and only if :

- Cross products $\overrightarrow{p_3p_1} \times \overrightarrow{p_3p_4}$ and $\overrightarrow{p_3p_2} \times \overrightarrow{p_3p_4}$ are of opposite signs.
- Cross products $\overrightarrow{p_1p_3} \times \overrightarrow{p_1p_1}$ and $p_1\overrightarrow{p4} \times \overrightarrow{p_1p_2}$ are of opposite signs.

As a result, it is sufficient to check the fulfillment of the last two conditions to test whether two line segments intersect. Note that we do not consider touching line segments as intersecting.

4.5 Curvature-based Features

Mathematically, the *curvature* of a continuously differentiable curve is defined as the rate of change of the tangential angle with respect to the arc length: $\kappa = d\phi/ds$.

Handling curvature of mouse curves is problematic because they are piecewise linear, so the curvature is zero inside the line segments and ill-defined on the corners. One approach to solve this problem is approximating the rough mouse curve by another smooth curve like B-spline (as done in [5]) or Bezier curves and then studying the curvature of the resulting smooth curves. We follow here a different approach.

First let us look at the the line integral of the curvature between two points of the curve

$$\Phi(a,b) = \int_a^b \kappa \, ds = \int_a^b d\phi = \phi(b) - \phi(a) \; . \tag{10}$$

It is called the *total curvature* and it is nothing but the total change in the tangential angle.

For mouse curves, the total curvature is well-defined and it is a step function that is constant along the line segments and jumps at the corners; The value of the jump at a corner equals to the change in the angle (see Fig. 2).

Since the curvature is the derivative of the total curvature, it is ill-defined at the jumps because of the discontinuity. However, it can still be defined in a

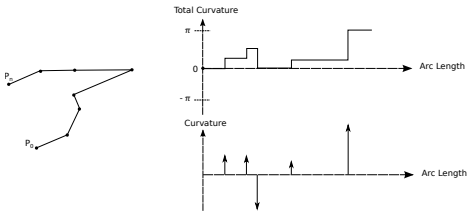


Fig. 2: Curvature and total curvature of a mouse curve (piecewise linear curve).

distributional sense as a Dirac delta function. The behavior of delta function is well-defined inside integrals and most curvature-based features are defined using integrals. The only feature that goes beyond integration, is the inflection profile but the problem is resolved then by replacing delta functions with Gaussian function.

Formally, the curvature of a mouse curve is then defined as a linear combination of delta functions positioned at the corners and weighted by the angles

$$\kappa(s) = \sum_{i=1}^{n-2} \theta_i \; \delta\left(s - \frac{s_i}{s_{n-1}}\right) \quad \text{where:} \tag{11}$$

$$s_i = \sum_{j=1}^{j=i} \sqrt{(x_j - x_{j+1})^2 + (y_j - y_{j+1})^2}$$

$$\theta_i = \text{sign}(\mathbf{v}_i \times \mathbf{v}_{i+1}) \arccos\left(\frac{\mathbf{v}_i . \mathbf{v}_{i+1}}{\|\mathbf{v}_i\| \|\mathbf{v}_{i+1}\|}\right)$$

$$\mathbf{v}_i = \overrightarrow{p_i p_{i+1}} \; .$$

Notice that in Eq. 11 the positions are rescaled by the length of the curve (s_{n-1}) in order to make the curvature scale-invariant.

Total Angle This feature measures the total change in angle between the first and last points of the curve. It equals to the integral of the curvature

$$\text{TA} = \int ds \; \kappa(s) = \int ds \sum_{i=1}^{n-2} \theta_i \; \delta(s - s_i) = \sum_{i=1}^{n-2} \theta_i \int ds \; \delta(s - s_i) = \sum_{i=1}^{n-2} \theta_i \; . \tag{12}$$

The last equality holds because the integral of a Dirac delta function is unity.

Bending Energy This feature measures the total change of angles regardless of the sign. It equals to the L^1-norm of the curvature

$$\text{BE} = \int ds \; |\kappa(s)| = \int ds | \sum_{i=1}^{n-2} \theta_i \; \delta(s - s_i)| = \sum_{i=1}^{n-2} |\theta_i| \int ds \; \delta(s - s_i) = \sum_{i=1}^{n-2} |\theta_i| \; . \tag{13}$$

Inflection Profile Inflection point is where the curve changes the sign of its curvatures. Computing the number of inflection points naively, by counting the number of sign changes of θ_i, results many spurious inflection points because of the noise on the curvature. Therefore, we need to smooth the curvature by replacing the delta functions with Gaussian functions of the same weight and certain width (see Fig. 3). The number of sign inflection points is the number of sign changes in the smoothed curvature. In order to avoid biasing the result to a certain smoothing level (certain width of the Gaussian), we use the different number of inflection points at five different smoothing levels as features. Taken together, we call these features *inflection profile*.

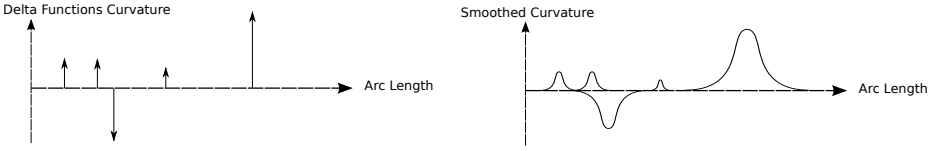


Fig. 3: Curvature can be smoothed by replacing delta functions with Gaussian functions of the same weight and at the same postion.

Sharpness Profile We define a sharp bend as an absolute change in the angle by more than $\pi/2$. A sharp bend may happen on a single corner or may take several corners. To find the number of sharp bends requiring m corners, we sum each m consecutive angles θ_i and then count the values that are greater than $\pi/2$ in absolute value.

We compute the number of sharp bends requiring five different number of corners. Taken together, these features are called *sharpness profile*.

Center and Central Moments The n-th moment of a normalized positive function around point c is defined as

$$\mu'_n(c) = \int ds\ (s-c)^n\ f(s)\ . \tag{14}$$

The first moment around 0 is called the mean $\mu := \mu'_1(0)$ and the moments around the mean $\mu_n := \mu'_n(\mu)$ are called central moments. Knowing the mean and the first few central moments of a function gives a rough idea about its shape.

We use the moments of the normalized absolute value of the curvature $f(s) = |\kappa(s)|/\int ds\ |\kappa(s)|$ as features. They are computed using the following relations which are obtained by substituting $f(s)$ in Eq. 14:

$$\mu = \frac{\sum_{i=1}^{n-2} s_i\ |\theta_i|}{\sum_{i=1}^{n-2} |\theta_i|} \tag{15}$$

$$\mu_n = \frac{\sum_{i=1}^{n-2} (s_i-\mu)^n\ |\theta_i|}{\sum_{i=1}^{n-2} |\theta_i|} \tag{16}$$

We used the values of the first 3 moments in our features vectors.

5 Behavior Comparison

As described earlier, we do not use the features themselves for classification but rather their probability distribution. This probability distribution can be approximated using a normalized histogram of the feature values. We determine the binning of the histogram using the equal-frequency discretization algorithm [9]. Given the feature values of all the users together, we sort all the values in ascending order. If the number of required bins is n, we use every nth value of the sorted data as a bin marker, plus the first and last values. Regarding the number of bins, we use eight bins per feature; this has heuristically given good results in approximating the underlying distributions of the features. As a result, our final signature vector has 152 values, which is an 8-value histogram for each one of our 19 features concatenated together.

Authenticating users using a signature is done via artificial neural networks. Each user has his/her own neural network which is trained to recognize his/her signature alone. A neural network automatically selects the most discriminating factors in the features vector by over-weighting the most discriminating features and ignoring the less important ones. The neural network used in our approach is a feed-forward multilayer perceptron network of three layers. The input layer consists of 152 nodes, corresponding to the length of our features vector. The hidden layer consists of 50 nodes whereas the network has only one node in the output layer. The output of the network is in the range [0,1]. During training, positive examples are assigned value 1 and negative examples are assigned value 0. However, due to the interpolating nature of the neural network, the output for new examples will lie in between. Therefore, a threshold limit is used to authorize the claimed identity.

To train a neural network to recognise sessions for a targeted user, we consider all the sessions that belong to that user in the training set as positive examples and all other sessions as negative examples. This was carried out in a similar manner for all users. In real life scenario, the first sessions of the user are used passively to build his/her signature. User authentication is the final step in the process; in order to verify the claimed identity of a session we first extract the features vector of that session and use it as an input for the network specifically trained to recognize that identity. Then, the output of the network is compared to an authentication threshold to ensure that it is sufficient enough to authenticate that captured behavior.

6 Evaluation

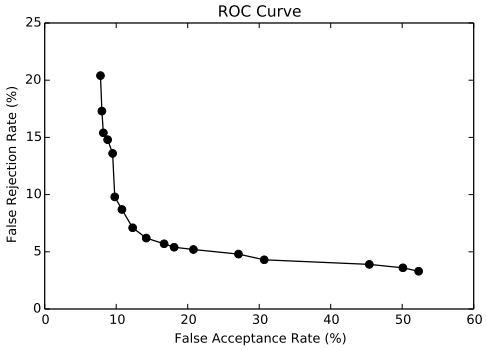
In order to test our model in normal working environment, our data set is collected from users during their regular workday activities without any kind of pre-defined tasks or restrictions. We developed a background app that intercepts raw mouse events passively and save them for later analysis. Ten users participated in this experiment using Mac OS without changing their mouse settings. The collected data is only mouse coordinates with an indicator to the current

performed action (point-and-click, move, drag-drop). Around 16,500 actions are collected for each user during about 24 working hours. Our evaluation is done using three measures: false acceptance rate (FAR), false rejection rate (FRR) and equal error rate (EER). FAR measures the likelihood that the system incorrectly accepts an access attempt by an intruder. Whereas, FRR measures the likelihood that the system incorrectly rejects an access attempt by an authorized user. EER is the value at which both acceptance and rejection errors are equal due to tuning the authentication threshold of the system. The sessions of each user are divided into two equal subsets: training set and testing set. After training the system on the training set only, we test the system by counting the number of misclassifications against all the sessions in the testing set. FRR is computed by counting the number of misclassifications when both the testing session and the trained neural network belong to the same user. On the other hand, FAR is computed by counting the number of misclassifications when the testing session and the trained neural network belong to different users. To make sure that FAR also covers the case of an attacker who has not been seen before by the system, the previous process is repeated ten times (according to the number of users). In each time, one of the users is considered as an outsider; his sessions are excluded during the training phase and are added to the testing set. The total number of misclassifications over the ten times is used to calculate FAR and FRR. To find the value of EER, we conduct the testing by varying the authentication threshold between 0.1 and 0.9. Figure 4 shows how the values of FAR decreases and the FRR increases as the authentication threshold increases for sessions of length 100 curves. The higher the threshold, the lower the probability the system incorrectly accepts an intruder and the higher the probability the system incorrectly rejects an authorized user. The shape of ROC curves for the two other values of session length (200,300) are similar to Figure 4. As expected, the EER decreases when we increase the length of the session. Increasing the session length allow the system to detect more behavioral patterns. However, long sessions give the attacker more time to finish his/her attack before the system detects him/her. Since the session length measured by the number of actions (curves), the actual needed time to authenticate a user may vary considerably even for sessions of the same user depending on how much time the user needs to generate the sufficient mouse actions. Table 1 shows the EER and the corresponding total average time our subjects took to generate the needed curves.

As we introduced before, [5] presented a similar model based on mouse curves too. By comparing results in Table 2 and our reported error rates (Table 1) we can see that our model achieves good results toward a reliable task-independence mouse based authentication system.

Table 1: EER values and the corresponding session length

Session Length (#Curves)	**Session Length (Time)**	**EER**	**Threshold**
100	5.6 min	9.8%	65%
200	14.3 min	7.2%	50%
300	18.7 min	5.3%	55%



Threshold	FRR(%)	FAR(%)
10%	3.3	52.3
15%	3.6	50.1
20%	3.9	45.4
25%	4.3	30.7
30%	4.8	27.1
35%	5.2	20.8
40%	5.4	18.1
45%	5.7	16.7
50%	6.2	14.2
55%	7.1	12.3
60%	8.7	10.8
65%	9.8	9.8
70%	13.6	9.5
75%	14.8	8.8
80%	15.4	8.2
85%	17.3	8.0
90%	20.4	7.8

Fig. 4: FAR and FRR values according to different authentication thresholds for sessions of length 100 curves and the corresponding ROC curve.

Table 2: EER values reported by Schulz [5].

#Curves	120	300	600	2400	3600
EER	20.6%	16.6%	13.8%	10.9%	11.1%

7 Conclusions

In this work we have proposed an approach to model the normal human-computer interaction via mouse device that can be used to control access to information assets, multimedia archives and personal devices by continuously authenticating the current user of the system. Our main contribution is introducing mouse-curve based features that satisfy certain mathematical properties related to task-independence. Experimental results of ten subjects achieved an EER of 5.3 percent. Future research will extend the subjects involved in the evaluation process and test consistency and invariance of the model over time. Besides, a statistical analysis will be useful to show the contribution of each feature in the final decision, which can also help us to know how comprehensive the features set is.

Acknowledgments This publication has emanated from research conducted with the financial support of Science Foundation Ireland (SFI) under grant number SFI/12/RC/2289. We would like to express our deep gratitude to Dr. Ammar Joukhadar and Mr. Khaldoon Ghanem for their valuable support and constructive recommendations on this project.

References

[1] Dobromir Todorov, ed. *Mechanics of User Identification and Authentication: Fundamentals of Identity Management*. Auerbach Publications, 2007.

[2] Maja Pusara and Carla E. Brodley. "User Re-authentication via Mouse Movements". In: *Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security*. USA: ACM, 2004, pp. 1–8.

[3] Ahmed Awad E. Ahmed and Issa Traore. "A New Biometric Technology Based on Mouse Dynamics". In: *IEEE Trans. Dependable Sec. Comput.* 4.3 (2007), pp. 165–179.

[4] Y. Nakkabi, I. Traore, and A.A.E. Ahmed. "Improving Mouse Dynamics Biometric Performance Using Variance Reduction via Extractors With Separate Features". In: *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on* 40.6 (Nov. 2010), pp. 1345–1353.

[5] D.A. Schulz. "Mouse Curve Biometrics". In: *Biometric Consortium Conference, 2006 Biometrics Symposium: Special Session on Research at the*. Sept. 2006, pp. 1–6.

[6] Dorothy E. Denning. "An Intrusion-Detection Model". In: *IEEE Trans. Softw. Eng.* 13.2 (Feb. 1987), pp. 222–232.

[7] Kenichi Hayashi, Eiji Okamoto, and Masahiro Mambo. "Proposal of User Identification Scheme Using Mouse". In: *Proceedings of the First International Conference on Information and Communication Security*. ICICS '97. London, UK, UK: Springer-Verlag, 1997, pp. 144–148.

[8] Zach Jorgensen and Ting Yu. "On Mouse Dynamics As a Behavioral Biometric for Authentication". In: *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*. ASIACCS '11. New York, NY, USA: ACM, 2011, pp. 476–482.

[9] Sotiris Kotsiantis and Dimitris Kanellopoulos. *Discretization Techniques: A recent survey*. 2006.