Keystroke Analysis for Remote Authentication

Gabriel Padis
School of Computing
Dublin City University
gabriel.padis2@mail.dcu.ie

Rim Zaafouri
School of Computing
Dublin City University
rim.zaafouri2@mail.dcu.ie

Dr Darragh O'Brien
Supervisor
School of Computing
Dublin City University

Date of publication: October 1, 2020

Abstract
Keystroke dynamics are used as a security method for authentication as a measure to reduce cyberattacks. By extracting data on a user's typing biometrics on a keyboard, the authentication system is able to recognize them as the legitimate user and therefore authorize the access to a device, a website or an application. Respectively, it blocks the authorization if the authentication system judges the user to be an intruder. While currently being an authentication system used on local machines, this tool has not yet been commonly researched or implemented over the network. This paper discusses the observation of the impact of network simulations on different machine learning keystroke authentication models. It covers both a reimplementation of key dynamics authentication on a local host machine with a stable network, and an evaluation of this authentication tool over a network under different unstable conditions.

I. Introduction

Real-time keystroke analysis is a dynamic biometric based authentication approach used to limit a user's access to an application or a database. It is an extra security feature during authentication; in the case of an intruder trying to obtain access with compromised or stolen credentials against a login form. The typing patterns of the intruder would not correspond to the genuine user's, and the hack attempt would be thwarted. Keystroke dynamics are a type of behavioural biometrics that measure attributes of a user including latencies between keystrokes, keystroke durations, finger placement and pressure, typing speed, and the order in which the keys are pressed. This technology is a non-invasive and user-friendly approach to reinforce the security of systems, as it requires no specific hardware equipment other than a keyboard. The authentication could be done at an entry-level when taking into account the password, or as a continuous verification when taking into account the typing behaviour of a user.

The applicability of keystroke dynamics is generally assessed for detecting intruders on a local machine. The research conducted here evaluates how using keystroke dynamics over the network impacts the quality of the detection made by different machine learning algorithms. Results from a password typed over an SSH connection will be compared in order to authenticate the incoming connection request. Multiple algorithms applied on the same dataset are compared to see how different network conditions can influence the data that is received and how the detectors are impacted in a real world application of keystroke dynamics for static authentication. The experiments described aim to show how detectors react to various network conditions, and which methods are to be preferred to have better results when using keystroke analysis for remote authentication.

A review of previous work in the area is first presented in a literature review in section II. The topic of remote authentication with keystroke dynamics is then defined in section III, as are the algorithms that are used. The study then presents the research and examines how the experiment was created with its features and limits in section IV. In section V, the results obtained from the research are compared, followed by a final overview and an interpretation of this authentication method over the network in section VI.

II. Literature review

Static keystroke dynamics is the study of fixed static data used in the purpose of authentication, such as passwords. Continuous authentication is the process of continuously identifying a user on a long period of time even after an initial authentication process has occurred.

A. *Static Authentication*

A survey in keystroke dynamics for biometric authentication [12] discusses the general work done in the area of keystroke dynamics. It describes the concept of biometrics which is a pattern recognition scheme that limits a user's access to applications or systems. Keystroke dynamics are behavioural biometrics collected as timing data from the user as it presses and releases the keys of a computer keyboard. This survey exposes the various methods of keystroke dynamics capturing. These methods include patents, statistical analysis, neural networks, fuzzy logic, data mining, and mobile phone keystroke analysis. The paper also gives an insight on the datasets collected for keystroke dynamics studies.

Monrose and Rubin [3] wrote the first paper in English and the most cited paper in 1990, on the topic of keystroke dynamics in a security context for user authentication. It is also one of the first to be written in English in 1990. It describes keystrokes as a biometric for identity verification, on the same par as hands, fingerprints and eyes. The pattern that is observed is how each user types, not what they type. From it the different steps of representation, extraction, classification and identification are defined. They used latency in digraphs as the single feature. They extended the research on fixed authentication using covariance matrix of latencies. They implemented many classifiers (Euclidean, Weighted probability, Bayesian, . . .) and calculated the Equal Error Rate (EER). They had their best result with the Bayesian detector with a 92% correct identification rate.

In a 2002 paper [5], the distance between two samples was computed using the relative positions of sample trigraphs. In this study, the samples were not thrown away because of typing errors, which increased the collected number of trigraphs. According to the number of samples given, the users were classified in 4 different categories. Users who had given 4 samples had a 100% accuracy rate, while users who had given just 1 sample had a 97% accuracy rate. As a result, a 4% False Alarm Rate and an Impostor Pass Rate of less than 0.01% were achieved.

Killourhy & Maxion wrote a paper [7] where they collected a keystroke-dynamics dataset, to develop an evaluation procedure, and to compare the performances of a range of different authentication algorithms. The literature is fragmented with each author using their own dataset, with their own metrics to compare different algorithms. The authors here provide a simple way to compare the different methods on a common set of data. The first step of this research was to collect a sample of keystroke-timing data. The second step was to implement 14 anomaly-detection algorithms from other authors that analyze the password-timing data, including algorithms of Manhattan, Euclidean, and Mahalanobis, neural network, fuzzy-logic, etc. The best equal-error rate was 0.096, obtained by the Manhattan detector, and the best zero-miss false-alarm rate was 0.468, obtained by the Nearest Neighbor with Mahalanobis distance detector.

In a research paper [9] referencing Killourhy & Maxion [7], the authors propose a new distance metric that combines the benefits of two classifiers. The new distance metric ensures that undesirable correlation and scale variations are accounted for, and that the influence of outliers is suppressed. As a result of conducted experiments using the nearest neighbor classifier, the new proposed distance metric was proven to improve the accuracy of keystroke dynamics using static text, as it reduced the EER to 8.7%.

Researchers [15] used Killourhy & Maxion's dataset [7], with different timings of press and release of the keys were captured and used as feature. The data was then evaluated by the statistical method of Support Vectors Machine (SVM) and by a deep learning model, whose architecture is not specified. The accuracy of the results was then analyzed: deep learning (92.6% accuracy) outperforms SVM (71.15% accuracy). This shows that user authentication may achieve better performance using deep learning, specifically using an optimizer NADAM for higher accuracy.

It is also possible to identify a legitimate user on their phone when the PIN is entered [8], by logging mobile keystroke in an application. It runs in the background so that it isn't intrusive to the user, and it logs all the keys pressed by the user along with the press and release times of the keys. A special dataset was collected. The features selected for the analysis of the data were key hold time, digraph time, and error rate. The data was then analyzed with 5 existing algorithms for evaluation, and then implemented with a proposed tri-mode system for identification which includes a learning mode using a Feed-Forward Fuzzy Classifier, a detection mode using dynamic optimizers like Particle Swarm Optimization, and a verification mode. As a result, the proposed system performed with an error rate of less than 2% and a False Rejection Rate of close to 0 after the verification mode.

B. Continuous Authentication

Patrick Bours and Soumik Mondal used Reinforcement learning algorithms in 2015 [10] to implement continuous authentication. The focus is aimed at two main requirements: the first is to not interrupt the user in their daily activity, and the second is for the system to use each keystroke to determine the genuineness of the user. They defined a trust model which evaluates the trust of the user based on a behaviour comparison of the current user with a template of the genuine user. The trust model uses key digraph classification protocols for the verification process, which include scaled Euclidean distance and correlation distance measurements. Depending on the retrieved results from the measurements, the trust model increases the system's trust (called Reward), or decreases it (Penalty). The results are classified in different categories: "Very Good" (all impostors are detected and the genuine user is never locked out), "Good" (the genuine user is never locked out but not all impostors were detected or the genuine user is locked out but all impostors are detected), "Bad" (the genuine user is locked out and not all impostors are detected) and "Ugly" (the false detection is drastically higher than the accurate one in all cases). Most of their results fall in the "Very Good" or "Good" categories, meaning their study was rather successful.

It is possible to consider a typing behaviour as a form of time series [14] while comparing the sample to a template, instead of using typing vector that requires to have a large amount of data, especially over time in a continuous authentication system. It underlines the fact that for big number of n-graphs there is a need of big sets of data per user so as to not have any holes in the n-graph possibilities. It could be completed with Neural Network for the missing values, but has not proven greatly successful. Digraph are more common so more data means more precision. To construct a template for each person, they only used digraphs as features. A subsequence consists of n time points, and a profile is a set of subsequences. They used a Dynamic Time Wrapping method to find similarities between points of subsequences, where a matrix of minimum warping distance is created, and its sum is the warping path. The smaller it is, the closest the two samples are. They used a similarity threshold taken from the average of the profile warping distance. They obtained a False Match Rate (False Acceptance Rate) of 0.54% and a False Non Match Rate (False Rejection Rate) of 1.64%. It is better than their comparison to a vector approach, respectively 8% and 6%.

In 2016, researchers applied a Gaussian Mixture Model (GMM) on raw keystroke data from 157 subjects [11]. It uses the mean and the standard deviation of two consecutive keystrokes digraphs. Their results show that using a GMM gives a EER of 0.39% whereas using a Gaussian Distribution yields an EER of 0.01%. It also shows their dataset has a greater precision that the one available previously with a GMM EER of 0.08% and a Gaussian distribution EER of 1.3%.

Encouraging results have been obtained by Daniel Gunetti and Claudia Picardi in 2005 [6]. They tried to authenticate personal identities of 205 individuals with a False Alarm Rate below 5% and and Imposter Pass Rate of less than 0.005%. They used pressed time for duration of n-graph, the time between pressed time of first key and the nth. The mean of each n-graph was calculated for the typed samples. Each n-graph are independent from the text and thus can be compared together. They calculated the distances of two typing samples using n-graphs with n going from 2 to 4.

KDE is a non-parametric way to estimate the probability density function (PDF) of a random variable. It needs 4 or more instances in sample and in digraph timings profile to have a good enough average after calculating their probability density. Its EER is slightly better than Gunetti & Picardi's method [6] on the tested datasets (0.04095% against 0.055225% EER) [13]. But KDE is better in truly uncontrolled environment with a stable and consistent performance, especially in noisy datasets.

C. Limits and Research Goals

There are many variations in keystroke dynamics analysis depending on formats, purposes, methods, measures of success, datasets and supports. Indeed the research papers currently existing are quite fragmented in their results, not all datasets are built the same way and are available to the public. Prior to 2009, almost all researchers collected their own data and created their own datasets for study purposes, until the 2009 Dependable Systems and Networks conference, where a requirement for common datasets was set in place [12].

Implementation can vary and is rarely discussed. Metrics for evaluation are also not consistent across the paper to evaluate the efficiency of an algorithm, even by using different names for the same metrics: FAR, FRR, FPR, FNR, EER, CER, ROC curves, Sensitivy, Recall, Precision, ... Dataset features can also be redundant and can reduce the overall capability of some detectors [7]. Outliers create variance and some researchers remove them from their data so that it does not impact the performance of their detectors [2]. Ensemble methods (like Bagging, Random Forest, ...) and Neural Network seem to yield better results [12]. It is also remarked of the difficulty of free text analysis due to the fact that it is less precise and harder to have a good model for [3]. None of the algorithms found in the literature have performance good enough to be accepted under the European Standard EN 50133-1, specifying that the false alarm rate must be under 1% and that the miss rate should not exceed 0.001% [16].

We evaluate how feature change and engineering might prove useful for an over the network case, while taking into account that the primary purpose is high security identification. As the context of application is network security, the number and complexity of features at our disposal is restricted. Using just latency as a feature for authentication is possible [3], it is a good comparison point to see how feature engineering may improve the results found.

Over the network is defined here as over an SSH connection. SSH is an application layer protocol that allows to setup a secured communication channel between two hosts to operate network services. It includes a login and upon successful authentication, remote command execution.

By sitting on the network an analysis of it is possible. In Dawn Xiaodong Song, David Wagner and Xuqing Tian paper in 2001 [4], they described how SSH packages can leak approximate size of data due to padding issues and how every individual keystroke that a user types is sent to the remote machine in a separate IP packet after the key is pressed. This leaks the keystroke timings and allows to do a Gaussian statistical approach to analyze the users typing rhythms. They then used a Hidden Markov Model and the n-Viterbi Algorithm to try to reduce the time it would take to brute force a password by eavesdropping on the network.

The paper [2] introduced two algorithms which were reproduced later by Killhoury and Maxion in their research [7]. They describe a simple k-Nearest Neighbor algorithm to compare it to their neural network, an autoassociative multilayer perceptron. They found that the neural network outperformed the nearest neighbor algorithm, but Killhoury and Maxion found the opposite. This could be due to the fact that Cho and al. discarded outliers in their dataset. We will use the same method of anomaly comparison of detectors for password authentication to re-implement these two methods and observe their performance in our test cases.

III. Keystroke Analysis for Remote Authentication

A. *Method*

Static keystroke dynamic authentication on a password is a problem of distinguishing the owner of the password to an impostor trying to hijack the account. The system will know the genuine user's keystroke pattern and will be able to authenticate it, detecting an impostor is an anomaly detection problem. We have a dataset of a user behaviour and we want to identify a new connection as a genuine (inlier) pattern or as an impostor (outlier) pattern. It is a one-class anomaly detection problem and not a multi-class classification of all the typing patterns into distinguishable users. Outlier detection algorithms are usually trained with data on both inlier and outlier points, but in this authentication case it is impossible to train the models on how a hacker might type a user's password without posing security issues, so the algorithm is not trained with outlier data points, it needs to compare the new observation only to the genuine user's keystroke dynamics, it is called a Novelty Detection algorithm. To do so the detection model will output for each password keystroke dynamics an anomaly score, which will be compared to a threshold and if the anomaly score is higher than the threshold then it is classified as an outlier. From the resulting classification multiple metrics can be extracted to evaluate the quality of the detection of the detectors. Comparison between detectors is then possible and a better performer in multiple situations emerge.

The research itself evaluates how the detector capabilities metrics vary when the data is simulated through an SSH connection. Each typing pattern can be played over an SSH connection, and the network conditions it which it happens will influence the resulting data and the models. The paper from Killourhy and Maxion [7] is used as a reference, one of the main reasons being that they provide the dataset on which they did their research. They did a meta analysis of detectors on the same dataset that they built, and they used two metrics to compare the algorithms evaluated on their dataset: Equal Error Rate and Zero Miss False Alarm Rate.

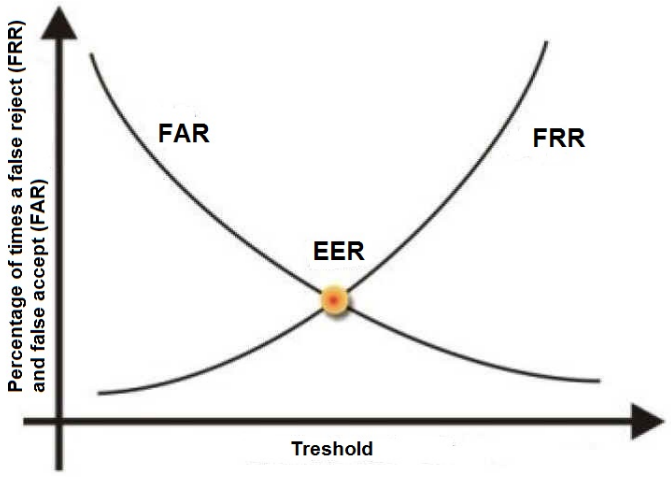


Fig. 1. Equal Error Rate (from Recogtech)

In Fig1 we can see how the Equal Error Rate (EER) is used in a biometric system for authentication, at any given threshold there will be a False Acceptance Rate (FAR) and a False Rejection Rate (FRR). These metrics depend on the chosen Null Hypothesis, here: an impostor cannot be distinguished from the genuine user based on its keystroke dynamics. The FAR represents the percentage of detection where unauthorised users are incorrectly accepted and the FRR represents the percentage of detection where authorised users are incorrectly rejected.

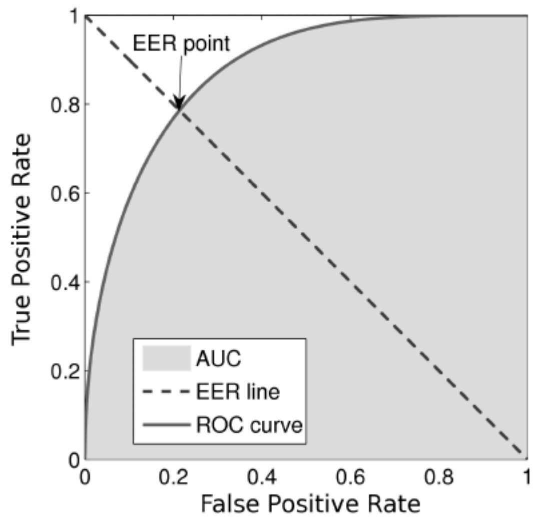


Fig. 2. ROC Cruve with Equal Error Rate (from ResearchGate)

In Fig2 we can see how the same EER is calculated based on a Receiver Operating Characteristic (ROC) curve. This curve represents the capabilities of classification of a detector based on its results compared to the ground truth. These results can be visualized with the help of a confusion matrix.

A ROC curve is dependent on the True Positive Rate (TPR) and the False Positive Rate (FPR). It is important to note the the TPR and FPR are also known as the Hit Rate (and Recall) and the Miss Alarm Rate respectively. In the classification problem, it is established that the condition positive (1) is the password's keystrokes are of an impostor, and that the condition negative (0) is the password's keystroke are of a genuine user. Under this classification model, the Hit Rate (TPR) is the frequency with which impostors correctly are detected, it is equal to 1 - Miss Rate (also called False Negative Rate (FNR)) that represents the percentage of impostor passwords that are not detected. The False Alarm Rate (FPR) is the frequency with which genuine users are mistakenly detected as impostors.

In this paper we are reimplementing two algorithms from the literature to compare our results and see how doing an authentication over the network is impacting the detectors' capabilities. The two algorithms were chosen so that they would be different enough to not react in the same way when impacted by network changes on the simulated connections. The first one is the k-Nearest Neighbor algorithm which uses the feature space to compare points to each other in order to find for a new point the k closest other points to decide the value it will be classified as. The second one is an autoassociative multilayer perceptron, a neural network which uses the input as a target to be reproduced, the distance between the input and output will serve to calculate its class.

B. Keystroke features

Since keystroke dynamics are the patterns of rhythm created when someone types on a computer, researchers use various different methods when sampling keystrokes to define the characteristics of keystroke dynamics. There are three that are most commonly used: Up-Down, Hold, and Down-Down. Up-Down means that the timing that is saved is the one between the moment when a key is lifted, and the moment when the next key is pressed. Hold means the time duration in which a key is pressed for, starting when a user pushes, and finishing when they lift their finger. Down-Down is the timing between the pressing of a key and the next one. Intuitively, Down-Down would be the sum of Hold and Up-Down.

C. Dataset Used

The dataset used for the study is the same dataset used in the referenced study of Killourhy and Maxion [7], as it is public, anonymous, and available for usage. The data collection methodology for that dataset consisted of recording keystrokes of subjects using a software application that presents the password to be typed and checks its correctness. The software records the time and the name of a key whenever a user presses and releases on it. In our case, only the Down-Down timing features will be used from the dataset.

The dataset has 400 typing patterns of the password per user, and there are 51 users in total. For each user, the tested detector was trained then tested with genuine and impostor user typing patterns. The relevant metrics were then calculated (EER and Zero Miss False Alarm Rate). For each detector, the average and standard deviations of the users were computed for each metric. The training dataset is composed of the first 200 typing patterns for each user. The test set is composed of two parts, the first is a test compared to the user's own typing patterns, the remaining 200 typing patterns of the user, and the second part is composed of the first 5 typing patterns of the other 50 subjects, they are impostor typing patterns.

D. Anomaly Score

During the testing phase, the output for each typing pattern is an anomaly score. We consider that this anomaly score follows a continuous random variable, and it is compared to a threshold. If the score is strictly superior to the threshold then it is classified as "positive" meaning it is considered an outlier, else it is inferior to the threshold and will be considered "negative". There are many threshold that are used during this evaluation, this is to have a maximum of points to draw the ROC Curve and determine the EER and Zero Miss False Alarm Rate.

E. Metrics

1) Equal Error Rate: The Equal Error Rate is for a chosen threshold when the False Acceptance Rate, here the Miss Rate, is equal to the False Rejection Rate, here the False Alarm Rate. It is an arbitrary metric, the closest it is to zero the better the performance is.
$EER = FPR(t)$ where t is computed as $FPR(t) = TPR(t)$

2) Zero Miss False Alarm Rate: The Zero Miss False Alarm Rate (ZMFAR) happens when the threshold is chosen so that the False Alarm Rate is minimum when the Miss Rate is equal to 0. It represents the percentage of genuine users that are considered impostor when none of the impostor are not detected.

F. k-Nearest Neighbor with Mahalanobis Distance

The k-Nearest-Neighbor algorithm represents the feature space with the number of dimensions being the number of features. During the training phase, the model will note the position of each typing patterns. During the testing phase, for a given test typing vector it will be placed in the feature space and find the k closest points to it. k is equal to 1 [7][2], the model looks for the closest vector to the test vector. The distance between the two points will be the anomaly score. This distance is the Mahalanobis distance between the two vectors, which uses the features' covariance matrix.

G. Neural Network

An autoassociative feed-forward multilayer perceptron with back propagation [2][1], is also called an autoencoder neural network. We could not find why the two names existed but the second one is more popular and we were able to find more documentation with this naming. An Autoencoder works by having as many input and output nodes as they are features in the data. It is constituted of one or more hidden layers, each with their own number of nodes. Usually the number of hidden nodes is less than the number of input/output nodes, it is useful for cases of compression algorithm and is called an undercomplete autoencoder. The literature[2][1] used a single hidden layer with the same number of nodes as the input/output. The goal of an autoencoder is to learn the underlying representation of the input vector to produce a condensed output vector. For typing behaviours, when training an autoencoder on genuine inputs it will learn to output vectors close to the originals. During the testing phase with impostor inputs, the autoencoder will output timing vectors that are far away for the input ones so the difference between the input and output will be greater. It is the difference, as an euclidean distance, between the input and output vectors that becomes the anomaly score which will be tested against a threshold to define the EER and ZMFAR. This discrimination of abnormal behaviours from good behaviours is possible due to having a bounded activation function such as step and by minimizing the error function during the training session. A ”normal” typing behaviour will have a small generalization error, so on the contrary a large error will probably come from an abnormal typing behaviour.

H. *The SSH connection*

SSH is a protocol that allows to securely transmit data between two machines. For that purpose, it enforces an authentication when a connection is trying to be established. The SSH protocol breaks the data down into packets through a reliable transport between a server and a client, usually through a TCP connection. A packet is composed of headers and of an encrypted payload with padding. The rest of the packets are encrypted continuously using vectors from the preceding packet. A SSH connection will open a channel that enable a connection to send the data between the two machines, allowing a reliable and secure communication.

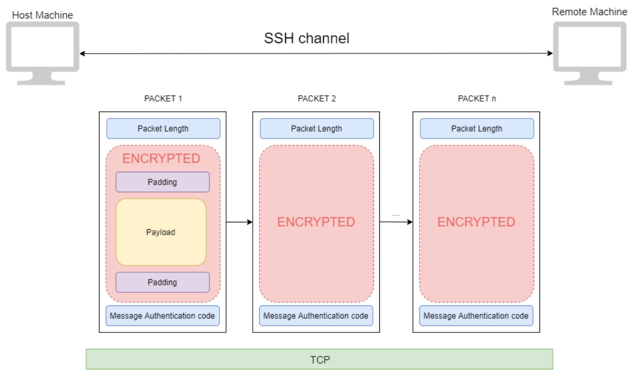


Fig. 3. SSH connection between host and remote machines

During the authentication, an SSH connection works in interactive mode by sending an IP packet immediately after the key is pressed, and the delay that the operating system has before sending the IP packet is negligible compared to the timing between two keystrokes [4].

It is thus possible to create an SSH connection through which the dataset keystrokes are sent to a remote machine. It is possible to receive each typing patterns packet per packet, the result being a dataset where network delay is significant. Therefore, packets will be sent through the secured channel simulating the key input of a singular keystroke, where its timing is the differences between the features. On the receiver end, the new dataset is composed of the received deltas between the key inputs, which will then be analyzed by the k-Nearest Neighbor and Neural Network algorithms.

However, all the features cannot be preserved as the collectable information is not the same when gathered on the local machine as opposed to when it is received through an SSH connection. Since the packet is sent directly after the SSH key is pressed, only the Down-Down keystroke dynamics are usable. Up-Down and Hold keystrokes are not available or deductible when the program is not running on the users' computer.

We observe the changes and draw conclusions from the effect that this kind of implementation has on different models performance for authentication over the network and the issues it raises.

IV. Implementation

A. *Technologies*

For our experiments and our implementation we used Python 3 with machine learning packages like scikit-learn for the k-Nearest Neighbor and TensorFlow with Keras for the autoencoder neural network. Since the technologies we used differ from the literature, which used R [7], small differences in global results may arise. Other parameters like the hardware may have an influence on the results.

B. *Reimplementation of the Algorithms*

Our steps in order are: clean the dataset of unwanted columns, create all the subjects with a specific detector, train the detector for the subject, test it, evaluate it. Then average the results of the equal error rate and the zero-miss false alarm rate. As explained previously, for each user, the first 200 typing patterns are used to train the detector, the other 200 are used to test with the first five typing patterns of every other user.

The hyper-parameters used in the two algorithms are the one stated in the literature by the author to obtain comparable results to the literature. Using cross validation techniques to determine the best hyper-parameters is not the goal.

- k-Nearest Neighbor:
  To create a scikit kNN detector, it is specified that only one (the closest), so k = 1, point will be taken in consideration when a new typing pattern arrives, the

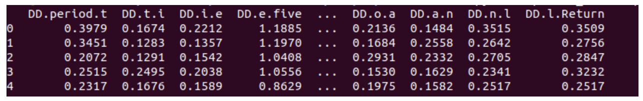
metric to use is ”mahalanobis”, to which one must specify the covariance matrix of the features for the user, and its inverse. Here the brute force algorithm is used, neither ”ball_tree”, nor ”kd_tree” worked in this case and errors happened during calculations which returned ”NaN”. See Appendix A.
To calculate the distance of new typing patterns to the ones already existing inside of the nearest neighbor, it looks for the k closest neighbors in the detector. The score is the mahalanobis distance and the index is the position of the closest point in the detector. See Appendix B.

- Autoencoder Neural Network:
  As per the literature the Neural Network has the same number of nodes as input, hidden and outputs which is the number of features, and the activation function of the hidden layer is a sigmoid. It uses the stochastic gradient descent with a learning rate of 0.0001 and a momentum of 0.0003 as an optimizer and the mean square error function as its loss function. When training the detector on the user’s typing pattern it has 500 epochs and uses its inputs as target outputs, it is what makes it an Autoencoder. See Appendix C.
  To calculate the distance of a test point, the Neural Network predict a set of typing patterns for the one being tested, it is then the euclidean distance between the original test point and the predicted one that is used to determine the quality of the detector. See Appendix D.

Once the scores for the user are gathered, the evaluation begins. Each distance must be evaluated to know if it is an outlier or not, depending on a threshold. This threshold varies from the smallest possible value to the highest possible value, for each of the user’s distances if the distance is superior to the threshold then it is an outlier, the same applies to the impostors’ distances. The predicted values are booleans (1 is outlier, 0 is an inlier). From the true values and these predicted values a confusion matrix can be created, out of which the False Positive Rate (False Alarm Rate) and the True Positive Rate (Hit Rate) can be determined. Since they are multiple thresholds, a lot of FPRs and TPRs are obtained, allowing us to draw a ROC curve for the user. From theses values the Equal Error Rate and the Zero Miss False Alarm Rate can determined.

C. Application over the Network

After implementing the algorithms, the focus is on the transmission of the packets over the network, simulating the keyboard inputs. To send the packets, the public dataset is used to provide each character typed in by the user sent as an individual packet, creating a new dataset with the received packets. The dataset is first loaded and filtered in a Python script leaving only the Down-Down key inputs. The table then holds the respective Down-Down data of the 400 passwords typed by the 51 users:



```
   DD.period.t  DD.t.i  DD.i.e  DD.e.five  ...  DD.o.a  DD.a.n  DD.n.l  DD.l.Return
0       0.3979  0.1674  0.2212     1.1885  ...  0.2136  0.1484  0.3515       0.3509
1       0.3451  0.1283  0.1357     1.1970  ...  0.1684  0.2558  0.2642       0.2756
2       0.2072  0.1291  0.1542     1.0408  ...  0.2931  0.2332  0.2705       0.2847
3       0.2515  0.2495  0.2038     1.0556  ...  0.1530  0.1629  0.2341       0.3232
4       0.2317  0.1676  0.1589     0.8629  ...  0.1975  0.1582  0.2517       0.2517
```

Fig. 4. Down-down keys used as deltas for Packets

Each one of these deltas is used as the time of delay between the transmission of two packets. Therefore, the wait time between two packets represents the time the user takes to press between a key and another, simulating their keyboard input speed.

A TCP connection is created between the host and the server. The local host acts as the packet sender and the server as the receiver. Although an SSH connection was initially planned, a simple TCP connection is set up here for the sake of simplifying the code and to get faster results. Indeed, an SSH traffic capture from Wireshark would have required an additional implementation. The packet sender goes through the table, storing each value as the wait time. After a packet gets sent, the server sleeps with its respective wait time, and then sends the following packet. On the receiver end, once the connection is initiated to bind the sockets to the local host server, it listens for the messages sent by the host. The receiver creates a new dataframe to store the new timing deltas. A timer is started when the connection is initiated. Once a message is received by the client, the delta between the previous time and the time of the reception of the message is measured and stored. The received time values are then appended creating the columns and rows of the new dataset. The dataset has the same number of features as the filtered dataset sent by the host.

At the end of the reception of all packets, the resulting dataframe with the new deltas is then exported and saved. The new data will then be analysed with the machine learning algorithms that were implemented as explained previously.

D. Network Simulation in Different Conditions

To see how the detectors would perform if they were implemented as security features on servers for authentication with an SSH connection it is needed to simulate multiple network conditions. To do so one can use real servers in different locations by acquiring or renting cloud services, but this incur problems relative to the control of the conditions in which the experiment operates, delay can only be inferred, packets can take different routes and/or can be lost, payload can be corrupted, down issues and outages can happen.

Instead, a network simulator like GNS3 can be used, it emulates real and virtual devices. From it, it is possible to draw a simple diagram representing two machines connected to the internet that have an SSH connection. The hosts can exchange IP packets like an SSH connection would during authentication and it is possible to handle more parameters. Packets will not be dropped, hosts cannot be down, and it is possible to control the delay over the network with two variables: latency and jitter. Latency is the time delay between the two hosts that is not dependent on the physical action of propagation, jitter is the variation from the previous latency.

Once the entire dataset of values passed through the network, a new dataset is created from which the network has had an influenced. The shift in keystroke timings is now dependent on the network conditions and the variations of the detectors performance can be related to the network conditions.

Different values were tested with this setup compared to the original data (with Down-Down keystrokes only):

- Localhost (0ms latency, 0ms jitter): this simulation is to see the change that the setup induce when compared to the original dataset, it is a control group.
- 10ms latency (0ms jitter and 5ms jitter): when simulating the network conditions during authentication tests were run for each latency value with no extra jitter in the network and with half of the latency as jitter.
- 25ms latency (0ms jitter and 12ms jitter)
- 50ms latency (0ms jitter and 25ms jitter)
- 100ms latency (0ms jitter and 50ms jitter)
- 250ms latency (0ms jitter and 125ms jitter)
- 500ms latency (0ms jitter and 250ms jitter)

This gives us 13 datasets that can be compared between each other. The average keystroke lasts 0.25s so it took more than 14 hours of uninterrupted runtime to obtain each dataset.

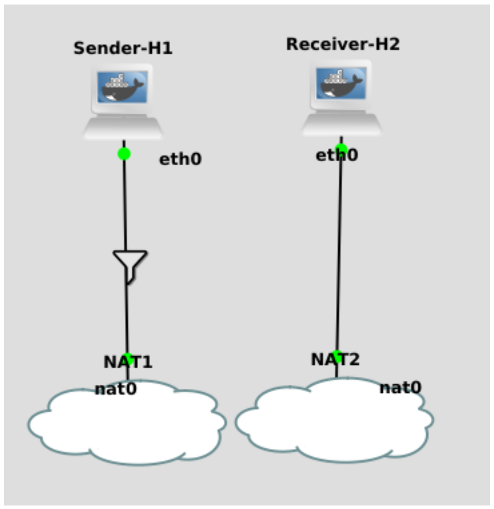


Fig. 5. GNS3 configuration

As Fig.5 shows, host 1 is the sender of data, it is connected to NAT1, on its connection is the packet filter tool that allows the mastery of the parameters of the simulation, data is then routed through the NATs and it goes out of NAT2 to finish in host 2, the receiver.

E. *Implementation in real life*

The mechanism of the keystroke database detection algorithms resembles the one of Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS), which typically compares network packets to an existing database. As opposed to an IPS, the keystroke algorithms do not actively block the intrusion but they rather scan the incoming traffic and alert the system in case of an anomaly detection, in this case, an imposter typing in the password. This would make the system closer to being an IDS than an IPS. This system has to be deployed over a network where the management of devices is possible.

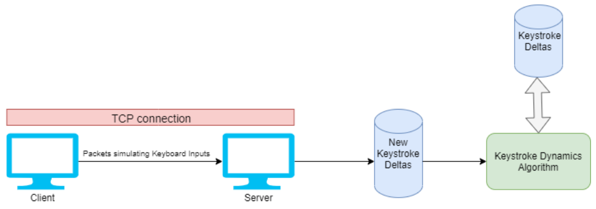

Keystroke Deltas
TCP connection
Packets simulating Keyboard Inputs
New Keystroke Deltas
Keystroke Dynamics Algorithm
Client
Server

Fig. 6. The keystroke dynamics IDS

Furthermore, as of now, the implemented algorithms work as a server-based inclusion detection system as it runs as a software on a critical resource, the server machine. The IDS here analyzes the activity on its own system and uses an external database as logs for detecting anomalies.

For further study, this keystroke dynamics IDS could be integrated in devices that are already used in a network, for instance a switch, a router, or a firewall. The system is then network-based, and would be able to see and analyze network traffic in real-time. It would also be possible to consider implementing an IPS or even make it complimentary to the existing SSH technology.

V. Results Obtained

The literature states the following values for the metrics of the tested algorithms [7]. These metrics are calculated using all the features of the dataset, in table I:

Detector	EER mean	EER std	ZMFAR mean	ZMFAR std
k-Nearest Neighbor	0.100	0.064	0.468	0.272
Neural Network	0.161	0.080	0.859	0.220

TABLE I
Original metrics table

When building the machine learning, models are both retrained and tested with the network impacted data. The results for the algorithms using only Down-Down features of the dataset, in table II:

Detector	EER mean	EER std	ZMFAR mean	ZMFAR std
k-Nearest Neighbor	0.142	0.078	0.793	0.290
Neural Network	0.170	0.117	0.753	0.314

TABLE II
DOWN-DOWN FEATURE METRICS TABLE

It is observed that the performance decreases on all metrics, except the ZMFAR of the Neural Network that decreases by 0.100. It can be deducted that using only Down-Down features has an adverse effect on the detectors even though it does not have a strong impact. The difference could also come from our implementation of the algorithms (which can be found here). Unfortunately we were not able to test the implementation of the detectors in the same conditions to observe the impact of our implementation when using all of the features, as opposed to only using the Down-Down subset.

Before testing the detectors in network conditions where delay matters, the detectors were tested in the network simulator GNS3 without any latency or jitter to see the impact of the software, in table III.

Detector	EER mean	EER std	ZMFAR mean	ZMFAR std
k-Nearest Neighbor	0.142	0.078	0.799	0.291
Neural Network	0.172	0.120	0.756	0.308

TABLE III
NO DELAY GNS3 METRICS TABLE

Here metrics did vary but these changes are not significant. It can be concluded that simulating the network compared to using a localhost implementation does not badly impact the metrics and thus network simulation can take place without bias coming from the simulation software.

With network simulation at different latency and jitter the following results, in table IV, was obtained:

Detector	Latency (ms)	Jitter (ms)	EER mean	EER std	ZMFAR mean	ZMFAR std
k-Nearest Neighbor	0	0	0.142	0.078	0.799	0.291
	10	0	0.142	0.078	0.8	0.290
	10	5	0.141	0.078	0.775	0.281
	25	0	0.141	0.078	0.78	0.295
	25	12	0.142	0.080	0.792	0.271
	50	0	0.142	0.078	0.783	0.309
	50	25	0.151	0.078	0.898	0.195
	100	0	0.196	0.059	0.988	0.065
	100	50	0.220	0.064	0.954	0.112
	250	0	0.254	0.074	1.0	0.0
	250	125	0.271	0.085	1.0	0.0
	500	0	0.287	0.080	1.0	0.0
	500	250	0.278	0.083	1.0	0.0
Neural Network	0	0	0.172	0.120	0.756	0.308
	10	0	0.169	0.117	0.748	0.318
	10	5	0.174	0.120	0.757	0.300
	25	0	0.173	0.118	0.755	0.313
	25	12	0.171	0.117	0.749	0.314
	50	0	0.172	0.122	0.743	0.323
	50	25	0.171	0.118	0.754	0.308
	100	0	0.175	0.124	0.836	0.230
	100	50	0.177	0.119	0.813	0.243
	250	0	0.187	0.124	0.853	0.207
	250	125	0.216	0.115	0.877	0.134
	500	0	0.235	0.126	0.812	0.199
	500	250	0.261	0.110	0.956	0.081

TABLE IV
NETWORK DELAY METRICS TABLE

Performance of the equal error rate and zero miss false alarm rate metrics decrease as delay increase.

A. *Result Evaluation*

The previous results can be graphed to help comparison:

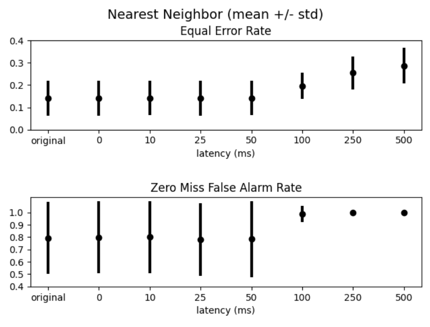


Fig. 7. k-Nearest Neighbor with Latency

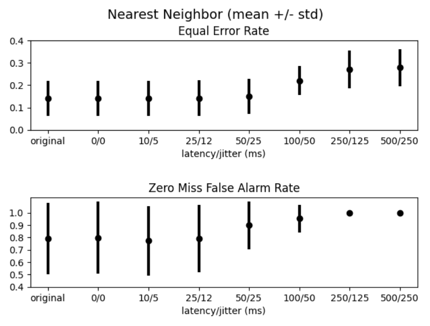


Fig. 8. k-Nearest Neighbor with Jitter

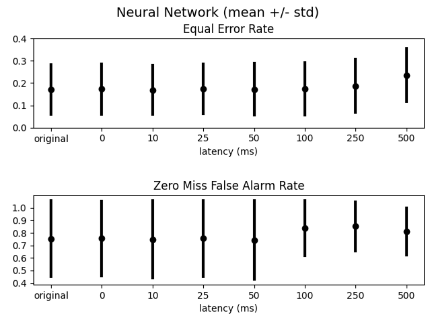


Fig. 9. Neural Network with Latency

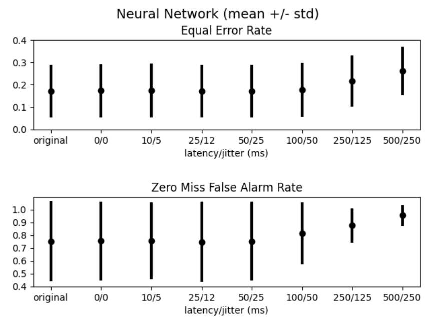


Fig. 10. Neural Network with Jitter

When comparing fig 7 with fig 8 and fig 9 with fig 10 it can be seen that jitter in delay does not seem to impact the average of the metrics, but it does increase the standard deviation quite significantly. Having a higher standard deviation shows that the user would be more impacted and have a less reliable system to use when trying to be authenticated.

It is also noticeable that even though the k-Nearest Neighbor detector starts with a better Equal Error Rate and Zero Miss False Alarm Rate, it deteriorates and end up at a point where detection capabilities are worse than the Neural Network detector at the same network conditions. It seems that an Auto-Associative Neural Network offers a better resistance to delay with latency and jitter than does a k-Nearest Neighbor algorithm. Having a more robust algorithm shows that it is able to be deployed in a greater variety of environment.

For example the k-Nearest Neighbor algorithm starting at 250ms of latency with 125ms of jitter has an Zero Miss False Alarm Rate of 1, it means that it is impossible to have a threshold to detect the impostors without considering at the same time some genuine users as impostors too. On the contrary, the Autoencoder Neural Network has its Zero Miss False Alarm Rate also increase from 0.756 to 0.956 but it does not goes up to 1, it is more stable than the k-Nearest Neighbor evolution from 0.799 to 1. The difference is more pronounced on the Equal Error Rate where the k-Nearest Neighbor goes from 0.142 to 0.248 whereas the Autoencoder only moves from 0.172 to 0.261.

From our experiments it is possible to conclude that not all algorithms will hold in the same way, and that it seems that a Neural Network provides better resistance to network latency and jitter. A remark can be made that performance seems to be stable until 50ms of latency with or without jitter, this is good news for network where some control is possible, such as a private network. When the delay is under this 50ms limit, it allows for some room in which the detectors can still work efficiently.

VI. Conclusion and Future Work

This conducted research demonstrates the feasibility of an application of keystroke dynamics for remote authentication over the network. We showed that the best detector in the literature, the k-Nearest Neighbor with Mahalanobis distance, is a method that is more sensitive to unsteady network conditions. Its performance decreases faster than the Autoencoder Neural Network method, which is more consistent in its performance even with high network delay.

While the quality of the detection decreases in networks with unstable conditions, the algorithms are still be able to authenticate the genuine users with an EER around 0.3, this measure is not unacceptable for a high security application. Detection quality decreases significantly after the 50ms latency limit, before that point performance are stable.

This study can be further improved by reimplementing new algorithms, especially other reported neural network algorithms that have promising metrics in the literature, and evaluating the resistance to network conditions, since our data shows that deep learning models seem to have more resistance to network noise. This would open the door to a more widely adaptable implementation of keystroke dynamics for authentication and remote authentication over the network.

Integrating this authentication system directly on network devices to analyze traffic in real time would also be a topic to explore. However, this would require the detection capabilities of the models used to drastically improve to match the European standard for access control systems. Once that standard rate is reached, keystroke dynamics for remote authentication over the network can become a normalized tool widely integrated in networks and commonly used for security enhancement.

References

[1] Byungho Hwang and Sungzoon Cho. “Characteristics of auto-associative MLP as a novelty detector”. In: *IJCNN’99. International Joint Conference on Neural Networks. Proceedings (Cat. No.99CH36339)* 5 (1999), 3086–3091 vol.5. DOI: 10.1109/IJCNN.1999.836051.

[2] Sungzoon Cho et al. "Web-Based Keystroke Dynamics Identity Verification Using Neural Network". In: *Journal of Organizational Computing and Electronic Commerce* 10 (2000), pp. 295–307. DOI: 10.1207/S15327744JOCE1004_07.

[3] Fabian Monrose and Aviel D. Rubin. "Keystroke dynamics as a biometric for authentication". In: *Future Generation Computer Systems* 16.4 (2000), pp. 351–359. ISSN: 0167-739X. DOI: https://doi.org/10.1016/S0167-739X(99)00059-X.

[4] Dawn Xiaodong Song, David Wagner, and Xuqing Tian. "Timing Analysis of Keystrokes and Timing Attacks on SSH". In: SSYM'01 (2001). DOI: 10.5555/1251327.1251352.

[5] Francesco Bergadano, Daniele Gunetti, and Claudia Picardi. "User Authentication through Keystroke Dynamics". In: *ACM Trans. Inf. Syst. Secur.* 5.4 (Nov. 2002), pp. 367–397. ISSN: 1094-9224. DOI: 10.1145/581271.581272.

[6] Daniele Gunetti and Claudia Picardi. "Keystroke Analysis of Free Text". In: *ACM Trans. Inf. Syst. Secur.* 8.3 (Aug. 2005), pp. 312–347. ISSN: 1094-9224. DOI: 10.1145/1085126.1085129.

[7] K. S. Killourhy and R. A. Maxion. "Comparing anomaly-detection algorithms for keystroke dynamics". In: *2009 IEEE/IFIP International Conference on Dependable Systems Networks* (Aug. 2009), pp. 125–134. DOI: 10.1109/DSN.2009.5270346.

[8] Saira Zahid et al. "Keystroke-Based User Identification on Smart Phones". In: *Recent Advances in Intrusion Detection* (2009). Ed. by Engin Kirda, Somesh Jha, and Davide Balzarotti, pp. 224–243. DOI: 10.1007/978-3-642-04342-0_12.

[9] Yu Zhong, Yunbin Deng, and Anil K. Jain. "Keystroke dynamics for user authentication". In: *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops* (2012), pp. 117–123. DOI: 10.1109/CVPRW.2012.6239225.

[10] Patrick Bours and Soumik Mondal. "Continuous Authentication with Keystroke Dynamics". In: (Jan. 2015). DOI: 10.13140/2.1.2642.5125.

[11] Yan Sun, Hayreddin Ceker, and Shambhu Upadhyaya. "Shared Keystroke Dataset for Continuous Authentication". In: *2016 IEEE International Workshop on Information Forensics and Security (WIFS)* (2016). DOI: 10.1109/WIFS.2016.7823894.

[12] Md Liakat Ali et al. "Keystroke Biometric Systems for User Authentication". In: *Journal of Signal Processing Systems* 86 (2017). DOI: 10.1007/s11265-016-1114-9.

[13] Chris Murphy et al. "Shared dataset on natural human-computer interaction to support continuous authentication research". In: *2017 IEEE International Joint Conference on Biometrics (IJCB)* (2017), pp. 525–530. DOI: 10.1109/BTAS.2017.8272738.

[14] Abdullah Alshehri, Frans Coenen, and Danushka Bollegala. "Iterative Keystroke Continuous Authentication: A Time Series Based Approach". In: *KI - Künstliche Intelligenz* 32 (2018). DOI: 10.1007/s13218-018-0526-z.

[15] Yohan Muliono, Hanry Ham, and Dion Darmawan. "Keystroke Dynamic Classification using Machine Learning for Password Authorization". In: *Procedia Computer Science* 135 (2018), pp. 564–569. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2018.08.209.

[16] European Committee for Electrotechnical Standardization (CENELEC). "European Standard EN 50133-1". In: ().

Appendix A
KNN Implementaion

```
self.nearest_neighbor = NearestNeighbors(n_neighbors=1, metric='mahalanobis',
    metric_params={'V': self.cov, 'VI': np.linalg.inv(self.cov)}, algorithm='brute')
```

Appendix B
KNN Distance Calculation

```
score, index = self.nearest_neighbor.kneighbors(self.x_test)
```

Appendix C
Neural Network Implemtatation

```
dim = self.x_train.shape[1]
input = Input(shape=(dim,))
hidden = Dense(dim, activation="sigmoid", kernel_initializer='random_normal',
    bias_initializer='zeros')(input)
output = Dense(dim, activation="linear", kernel_initializer='random_normal',
    bias_initializer='zeros')(hidden)
self.neural_network = Model(inputs=input, outputs=output)
opt = tf.keras.optimizers.SGD(learning_rate=0.0001, momentum=0.0003)
self.neural_network.compile(optimizer=opt, loss='mse')
self.neural_network.fit(self.x_train, self.x_train, epochs=500, verbose=0)
```

Appendix D
Neural Network Distance Calculation

```
predictions = self.neural_network.predict(self.x_test)
distance.euclidean(self.x_test.iloc[i], predictions[i])
```