AROONAV MISHRA

COMPARING ANOMALY DETECTION ALGORITHMS FOR KEYSTROKE DYNAMICS

UNDER THE GUIDANCE OF
PROF. PUSPANJALI MOHAPATRA

in partial fulfillment for the award of the degree of Bachelor in Technology in Computer Science & Engineering

Department of Computer Science & Engineering International Institute of Information Technology Bhubaneswar, India

WEBSITE:
aroonav.github.io

E-MAIL:
aroonav11@gmail.com

ABSTRACT

Legitimate user authentication is an important part of the problem related to the computer and system security. The maintenance of security becomes even more difficult when an invalid user gets the system access information. The variables that help make a handwritten signature a unique human identifier also provides a unique digital signature in the form of a stream of latency periods between keystrokes.

Keystroke dynamics - the analysis of typing rhythms to discriminate among users has been proposed for detecting impostors (i.e., both insiders and external attackers). Such a system will determine if the current user is the genuine one or not. The approaches use typing biometrics of a user during login to identify a user. We used a data-set of 51 subjects, each one typing a typical, strong password of 10 characters 400 times over multiple sessions. This thesis presents a suite of techniques for password authentication using statistical methods, neural networks and fuzzy logic.

ACKNOWLEDGEMENTS

First of all, I wish to thank our project guide Prof. Puspanjali Mohapatra for her invaluable guidance in this project, the detailed explanations, the patience and the precision in the suggestions. Second, I wish to thank my team members, Goutam Das and Asmi Pattnaik for their competence and kindness. My sincere thanks to all the people who have contributed to this thesis through their research and knowledge.

Department of Computer Science and Engineering

International Institute of Information Technology Bhubaneswar

Certificate

This is to certify that the dissertation entitled **Comparing Anomaly Detection Algorithms for Keystroke Dynamics** submitted by **Aroonav Mishra** is approved for the award of Degree of Bachelor of Technology in Computer Science & Engineering

Prof. Puspanjali Mohapatra
Project Guide

Date:

CONTENTS

1 INTRODUCTION

Compromised passwords and shared accounts are frequently exploited by both external attackers and insiders. External attackers test whether accounts use default or common passwords. Insiders compile lists of shared or compromised passwords for later use, e.g., to damage the system in the event that they are fired or demoted. If we had some means, other than knowledge of a password, with which to identify exactly who is logging into an account, and to discriminate between the genuine user of an account and an impostor, we could significantly curb these security threats.

One proposed approach is the use of keystroke dynamics - the analysis of typing rhythms to discriminate among users as a biometric identifier. Each person has an almost unique pattern of typing. This pattern can be learned and then can be used for identifying the user. With keystroke dynamics, impostor attempts to authenticate using a compromised password could be detected and rejected because their typing rhythms differ from those of the genuine user.

The thesis uses statistical techniques, neural networks and fuzzy logic to learn the typing behavior of a user. The objective is to verify the user based on the learned information and to compare the performance of these approaches. This is to drive progress toward better results. By establishing which anomaly-detection strategies outperform others, the research community gains insight into what detector characteristics reduce the error rate, identifying promising directions for further study.

Unfortunately, although researchers have conducted experiments to measure the performance of various anomaly detectors, these experimental results are impossible to compare. Too many factors vary from one evaluation to another. If we are to assess the state of the art in keystroke dynamics, and to measure future progress, we need a shared benchmark data set and a repeatable procedure for conducting evaluations. Only then can the error rates of anomaly detectors be properly measured and compared against one another.

The European standard for access-control systems (EN-50133-1) specifies a false-alarm rate of less than 1%, with a miss rate of no more than 0.001% [CENELEC, 2002]. At present, no anomaly detector has achieved these error rates in repeated evaluation, and so keystroke dynamics could not be deployed as a sole access-control technology.

2 LITERATURE SURVEY

Since Forsen[Forsen et al., 1977] first investigated in 1977 whether users could be distinguished by the way they type their names, many different techniques and uses for keystroke dynamics have been proposed.
Peacock[Peacock et al., 2004] conducted an extensive survey of the keystroke-dynamics literature. Not all of that research is relevant to the use considered in this work, namely, analyzing password-typing times. For instance, Gaines[Gaines et al., 1980] considered whether typists could be identified by analyzing keystroke times as they transcribed long passages of text. Techniques for analyzing passages are different from those for analyzing passwords, and the same evaluation data cannot always be used to assess both.

Further, even among studies of password-typing times, not all of the extant techniques can be evaluated using the same procedure. One class of techniques is *anomaly detection*. The typing samples of a single, genuine user are used to build (or train) a model of the user's typing behavior. When a new typing sample is presented, the detector tests the sample's similarity to the model, and outputs an anomaly score. In contrast, another class of techniques is *multi-class classification*. The typing samples of multiple users are used to find decision boundaries that can be used to distinguish each user from the others. Since anomaly detectors train on a single user's data, while multi-class classifiers train on multiple users' data, these two techniques require different evaluation procedures.

Haider[Haider et al., 2000] in 2000 presented a suite of techniques for password authentication using neural networks, fuzzy logic, statistical methods, and several hybrid combinations of these approaches. The approaches presented in their paper used typing biometrics of a user, in addition to conventional login information, to identify a user. The system limited the password length to 7 characters, which is considered a standard password length. During the learning process users were required to enter the password 15 times. The delays between each of the characters of the password string were recorded. At the end of the learning process the system has 6 vectors of length 15 representing inter-character delays. These values are then passed to the neural, statistical, and fuzzy unit.

3 IMPLEMENTATION

Our objective is to develop a keystroke-dynamics evaluation procedure, and to measure the performance of the algorithms. Our approach is as follows:

1. Selecting a dataset.
2. Detector implementation - Three anomaly detectors from existing literature were implemented, namely a statistical, neural and fuzzy implementation.
3. Evaluation methodology - We used a procedure to evaluate each detector, on the same data, under the same conditions, so that performances can be compared.

Each of the above parts have been explained in detail in this chapter.

3.1 DATA SET AND FEATURE VECTOR

We used a publically available data-set[K. Killourhy, 2009] developed by Killourhy [Killourhy and Maxion, 2000]. They recruited 51 subjects from a university. Subjects completed 8 data-collection sessions (of 50 passwords each), for a total of 400 password typing samples. They waited at least one day between sessions, to capture some of the day-to-day variation of each subject's typing.
Some researchers have suggested that letting users choose their own passwords makes it easier to discriminate them [Killourhy and Maxion, 2000]. If true, then letting users choose their own passwords would bias the results of an experiment designed to evaluate performance on an arbitrary password. Thus they decided that the same password would be typed by all of the subjects. The following password was used for creating the data set:

.tie5Roanl

This password was chosen to be representative of a strong 10-character password because it contains more than 7 characters, a capital letter, a number, and punctuation.

The data are arranged as a table with 34 columns. Each row of data corresponds to the timing information for a single repetition of the password by a single subject. The first column, subject, is a unique identifier for each subject (e.g., s002 or s057). Even though the data set contains 51 subjects, the identifiers do not range from s001 to s051; subjects had been assigned unique IDs across a range of keystroke experiments, and not every subject participated in every experiment. For instance, Subject 1 did not perform the password typing task and so s001 did not appear in the data set. The second column, sessionIndex, is the session in which the password was typed (ranging from 1 to 8). The third column, rep, is the repetition of the password within the session (ranging from 1 to 50).

The remaining 31 columns present the timing information for the password. The name of the column encodes the type of timing information. Column names of the form H.key designate a hold time for the named key (i.e., the time from when key was pressed to when it was released). Column names of the form DD.key1.key2 designate a keydown-keydown time for the named digraph(a combination of two letters representing one sound, as in *ti* and *Ro*). Column names of the form UD.key1.key2 designate a keyup-keydown time for the named digraph (i.e., the time from when key1 was released to when key2 was pressed). Note that UD times can be negative, and that H times and UD times add up to DD times.

Consider the following one-line example of what you will see in the data:

subject	sessionIndex	rep	H.period	DD.period.t	UD.period.t	...
s002	1	1	0.1491	0.3979	0.2488	...

The example presents typing data for subject 2, session 1, repetition 1. The period key was held down for 0.1491 seconds (149.1 milliseconds); the time between pressing the period key and the t key (keydown-keydown time) was 0.3979 seconds; the time between releasing the period and pressing the t key (keyup-keydown time) was 0.2488 seconds; and so on.

The raw typing data (e.g., key events and timestamps) cannot be used directly by an anomaly detector. Instead, sets of timing features are extracted from the raw data.
These features are typically organized into a vector of times called a *feature vector*.
We decided to use all the features in the dataset. Specifically, we considered the Enter key to be part of the password (effectively making the 10-character password 11 keystrokes long), and we extracted keydown-keydown times, keyup-keydown times, and hold times for all keys in the password. For each password, 31 timing features were extracted and organized into a vector. The times are stored in seconds (as floating-point numbers).

3.2 IMPLEMENTATION

3.2.1 Statistical Unit

During the learning mode the statistics unit receives the inter-key delays vectors from the system and calculates the average and standard deviation of each of the set of values. Based on these averages and standard deviations, confidence intervals for each of the delay are formed. The equation below determines the confidence interval for just one delay 1.

$$\text{Confidence Interval} = X_i \pm (Z * sd)$$

X_i is the average of the first delay vector,
sd is the standard deviation of the vector and
Z is the standard normal distribution value.

At the end of the learning mode, confidence intervals of all the delays are saved in the corresponding user profile.
When the user enters the password during working mode, the unit loads

stored confidence intervals values from the user profile and matches the value of each inter-key delay with the corresponding confidence interval.
If a particular inter-key delay lies within the confidence interval then it is assumed as a valid delay. The process is repeated for all inter-key delays entered during the working mode. After matching all the delays the authenticity of the user is decided.

For the password ".tie5Roanl", the inter-key delays are collected. The vector represents the typing pattern for a valid user.

x = [0.1491, 0.3979, 0.2488, 0.1069, 0.1674, 0.0605, 0.1169, 0.2212, 0.1043, 0.1417, 1.1885, 1.0468, 0.1146, 1.6055, 1.4909, 0.1067, 0.7590, 0.6523, 0.1016, 0.2136, 0.1120, 0.1349, 0.1484, 0.0135, 0.0932, 0.3515. 0.2583, 0.1338, 0.3509, 0.2171, 0.0742]
(The timing values are in seconds)

For example, for the password “qsvldno”, the confidence interval is shown Vector t represents the password-input delays by an intruder. t = [33,27,22,38,39,44]
These values are compared with the confidence interval shown. The table below shows the result of the comparison. Since only 2 of the values matched with the stored one, the user is considered as an invalid user.

3.2.2 Neural Network Unit

This detector was described by Haider[Haider et al., 2000]. It incorporates a 3 layer feed-forward neural-network trained with the back-propagation algorithm [R. O. Duda and Stork., 2001].

The algorithm uses supervised mode of learning for memorizing the inter-key delays.
The original detector was designed specifically with 6 input nodes, 4 hidden nodes and 1 output nodes.
Our vectors are 31-features long. Thus we created an input unit of 31 nodes, 1 output node and 21 hidden nodes. The number of hidden nodes were selected by maintaining the original ratio of 2 hidden nodes for every 3 output nodes.

* A sigmoid activation function was used both in the hidden and output layers.
* At the beginning of the learning process, the weight matrices between input and hidden layer(M1) and between hidden and output layer(M2) are initialized to 0.1
* The detector was trained to produce a 1.0 on the output node for every training vector.
* The learning-rate parameter was set to 0.0001.
* We trained for 500 epochs.

During testing, the test vector was run through the network, and the output of the network was detected. The error was calculated as $(target - output)^2$
For the password ".tie5Roanl", input neurons were initialized with the feature vector(x). The vector represents the typing pattern for a valid user.

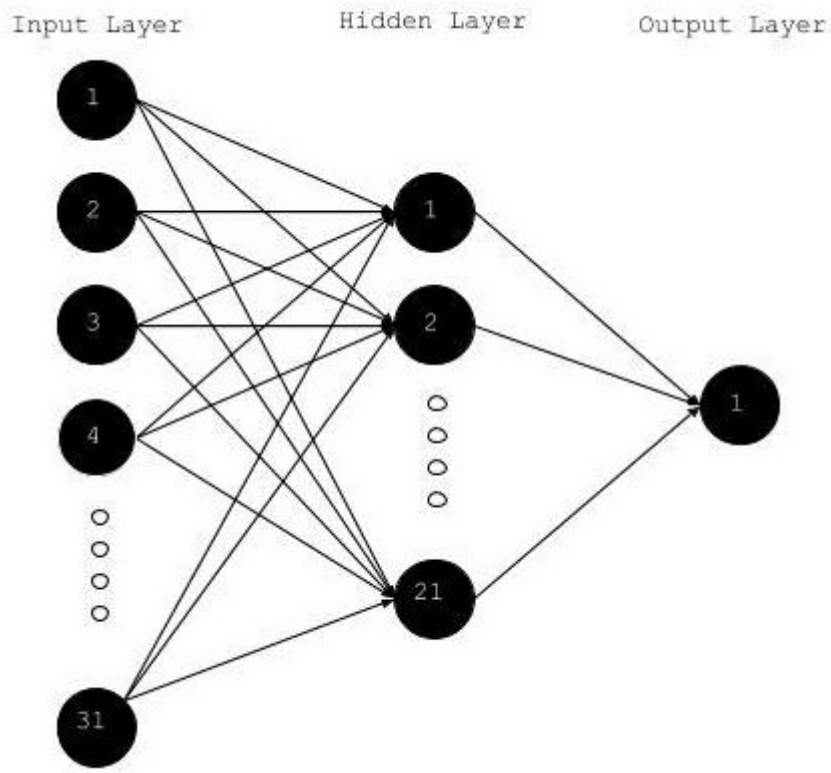

Figure 1: Neural Network

x = [0.1491, 0.3979, 0.2488, 0.1069, 0.1674, 0.0605, 0.1169, 0.2212, 0.1043, 0.1417, 1.1885, 1.0468, 0.1146, 1.6055, 1.4909, 0.1067, 0.7590, 0.6523, 0.1016, 0.2136, 0.1120, 0.1349, 0.1484, 0.0135, 0.0932, 0.3515. 0.2583, 0.1338, 0.3509, 0.2171, 0.0742]
The timing values of the vector are in seconds.

The network learns the typing pattern of the user after a fixed number of iterations. The values of M1 and M2 are stored in the user profile.
These values are used to validate the user or to block an intruder. During the working mode the neural net is initialize with these values. Newly entered password delays are passed as an input to input neurons. If the output vector produces the value within a threshold value then the user is considered as a legitimate one.

3.2.3 Fuzzy Unit

This detector was described by Haider et al.[Haider et al., 2000]. It incorporates a fuzzy-logic inference procedure. The key idea is that ranges of typing times are assigned to fuzzy sets (e.g., the times in the range of 0.50–0.82 seconds are part of a set named moderate"). The sets are called fuzzy because elements can partially belong to a set (e.g., the time 0.66 is strongly in the "moderate" set while 0.51 is only weakly a member of this set).

In the training phase, the detector determines how strongly each feature belongs to each set, and each feature is matched with the set in which its membership is strongest (e.g. the t-hold-time feature will be matched with

Classifiers	Low Value(seconds)	Middle Value(seconds)	Higher value(seconds)
Very Fast (VS)	0.060	0.220	0.380
Fast (F)	0.280	0.440	0.600
Moderate (M)	0.500	0.660	0.820
Slow (S)	0.720	0.880	1.040
Very Slow (VS)	0.940	1.100	1.260

Figure 2: Classifiers in Fuzzy unit

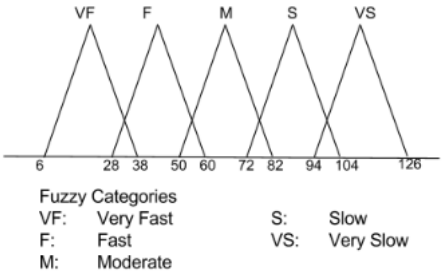

Figure 3: Fuzzy Categories

the "moderate" set if most t hold times are around 0.66 seconds). In the test phase, each timing feature is checked to see if it belongs to the same set as the training data (e.g., the test vector's t hold time is checked for membership in the "moderate" set). The anomaly score is calculated as the average lack of membership across all test vector timing features.

The fuzzy classifiers used in the inference system are shown in figure 2 and figure 3.

Calculation of Membership Function in the inference system:

```
If (Input < LowerBound OR Input > UpperBound)
    Then 0
Else If (Input < Midvalue)
    Then (Input - LowerBound) / (Midvalue - LowerBound)
Else If (Input = Midvalue)
    Then 1
Else (UpperBound - Input) / (UpperBound - Midvalue)
```

The steps involved for training are as follows:

1. For each user out of 51 users:
   1.1 For each feature out of the 31 features for this user:
      1.1.1 Extract vectors consisting of a particular number of values(first 200 attempts) for each feature. These values are used for training.
      1.1.2 Use this vector to assign a particular fuzzy category to this feature.
   1.2 Store this profile of fuzzy categories.

Lets see an example for training for a particular user's particular feature. For user "s002", let the no. of attempts used for training be 15 (only for this example). In the system, 200 attempts are used for training. The feature for which the system is training is the "H.period" feature.

Value	Very Fast	Fast	Moderate	Slow	Very Slow
1.1885	0	0	0	0	0.446875
1.197	0	0	0	0	0.39375
1.0408	0	0	0	0	0.63
1.0556	0	0	0	0	0.7225
0.8629	0	0	0	0.893125	0
0.9373	0	0	0	0.641875	0
0.7967	0	0	0.145625	0.479375	0
0.6447	0	0	0.904375	0	0
0.7357	0	0	0.526875	0.098125	0
0.7550	0	0	0.40625	0.21875	0
0.6927	0	0	0.795625	0	0
0.9155	0	0	0	0.778125	0
0.7028	0	0	0.7325	0	0
0.9165	0	0	0	0.771875	0
1.3501	0	0	0	0	0

Figure 4: Membership Values

COG(sec)	Very Fast	Fast	Moderate	Small	Very Small
0.870113	0	0	0	0.938205	0

Figure 5: Membership Values of COG

So the vector extracted from the first 15 attempts of user "s002" for Session Index 1 (repetitions 1 to 15) for the feature vector "DD.e.five" is : [1.1885, 1.197, 1.0408, 1.0556, 0.8629, 0.9373, 0.7967, 0.6447, 0.7357, 0.755, 0.6927, 0.9155, 0.7028, 0.9165, 1.3501]

We then find the Centre of Gravity (COG) for these values using the formula:

$$I = \frac{\sum \mu_i x}{\sum \mu_i} \tag{1}$$

The COG came out to be 0.870113 seconds.

The COG is again fuzzified and its membership values are shown in Figure 5.
The final fuzzy category for the "DD.e.five" feature for the user "s002" is "Small". The process of finding center of gravity is repeated for all vectors. At the end of this process, this information is obtained and stored in the user profile.

3.3 EVALUATION METHODOLOGY

Consider a scenario in which a user's long-time password has been compromised by an impostor. The user is assumed to be practiced in typing their password, while the impostor is unfamiliar with it (e.g., typing it for the first time). We measure how well each of our detectors is able to discriminate between the impostor's typing and the genuine user's typing in this scenario.
We start by designating one of our 51 subjects as the genuine user, and the rest as imposters.

3.3.1 Training and testing

The steps in training and testing include:

1. We run the training phase of the detector on the timing vectors from the first 200 password repetitions typed by the the genuine user. The detector builds a model of the user's typing behavior.

2. Then, we run the test phase of the detector on the timing vectors from the remaining 200 repetitions typed by the genuine user. We record the anomaly scores assigned to each timing vector as *user scores*.

3. Finally, we run the test phase of the detector on the timing vectors from 5 repetitions typed by each of the 50 impostors. We record the anomaly scores assigned to each timing vector as *impostor scores*.

3.3.2 Calcuating performance

An imposter must have an anomaly score greater than that of a genuine user. We take the mean of the user scores to get a single anomaly score(*mean user score*) and also take the mean of imposter scores(*mean imposter score*). By looking at the anomaly scores, we experimentally choose a threshold value to distinguish between the anomaly scores of a genuine user and the imposter.

4 RESULTS

The said threshold value can only be set when there is a significant difference between the *mean user score* and *mean imposter score*.

The results suggested that we were unable to set a clear threshold that differentiated the genuine user from the imposters. For example, for the FIS, the mean user score and the mean imposter score for user s002 came to be 0.00315449 and 0.00540065. Thus although the imposter score is greater than the user score, they are not significantly different.

As the mean user score and the mean imposter score of the system are similar, hence we couldn't set a threshold value for this dataset of users. The possible reasons for this behaviour can be attributed to the following:

1. During the creation of the dataset, the users entered in 8 sessions of 50 password attempts in each session. At the start of the session, the speed of the user is less. As the user starts getting accustomed to the password, his/her speed increases. If the number of sessions is increased then the difference between the *mean user score* and *mean imposter score* should be high enough to establish a threshold value.
2. In the described models, there are many variables that will affect the performance of the system. Some factors in the FIS are number of fuzzy sets, the overlap between the fuzzy sets, etc.

5 CONCLUSION AND FUTURE SCOPE

Our objective in this work has been to use an available data set, develop an evaluation procedure for implementing the statistical, neural and fuzzy units, and measure the performance of the algorithms on an equal basis. The system uses inter-key delays of the password for user identification.

There are some commercially available systems that characterize the password from simple to complex based on the position of characters on the keyboard. An extension of the presented approach that also incorporates these techniques will definitely improve the performance manifold. The increase in precision of the calculated delays may further refine the results.

The future scope of our work includes:
* Developing a new data set by taking into consideration the visible defects of the one tested in our work.
* Other anomaly detection algorithms which have been proved to provide a good performance in literature (Auto-Associative Neural network, K-means, etc) can be implemented such that we can test the performance of each one against the new and current data sets.
* The algorithms presented in the study can be used for long text after introducing more features like, frequency of characters and words. The analysis of long text can be used in continuous authentication systems.
* The combination of the results of anomaly detection algorithms for keystroke dynamics with mouse usage patterns and application usage will give better identification for the genuine users.

BIBLIOGRAPHY

CENELEC

2002 *European Standard EN 50133-1: Alarm systems. Access control systems for use in security applications. Part 1: System requirements, 2002.* Standard Number EN 50133-1:1996/A1:2002, Technical Body CLC/TC79, European Committee for Electrotechnical Standardization (CENELEC).

Forsen, G., M. Nelson, and R. Staron

1977 *Jr. Personal attributes authentication techniques. Technical Report RADC-TR-77-333, Rome Air Development Center, October 1977.*

Gaines, R. S., W. Lisowski, S. J. Press, and N. Shapiro

1980 *Authentication by keystroke timing: Some preliminary results. Technical Report R-2526-NSF, RAND Corporation, May 1980.*

Haider, S., A. Abbas, and A. K. Zaidi

2000 *A multi-technique approach for user identification through keystroke dynamics.* IEEE International Conference on Systems, Man and Cybernetics, pages 1336–1341.

K. Killourhy, R. Maxion

2009 *Dataset*, http://www.cs.cmu.edu/~keystroke/DSL-StrongPasswordData.csv.

Killourhy, K.S. and R.A. Maxion

2000 *Comparing Anomaly Detectors for Keystroke Dynamics*, Proc. 39th Ann. Int'l Conf. Dependable Systems and Networks (DSN 09), IEEE CS, 2009, pp. 125–134.

Peacock, A., X. Ke, and M. Wilkerson

2004 *Typing patterns: A key to user identification.* IEEE Security and Privacy, 2(5):40–47.

R. O. Duda, P. E. Hart and D. G. Stork.

2001 *Pattern Classification*, second, John Wiley & Sons, Inc., Point Roberts, Washington, USA.