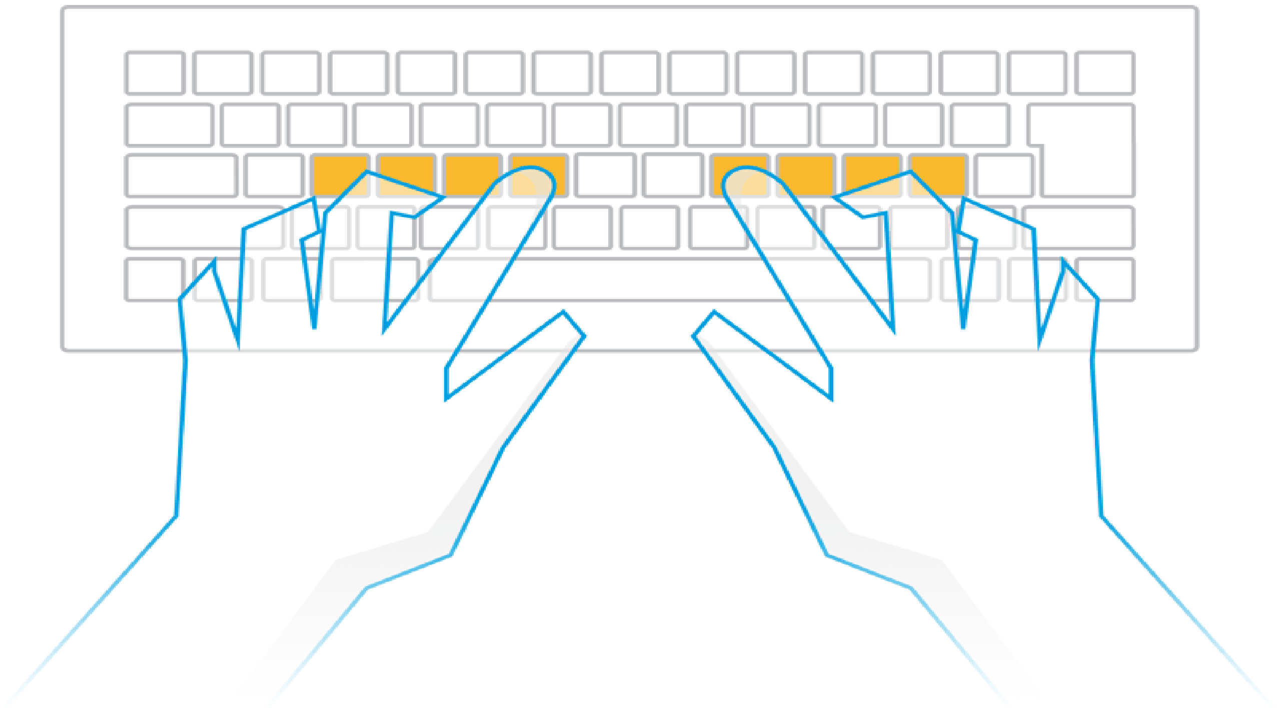
PROJECT REPORT

User Verification based on Keystroke Dynamics



Urvi Chandreshkumar Sheth [A20430232]
Rakshith Churchagundi Amarnath [A20424771]
Aditya Yaji [A20426486]

Contents:

Introduction

Keystroke dynamics is the study of whether people can be distinguished by their typing rhythms, much like handwriting is used to identify the author of a written text. A digital fingerprint would tie a person to a computer-based crime in the same manner that a physical fingerprint ties a person to the scene of a physical crime. Access control could incorporate keystroke dynamics both by requiring a legitimate user to type a password with the correct rhythm, and by continually authenticating that user while they type on the keyboard. Keystroke dynamics is a behavioral biometric method.

Purpose

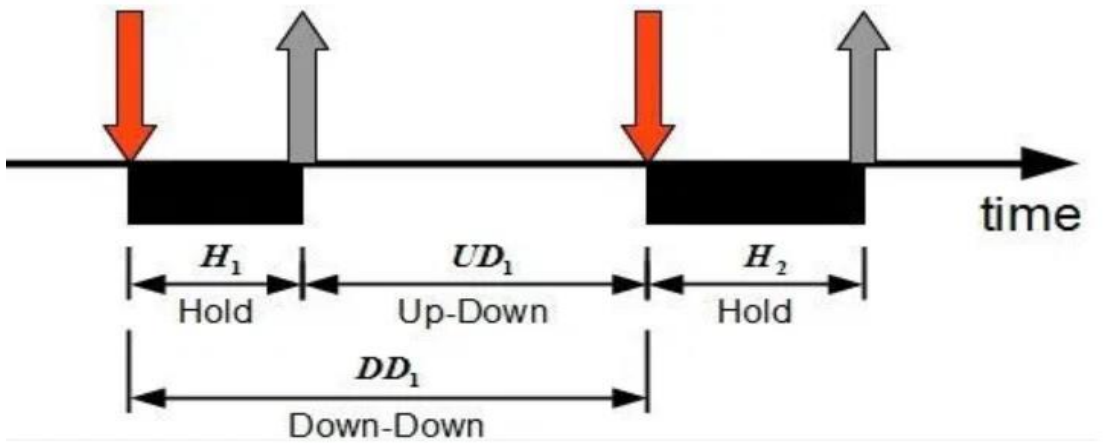
One-time authentication like passwords can be breached or hacked. An extra layer of security for the user to confirm that the user is an authorized user, on an ongoing basis like Keystroke dynamics. Its unobtrusive. There is no requirement for additional hardware, which makes it economical.

One famous application is the Coursera website using a typing test as the method to verify the students. No additional hardware is required to collect keystrokes; only a keyboard. Other possible applications include acting as an electronic fingerprint, or in an access-control mechanism. Keystroke dynamics, together with security measures already in place (e.g., password) can hence serve as a convenient 2-factor authentication.

Design

The keystroke dynamics is collection of timing data for the typing events. Here we are evaluating basic keystroke feature which can be used further to implement the anomaly detection.

The keystroke dataset provides us with these three features. The below picture depicts the three time periods used as features in keystroke data-set.



The features we are extracting from the keystroke dynamics are:

- Hold time – time between press and release of a key.
- Keydown-Keydown time – time between the pressing of consecutive keys.
- Keyup-Keydown time – time between the release of one key and the press of next key.

After the Features has been extracted from the timing data for keystroke, these data are used further to train the learning models to understand he characteristics of a user.

Implementation

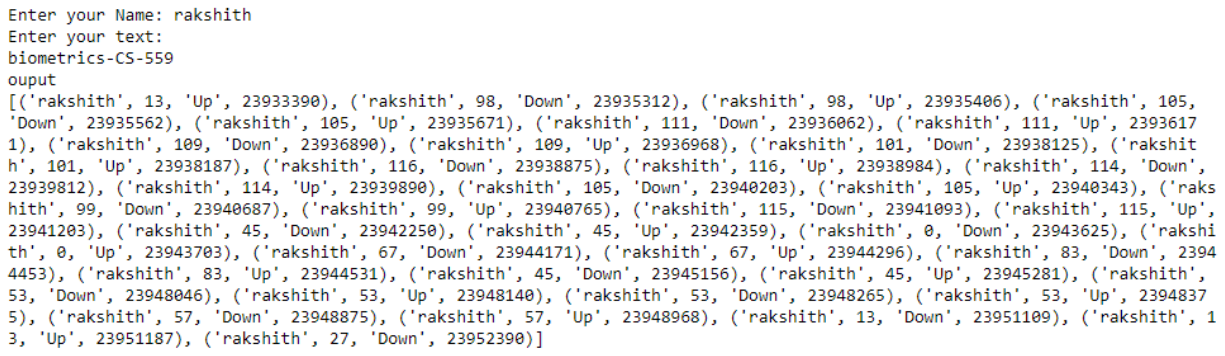
The implementation for generating key stroke dynamics and model training is done is three phases in this project using python 3.6/3.7:

- Data Acquisition
- Feature Extraction
- Feature Engineering

Data Acquisition

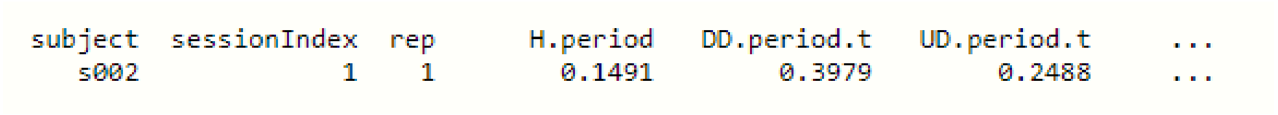
This project mainly uses two datasets, **first dataset** is collected by running the ipynb named Collecting keystroke. This file takes input from user and stores it into the collecting_keystroke.csv. The first input is the name of the user and after that the user is asked to enter alphanumeric data with some special characters.

To get the key stroke dynamics, pyhook python library is used. This library allows to write own event listeners to key up and key down events with the other information such as time, the key id of the key pressed, ascii value of the key pressed etc. [1]. The code overwrites the key Up and key Down event and store the data of username, the ascii value of the key pressed, the event for the key (key up or key down) and the time when the key event happened in milliseconds with reference to the UNIX time.



```
Enter your Name: rakshith
Enter your text:
biometrics-CS-559
ouput
[('rakshith', 13, 'Up', 23933390), ('rakshith', 98, 'Down', 23935312), ('rakshith', 98, 'Up', 23935406), ('rakshith', 105,
'Down', 23935562), ('rakshith', 105, 'Up', 23935671), ('rakshith', 111, 'Down', 23936062), ('rakshith', 111, 'Up', 2393617
1), ('rakshith', 109, 'Down', 23936890), ('rakshith', 109, 'Up', 23936968), ('rakshith', 101, 'Down', 23938125), ('rakshit
h', 101, 'Up', 23938187), ('rakshith', 116, 'Down', 23938875), ('rakshith', 116, 'Up', 23938984), ('rakshith', 114, 'Down',
23939812), ('rakshith', 114, 'Up', 23939890), ('rakshith', 105, 'Down', 23940203), ('rakshith', 105, 'Up', 23940343), ('raks
hith', 99, 'Down', 23940687), ('rakshith', 99, 'Up', 23940765), ('rakshith', 115, 'Down', 23941093), ('rakshith', 115, 'Up',
23941203), ('rakshith', 45, 'Down', 23942250), ('rakshith', 45, 'Up', 23942359), ('rakshith', 0, 'Down', 23943625), ('rakshi
th', 0, 'Up', 23943703), ('rakshith', 67, 'Down', 23944171), ('rakshith', 67, 'Up', 23944296), ('rakshith', 83, 'Down', 2394
4453), ('rakshith', 83, 'Up', 23944531), ('rakshith', 45, 'Down', 23945156), ('rakshith', 45, 'Up', 23945281), ('rakshith',
53, 'Down', 23948046), ('rakshith', 53, 'Up', 23948140), ('rakshith', 53, 'Down', 23948265), ('rakshith', 53, 'Up', 2394837
5), ('rakshith', 57, 'Down', 23948875), ('rakshith', 57, 'Up', 23948968), ('rakshith', 13, 'Down', 23951109), ('rakshith', 1
3, 'Up', 23951187), ('rakshith', 27, 'Down', 23952390)]
```

The **Second dataset** is called CMU Keystroke Dynamics Benchmark Data-set [2] which comprised of keystroke information for 51 users, each user typing the password ".tie5Roanl" 400 times. The recruited 51 subjects (typists) fully completed the study. All subjects typed the same password, and each subject typed the password 400 times over 8 sessions (50 repetitions per session). The password (.tie5Roanl) was chosen to be representative of a strong 10-character password [3].



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

Feature Extraction

To calculate timing features for the **first dataset**, for each user the key down and key up event for each key id is identified. For the key pressing event, the event of down-up pair is one key stroke. So, from the millisecond value which is stored in collecting keystroke csv file is taken into calculation to find values of Hold, Down-Down and Up-Down. We are considering ascii value of 33 to 122 (a-z A-Z 0-9 and special characters). The values for each key and user combination taken from the input file and then the values are calculated and stored in keystrokedistance.csv.

For **second dataset,** the raw records of all the subjects' keystrokes and timestamps were analyzed to create a password-timing table. The password-timing containing timings of Hold (H), Down-Down (DD) and Up-Down (UD) is stored in table encodes the timing features for each of the 400 passwords that each subject typed. [3]. The second dataset is stored in keystroke.csv

Feature Engineering

After having all the dataset timing values, now the project will train the model using this data. We have used Manhattan, Euclidean and K-means algorithmic models. We take both the dataset as input to the algorithm and divide the dataset into 70-30% for training and testing datasets. The training data is applied to each algorithm and the test data is used to predict if the model can predict the user is valid or not. These models are written in the file named analysis ipynb.

For Manhattan and Euclidean algorithm, we consider mean vector for the training dataset and using that we are generating distance between the testing dataset and the mean value using python function called city block and Euclidean to calculate the both distances respectively. For K-means, The dataset only uses the timing variables and we apply K-means algorithm for n=5 which uses predict method of GridSearchCV python package.

Result

The result here is compared in the form of ROC curve. There were two datasets in which one was controlled, and another is self-collected gibberish dataset. The three models which are applied are Manhattan distance, Euclidean distance and k-means algorithm. The comparison of equal error rate (ERR) is shown below in the table.

Methods	**Equal Error rate from different dataset**	
	Dataset 1	**Dataset 2**
Manhattan	0.383	0.195
Euclidean	0.444	0.218
K-Means	**0.306**	**0.155**

As the table states as above, the equal error rate for all the methodology for dataset1 is as twice as for dataset 2. Though the error rate seems bad in comparison, it is still a better result as it gives minimum of 55-60% accuracy for the worst case.

References

[1] https://sourceforge.net/p/pyhook/wiki/PyHook_Tutorial/

[2] http://www.cs.cmu.edu/~keystroke/KillourhyMaxion09.pdf

[3] http://www.cs.cmu.edu/~keystroke/