**Data Collection**

We wanted our data set to include movements that were natural to best simulate day to day actions a user would perform on the computer in order to have a more relevant set of data to what our authenticator would see in a practical environment. To do this we asked a class of students to record themselves and two other friends or family members. The students were told to instruct whomever was being recorded to go about any activity on the computer they normally would for an hour period.

In order to log the mouse activity of users we created a simple mouse logger. The logger was a small piece of software that ran on the windows operating system. The logger, when run on a machine, logged the position of the cursor using the windows method GetCursorPos(), the state of the left and right mouse buttons (up or down), and a timestamp every 15.6ms. 15.6ms being the default timer resolution that windows supports. The resulting log was then placed into a text file.

We then screened the samples and removed samples with very few (0 – 10) movements and samples that weren't an hour long. Post-screening we had 25 valid user samples. Our samples consisted of university students and their friends and family collected from late November to early December 2014.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Maximum | Minimum | Average |
| Left Clicks | 716 | 37 | 547.84 |
| Right Clicks | 690 | 0 | 122.8 |
| Double Clicks | 1282 | 1 | 181.64 |

Table 1. Description of the data collected.

**Feature Extraction**

We defined a mouse movement as a period of cursor movement followed by a click. This isn't to say that a click without any movement prior is not considered a mouse movement. It would be a mouse movement with no speed, acceleration, or jerk. If the cursor sustains no action for a period of 500ms or longer we classify that period as a pause to allow for periods where the user is not concentrating on moving the mouse or not moving it with conscious effort.

1. Features

Some of the features we used were found in [sources here] and showed promise in accurately identifying a user's activities. The features used were:

|  |  |
| --- | --- |
| Feature Name | Description |
| Single Left Click Length | The length of a click from the first record the mouse button is found to be down to the first record the mouse button is found to be up. |
| Single Right Click length |
| Double Click Length | Split into 4 sub-features: Total Length and intervals 1,2, and 3.  Total length: The time between the first record the mouse is found to be down and the first record the mouse is found to be up after the second click.  Interval 1: The first single left click length in a double click.  Interval 2: The time between the first record of the left mouse button being found up after the first click to the first record of the left mouse button being found down for the second left click.  Interval 3: The second single left click length in a double click. |
| Speed | Distance the cursor travelled divided by the total time of the movement up until the click. |
| Acceleration | Change in Speed over time |
| Jerk | Change in Acceleration over time |

Table 2. Definitions of features used.

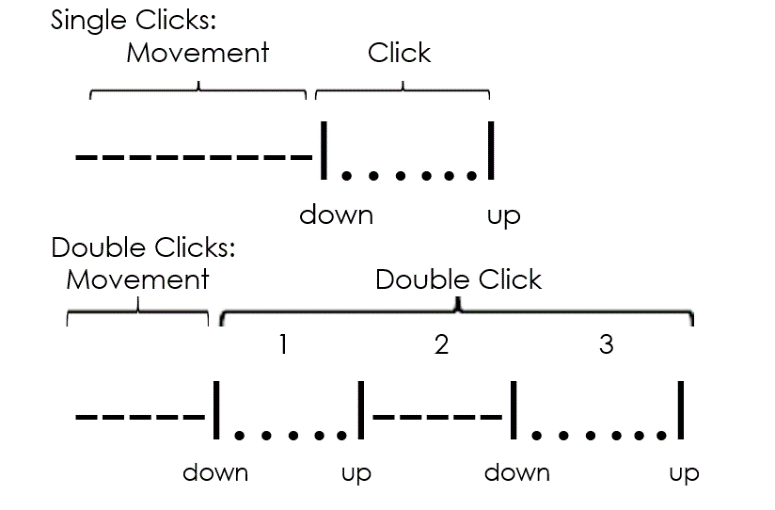


Figure 1. Graphic representations of a single click and a double click across time. Dashes represent mouse buttons being up, Dots are periods of a mouse button being pressed, and vertical bars are periods of transition where the user presses or releases a button.

**Methodology**

**Outlier filtering**

For our setup we first went through each of the user samples and extracted each of the features, as defined above, from each mouse movement. To remove rare cases of mouse movements where the user provides a mouse movement far from the mean we implemented an outlier removal scheme. We define a mouse feature to be an outlier if it is more eccentric than 66% of the population of the same mouse features for a single user. For example, if we are looking to find if a particular mouse feature M is an outlier:

IsAnOutlier(M):

count = 0

for each feature i in the population of type M

if |M - i| > r

increment count

if (count / Population.size) > .33

return True

else

return False

We defined the variable r, for each mouse feature, a number large enough to include most of the population while small enough to remove the noise from the data. We found optimal values for our r values after extensive experimentation we will describe later on.

**Our verifier/Verification/Authentication method**

To authenticate a user based on their mouse movements we analyze six mouse features over the course of a sample. Our method is a simplistic user-authentication model. We create a profile based on a user's past behavior then test further samples against that profile to determine if the sample we test with is close enough to the previous profile to be considered the same user.

We then split the edited list of features into a testing set and a training set. For each file we create user profiles based on half of the user's sample (about 50 mouse movements on average for our samples). The profiles being simply the averages and standard deviations of each of the mouse features. The mouse movements from the sample that we did not use for the profile go into the user's testing set.

Using a simple classification scoring method we can see how well a user profile and a testing set of data match up. For our entire set of data we took every user profile and tested each against every testing set. Our custom testing method generated a score for each test between 0 and 1. 1 being a perfect match and 0 being a perfect mismatch.

We tested by comparing each of the mouse movements in a testing set against the users profile. If the features in the mouse movement are within m standard deviations of the corresponding means in the user profile then the test gets a point. The total amount of points accrued is divided by the total amount of mouse movements in the file to generate the score.

**Figure 2. Flowchart depicting the steps in our method.**

**Experimental Setup**

We had a few parameters where it wasn’t immediately apparent how to set them. For the R variables we needed values that can discriminate mouse feature values that are too eccentric while keeping as much unique data as possible. To solve this we ran a script that ran the tests and found the EER over an array of values for each parameter to determine the optimal values where the EER is at its lowest.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Single Click R | 15 | 20 | 25 | 30 | 35 |
| Double Click R | 175 | 180 | 185 | 190 | 195 |
| M Values | .7 | 1 | 1.3 | 1.6 | 1.9 |

Table 3. Values tested for our parameters.

We found that an M value of 1.9, a Single Click R of 15, and a Double Click R of 175 to produce the lowest EER for our set of data.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Single R Val 15** | 175 | 180 | 185 | 190 | 195 |
| .7 | 24.92% | 14.85% | 11.76% | 11.76% | 11.93% |
| 1 | 16.00% | 26.09% | 26.09% | 27.96% | 24.60% |
| 1.3 | 14.37% | 15.83% | 15.83% | 12.18% | 12.15% |
| 1.6 | 14.50% | 12.00% | 12.00% | 11.26% | 11.71% |
| 1.9 | 10.83% | 12.00% | 12.67% | 12.02% | 12.00% |
| 2.2 | 8.33% | 10.57% | 14.51% | 12.97% | 14.42% |
| 2.5 | 9.44% | 11.65% | 10.74% | 11.65% | 10.79% |

**Experimental Results and Analysis**

To calculate the results of our experiment we took each users profile and tested that against every other set of testing data. This values in this generated confusion matrix are then used to calculate the false rejection rate and false acceptance rate. After finding the best settings for our parameters we obtained 10.14% equal error rate.