Jon Dycaico; Iris Favoreal

CSS 337: Assignment 2

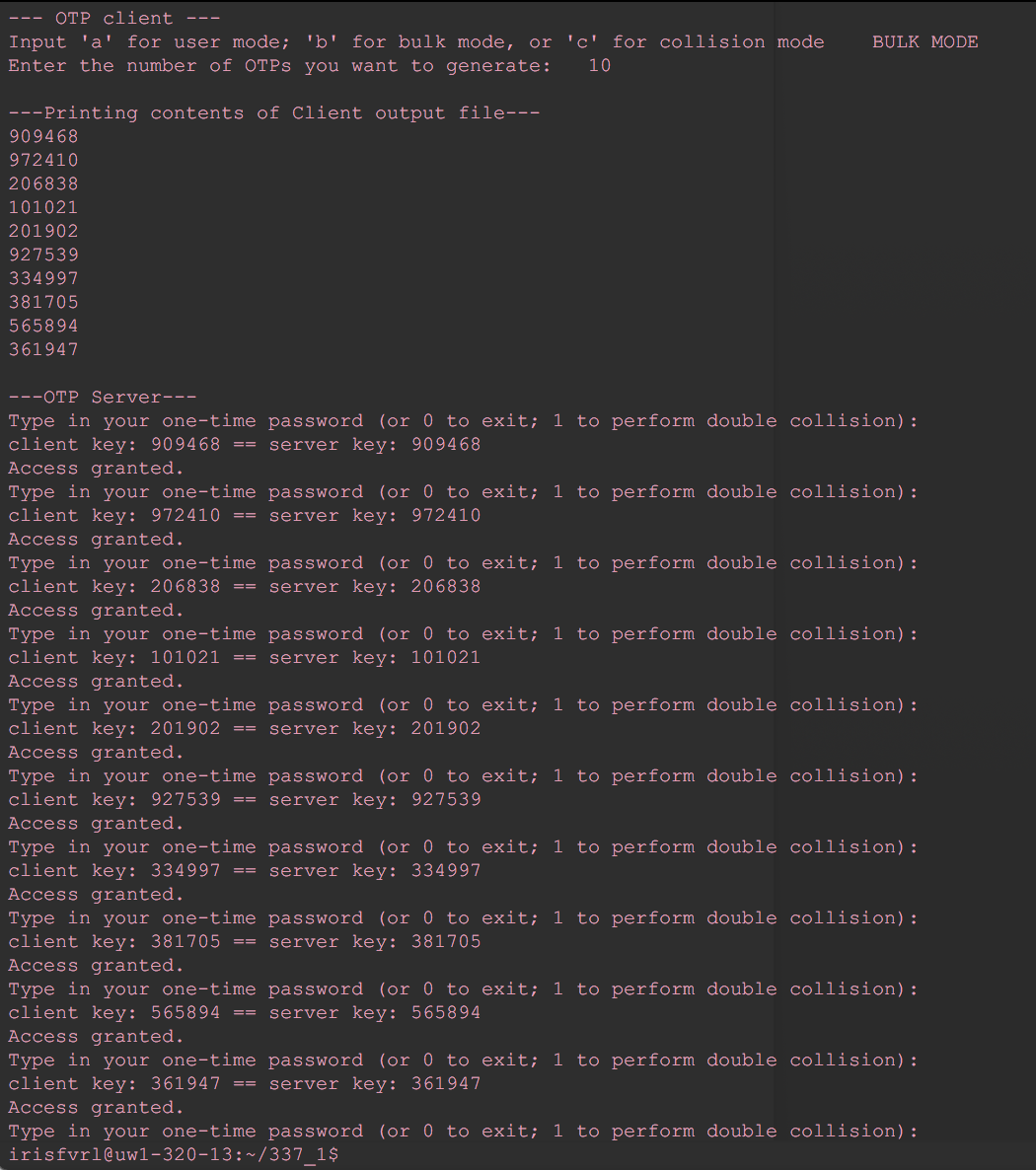
November 7, 2018

**Overview**

This is a client-server OTP program wherein the client generates a one-time password every time a user prompts it to. This can then be validated by the server program; if the user enters the proper client-supplied one-time password, the server will grant the user access.

**Test.sh**

Test.sh is a script file that automates the validation of the proper generation of one-time passwords between client and server, using a test size of 10 keys. Below is a screenshot of running the script. The OTP client is prompted to generate 10 keys using the ‘Bulk Mode’ which is then saved into a newly-created file named ‘output.txt’. This file is then parsed line-by-line as the input into the server program, allowing the server to check the validity of each key generated by the client.

****

**OTP Client**

A python script consisting of three main functions:

1. user mode

* The script prints an OTP to the terminal each time it is prompted by the user

b) bulk mode

* The script generates the number of OTPs corresponding to the value it is passed
  + Good for testing, or leveraging for other internal functions

c) collision

* The script returns the number of collisions detected at intervals of 1000 keys, up to 10000. It then graphs this relationship using the matplotlib library

**OTP Server**

A java program that takes in the one-time password string as standard input and validates it by comparing it with the server-side keys.

* Synchronization mechanism
  + Case: User OTP is ahead of Server OTP
    - The server checks the next 10 keys for a match. It'll check the pre-generated keys first, then generate more keys as needed so that a total of 10 keys is checked if there is no match found.
    - If there is a match, the user is prompted to type in the next OTP. If it's the proper one, the user is granted access.
    - If there is no match, the user is not granted access.
  + Case: Invalid OTP not matching the next 10 keys, or just at all
    - The server checks the next 10 keys for a match. It'll check the pre-generated keys first, if any, then generate more keys as needed so that a total of 10 keys is checked.
    - User is not granted access. And the server remains to expect key #1.
* Multiple collision test - user input is "1"
  + User can perform a multiple collision test which analyzes how many times two consecutive keys occur again while traversing down a dataset. This is done with a dataset of 10000 keys.

## 

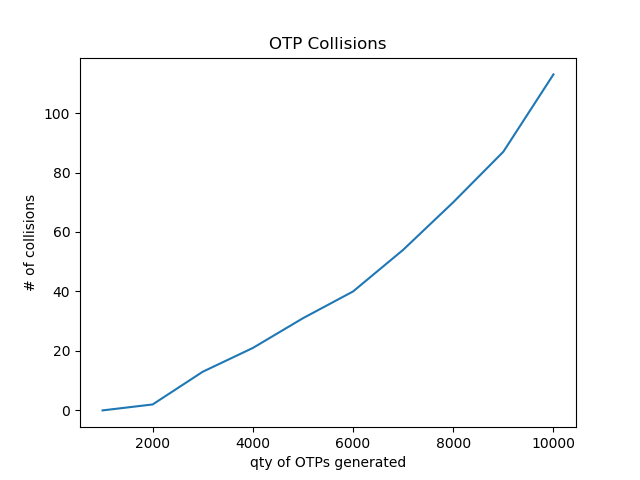
## 

## 

## Collision tests

Single collision testing takes place in the client, while “paired” collision testing takes place in the server script.

Single collision tests



For each key in the list, go through the list and find any duplicates, adding them to the counter.

The output shows evidence of a logarithmic relationship between the quantity of OTPs and the number of collisions detected.

Multiple collision tests

Both researchers were unable to find any collisions of sequential pairs of OTPs, even as *n* (the number of OTPs generated) passed the 10,000 mark.

If the probability of randomly generating any particular single 6-digit number is 1/k, the probability of generating a sequence of two particular 6-digit numbers is 1/k^2, or perhaps even smaller.

If we use a middling data point as our reference, we have the probability of getting a single 6-digit value randomly as 40/6,000, or 1/150. Judging by the logic above, the probability of getting a sequential pair of random 6-digit numbers could easily be as low as, or lower than 1/22500.

Thus, even at the 10,000 mark, we had only a 0.44 chance or below of finding one of these matched pairs. This could explain our findings. I would guess that the above calculations are incorrect and the probability of such a pair is even lower, as we were unable to generate one despite multiple trials.