Homework 3: this homework consists of only 1 question, and all parts are mandatory. All 5 parts require coding. Coding is required in Python. A long with your code, provide a sheet including steps followed and comments (where requested). The code will be automatically graded, so provide what is requested. If in doubt, as on Piazza.

Deadline and notes: 1 week from the time stamp of the posting of the homework on Piazza. No late homework is accepted. No exceptions. Academic integrity is taken seriously, and cheating is not tolerated. Note about data: do not distribute the file. When unzipped, the file is 2.8GB

Question 1 (100pts) attached you will find a large file with a large number of repeating ASCII passwords (320,412,510 passwords) corresponding to 2,151,219 unique passwords. Based on the contents of the file and what you learned in the class, perform the following (using python):

1. **Frequency analysis:**
   1. Perform a frequency analysis of the passwords. The outcome of this analysis should be a sorted file with two columns. The first column in the resulting file is the number of times a password appears in the file, and the second column is the password itself. The second column is the set of unique passwords in the file.
   2. What do you conclude about the distribution of the passwords and their frequency? Comment on whether that is surprising or in line with your expectations.
2. **Strength analysis:** 
   1. Perform a strength analysis of the passwords in the file. The strength analysis should follow the Shannon entropy. Use the entire set of passwords in the file as your space of probability calculations. Here is an example of how the Shannon entropy works with the entire file as your calculation base. Let PWD = {Aziz, Mohaisen, is, a, har$h, professor} be your dictionary of passwords. First you calculate the probability for any character (case sensitive) to happen in any password (Aziz, Mohaisen, etc.). For example, all passwords have 29 characters in total, and A happens only once, so the probability for A is 1/29. Probability of a, on the other hand is 3/29, and so forth. Then, for a given password w, you construct a probability distribution over the characters, plug that distribution in the Shannon formula explained in the class, and done. The outcome should be a sorted file of two columns. The first column is entropy value, and the second column should be a unique password in the passwords dictionary (there is no point of including the same password multiple times, since they will have the same entropy).
   2. Comment on the strength analysis of the passwords and any obvious trends
3. **Strength analysis of unique passwords:**
   1. Redo question 2, but this time using the set of unique passwords in column 2 of question 1. This is, when calculating the probability as in question 2, consider your space of probability to be driven from the unique passwords dictionary. Compare the results with question 3.
   2. Comment on the comparison, and how the frequency of a password in a dictionary make that password a good (or a bad) choice with respect to security.
4. **Offline dictionary attacks:** The hash values (MD5 hashes, installed) below were revealed when a DB was compromised. Answer the following
   1. If you are to try to find the original passwords in those hashes, how would you approach the problem?
   2. If you are told that the passwords are already in the dictionary you are given, how your approach would be different?
   3. Find the passwords of those hashes, given that they are in the dictionary.
      1. ba931c15ec0163c4bb339f41571694ce
      2. c9cd905fc459e5550b8c3b01d4346c25
      3. e9269d9e52a692f52caece9d0e7cdae1
      4. 660719b4a7591769583a7c8d20c6dfa4
5. **Contexts for targeted password cracking:** Jim is a fan of the Buffalo Bills. This, in turn, is reflected in his choice of passwords. Often time, he would use his passwords as words related to the Buffalo Bills, including names of players, years of wins, etc. The MD5 (unsalted) hash value of Jim’s password is “83bfc234f88cc75d52e9ec24e54bc8be”. Answer the following:
   1. Provide 3 possible candidates for what the password could be
   2. If you are told that the password is a family name of a player playing for the Bills, how would this help you find the password?
   3. If you are told that the password is a family name of a player who retired on April 7th, how would this help you find the password?
   4. Use a side channel to infer the password, without knowing the context above (e.g., search Google).
   5. What does this tell you? About salting, using common passwords, etc.

**Solutions:** Armaan Goyal

armaango(50170093)

1. See the attached output files, code, and readme file.
2. Based on the output of the previous part, following are the observations I made about the frequency of the passwords:

* The most frequent passwords consist of common names of people or places or objects like Johnson, California or C0mputer. These passwords seem pretty simple and in my opinion are easier to break using a brute force attack
* The least frequent passwords seem comparatively complex containing a mix of uppercase lowercase characters along with special characters and numbers. Also long strings of numbers are also among unique passwords which could effectively either be any random sequence of numbers created or even a unique string of numbers like somebody’s cell number or some serial number for that matter. These passwords can prove to be tougher to crack
* Somewhere in the middle lie passwords that are made of combining common words and a pair of numbers appended behind the words. These passwords mainly vary based on the different numeric part whereas the words in them seem to be easy to break.

These observations are in line with what I would expect in such a large database of passwords.

2)

a. See the attached output files, code, and readme file.

b. Based on the output of the above file, the obvious trend was that longer passwords that have all unique characters have a higher entropy value thus they can be expected to be much more secure. As the password length shortens or the characters in the password start to repeat, the entropy value falls. For passwords that have a single character repeatedly, the entropy value came out same as if the single character was used which is also obvious based on the formula we used to calculate.

3)

a. See the attached output files, code, and readme file.

b. After comparing the output above and for the previous Q 2(a), I noticed the similar pattern as mentioned above, only observable difference was in the entropy values of the individual passwords.

This tells us that even if a password used was too frequent, it doesn’t really define the strength of the password on its own. If a long password with unique characters is used, it will always be harder to crack.

4)

a. First step would be to try and use any available tools online. There are a lot of tools available which keep a database of strings and their hashes and those tools simply run the provided hash value against those databases and if found, return the original string. It is a very primitive approach but since hashing has the one-wayness property, there is no other way to reverse the hash value to get the original password

b. In case I know that the passwords are already in the dictionary of passwords that I have, my approach would then turn to a simple brute force attack. I will keep computing the hash for each password in my dictionary and keep comparing it with the hash whose reverse password I need to find. If the password is present in my dictionary, soon it will match and I will have my required output.

c. See the attached output files, code, and readme file.

5)

a. 1. Bills1964Buffalo 2. JimKelly12 3. 2004JimRitcher

b. If I have the said information that the password is a family name of a player, I would simply start to compile the family names of all players who have ever played for Buffalo Bills, information that is freely and easily available online, compute the hashes for these names and start comparing them against my input hash value

c. The said information would narrow down my data to a very small set, preferably even a single name based on which I can easily decipher the password without too much effort as the given data about the context would add to my search strings and leave me with a very small probable data set.

d. I used Google to find an online tool to decrypt the hash and found the password as Tarpley

e. Since I was able to find the password easily online, this tells us that simple passwords like names of people, especially who are public and whose names are available online, should not be used as passwords. Even if one needs to use any such names or phrases as part of their passwords in order to make them easy to remember, it is absolutely advisable to couple them up with either unrelated numbers or special characters or both. Single words in isolation used as passwords are never secure and should be avoided at all costs.