**Lab 3: MD5, Rainbow Tables**

**Deadline: 7 June 2023 11:59PM**

* [Lab 3: MD5, Rainbow Tables](https://github.com/istd50042/lab2022_student/blob/main/lab3/README.md#lab-3-md5-rainbow-tables)
  + [Objectives](https://github.com/istd50042/lab2022_student/blob/main/lab3/README.md#objectives)
  + [Part I: Hashing Using MD5](https://github.com/istd50042/lab2022_student/blob/main/lab3/README.md#part-i-hashing-using-md5)
  + [Part II: Break Hashes with Brute Force](https://github.com/istd50042/lab2022_student/blob/main/lab3/README.md#part-ii-break-hashes-with-brute-force)
  + [Part III: Creating Rainbow Tables](https://github.com/istd50042/lab2022_student/blob/main/lab3/README.md#part-iii-creating-rainbow-tables)
  + [Part IV: Salt](https://github.com/istd50042/lab2022_student/blob/main/lab3/README.md#part-iv-salt)
    - [Generating New Hashes](https://github.com/istd50042/lab2022_student/blob/main/lab3/README.md#generating-new-hashes)
    - [Cracking the Hashes](https://github.com/istd50042/lab2022_student/blob/main/lab3/README.md#cracking-the-hashes)
  + [Part V: Hash Breaking Competition](https://github.com/istd50042/lab2022_student/blob/main/lab3/README.md#part-v-hash-breaking-competition)
  + [Submission](https://github.com/istd50042/lab2022_student/blob/main/lab3/README.md#submission)
    - [eDimension Submission](https://github.com/istd50042/lab2022_student/blob/main/lab3/README.md#edimension-submission)

**Objectives**

* Hashing using MD5
* Crack MD5 hashes using brute-force and rainbow tables
* Strengthen MD5 hash using salt and crack again the salted hashes
* Crack some more difficult hashes with available tools

The submission from this lab will include a short **report** to be submitted in **PDF or HTML** form.

A template for the report has been provided in the form of a Jupyter Notebook. Export it as HTML (preferred) for the submission. You can also use any other document processing tool to make the report, as long as you answer all the questions in each section.

**Part I: Hashing Using MD5**

In this section, we will explore hashing using [MD5](https://en.wikipedia.org/wiki/MD5).

Compute a couple of MD5 hashes of a few strings of your choice. The strings should be of different lengths.

There are several methods to generate MD5 hashes. Two are provided here for you to explore:

1. Shell: echo -n "foobar" | md5sum
2. Python: hashlib library (see example below)

import hashlib

plaintext = "Pancakes"

result = hashlib.md5(plaintext.encode())

print(result.hexdigest())

Answer the following questions in your report:

* How does the length of the hash correspond to the input string?
* Are there any visible correlations between the hash and the input string?
* What are the issues related to the cryptographic weakness of MD5?

**Part II: Break Hashes with Brute Force**

There are fifteen hash values in the file hash5.txt (newline separated).

For this exercise, create a Python 3 script ex2.py to reverse the hashes via brute force. You can achieve this by hashing different input values until you find a hash that matches one in hash5.txt.

You need only to consider strings that fulfill the following criteria:

* Length of 5 characters
* Each character can be a lowercase alphabet or numeric

Take note of the computation time of your script to reverse all fifteen hashes.

**Save all the hashes you recover as ex2\_hash.txt**

Answer the following questions in your report:

* How much time did you take in total?
* How much time does it take to crack each string, on average?
* Is it possible to amortize (gradually write off the initial cost of) the brute forcing attempts?

**Part III: Creating Rainbow Tables**

For this exercise, you need to install the CLI tool rainbowcrack from <http://project-rainbowcrack.com/>.

Installing Rainbowcrack:

# download the latest rainbowcrack

wget http://project-rainbowcrack.com/rainbowcrack-1.8-linux64.zip

unzip rainbowcrack-1.8-linux64.zip

cd rainbowcrack-1.8-linux64

# somewhat unsafe, haha

chmod +x \*

# now you can run ./rtgen , ./rtsort , ./rcrack etc

Tasks:

* Use [rtgen](http://project-rainbowcrack.com/generate.htm) to generate rainbow tables with the characteristics shown below.
  + Five characters input
  + Only lower case letters and numeric characters.
  + Chain length is 3800.
  + Chain number is 600000.
  + Part index is 0.
  + Table index is 0.
* Use [rtsort](http://project-rainbowcrack.com/generate.htm) to sort the rainbow table to make searchable by rcrack. (sorts the rainbow chains by end point to make binary search possible)
* Use [rcrack](http://project-rainbowcrack.com/crack.htm) to crack the list of fifteen strings from hash5.txt.

Answer the following questions in your report:

* Is rcrack faster/slower than your script ex2.py? By how much faster/slower is it?
* Do you observe any advantages or disadvantages of using rainbowcrack?

**Part IV: Salt**

In this section, we deal with salt, but not the kind we are most familiar with.

**Generating New Hashes**

You will need to write a script ex4.py to do the following:

1. Load the hashes from ex2\_hash.txt and append one **random** lowercase character as salt value to all the elements of the list of passwords you recovered in the previous part of this exercise.
2. Rehash the password using MD5, and store the newly hashed passwords and their salt values into a new file called salted6.txt. Store the new plaintexts as well in a plain6.txt file.

The functional definition of our salt strategy is the following:

saltedhash(plaintext) = hash(plaintext||salt)

Where || denotes concatenation.

**Cracking the Hashes**

* Generate a new rainbow table using rtgen (with new parameters - up to you do decide) to break the hash values. As before, sort the table using rtsort.
* Compare the timing of the new table generation and lookup vs the previous values.
* Try to break as many salted hashes as possible.

Answer the following questions in your report:

* What is the observed differences between your ease of cracking the salted vs the unsalted plaintexts?
* Report the difference in time observed to crack.
* Explain any differences between salted and non salted rcrack strategies.

**Part V: Hash Breaking Competition**

We provide a list of hashes in hashes.txt:

* They are of different difficulty, not all are equally hard.
* There are no easy rules about length or characters allowed any more!
* Try to reverse as many of those hashes as possible. You can also use any other tools as you want. (There are no limitations here)

Submit the answers as ex5.csv file containing two columns. The first column is the md5 hash of the plaintext you break, and the second column is the plaintext.

Answer the following questions in your report:

* What is the approach you used to crack the hashes
* How you decided or designed your approach
* Main challenges and limitations of your approach
* **How many hashes out of the total did you manage to crack** (there is no prize)

**Submission**

**eDimension Submission**

Lab 3 submission:

Upload a **zip file** with the following: file name format: **lab3\_name\_studentid**

Report (PDF or HTML format)

* ex2.py
* ex2\_hash.txt
* ex4.py
* salted6.txt
* plain6.txt
* ex5.csv

**Deadline: 7 June 2023**