**Lab 3: MD5, Hashing**

**Deadline: 15 June 2024 11:59PM**

* + [Lab 3: MD5,](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: 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 IV: 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
* 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.

**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: 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 ex3.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.

Answer the following questions in your report:

* For this exercise, modify the previous Python 3 script **ex2.py**to **ex3.py** to reverse the salted hashes via brute force.
* What are the observed differences between your ease of cracking the salted vs the unsalted plaintexts?
* Report the difference in time taken to crack the salted and the unsalted hash values.
* Explain any differences between salted and un-salted crack strategies.

**Part IV: Hash Breaking Competition**

We have provided 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 ex4.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** .

**NOTE**: **The file names must not be changed, and you must name the files as stated in this document. File names other than the ones requested in this document will not be marked.**

**Report (PDF or HTML format)**

* ex2.py
* ex2\_hash.txt
* ex3.py
* salted6.txt
* plain6.txt
* ex4.csv
* Report is a must.

**Deadline: 15 June 2024**