51.506 Security Tools Lab 1

Assignment 1 – Hashing & passwords

Hand-out : 22-May-2023

Hand-in : 30-May-2023 (2359hrs)

1. Objectives

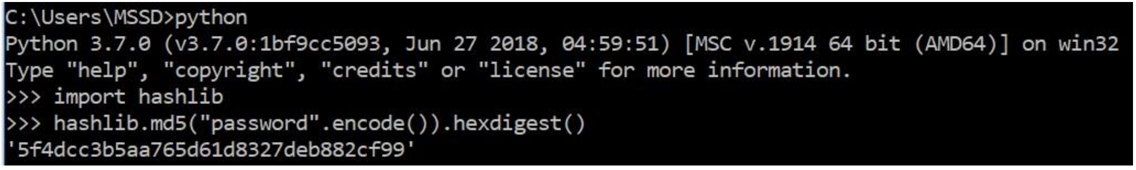
- Hash password using MD5
- Crack MD5 hashes using brute-force and rainbow tables
- Strengthen MD5 hash using salt and crack again the salted hashes by rainbow tables and rule-based extension of dictionary attack using hashcat
- Compete in the hash breaking competition

2. Setup

- This lab can be done in Windows or Linux. We'll be using mostly Linux to do the demo.
- Note that in this laboratory you should use python3

3. Hashing password using MD5

To warm up, compute a couple of MD5 hashes of strings of your choice using python's command line. use import hashlib module and its md5() function. For example:

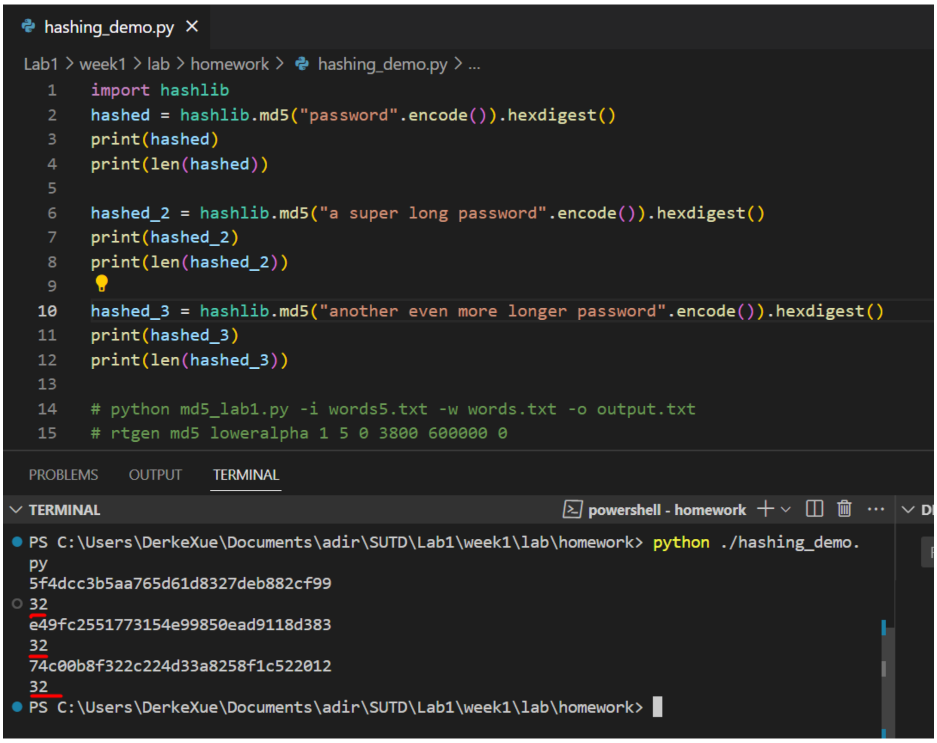


```
C:\Users\MSSD>python
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import hashlib
>>> hashlib.md5("password".encode()).hexdigest()
'5f4dcc3b5aa765d61d8327deb882cf99'
```

You can also use the provided demo file called hashing_demo.py

Observe the length of the output, and whether it depends on length of input.

Answer:



```python
import hashlib
hashed = hashlib.md5("password".encode()).hexdigest()
print(hashed)
print(len(hashed))

hashed_2 = hashlib.md5("a super long password".encode()).hexdigest()
print(hashed_2)
print(len(hashed_2))

hashed_3 = hashlib.md5("another even more longer password".encode()).hexdigest()
print(hashed_3)
print(len(hashed_3))

# python md5_lab1.py -i words5.txt -w words.txt -o output.txt
# rtgen md5 loweralpha 1 5 0 3800 600000 0
```

```
PS C:\Users\DerkeXue\Documents\adir\SUTD\Lab1\week1\lab\homework> python ./hashing_demo.
py
5f4dcc3b5aa765d61d8327deb882cf99
32
e49fc2551773154e99850ead9118d383
32
74c00b8f322c224d33a8258f1c522012
32
PS C:\Users\DerkeXue\Documents\adir\SUTD\Lab1\week1\lab\homework>
```

With different input string length, the hash function will always produce the hash digest of constant length.

So, it’s not depends on the length of the input.

4. Brute-Force VS Dictionary Attack

For this exercise, use the 15 hash values from the <STUDENT_ID>-hash5.txt

Create a Python 3 script called md5_lab1.py, which will find the corresponding input plaintext that was used for making the hash values in file <STUDENT_ID>-hash5.txt.

Consider only passwords with 5 characters (lowercase and/or numeric characters).

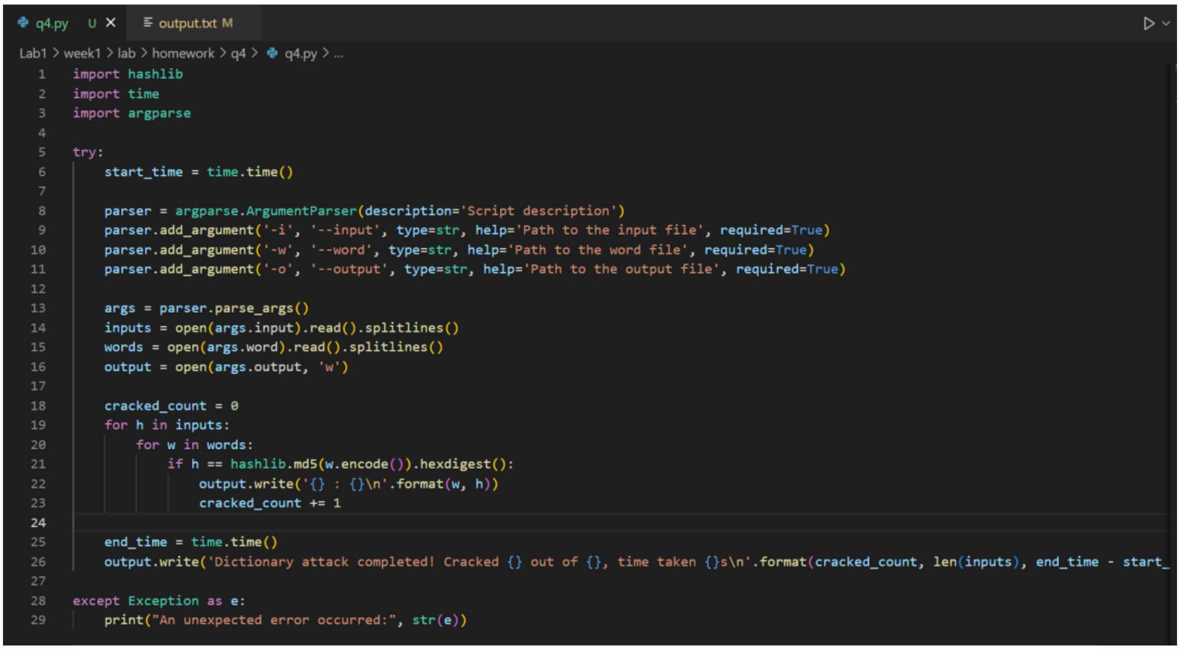
To help reduce the search space we provide a dictionary with newline separated common words in words5.txt. Use the dictionary as the first resort.

It might not be enough to crack all hashed plaintexts, then compute a hash value for each possible combination of lowercase letters, and then with each possible combination from union of all lowercase letters and digits.

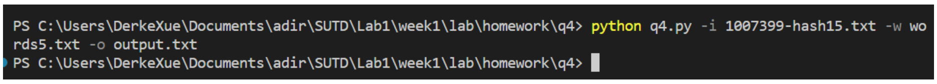
Take note of the computation time of your algorithm to reverse all 15 hashes. You will need it in later step. Consider bash utility time (only for Linux users) or the timeit python module: https://docs.python.org/3.6/library/timeit.html

Final script should support three mandatory arguments:
-i <INPUT_FILE>, -w <DICTIONARY-_FILE>, and -o <OUTPUT_FILE>
Respect the format: one hash/plaintext/dictionary entry per line

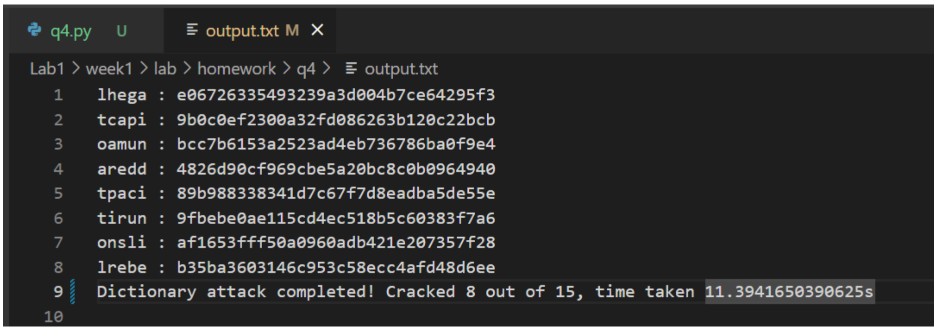
Answer:



```python
import hashlib
import time
import argparse

try:
    start_time = time.time()

    parser = argparse.ArgumentParser(description='Script description')
    parser.add_argument('-i', '--input', type=str, help='Path to the input file', required=True)
    parser.add_argument('-w', '--word', type=str, help='Path to the word file', required=True)
    parser.add_argument('-o', '--output', type=str, help='Path to the output file', required=True)

    args = parser.parse_args()
    inputs = open(args.input).read().splitlines()
    words = open(args.word).read().splitlines()
    output = open(args.output, 'w')

    cracked_count = 0
    for h in inputs:
        for w in words:
            if h == hashlib.md5(w.encode()).hexdigest():
                output.write('{} : {}\n'.format(w, h))
                cracked_count += 1

    end_time = time.time()
    output.write('Dictionary attack completed! Cracked {} out of {}, time taken {}s\n'.format(cracked_count, len(inputs), end_time - start_

except Exception as e:
    print("An unexpected error occurred:", str(e))
```



```
PS C:\Users\DerkeXue\Documents\adir\SUTD\Lab1\week1\lab\homework\q4> python q4.py -i 1007399-hash15.txt -w wo
rds5.txt -o output.txt
PS C:\Users\DerkeXue\Documents\adir\SUTD\Lab1\week1\lab\homework\q4> 
```



```
lhega : e06726335493239a3d004b7ce64295f3
tcapi : 9b0c0ef2300a32fd086263b120c22bcb
oamun : bcc7b6153a2523ad4eb736786ba0f9e4
aredd : 4826d90cf969cbe5a20bc8c0b0964940
tpaci : 89b988338341d7c67f7d8eadba5de55e
tirun : 9fbebe0ae115cd4ec518b5c60383f7a6
onsli : af1653fff50a0960adb421e207357f28
lrebe : b35ba3603146c953c58ecc4afd48d6ee
Dictionary attack completed! Cracked 8 out of 15, time taken 11.3941650390625s
```

(q4.py attached inside zip folder)

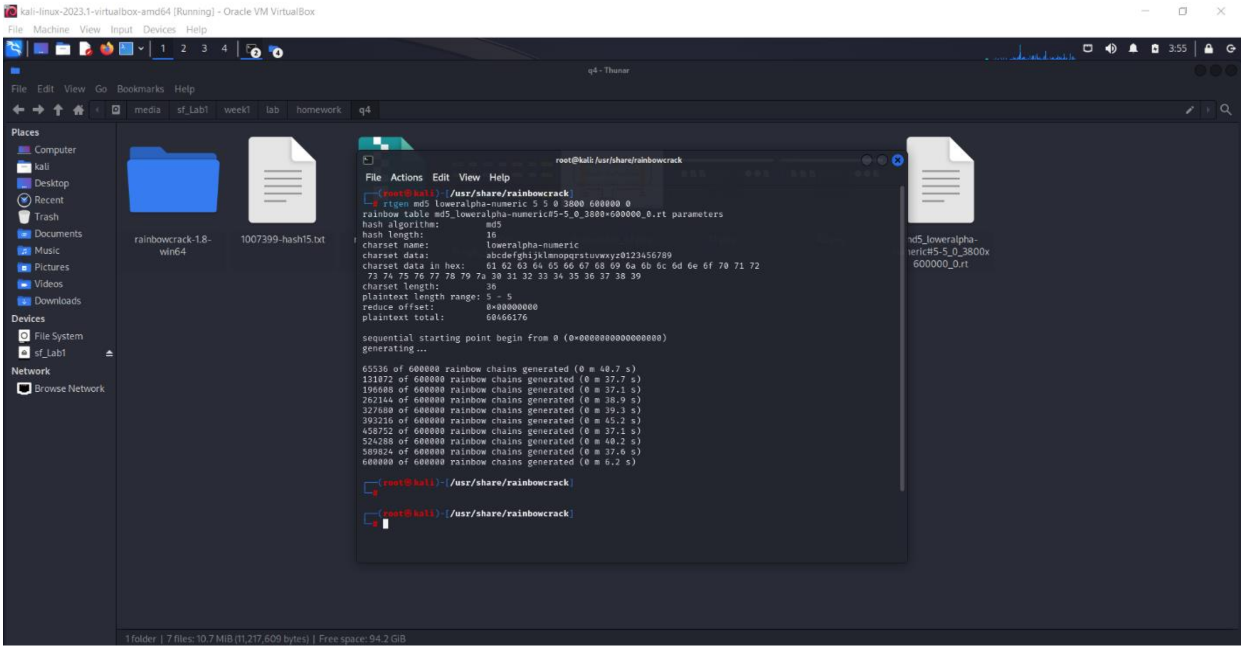
5. Creating Rainbow Tables

Install the program rainbowcrack : http://project-rainbowcrack.com/

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. (d, screen shot)

The chain length and chain number values maybe suboptimal? If yes, find better ones.

Answer 1:



kali-linux-2023.1-virtualbox-amd64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
q4 - Thunar
File Edit View Go Bookmarks Help
media sf_Lab1 week1 lab homework q4
Places
Computer
kali
Desktop
Recent
Trash
Documents
Music
Pictures
Videos
Downloads
Devices
File System
sf_Lab1
Network
Browse Network
rainbowcrack-1.8-win64
1007399-hash15.txt
md5_loweralpha-numeric#5-5_0_3800x600000_0.rt
root@kali: /usr/share/rainbowcrack
File Actions Edit View Help
1 folder | 7 files: 10.7 MiB (11,217,609 bytes) | Free space: 94.2 GiB

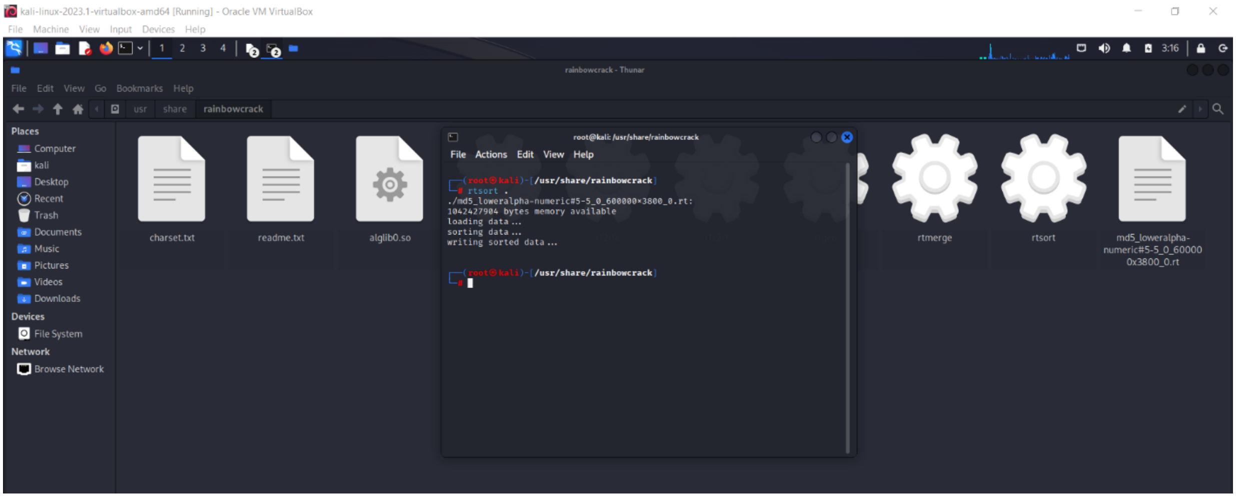
```
┌──(root㉿kali)-[/usr/share/rainbowcrack]
└─# rtgen md5 loweralpha-numeric 5 5 0 3800 600000 0
rainbow table md5_loweralpha-numeric#5-5_0_3800x600000_0.rt parameters
hash algorithm:         md5
hash length:            16
charset name:           loweralpha-numeric
charset data:           abcdefghijklmnopqrstuvwxyz0123456789
charset data in hex:    61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72
 73 74 75 76 77 78 79 7a 30 31 32 33 34 35 36 37 38 39
charset length:         36
plaintext length range: 5 - 5
reduce offset:          0x00000000
plaintext total:        60466176

sequential starting point begin from 0 (0x0000000000000000)
generating...

65536 of 600000 rainbow chains generated (0 m 40.7 s)
131072 of 600000 rainbow chains generated (0 m 37.7 s)
196608 of 600000 rainbow chains generated (0 m 37.1 s)
262144 of 600000 rainbow chains generated (0 m 38.9 s)
327680 of 600000 rainbow chains generated (0 m 39.3 s)
393216 of 600000 rainbow chains generated (0 m 45.2 s)
458752 of 600000 rainbow chains generated (0 m 37.1 s)
524288 of 600000 rainbow chains generated (0 m 40.2 s)
589824 of 600000 rainbow chains generated (0 m 37.6 s)
600000 of 600000 rainbow chains generated (0 m 6.2 s)

┌──(root㉿kali)-[/usr/share/rainbowcrack]
└─#

┌──(root㉿kali)-[/usr/share/rainbowcrack]
└─# 
```

yes, Chain length is 3800, Chain number is 600000 is suboptimal,

base on the better one can find is Chain length of 1900, Chain number of 300000 with better time value ratio baes on 2nd result screen shoot in answer 4.

Use rtsort to sort the rainbow table to make searchable by rcrack.

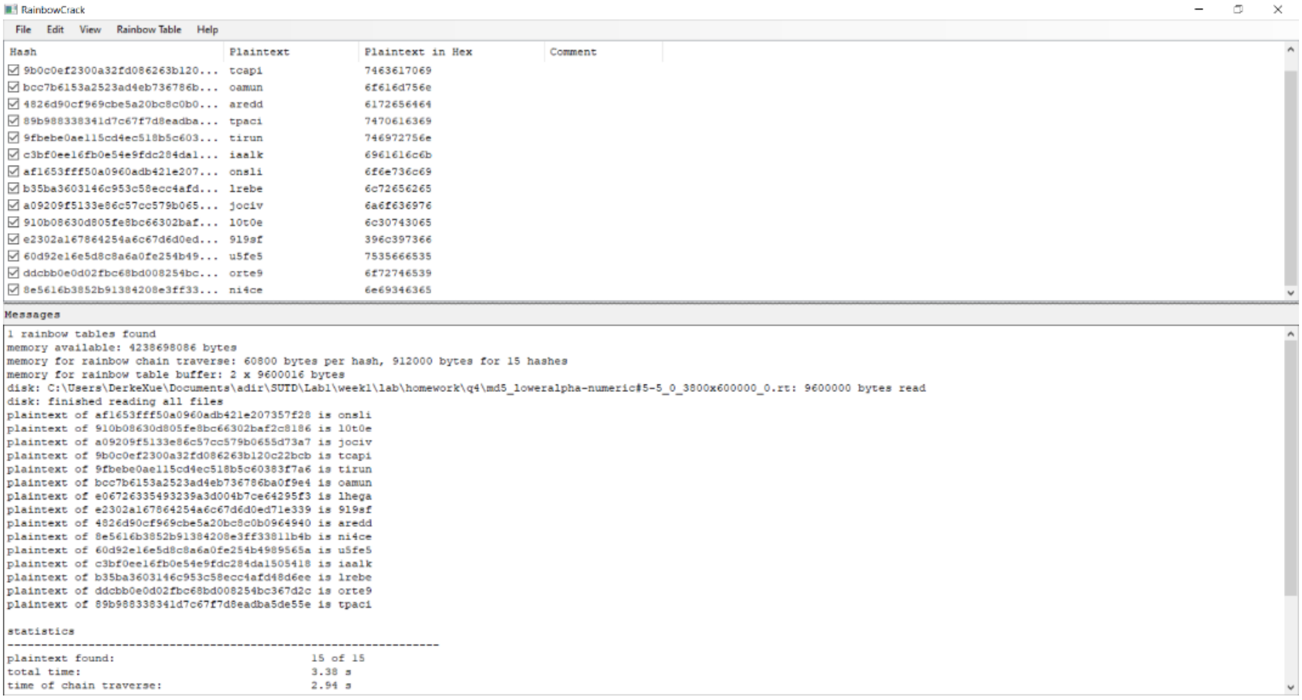
Answer 2:


kali-linux-2023.1-virtualbox-amd64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
rainbowcrack - Thunar
File Edit View Go Bookmarks Help
usr share rainbowcrack
Places
Computer
kali
Desktop
Recent
Trash
Documents
Music
Pictures
Videos
Downloads
Devices
File System
Network
Browse Network
charset.txt
readme.txt
alglib0.so
rtmerge
rtsort
md5_loweralpha-numeric#5-5_0_600000x3800_0.rt
3:16
root@kali: /usr/share/rainbowcrack
File Actions Edit View Help

```
┌──(root㉿kali)-[/usr/share/rainbowcrack]
└─# rtsort .
./md5_loweralpha-numeric#5-5_0_600000×3800_0.rt:
1042427904 bytes memory available
loading data ...
sorting data ...
writing sorted data ...


┌──(root㉿kali)-[/usr/share/rainbowcrack]
└─# 
```

Use rcrack (http://project-rainbowcrack.com/crack.htm) to crack the list of fifteen passwords from hash5.txt

Answer 3:



Hash	Plaintext	Plaintext in Hex	Comment
☑ 9b0c0ef2300a32fd086263b120...	tcapi	7463617069	
☑ bcc7b6153a2523ad4eb736786b...	oamun	6f616d756e	
☑ 4826d90cf969cbe5a20bc8c0b0...	aredd	6172656464	
☑ 89b988338341d7c67f7d8eadba...	tpaci	7470616369	
☑ 9fbebe0ae115cd4ec518b5c603...	tirun	746972756e	
☑ c3bf0ee16fb0e54e9fdc284da1...	iaalk	6961616c6b	
☑ af1653fff50a0960adb421e207...	onsli	6f6e736c69	
☑ b35ba3603146c953c58ecc4afd...	lrebe	6c72656265	
☑ a09209f5133e86c57cc579b065...	jociv	6a6f636976	
☑ 910b08630d805fe8bc66302baf...	l0t0e	6c30743065	
☑ e2302a167864254a6c67d6d0ed...	9l9sf	396c397366	
☑ 60d92e16e5d8c8a6a0fe254b49...	u5fe5	7535666535	
☑ ddcbb0e0d02fbc68bd008254bc...	orte9	6f72746539	
☑ 8e5616b3852b91384208e3ff33...	ni4ce	6e69346365	

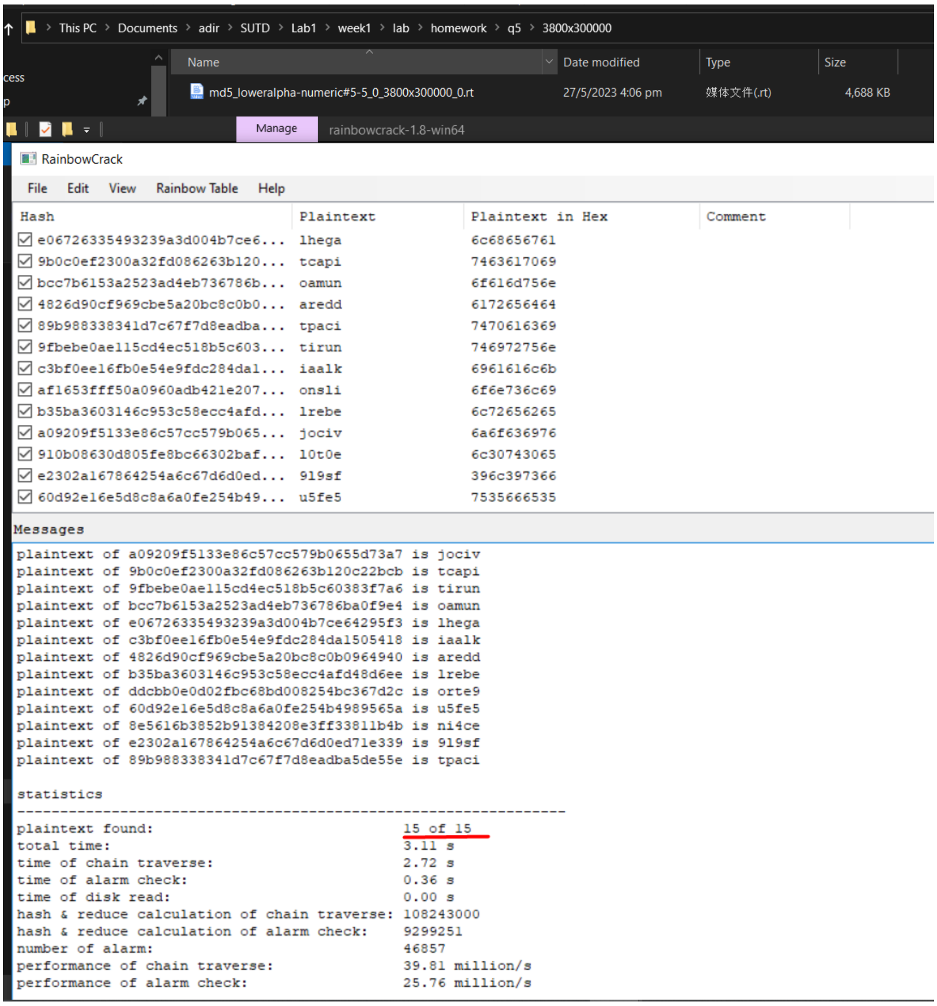
Messages

```
1 rainbow tables found
memory available: 4238698086 bytes
memory for rainbow chain traverse: 60800 bytes per hash, 912000 bytes for 15 hashes
memory for rainbow table buffer: 2 x 9600016 bytes
disk: C:\Users\DerkeXue\Documents\adir\SUTD\Lab1\week1\lab\homework\q4\md5_loweralpha-numeric#5-5_0_3800x600000_0.rt: 9600000 bytes read
disk: finished reading all files
plaintext of af1653fff50a0960adb421e207357f28 is onsli
plaintext of 910b08630d805fe8bc66302baf2c8186 is l0t0e
plaintext of a09209f5133e86c57cc579b0655d73a7 is jociv
plaintext of 9b0c0ef2300a32fd086263b120c22bcb is tcapi
plaintext of 9fbebe0ae115cd4ec518b5c60383f7a6 is tirun
plaintext of bcc7b6153a2523ad4eb736786ba0f9e4 is oamun
plaintext of e06726335493239a3d004b7ce64295f3 is lhega
plaintext of e2302a167864254a6c67d6d0ed71e339 is 9l9sf
plaintext of 4826d90cf969cbe5a20bc8c0b0964940 is aredd
plaintext of 8e5616b3852b91384208e3ff33811b4b is ni4ce
plaintext of 60d92e16e5d8c8a6a0fe254b4989565a is u5fe5
plaintext of c3bf0ee16fb0e54e9fdc284da1505418 is iaalk
plaintext of b35ba3603146c953c58ecc4afd48d6ee is lrebe
plaintext of ddcbb0e0d02fbc68bd008254bc367d2c is orte9
plaintext of 89b988338341d7c67f7d8eadba5de55e is tpaci

statistics
----------------------------------------------------------------
plaintext found:                          15 of 15
total time:                               3.38 s
time of chain traverse:                   2.94 s
```

You can see that there are 35^5 ~= 60M combinations for brute force attack.
and the number of plaintexts covered by rainbow table is 3:800 600:000 = 2:280:000:000. So, the ratio is ~ 40:1. Try to decrease size of the rainbow table e.g., to ratio 20, 10 and 5, and observe whether all hashes are cracked (use chain number parameter).

Answer 4:
∵
– Chain length is 3800.
– Chain number is 600000.
=> ratio is ~ 40:1
∴

– Chain length is 3800.
– Chain number is 300000.
=> ratio is ~ 20:1



This PC › Documents › adir › SUTD › Lab1 › week1 › lab › homework › q5 › 3800x300000

Name	Date modified	Type	Size
md5_loweralpha-numeric#5-5_0_3800x300000_0.rt	27/5/2023 4:06 pm	媒体文件(.rt)	4,688 KB

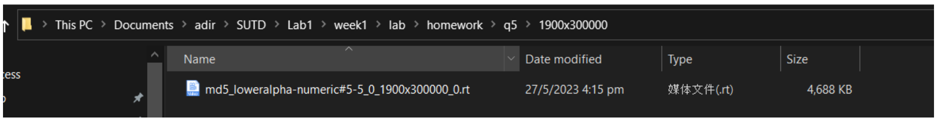
Hash	Plaintext	Plaintext in Hex	Comment
☑ e06726335493239a3d004b7ce6...	lhega	6c68656761	
☑ 9b0c0ef2300a32fd086263b120...	tcapi	7463617069	
☑ bcc7b6153a2523ad4eb736786b...	oamun	6f616d756e	
☑ 4826d90cf969cbe5a20bc8c0b0...	aredd	6172656464	
☑ 89b988338341d7c67f7d8eadba...	tpaci	7470616369	
☑ 9fbebe0ae115cd4ec518b5c603...	tirun	746972756e	
☑ c3bf0ee16fb0e54e9fdc284da1...	iaalk	6961616c6b	
☑ af1653fff50a0960adb421e207...	onsli	6f6e736c69	
☑ b35ba3603146c953c58ecc4afd...	lrebe	6c72656265	
☑ a09209f5133e86c57cc579b065...	jociv	6a6f636976	
☑ 910b08630d805fe8bc66302baf...	l0t0e	6c30743065	
☑ e2302a167864254a6c67d6d0ed...	919sf	396c397366	
☑ 60d92e16e5d8c8a6a0fe254b49...	u5fe5	7535666535	

Messages

```
plaintext of a09209f5133e86c57cc579b0655d73a7 is jociv
plaintext of 9b0c0ef2300a32fd086263b120c22bcb is tcapi
plaintext of 9fbebe0ae115cd4ec518b5c60383f7a6 is tirun
plaintext of bcc7b6153a2523ad4eb736786ba0f9e4 is oamun
plaintext of e06726335493239a3d004b7ce64295f3 is lhega
plaintext of c3bf0ee16fb0e54e9fdc284da1505418 is iaalk
plaintext of 4826d90cf969cbe5a20bc8c0b0964940 is aredd
plaintext of b35ba3603146c953c58ecc4afd48d6ee is lrebe
plaintext of ddcbb0e0d02fbc68bd008254bc367d2c is orte9
plaintext of 60d92e16e5d8c8a6a0fe254b4989565a is u5fe5
plaintext of 8e5616b3852b91384208e3ff33811b4b is ni4ce
plaintext of e2302a167864254a6c67d6d0ed71e339 is 919sf
plaintext of 89b988338341d7c67f7d8eadba5de55e is tpaci

statistics
----------------------------------------------------------------
plaintext found:                          15 of 15
total time:                               3.11 s
time of chain traverse:                   2.72 s
time of alarm check:                      0.36 s
time of disk read:                        0.00 s
hash & reduce calculation of chain traverse: 108243000
hash & reduce calculation of alarm check:    9299251
number of alarm:                          46857
performance of chain traverse:            39.81 million/s
performance of alarm check:               25.76 million/s
```

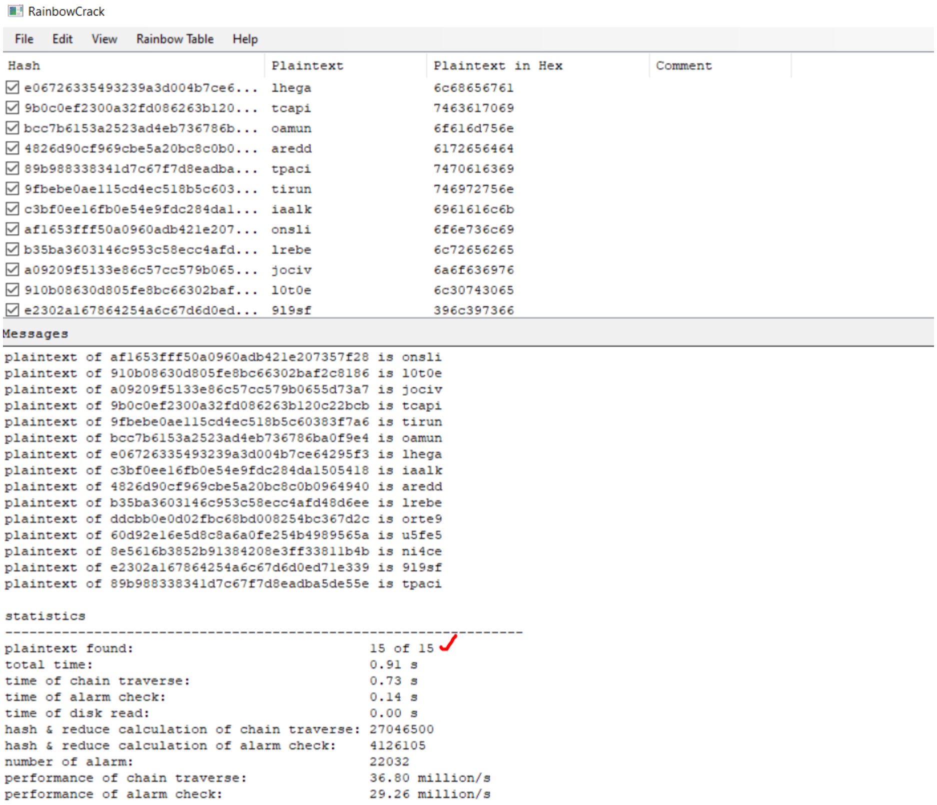
∴

– Chain length is 1900.
– Chain number is 300000.
=> ratio is ~ 10:1



This PC › Documents › adir › SUTD › Lab1 › week1 › lab › homework › q5 › 1900x300000

Name	Date modified	Type	Size
md5_loweralpha-numeric#5-5_0_1900x300000_0.rt	27/5/2023 4:15 pm	媒体文件(.rt)	4,688 KB



Hash	Plaintext	Plaintext in Hex	Comment
☑ e06726335493239a3d004b7ce6...	lhega	6c68656761	
☑ 9b0c0ef2300a32fd086263b120...	tcapi	7463617069	
☑ bcc7b6153a2523ad4eb736786b...	oamun	6f616d756e	
☑ 4826d90cf969cbe5a20bc8c0b0...	aredd	6172656464	
☑ 89b988338341d7c67f7d8eadba...	tpaci	7470616369	
☑ 9fbebe0ae115cd4ec518b5c603...	tirun	746972756e	
☑ c3bf0ee16fb0e54e9fdc284da1...	iaalk	6961616c6b	
☑ af1653fff50a0960adb421e207...	onsli	6f6e736c69	
☑ b35ba3603146c953c58ecc4afd...	lrebe	6c72656265	
☑ a09209f5133e86c57cc579b065...	jociv	6a6f636976	
☑ 910b08630d805fe8bc66302baf...	l0t0e	6c30743065	
☑ e2302a167864254a6c67d6d0ed...	919sf	396c397366	

Messages

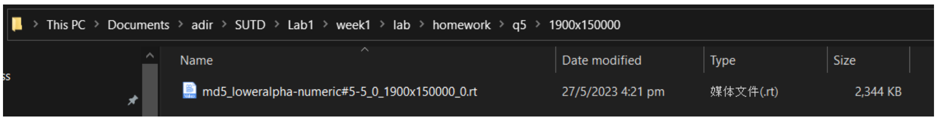
```
plaintext of af1653fff50a0960adb421e207357f28 is onsli
plaintext of 910b08630d805fe8bc66302baf2c8186 is l0t0e
plaintext of a09209f5133e86c57cc579b0655d73a7 is jociv
plaintext of 9b0c0ef2300a32fd086263b120c22bcb is tcapi
plaintext of 9fbebe0ae115cd4ec518b5c60383f7a6 is tirun
plaintext of bcc7b6153a2523ad4eb736786ba0f9e4 is oamun
plaintext of e06726335493239a3d004b7ce64295f3 is lhega
plaintext of c3bf0ee16fb0e54e9fdc284da1505418 is iaalk
plaintext of 4826d90cf969cbe5a20bc8c0b0964940 is aredd
plaintext of b35ba3603146c953c58ecc4afd48d6ee is lrebe
plaintext of ddcbb0e0d02fbc68bd008254bc367d2c is orte9
plaintext of 60d92e16e5d8c8a6a0fe254b4989565a is u5fe5
plaintext of 8e5616b3852b91384208e3ff33811b4b is ni4ce
plaintext of e2302a167864254a6c67d6d0ed71e339 is 919sf
plaintext of 89b988338341d7c67f7d8eadba5de55e is tpaci

statistics
-------------------------------------------------------------
plaintext found:                            15 of 15 ✓
total time:                                 0.91 s
time of chain traverse:                     0.73 s
time of alarm check:                        0.14 s
time of disk read:                          0.00 s
hash & reduce calculation of chain traverse: 27046500
hash & reduce calculation of alarm check:   4126105
number of alarm:                            22032
performance of chain traverse:              36.80 million/s
performance of alarm check:                 29.26 million/s
```

∴

– Chain length is 1900.

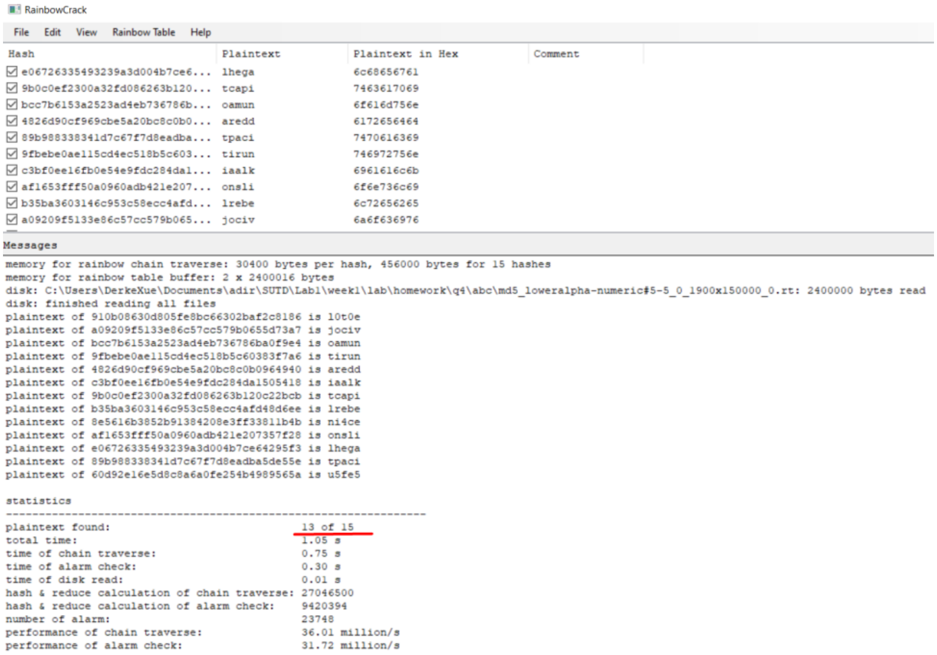
– Chain number is 150000.

=> ratio is ~ 5:1



This PC > Documents > adir > SUTD > Lab1 > week1 > lab > homework > q5 > 1900x150000

Name	Date modified	Type	Size
md5_loweralpha-numeric#5-5_0_1900x150000_0.rt	27/5/2023 4:21 pm	媒体文件(.rt)	2,344 KB



Hash	Plaintext	Plaintext in Hex	Comment
☑ e06726335493239a3d004b7ce6...	lhega	6c68656761	
☑ 9b0c0ef2300a32fd086263b120...	tcapi	7463617069	
☑ bcc7b6153a2523ad4eb736786b...	oamun	6f616d756e	
☑ 4826d90cf969cbe5a20bc8c0b0...	aredd	6172656464	
☑ 89b988338341d7c67f7d8eadba...	tpaci	7470616369	
☑ 9fbebe0ae115cd4ec518b5c603...	tirun	746972756e	
☑ c3bf0ee16fb0e54e9fdc284da1...	iaalk	6961616c6b	
☑ af1653fff50a0960adb421e207...	onsli	6f6e736c69	
☑ b35ba3603146c953c58ecc4afd...	lrebe	6c72656265	
☑ a09209f5133e86c57cc579b065...	jociv	6a6f636976	

Messages

```
memory for rainbow chain traverse: 30400 bytes per hash, 456000 bytes for 15 hashes
memory for rainbow table buffer: 2 x 2400016 bytes
disk: C:\Users\DerkeXue\Documents\adir\SUTD\Lab1\week1\lab\homework\q4\abc\md5_loweralpha-numeric#5-5_0_1900x150000_0.rt: 2400000 bytes read
disk: finished reading all files
plaintext of 910b08630d805fe8bc66302baf2c8186 is l0t0e
plaintext of a09209f5133e86c57cc579b0655d73a7 is jociv
plaintext of bcc7b6153a2523ad4eb736786ba0f9e4 is oamun
plaintext of 9fbebe0ae115cd4ec518b5c60383f7a6 is tirun
plaintext of 4826d90cf969cbe5a20bc8c0b0964940 is aredd
plaintext of c3bf0ee16fb0e54e9fdc284da1505418 is iaalk
plaintext of 9b0c0ef2300a32fd086263b120c22bcb is tcapi
plaintext of b35ba3603146c953c58ecc4afd48d6ee is lrebe
plaintext of 8e5616b3852b91384208e3ff33811b4b is ni4ce
plaintext of af1653fff50a0960adb421e207357f28 is onsli
plaintext of e06726335493239a3d004b7ce64295f3 is lhega
plaintext of 89b988338341d7c67f7d8eadba5de55e is tpaci
plaintext of 60d92e16e5d8c8a6a0fe254b4989565a is u5fe5

statistics
-------------------------------------------------------------
plaintext found:                            13 of 15
total time:                                 1.05 s
time of chain traverse:                     0.75 s
time of alarm check:                        0.30 s
time of disk read:                          0.01 s
hash & reduce calculation of chain traverse: 27046500
hash & reduce calculation of alarm check:   9420394
number of alarm:                            23748
performance of chain traverse:              36.01 million/s
performance of alarm check:                 31.72 million/s
```

6. Salting

Extend your Python script to 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.

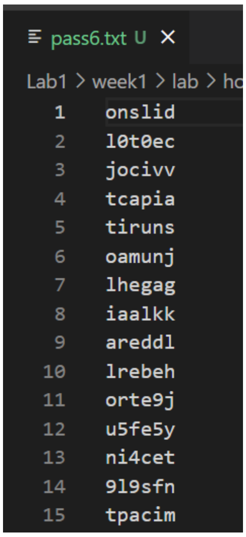
Rehash the password using MD5, and store the newly hashed passwords into a new file called salted6.txt (remember to store the new password as well, maybe in a pass6.txt file). The functional definition of our salt strategy is the following: saltedhash(password) = hash(password||salt), where operator || represents concatenation.

Generate a new rainbow table using **rtgen** (with new parameters) to break the hash values. As before, sort the table using **rtsort**.

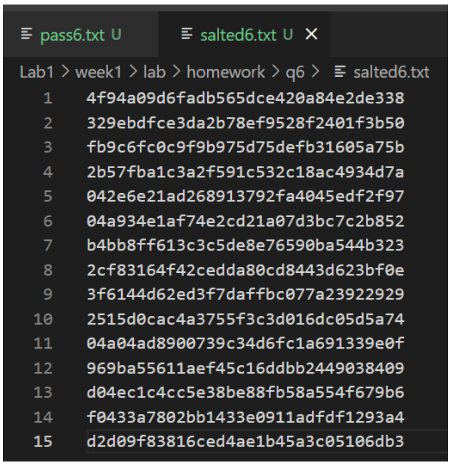
Compare the timing of the new table generation and lookup vs the previous values Try to break as many salted hashes as possible.

In your writeup explain the differences between salted and non-salted rcrack strategies and compare the timings.

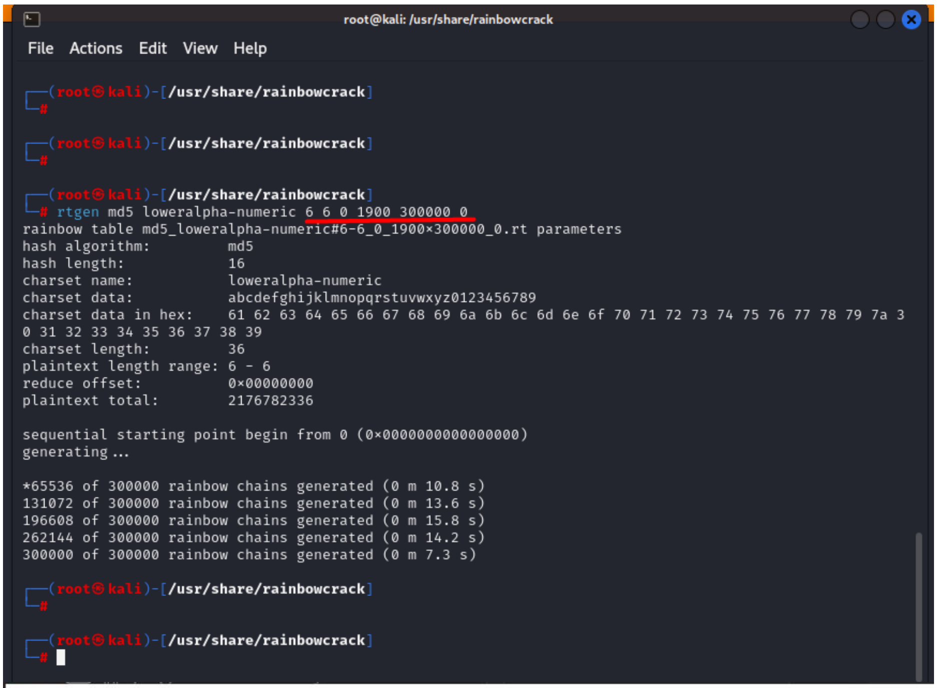
Answer:



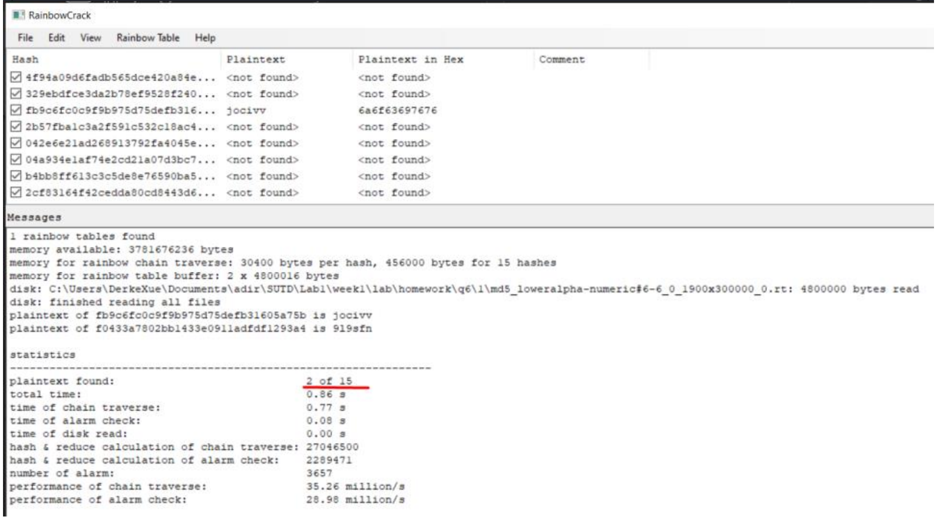
```
onslid
l0t0ec
jocivv
tcapia
tiruns
oamunj
lhegag
iaalkk
areddl
lrebeh
orte9j
u5fe5y
ni4cet
9l9sfn
tpacim
```



```
4f94a09d6fadb565dce420a84e2de338
329ebdfce3da2b78ef9528f2401f3b50
fb9c6fc0c9f9b975d75defb31605a75b
2b57fba1c3a2f591c532c18ac4934d7a
042e6e21ad268913792fa4045edf2f97
04a934e1af74e2cd21a07d3bc7c2b852
b4bb8ff613c3c5de8e76590ba544b323
2cf83164f42cedda80cd8443d623bf0e
3f6144d62ed3f7daffbc077a23922929
2515d0cac4a3755f3c3d016dc05d5a74
04a04ad8900739c34d6fc1a691339e0f
969ba55611aef45c16ddbb2449038409
d04ec1c4cc5e38be88fb58a554f679b6
f0433a7802bb1433e0911adfdf1293a4
d2d09f83816ced4ae1b45a3c05106db3
```


root@kali: /usr/share/rainbowcrack
File Actions Edit View Help

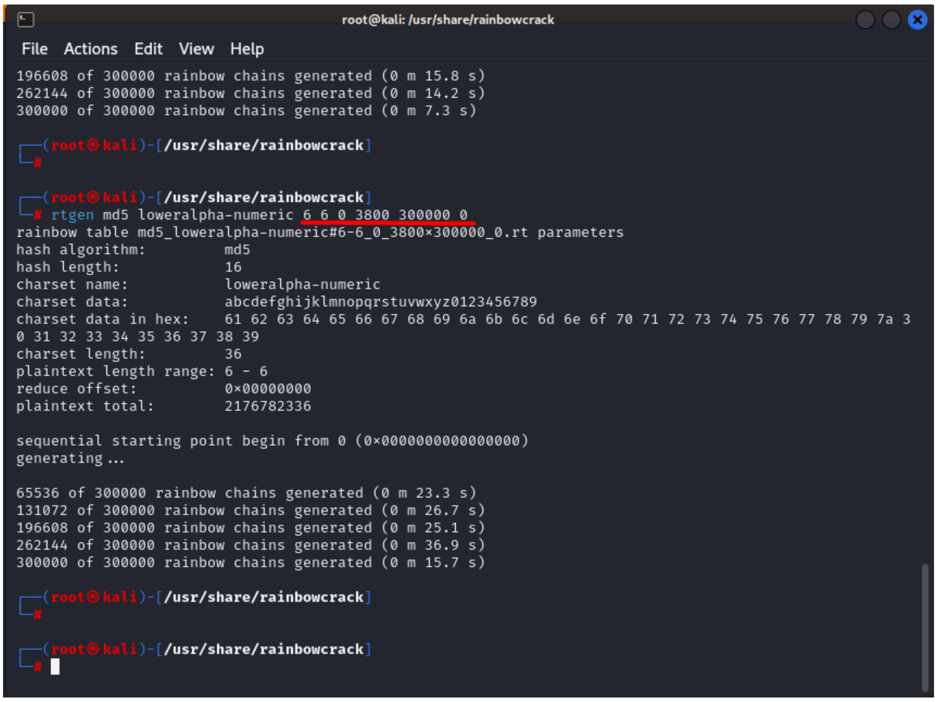
```
┌──(root㉿kali)-[/usr/share/rainbowcrack]
└─#

┌──(root㉿kali)-[/usr/share/rainbowcrack]
└─#

┌──(root㉿kali)-[/usr/share/rainbowcrack]
└─# rtgen md5 loweralpha-numeric 6 6 0 1900 300000 0
rainbow table md5_loweralpha-numeric#6-6_0_1900×300000_0.rt parameters
hash algorithm:         md5
hash length:            16
charset name:           loweralpha-numeric
charset data:           abcdefghijklmnopqrstuvwxyz0123456789
charset data in hex:    61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76 77 78 79 7a 3
0 31 32 33 34 35 36 37 38 39
charset length:         36
plaintext length range: 6 - 6
reduce offset:          0×00000000
plaintext total:        2176782336

sequential starting point begin from 0 (0×0000000000000000)
generating...

*65536 of 300000 rainbow chains generated (0 m 10.8 s)
131072 of 300000 rainbow chains generated (0 m 13.6 s)
196608 of 300000 rainbow chains generated (0 m 15.8 s)
262144 of 300000 rainbow chains generated (0 m 14.2 s)
300000 of 300000 rainbow chains generated (0 m 7.3 s)

┌──(root㉿kali)-[/usr/share/rainbowcrack]
└─#

┌──(root㉿kali)-[/usr/share/rainbowcrack]
└─# █
```



Hash	Plaintext	Plaintext in Hex	Comment
☑ 4f94a09d6fadb565dce420a84e...	<not found>	<not found>	
☑ 329ebdfce3da2b78ef9528f240...	<not found>	<not found>	
☑ fb9c6fc0c9f9b975d75defb316...	jocivv	6a6f63697676	
☑ 2b57fba1c3a2f591c532c18ac4...	<not found>	<not found>	
☑ 042e6e21ad268913792fa4045e...	<not found>	<not found>	
☑ 04a934e1af74e2cd21a07d3bc7...	<not found>	<not found>	
☑ b4bb8ff613c3c5de8e76590ba5...	<not found>	<not found>	
☑ 2cf83164f42cedda80cd8443d6...	<not found>	<not found>	

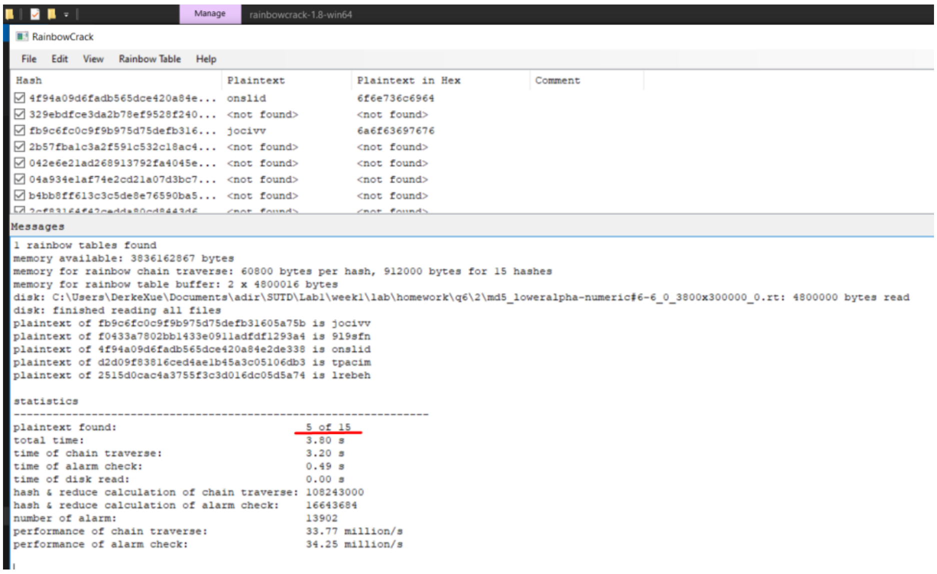
Messages

```
1 rainbow tables found
memory available: 3781676236 bytes
memory for rainbow chain traverse: 30400 bytes per hash, 456000 bytes for 15 hashes
memory for rainbow table buffer: 2 x 4800016 bytes
disk: C:\Users\DerkeXue\Documents\adir\SUTD\Lab1\week1\lab\homework\q6\1\md5_loweralpha-numeric#6-6_0_1900x300000_0.rt: 4800000 bytes read
disk: finished reading all files
plaintext of fb9c6fc0c9f9b975d75defb31605a75b is jocivv
plaintext of f0433a7802bb1433e0911adfdf1293a4 is 919sfn

statistics
----------------------------------------------------------------
plaintext found:                              2 of 15
total time:                                   0.86 s
time of chain traverse:                       0.77 s
time of alarm check:                          0.08 s
time of disk read:                            0.00 s
hash & reduce calculation of chain traverse:  27046500
hash & reduce calculation of alarm check:     2289471
number of alarm:                              3657
performance of chain traverse:                35.26 million/s
performance of alarm check:                   28.98 million/s
```



root@kali: /usr/share/rainbowcrack
File Actions Edit View Help

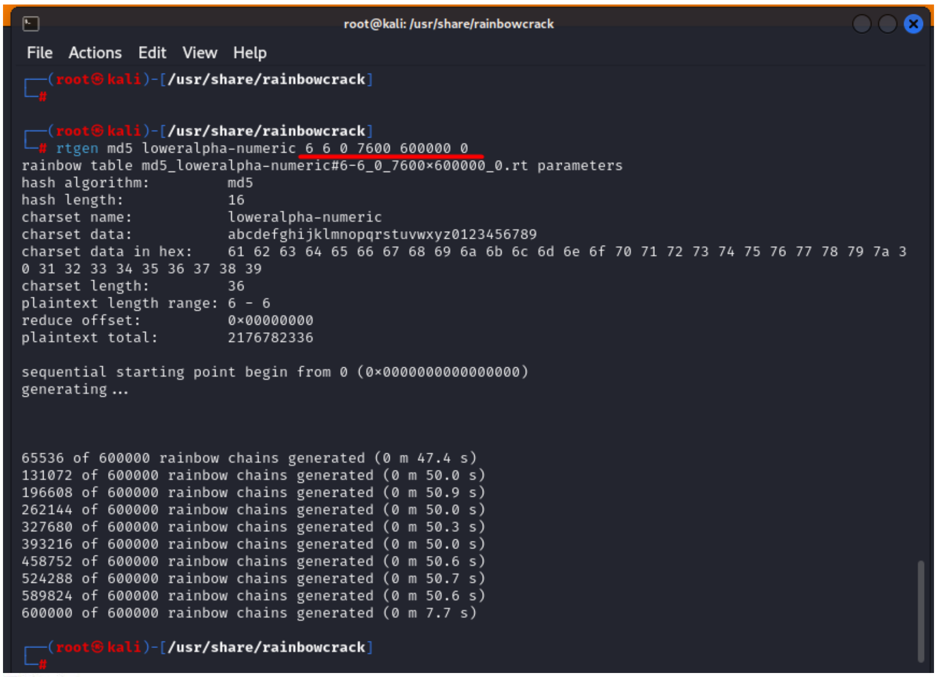
```
196608 of 300000 rainbow chains generated (0 m 15.8 s)
262144 of 300000 rainbow chains generated (0 m 14.2 s)
300000 of 300000 rainbow chains generated (0 m 7.3 s)

┌──(root㉿kali)-[/usr/share/rainbowcrack]
└─#

┌──(root㉿kali)-[/usr/share/rainbowcrack]
└─# rtgen md5 loweralpha-numeric 6 6 0 3800 300000 0
rainbow table md5_loweralpha-numeric#6-6_0_3800x300000_0.rt parameters
hash algorithm:         md5
hash length:            16
charset name:           loweralpha-numeric
charset data:           abcdefghijklmnopqrstuvwxyz0123456789
charset data in hex:    61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76 77 78 79 7a 3
0 31 32 33 34 35 36 37 38 39
charset length:         36
plaintext length range: 6 - 6
reduce offset:          0x00000000
plaintext total:        2176782336

sequential starting point begin from 0 (0x0000000000000000)
generating...

65536 of 300000 rainbow chains generated (0 m 23.3 s)
131072 of 300000 rainbow chains generated (0 m 26.7 s)
196608 of 300000 rainbow chains generated (0 m 25.1 s)
262144 of 300000 rainbow chains generated (0 m 36.9 s)
300000 of 300000 rainbow chains generated (0 m 15.7 s)

┌──(root㉿kali)-[/usr/share/rainbowcrack]
└─#

┌──(root㉿kali)-[/usr/share/rainbowcrack]
└─# 
```



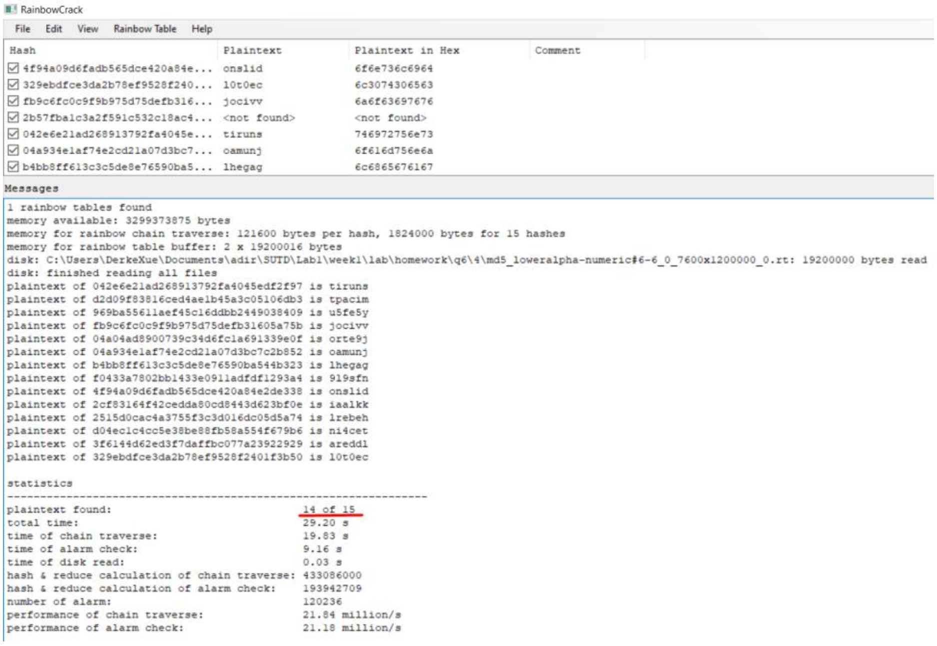
Hash	Plaintext	Plaintext in Hex	Comment
☑ 4f94a09d6fadb565dce420a84e...	onslid	6f6e736c6964	
☑ 329ebdfce3da2b78ef9528f240...	<not found>	<not found>	
☑ fb9c6fc0c9f9b975d75defb316...	jocivv	6a6f63697676	
☑ 2b57fba1c3a2f591c532c18ac4...	<not found>	<not found>	
☑ 042e6e21ad268913792fa4045e...	<not found>	<not found>	
☑ 04a934e1af74e2cd21a07d3bc7...	<not found>	<not found>	
☑ b4bb8ff613c3c5de8e76590ba5...	<not found>	<not found>	
☑ 2cf83164f42cedda80cd8443d6...	<not found>	<not found>	

Messages

```
1 rainbow tables found
memory available: 3836162867 bytes
memory for rainbow chain traverse: 60800 bytes per hash, 912000 bytes for 15 hashes
memory for rainbow table buffer: 2 x 4800016 bytes
disk: C:\Users\DerkeXue\Documents\adir\SUTD\Lab1\week1\lab\homework\q6\2\md5_loweralpha-numeric#6-6_0_3800x300000_0.rt: 4800000 bytes read
disk: finished reading all files
plaintext of fb9c6fc0c9f9b975d75defb31605a75b is jocivv
plaintext of f0433a7802bb1433e0911adfdf1293a4 is 919sfn
plaintext of 4f94a09d6fadb565dce420a84e2de338 is onslid
plaintext of d2d09f83816ced4ae1b45a3c05106db3 is tpacim
plaintext of 2515d0cac4a3755f3c3d016dc05d5a74 is lrebeh

statistics
----------------------------------------------------------------
plaintext found:                          5 of 15
total time:                               3.80 s
time of chain traverse:                   3.20 s
time of alarm check:                      0.49 s
time of disk read:                        0.00 s
hash & reduce calculation of chain traverse: 108243000
hash & reduce calculation of alarm check:    16643684
number of alarm:                          13902
performance of chain traverse:            33.77 million/s
performance of alarm check:               34.25 million/s
```


root@kali: /usr/share/rainbowcrack
File Actions Edit View Help

```
┌──(root㉿kali)-[/usr/share/rainbowcrack]
└─#

┌──(root㉿kali)-[/usr/share/rainbowcrack]
└─# rtgen md5 loweralpha-numeric 6 6 0 7600 600000 0
rainbow table md5_loweralpha-numeric#6-6_0_7600×600000_0.rt parameters
hash algorithm:         md5
hash length:            16
charset name:           loweralpha-numeric
charset data:           abcdefghijklmnopqrstuvwxyz0123456789
charset data in hex:    61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76 77 78 79 7a 3
0 31 32 33 34 35 36 37 38 39
charset length:         36
plaintext length range: 6 - 6
reduce offset:          0×00000000
plaintext total:        2176782336

sequential starting point begin from 0 (0×0000000000000000)
generating ...



65536 of 600000 rainbow chains generated (0 m 47.4 s)
131072 of 600000 rainbow chains generated (0 m 50.0 s)
196608 of 600000 rainbow chains generated (0 m 50.9 s)
262144 of 600000 rainbow chains generated (0 m 50.0 s)
327680 of 600000 rainbow chains generated (0 m 50.3 s)
393216 of 600000 rainbow chains generated (0 m 50.0 s)
458752 of 600000 rainbow chains generated (0 m 50.6 s)
524288 of 600000 rainbow chains generated (0 m 50.7 s)
589824 of 600000 rainbow chains generated (0 m 50.6 s)
600000 of 600000 rainbow chains generated (0 m 7.7 s)

┌──(root㉿kali)-[/usr/share/rainbowcrack]
└─#
```



Hash	Plaintext	Plaintext in Hex	Comment
☑ 4f94a09d6fadb565dce420a84e...	onslid	6f6e736c6964	
☑ 329ebdfce3da2b78ef9528f240...	10t0ec	6c3074306563	
☑ fb9c6fc0c9f9b975d75defb316...	jocivv	6a6f63697676	
☑ 2b57fba1c3a2f591c532c18ac4...	<not found>	<not found>	
☑ 042e6e21ad268913792fa4045e...	tiruns	746972756e73	
☑ 04a934e1af74e2cd21a07d3bc7...	oamunj	6f616d756e6a	
☑ b4bb8ff613c3c5de8e76590ba5...	lhegag	6c6865676167	

Messages

```
1 rainbow tables found
memory available: 3299373875 bytes
memory for rainbow chain traverse: 121600 bytes per hash, 1824000 bytes for 15 hashes
memory for rainbow table buffer: 2 x 19200016 bytes
disk: C:\Users\DerkeXue\Documents\adir\SUTD\Lab1\week1\lab\homework\q6\4\md5_loweralpha-numeric#6-6_0_7600x1200000_0.rt: 19200000 bytes read
disk: finished reading all files
plaintext of 042e6e21ad268913792fa4045edf2f97 is tiruns
plaintext of d2d09f83816ced4ae1b45a3c05106db3 is tpacim
plaintext of 969ba55611aef45c16ddbb2449038409 is u5fe5y
plaintext of fb9c6fc0c9f9b975d75defb31605a75b is jocivv
plaintext of 04a04ad8900739c34d6fc1a691339e0f is orte9j
plaintext of 04a934e1af74e2cd21a07d3bc7c2b852 is oamunj
plaintext of b4bb8ff613c3c5de8e76590ba544b323 is lhegag
plaintext of f0433a7802bb1433e0911adfdf1293a4 is 919sfn
plaintext of 4f94a09d6fadb565dce420a84e2de338 is onslid
plaintext of 2cf83164f42cedda80cd8443d623bf0e is iaalkk
plaintext of 2515d0cac4a3755f3c3d016dc05d5a74 is lrebeh
plaintext of d04ec1c4cc5e38be88fb58a554f679b6 is ni4cet
plaintext of 3f6144d62ed3f7daffbc077a23922929 is areddl
plaintext of 329ebdfce3da2b78ef9528f2401f3b50 is 10t0ec

statistics
-------------------------------------------------------------------
plaintext found:                            14 of 15
total time:                                 29.20 s
time of chain traverse:                     19.83 s
time of alarm check:                        9.16 s
time of disk read:                          0.03 s
hash & reduce calculation of chain traverse: 433086000
hash & reduce calculation of alarm check:   193942709
number of alarm:                            120236
performance of chain traverse:              21.84 million/s
performance of alarm check:                 21.18 million/s
```

As can see with password with salt of length 6 greatly increase the cracking timing from 0.9s to 29.2s compared to non-salted password of length 5.

7. Hashcat

For windows users,
Download hashcat tool https://hashcat.net/files/hashcat-5.1.0.7z

Extract to a folder and navigate to that folder to find your hashcat executable. For Windows 64 bit, it will be hashcat64.exe

You can test by running the following command in the directory.

○ hashcat -m 0 -a 0 -o cracked.txt target_hashes.txt /usr/share/wordlists/rockyou.txt

Open the file cracked.txt to reveal the cracked hash.

$1$uOM6WNc4$r3ZGeSB11q6UUSILqek3J1:hash234

Here, -m is the mode of hashcat (500 = md5crypt), -a is the attack mode (0 = straight mode), -o is the output file, and then following files contain hash values and dictionary, respectively. Note that you may try to use the -force if hashcat will complain about OpenCL drivers/unsupported OS. See https://hashcat.net/wiki/doku.php?id=hashcat for quick documentation and parameter description.

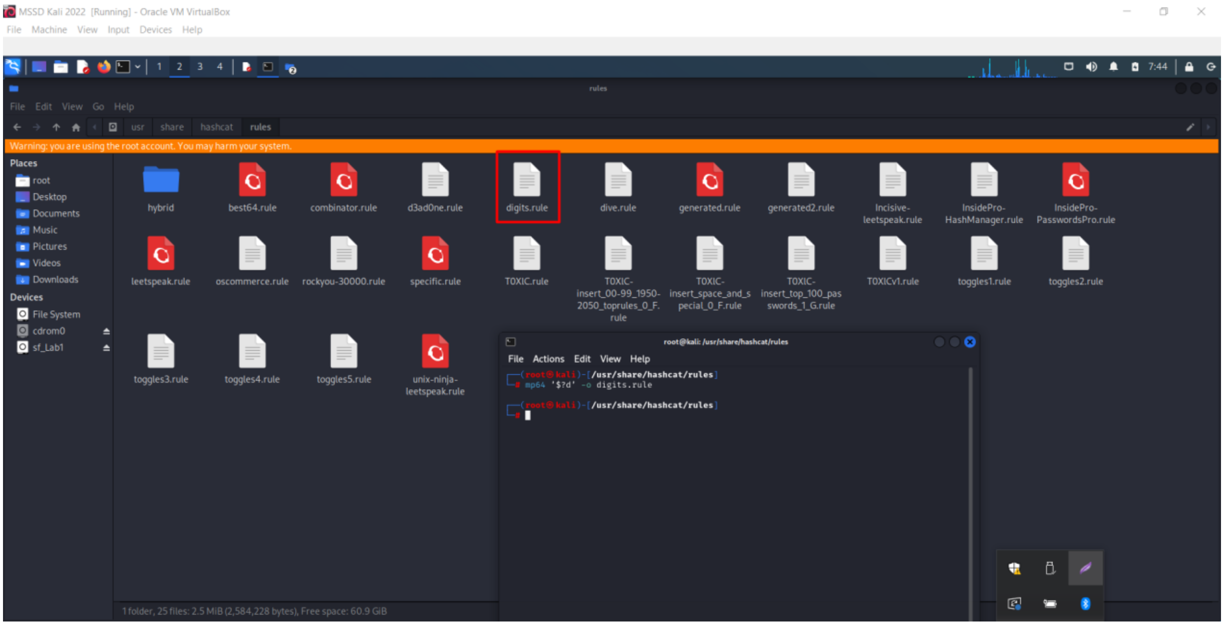
First, try to write simple hashcat **rule-based attack** using dictionary, which will exploit the knowledge of how salting was performed. You should consider -r parameter for rule file and -m equal to 0, as you are cracking raw hash of MD5.

See description for creating rules at URL https://hashcat.net/wiki/doku.php?id=rule_based_attack, and please consider using maskprocessor that can generate rule set according to input mask – https://github.com/hashcat/maskprocessor/releases. For example, to generate rules file covering salted passwords from a dictionary by salting strategy hash(password||any_digit), use:

○ mp64 '$?d' -o digits.rule

where -o specifies output file for generated rules, operator $ represents concatenation, and ?d is the wildcard representing all digits.

Answer:

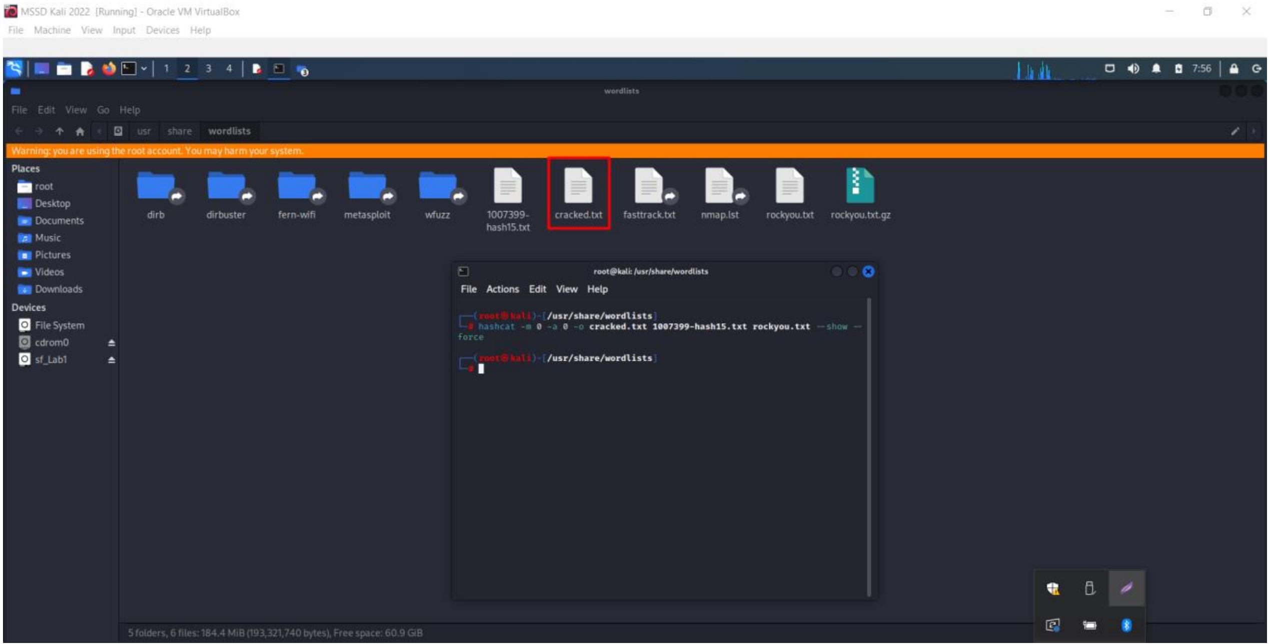

MSSD Kali 2022 [Running] - Oracle VM VirtualBox
rules
usr share hashcat rules
Warning: you are using the root account. You may harm your system.
hybrid
best64.rule
combinator.rule
d3ad0ne.rule
digits.rule
dive.rule
generated.rule
generated2.rule
Incisive-leetspeak.rule
InsidePro-HashManager.rule
InsidePro-PasswordsPro.rule
leetspeak.rule
oscommerce.rule
rockyou-30000.rule
specific.rule
T0XIC.rule
T0XIC-insert_00-99_1950-2050_toprules_0_F.rule
T0XIC-insert_space_and_special_0_F.rule
T0XIC-insert_top_100_passwords_1_G.rule
T0XICv1.rule
toggles1.rule
toggles2.rule
toggles3.rule
toggles4.rule
toggles5.rule
unix-ninja-leetspeak.rule
1 folder, 25 files: 2.5 MiB (2,584,228 bytes), Free space: 60.9 GiB
root@kali: /usr/share/hashcat/rules

```
┌──(root㉿kali)-[/usr/share/hashcat/rules]
└─# mp64 '$?d' -o digits.rule

┌──(root㉿kali)-[/usr/share/hashcat/rules]
└─# 
```

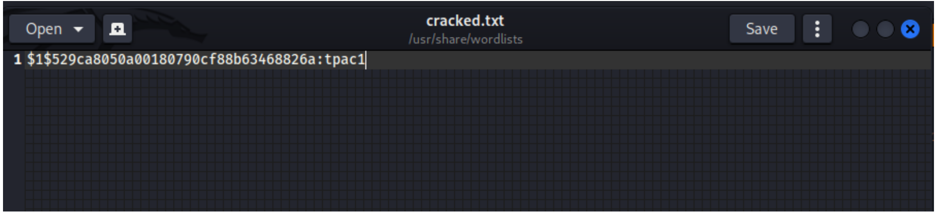
Nice tutorial how to crack password by hashcat using rules is at URL https://labs.mwrinfosecurity.com/blog/a-practical-guide-to-cracking-password-hashes/

How many passwords did you crack? Why you did not crack some passwords?

Answer:


MSSD Kali 2022 [Running] - Oracle VM VirtualBox
wordlists
usr share wordlists
Warning: you are using the root account. You may harm your system.
dirb
dirbuster
fern-wifi
metasploit
wfuzz
1007399-hash15.txt
cracked.txt
fasttrack.txt
nmap.lst
rockyou.txt
rockyou.txt.gz
5 folders, 6 files: 184.4 MiB (193,321,740 bytes), Free space: 60.9 GiB
root@kali: /usr/share/wordlists

```
┌──(root㉿kali)-[/usr/share/wordlists]
└─# hashcat -m 0 -a 0 -o cracked.txt 1007399-hash15.txt rockyou.txt --show --
force

┌──(root㉿kali)-[/usr/share/wordlists]
└─# 
```



```
1 $1$529ca8050a00180790cf88b63468826a:tpac1
```

1 password cracked, not all cracked may because:

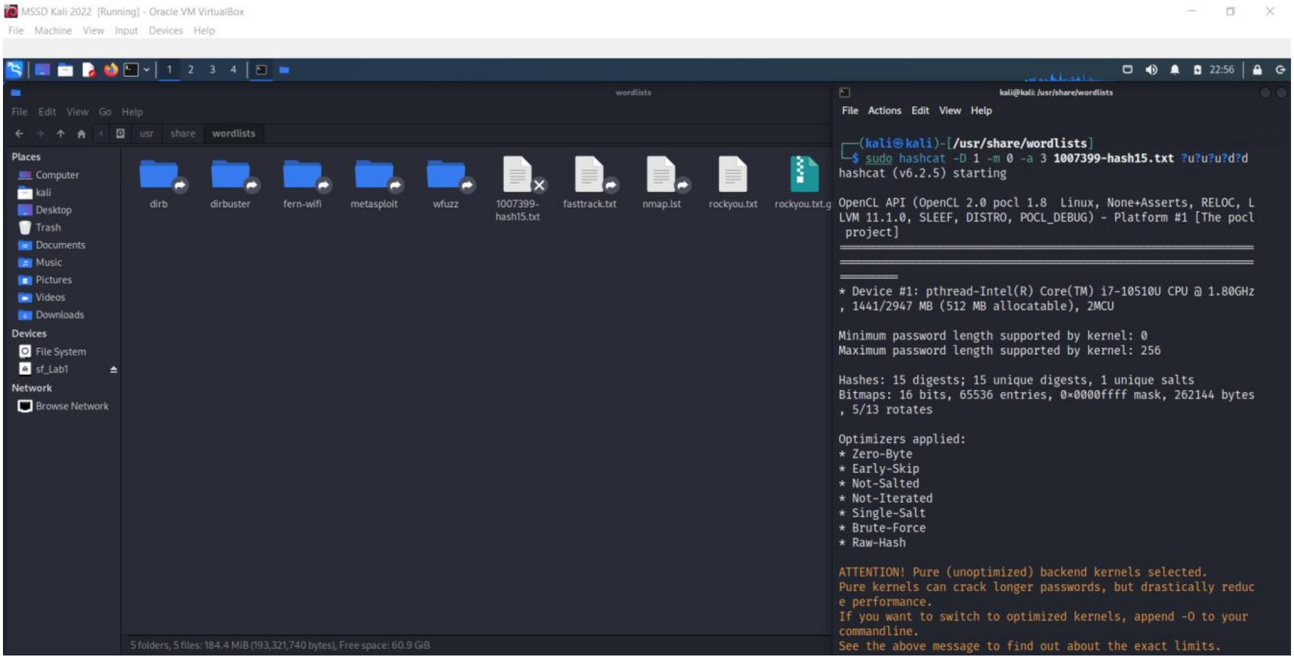
1. digits.rule matching pattern is not fully covered
2. rockyou.txt password source is not completely covered.

Second, try to execute **mask attack** that considers knowledge of character set in particular positions (including knowledge of how salting was performed) and also leverages parallelism of you CPU/GPU thanks to openCL library.
See URL https://hashcat.net/wiki/doku.php?id=mask_attack for quick introduction.
For example, to crack passwords having 3 upper case characters followed by 2 digits, hashcat can be run as:

- hashcat -D 1 -m 0 -a 3 some_hashes.txt ?u?u?u?d?d

Compare the timings with brute force that you implemented in your custom python script. If your computer has Intel graphic card, then you can also try parameter -D 2 and take note about the differences in timings.

Answer:



```
┌──(kali㉿kali)-[/usr/share/wordlists]
└─$ sudo hashcat -D 1 -m 0 -a 3 1007399-hash15.txt ?u?u?u?d?d
hashcat (v6.2.5) starting

OpenCL API (OpenCL 2.0 pocl 1.8  Linux, None+Asserts, RELOC, L
LVM 11.1.0, SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [The pocl
 project]
==============================================================
==============================================================
========
* Device #1: pthread-Intel(R) Core(TM) i7-10510U CPU @ 1.80GHz
, 1441/2947 MB (512 MB allocatable), 2MCU

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

Hashes: 15 digests; 15 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes
, 5/13 rotates

Optimizers applied:
* Zero-Byte
* Early-Skip
* Not-Salted
* Not-Iterated
* Single-Salt
* Brute-Force
* Raw-Hash

ATTENTION! Pure (unoptimized) backend kernels selected.
Pure kernels can crack longer passwords, but drastically reduc
e performance.
If you want to switch to optimized kernels, append -O to your
commandline.
See the above message to find out about the exact limits.
```

Compare timings with Rainbowcrack and include it into the previous writeup. Also mention rules and commands that you used for cracking the hashed and salted passwords.

Answer:

Cracking Speed:

RainbowCrack: RainbowCrack is known for its speed in cracking password hashes. It utilizes precomputed rainbow tables, which are generated in advance to accelerate the cracking process. These tables allow RainbowCrack to quickly match password hashes, resulting in faster cracking speeds for certain types of hashes.

Hashcat: Hashcat is a highly optimized password cracking tool that supports various cracking techniques, such as brute-force, dictionary attacks, and rule-based attacks. It utilizes the computational power of modern GPUs (Graphics Processing Units) to accelerate the cracking process. With the right hardware setup, Hashcat can achieve impressive speeds for cracking a wide range of password hashes.

Flexibility:

RainbowCrack: RainbowCrack primarily focuses on cracking LM and NTLM hashes, which are commonly used in older Windows operating systems. It excels in cracking these specific types of hashes due to its efficient use of precomputed tables. However, it may not be as versatile when it comes to cracking other types of hashes.

Hashcat: Hashcat is highly flexible and supports a wide range of hash types, including popular ones like MD5, SHA1, bcrypt, and more. It can handle various hashing algorithms and formats, making it suitable for cracking passwords from different sources and platforms. Additionally, Hashcat offers extensive customization options, allowing users to define specific attack modes, rules, and masks to adapt to different password cracking scenarios.

In summary, RainbowCrack is known for its speed when cracking LM and NTLM hashes using precomputed tables, while Hashcat is a versatile and powerful tool that can handle multiple hash types and provides customization options for different cracking techniques. The choice between the two tools depends on the specific hashes you want to crack and the level of flexibility and customization requirements.

8. Hash breaking competition

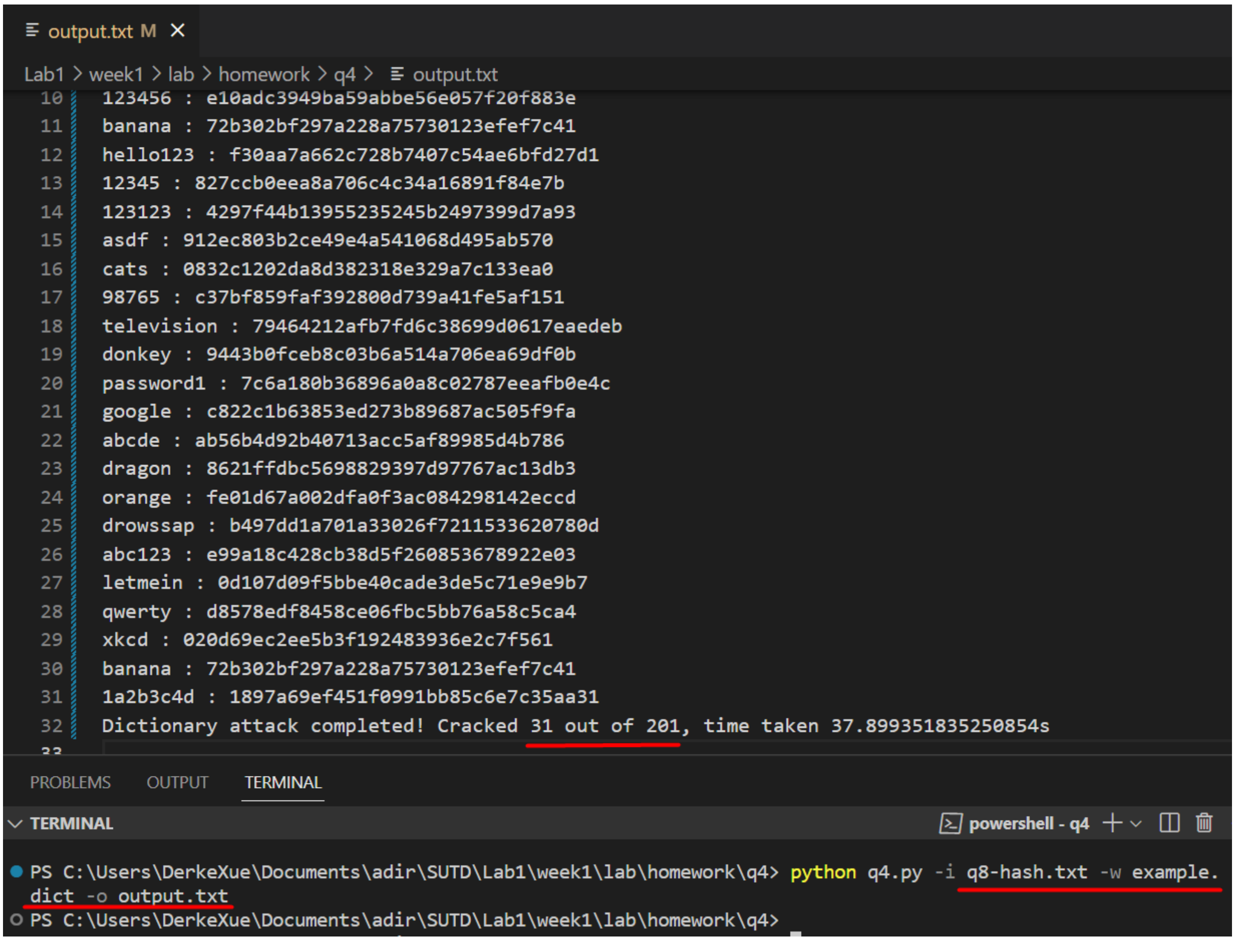
We provide a list of hashes in hashes.txt
They are of various difficulty – not all are equally hard. There are no easy rules about length or characters allowed anymore!

Implement an optimized script and try to reverse as many of those hashes as possible. You can also use other tools as you want (hashcat, rainbowcrack)

Write a short explanation on the approach you use to crack those passwords. Submit the answers as a CSV file called competition.csv containing two columns. The first column is the md5 hash of the password you break, and the second column is the plain text password.

Answer:

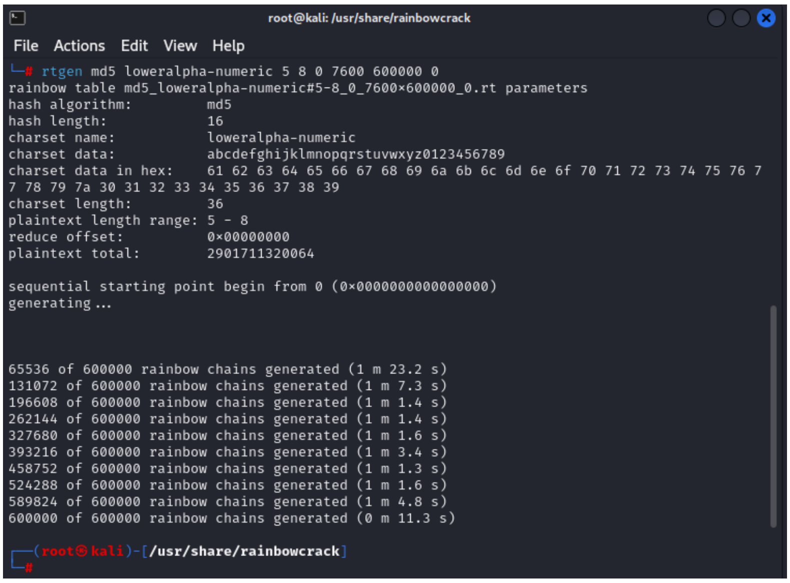
1. dictionary attack (reuse q4.py in question 4, together with example.dict):



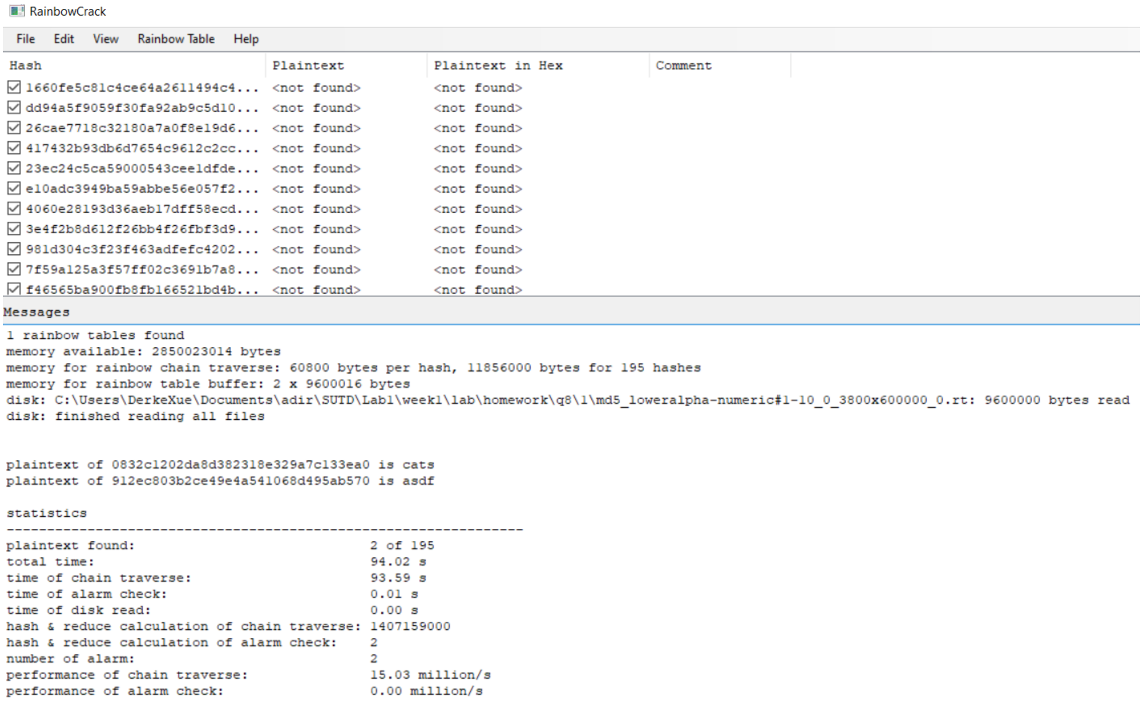
```
123456 : e10adc3949ba59abbe56e057f20f883e
banana : 72b302bf297a228a75730123efef7c41
hello123 : f30aa7a662c728b7407c54ae6bfd27d1
12345 : 827ccb0eea8a706c4c34a16891f84e7b
123123 : 4297f44b13955235245b2497399d7a93
asdf : 912ec803b2ce49e4a541068d495ab570
cats : 0832c1202da8d382318e329a7c133ea0
98765 : c37bf859faf392800d739a41fe5af151
television : 79464212afb7fd6c38699d0617eaedeb
donkey : 9443b0fceb8c03b6a514a706ea69df0b
password1 : 7c6a180b36896a0a8c02787eeafb0e4c
google : c822c1b63853ed273b89687ac505f9fa
abcde : ab56b4d92b40713acc5af89985d4b786
dragon : 8621ffdbc5698829397d97767ac13db3
orange : fe01d67a002dfa0f3ac084298142eccd
drowssap : b497dd1a701a33026f7211533620780d
abc123 : e99a18c428cb38d5f260853678922e03
letmein : 0d107d09f5bbe40cade3de5c71e9e9b7
qwerty : d8578edf8458ce06fbc5bb76a58c5ca4
xkcd : 020d69ec2ee5b3f192483936e2c7f561
banana : 72b302bf297a228a75730123efef7c41
1a2b3c4d : 1897a69ef451f0991bb85c6e7c35aa31
Dictionary attack completed! Cracked 31 out of 201, time taken 37.899351835250854s
```

```
PS C:\Users\DerkeXue\Documents\adir\SUTD\Lab1\week1\lab\homework\q4> python q4.py -i q8-hash.txt -w example.
dict -o output.txt
PS C:\Users\DerkeXue\Documents\adir\SUTD\Lab1\week1\lab\homework\q4>
```

2. rainbow crack:



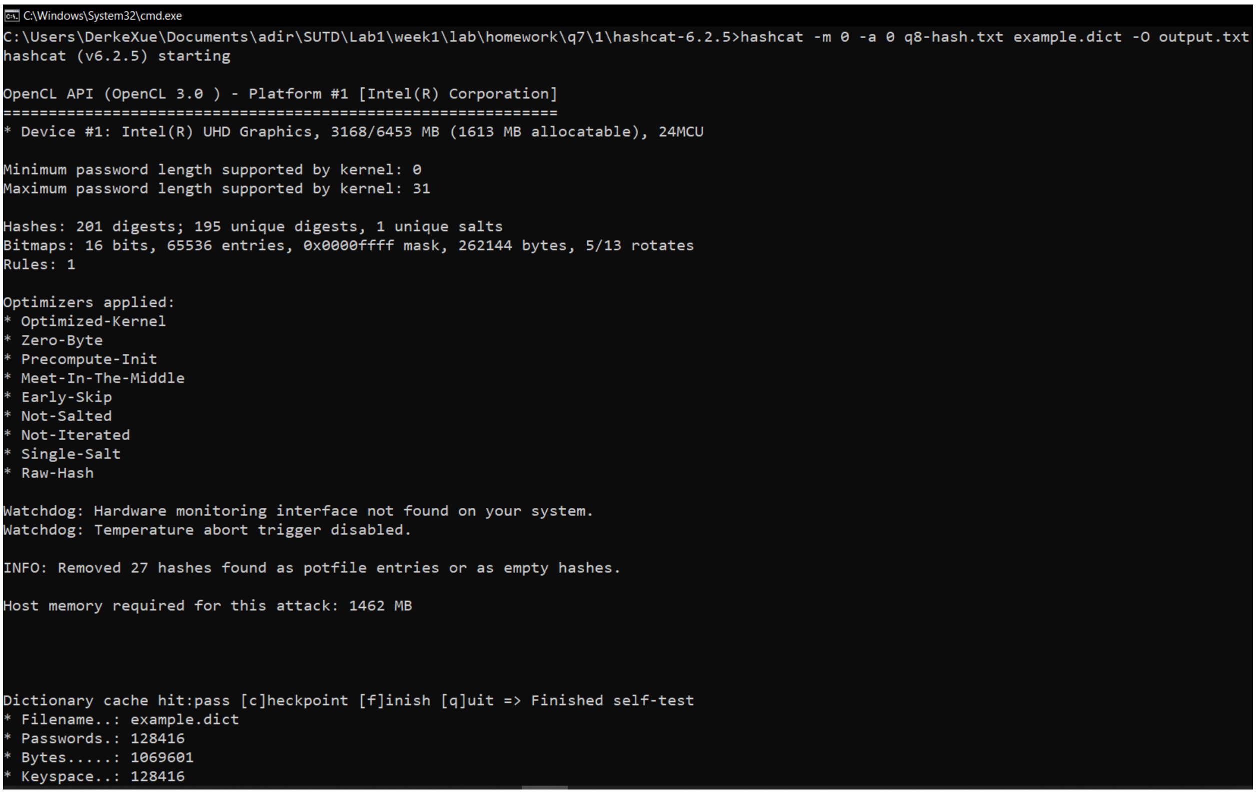
```
└─# rtgen md5 loweralpha-numeric 5 8 0 7600 600000 0
rainbow table md5_loweralpha-numeric#5-8_0_7600×600000_0.rt parameters
hash algorithm:         md5
hash length:            16
charset name:           loweralpha-numeric
charset data:           abcdefghijklmnopqrstuvwxyz0123456789
charset data in hex:    61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76 7
7 78 79 7a 30 31 32 33 34 35 36 37 38 39
charset length:         36
plaintext length range: 5 - 8
reduce offset:          0×00000000
plaintext total:        2901711320064

sequential starting point begin from 0 (0×0000000000000000)
generating ...



65536 of 600000 rainbow chains generated (1 m 23.2 s)
131072 of 600000 rainbow chains generated (1 m 7.3 s)
196608 of 600000 rainbow chains generated (1 m 1.4 s)
262144 of 600000 rainbow chains generated (1 m 1.4 s)
327680 of 600000 rainbow chains generated (1 m 1.6 s)
393216 of 600000 rainbow chains generated (1 m 3.4 s)
458752 of 600000 rainbow chains generated (1 m 1.3 s)
524288 of 600000 rainbow chains generated (1 m 1.6 s)
589824 of 600000 rainbow chains generated (1 m 4.8 s)
600000 of 600000 rainbow chains generated (0 m 11.3 s)

┌──(root㉿kali)-[/usr/share/rainbowcrack]
└─#
```



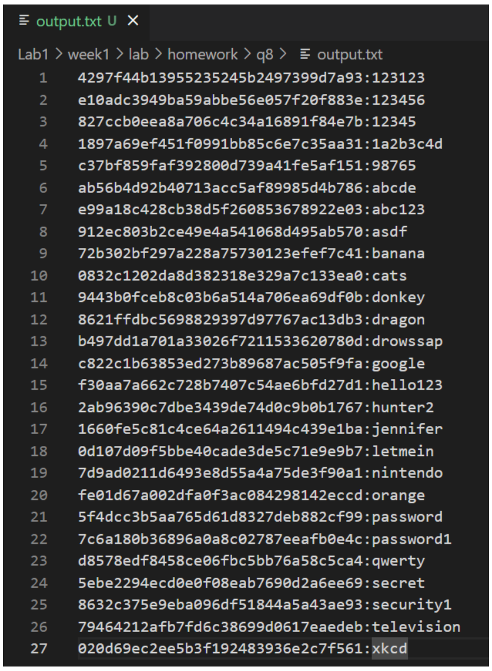
Hash	Plaintext	Plaintext in Hex	Comment
☑ 1660fe5c81c4ce64a2611494c4...	<not found>	<not found>	
☑ dd94a5f9059f30fa92ab9c5d10...	<not found>	<not found>	
☑ 26cae7718c32180a7a0f8e19d6...	<not found>	<not found>	
☑ 417432b93db6d7654c9612c2cc...	<not found>	<not found>	
☑ 23ec24c5ca59000543cee1dfde...	<not found>	<not found>	
☑ e10adc3949ba59abbe56e057f2...	<not found>	<not found>	
☑ 4060e28193d36aeb17dff58ecd...	<not found>	<not found>	
☑ 3e4f2b8d612f26bb4f26fbf3d9...	<not found>	<not found>	
☑ 981d304c3f23f463adfefc4202...	<not found>	<not found>	
☑ 7f59a125a3f57ff02c3691b7a8...	<not found>	<not found>	
☑ f46565ba900fb8fb166521bd4b...	<not found>	<not found>	

```
1 rainbow tables found
memory available: 2850023014 bytes
memory for rainbow chain traverse: 60800 bytes per hash, 11856000 bytes for 195 hashes
memory for rainbow table buffer: 2 x 9600016 bytes
disk: C:\Users\DerkeXue\Documents\adir\SUTD\Lab1\week1\lab\homework\q8\1\md5_loweralpha-numeric#1-10_0_3800x600000_0.rt: 9600000 bytes read
disk: finished reading all files


plaintext of 0832c1202da8d382318e329a7c133ea0 is cats
plaintext of 912ec803b2ce49e4a541068d495ab570 is asdf

statistics
----------------------------------------------------------------
plaintext found:                             2 of 195
total time:                                  94.02 s
time of chain traverse:                      93.59 s
time of alarm check:                         0.01 s
time of disk read:                           0.00 s
hash & reduce calculation of chain traverse: 1407159000
hash & reduce calculation of alarm check:    2
number of alarm:                             2
performance of chain traverse:               15.03 million/s
performance of alarm check:                  0.00 million/s
```

3. hashcat:



```
C:\Users\DerkeXue\Documents\adir\SUTD\Lab1\week1\lab\homework\q7\1\hashcat-6.2.5>hashcat -m 0 -a 0 q8-hash.txt example.dict -O output.txt
hashcat (v6.2.5) starting

OpenCL API (OpenCL 3.0 ) - Platform #1 [Intel(R) Corporation]
=============================================================
* Device #1: Intel(R) UHD Graphics, 3168/6453 MB (1613 MB allocatable), 24MCU

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 31

Hashes: 201 digests; 195 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Optimizers applied:
* Optimized-Kernel
* Zero-Byte
* Precompute-Init
* Meet-In-The-Middle
* Early-Skip
* Not-Salted
* Not-Iterated
* Single-Salt
* Raw-Hash

Watchdog: Hardware monitoring interface not found on your system.
Watchdog: Temperature abort trigger disabled.

INFO: Removed 27 hashes found as potfile entries or as empty hashes.

Host memory required for this attack: 1462 MB




Dictionary cache hit:pass [c]heckpoint [f]inish [q]uit => Finished self-test
* Filename..: example.dict
* Passwords.: 128416
* Bytes.....: 1069601
* Keyspace..: 128416
```



```
4297f44b13955235245b2497399d7a93:123123
e10adc3949ba59abbe56e057f20f883e:123456
827ccb0eea8a706c4c34a16891f84e7b:12345
1897a69ef451f0991bb85c6e7c35aa31:1a2b3c4d
c37bf859faf392800d739a41fe5af151:98765
ab56b4d92b40713acc5af89985d4b786:abcde
e99a18c428cb38d5f260853678922e03:abc123
912ec803b2ce49e4a541068d495ab570:asdf
72b302bf297a228a75730123efef7c41:banana
0832c1202da8d382318e329a7c133ea0:cats
9443b0fceb8c03b6a514a706ea69df0b:donkey
8621ffdbc5698829397d97767ac13db3:dragon
b497dd1a701a33026f7211533620780d:drowssap
c822c1b63853ed273b89687ac505f9fa:google
f30aa7a662c728b7407c54ae6bfd27d1:hello123
2ab96390c7dbe3439de74d0c9b0b1767:hunter2
1660fe5c81c4ce64a2611494c439e1ba:jennifer
0d107d09f5bbe40cade3de5c71e9e9b7:letmein
7d9ad0211d6493e8d55a4a75de3f90a1:nintendo
fe01d67a002dfa0f3ac084298142eccd:orange
5f4dcc3b5aa765d61d8327deb882cf99:password
7c6a180b36896a0a8c02787eeafb0e4c:password1
d8578edf8458ce06fbc5bb76a58c5ca4:qwerty
5ebe2294ecd0e0f08eab7690d2a6ee69:secret
8632c375e9eba096df51844a5a43ae93:security1
79464212afb7fd6c38699d0617eaedeb:television
020d69ec2ee5b3f192483936e2c7f561:xkcd
```

(Cracked 27 out of 201)

competition.csv (attached in zip folder):

dictionary attack:

1660fe5c81c4ce64a2611494c439e1ba	jennifer
e10adc3949ba59abbe56e057f20f883e	123456
5ebe2294ecd0e0f08eab7690d2a6ee69	secret
5f4dcc3b5aa765d61d8327deb882cf99	password
2ab96390c7dbe3439de74d0c9b0b1767	hunter2
72b302bf297a228a75730123efef7c41	banana
8632c375e9eba096df51844a5a43ae93	security1
7d9ad0211d6493e8d55a4a75de3f90a1	nintendo
d8578edf8458ce06fbc5bb76a58c5ca4	qwerty
e10adc3949ba59abbe56e057f20f883e	123456
72b302bf297a228a75730123efef7c41	banana
f30aa7a662c728b7407c54ae6bfd27d1	hello123
827ccb0eea8a706c4c34a16891f84e7b	12345
4297f44b13955235245b2497399d7a93	123123
912ec803b2ce49e4a541068d495ab570	asdf
0832c1202da8d382318e329a7c133ea0	cats
c37bf859faf392800d739a41fe5af151	98765
79464212afb7fd6c38699d0617eaedeb	television
9443b0fceb8c03b6a514a706ea69df0b	donkey
7c6a180b36896a0a8c02787eeafb0e4c	password1
c822c1b63853ed273b89687ac505f9fa	google
ab56b4d92b40713acc5af89985d4b786	abcde
8621ffdbc5698829397d97767ac13db3	dragon
fe01d67a002dfa0f3ac084298142eccd	orange
b497dd1a701a33026f7211533620780d	drowssap
e99a18c428cb38d5f260853678922e03	abc123

0d107d09f5bbe40cade3de5c71e9e9b7	letmein
d8578edf8458ce06fbc5bb76a58c5ca4	qwerty
020d69ec2ee5b3f192483936e2c7f561	xkcd
72b302bf297a228a75730123efef7c41	banana
1897a69ef451f0991bb85c6e7c35aa31	1a2b3c4d
rainbow crack:	
0832c1202da8d382318e329a7c133ea0	cats
912ec803b2ce49e4a541068d495ab570	asdf
hashcat:	
4297f44b13955235245b2497399d7a93	123123
e10adc3949ba59abbe56e057f20f883e	123456
827ccb0eea8a706c4c34a16891f84e7b	12345
1897a69ef451f0991bb85c6e7c35aa31	1a2b3c4d
c37bf859faf392800d739a41fe5af151	98765
ab56b4d92b40713acc5af89985d4b786	abcde
e99a18c428cb38d5f260853678922e03	abc123
912ec803b2ce49e4a541068d495ab570	asdf
72b302bf297a228a75730123efef7c41	banana
0832c1202da8d382318e329a7c133ea0	cats
9443b0fceb8c03b6a514a706ea69df0b	donkey
8621ffdbc5698829397d97767ac13db3	dragon
b497dd1a701a33026f7211533620780d	drowssap
c822c1b63853ed273b89687ac505f9fa	google
f30aa7a662c728b7407c54ae6bfd27d1	hello123
2ab96390c7dbe3439de74d0c9b0b1767	hunter2
1660fe5c81c4ce64a2611494c439e1ba	jennifer
0d107d09f5bbe40cade3de5c71e9e9b7	letmein
7d9ad0211d6493e8d55a4a75de3f90a1	nintendo
fe01d67a002dfa0f3ac084298142eccd	orange
5f4dcc3b5aa765d61d8327deb882cf99	password
7c6a180b36896a0a8c02787eeafb0e4c	password1
d8578edf8458ce06fbc5bb76a58c5ca4	qwerty
5ebe2294ecd0e0f08eab7690d2a6ee69	secret
8632c375e9eba096df51844a5a43ae93	security1
79464212afb7fd6c38699d0617eaedeb	television
020d69ec2ee5b3f192483936e2c7f561	xkcd

9. Hand-in

Submit your md5_lab1.py script that breaks the supplied hash values in hash5.txt, generate the salted hashes and the relevant files. Put your username and mention the timings in your header.

Prepare the writeups for all sections where explanations are requested. Include your conclusions and learning points.

Include the found plaintexts/hashes (competition.csv) in your writeup.