

**Jaleel Rogers**

**Professor Mukhopadhyay**

**CIS 4204.01**

**02 February 2024**

```
ATTENTION! Pure (unoptimized) backend kernels selected.
Pure kernels can crack longer passwords, but drastically reduce 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.

Watchdog: Temperature abort trigger set to 90C

Host memory required for this attack: 1 MB

Dictionary cache hit:
* Filename..: rockyou.txt
* Passwords.: 14344384
* Bytes.....: 139921497
* Keyspace..: 14344384

5f4dcc3b5aa765d61d8327deb882cf99:password
482c811da5d5b4bc6d497ffa98491e38:password123
7ecc19e1a0be36ba2c6f05d06b5d3058:weak
```

***Figure 1 - Executing Hashcat***

For the entirety of the assignment, I used the Kali OS. The objective of the first part of the assignment is to create hash collisions by using a dictionary executed on hashcat. For the dictionary of my choice I used a total of three; rockyou, crackstation, and md5encrypt. For figure 1, I inserted the text file hash.txt containing the hashes 7ecc19e1a0be36ba2c6f05d06b5d3058, 5f4dcc3b5aa765d61d8327deb882cf99, 482c811da5d5b4bc6d497ffa98491e38, and f7bbdf9e9e4d3112c852f142cd6ddc7a using the rockyou dictionary. The command used was hashcat -a0 -m0 hashes.txt rockyou.txt. The reason I used three dictionaries is that not all hashes had a corresponding plaintext value in rockyou dictionary. Specifically, the hash f7bbdf9e9e4d3112c852f142cd6ddc7a in both rockyou and crackstation dictionaries.

Jaleel Rogers

Professor Mukhopadhyay

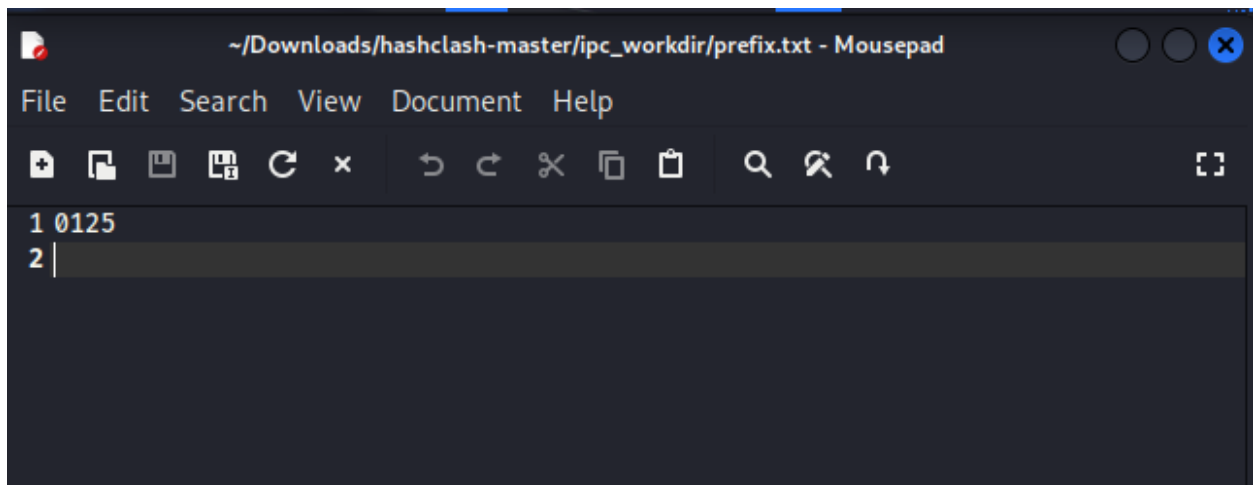
CIS 4204.01

02 February 2024



*Figure 2 - Executing md5encrypt*

In Figure 2, I used the third dictionary, md5 encrypt, which is a browser-based dictionary in order to get the plaintext. The plaintext for 7ecc19e1a0be36ba2c6f05d06b5d3058 is weak, 5f4dcc3b5aa765d61d8327deb882cf99 is password, 482c811da5d5b4bc6d497ffa98491e38 is password123, and f7bbdf9e9e4d3112c852f142cd6ddc7a is Smith@1998.



*Figure 3 - prefix.txt file*

Jaleel Rogers

Professor Mukhopadhyay

CIS 4204.01

02 February 2024

In the second part of the assignment the objective is to get two outputs in the form of binary files from a string input then hash those binary values to see if they result in the same hash. Figure 3, is the prefix.txt containing the string 0125.

```
Q-3: |01100111 01000101 00100011 00000001|
Q-2: |00010000 00110010 01010100 01110110|
Q-1: |10011000 10111010 11011100 11111110|
Q0:  |11101111 11001101 10101011 10001001| ok p=1
Q1:  |00111110 00111000 01111111 00001111| ok p=1
Q2:  |10010000 10111000 11100000 01001110| ok p=1
Q3:  |10101+-0 01010100 01000011 11000010| ok p=0.994141
Q4:  |.....0+. .1..... -. ....| ok p=1
Q5:  |.....+- .+..... -. ....| ok p=0.967773
Q6:  |.....-. .+..... -. ....| ok p=0.988281
Q7:  |.....+. .+..... -. ....|
Saving 1280000 paths ... done.
Autobalance parameters: maxcond=254
Verified: 0 bad out of 2141859
Estimating maxcond for upper bound 5120000 (=1280000 * 4) ...
t=7: 0% 10 20 30 40 50 60 70 80 90 100%
|-----|
e *****
Found maxcond = 258
t=7: 0% 10 20 30 40 50 60 70 80 90 100%
|-----|
*****
Q-3: |01100111 01000101 00100011 00000001|
Q-2: |00010000 00110010 01010100 01110110|
Q-1: |10011000 10111010 11011100 11111110|
Q0:  |11101111 11001101 10101011 10001001| ok p=1
Q1:  |00111110 00111000 01111111 00001111| ok p=1
Q2:  |10010000 10111000 11100000 01001110| ok p=1
Q3:  |10101+-0 01010100 01000011 11000010| ok p=0.988281
Q4:  |.....0+. .1..... -. ....| ok p=1
Q5:  |.....1+. .+..... -. ....V| ok p=0.976562
Q6:  |.....-. .+..... -. ....| ok p=1
Q7:  |.....-. .+..... -. ....| ok p=0.986328
Q8:  |.....+. .+..... ...+.... -. ....|
Saving 1280000 paths ... done.
Autobalance parameters: maxcond=256
Verified: 16 bad out of 1982357
Estimating maxcond for upper bound 5120000 (=1280000 * 4) ...
t=8: 0% 10 20 30 40 50 60 70 80 90 100%
|-----|
e *****
Found maxcond = 263
t=8: 0% 10 20 30 40 50 60 70 80 90 100%
|-----|
***
```

*Figure 4 - Executing Hashclash*

I used hashclash, a hash collision software to create the two binary files. However, in Figure 4 it kept looping for hours and would not stop, creating no output in the process.

Jaleel Rogers

Professor Mukhopadhyay

CIS 4204.01

02 February 2024

```
(kali㉿kali)-[~/md5collgen]
$ ./md5collgen -p ~/Downloads/hashclash-master/ipc_workdir/prefix.txt -o out1_class.bin out2_class.bin
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: '/home/kali/Downloads/hashclash-master/ipc_workdir/prefix_msg1.txt' and '/home/kali/Downloads/h
ashclash-master/ipc_workdir/prefix_msg2.txt'
Using prefixfile: '/home/kali/Downloads/hashclash-master/ipc_workdir/prefix.txt'
Using initial value: 55ac2514e4839e66c9f7d0ae26026e6f

Generating first block: .
Generating second block: S00.
Running time: 0.141878 s

(kali㉿kali)-[~/md5collgen]
$ ./md5collgen -p ~/Downloads/hashclash-master/ipc_workdir/prefix.txt -o out1_class.bin out2_class.bin
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)
```

*Figure 5 - Executing md5collgen*

```
~/Downloads/hashclash-master/ipc_workdir/prefix_msg1.txt - Mousepad
File Edit Search View Document Help
[Icons]
1 0125
2 .....:g+zX0)BZ+s%.
3 ..-
4 '0+W::4l{+6[V:U T++NLTL-/xb*W+++hY+5i++ +e_mJ*VB+}++++!?!?++++
```

*Figure 6 - msg1.txt file*

```
~/Downloads/hashclash-master/ipc_workdir/prefix_msg2.txt - Mousepad
File Edit Search View Document Help
[Icons]
1 0125
2 .....:g+zX0)BZ+s%.
3 ..-
4 '0+W::4l{+6[+:U T++NLTL-/xb*W+++hY+5i++ +e_mJ*UB+}++++!?!?++++
```

*Figure 7 - msg2.txt file*

Jaleel Rogers

Professor Mukhopadhyay

CIS 4204.01

02 February 2024

```
(kali@kali)-[~/Downloads/hashclash-master/ipc_workdir]
$ xxd prefix_msg1.txt
00000000: 3031 3235 0a00 0000 0000 0000 0000 0000 0125.....
00000010: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000020: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000030: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000040: bb8a 3a01 6791 7afb e817 9e58 8630 2942 ..:g.z...X.0)B
00000050: 5ad0 fe9f 8712 d473 2506 84a6 082e 0dc8 Z.....s%.....
00000060: 1505 ace7 de2d 0a27 0730 d7d4 9c3d c5fe .....-.'.0...=..
00000070: a8de dbfc 3a34 6c7b e836 5b56 3a55 e0b2 ....:4l{.6[V:U..
00000080: bb20 c954 04eb 8307 d117 c5c3 cdb8 a64e ..T.....N
00000090: 4c54 e12d 2f78 0162 cd57 f6cc d968 59a2 LT../x.b.W...hY.
000000a0: 3569 9fbe 202b a265 5f6d d40f 4a2a 5642 5i..+.e_m..J*VB
000000b0: f21a 9b7d 82fc 86e2 e221 3f9c a494 b503 ...}.....!?.....

(kali@kali)-[~/Downloads/hashclash-master/ipc_workdir]
$ xxd prefix_msg2.txt
00000000: 3031 3235 0a00 0000 0000 0000 0000 0000 0125.....
00000010: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000020: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000030: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000040: bb8a 3a01 6791 7afb e817 9e58 8630 2942 ..:g.z...X.0)B
00000050: 5ad0 fe1f 8712 d473 2506 84a6 082e 0dc8 Z.....s%.....
00000060: 1505 ace7 de2d 0a27 0730 d7d4 9cbd c5fe .....-.'.0.....
00000070: a8de dbfc 3a34 6c7b e836 5bd6 3a55 e0b2 ....:4l{.6[:U..
00000080: bb20 c954 04eb 8307 d117 c5c3 cdb8 a64e ..T.....N
00000090: 4c54 e1ad 2f78 0162 cd57 f6cc d968 59a2 LT../x.b.W...hY.
000000a0: 3569 9fbe 202b a265 5f6d d40f 4aaa 5542 5i..+.e_m..J.UB
000000b0: f21a 9b7d 82fc 86e2 e221 3f1c a494 b503 ...}.....!?.....
```

Figure 8 - msg1 and msg2 as binary files

I then used the md5collgen script to run the prefix.txt as seen in Figure 5. Figure 6 and Figure 7 are the output of prefix.txt. In Figure 8, I used the xxd command to view the text files as binary files.

```
(kali@kali)-[~/Downloads/hashclash-master/ipc_workdir]
$ diff prefix_msg1.txt prefix_msg2.txt
Binary files prefix_msg1.txt and prefix_msg2.txt differ
```

Figure 9 - Binary file comparison

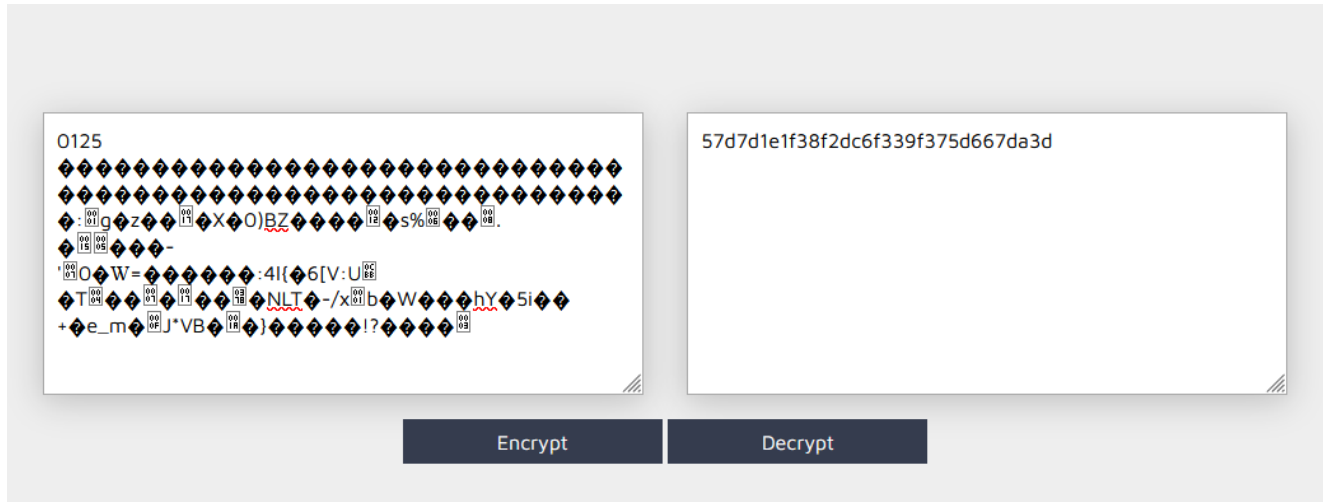
In Figure 9, I then used the diff command to compare the two binary files as txt files. Both files have different values.

Jaleel Rogers

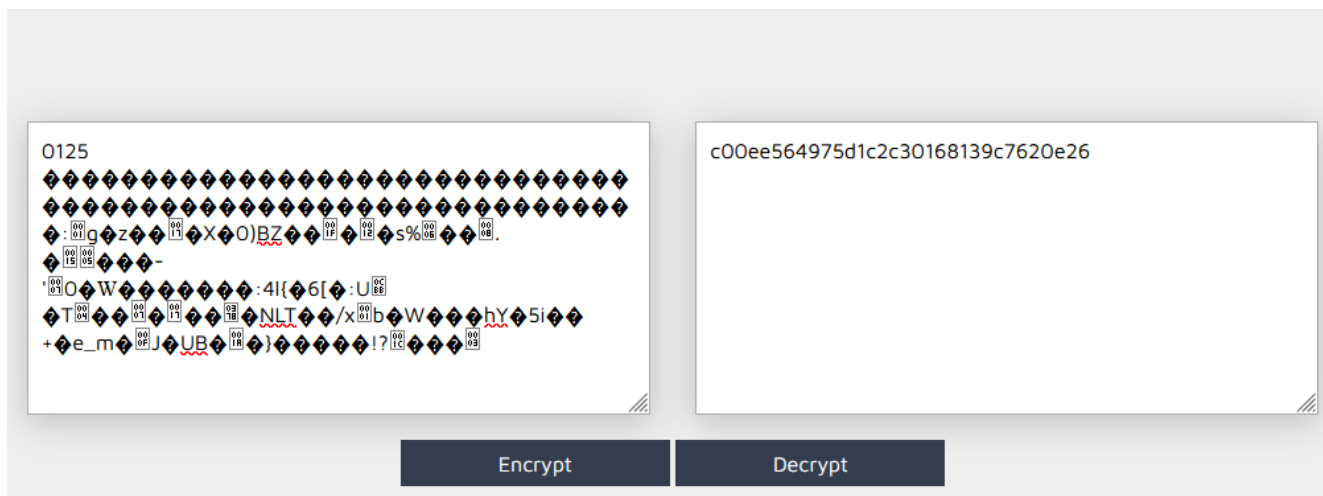
Professor Mukhopadhyay

CIS 4204.01

02 February 2024



*Figure 10 - msg1.txt hash*



*Figure 11 - msg2.txt hash*

For both files they have different hash values as seen in Figures 10 and 11. I wasn't successfully able to hash the files to have the same output.