

Introduction to Information Security - CS 458

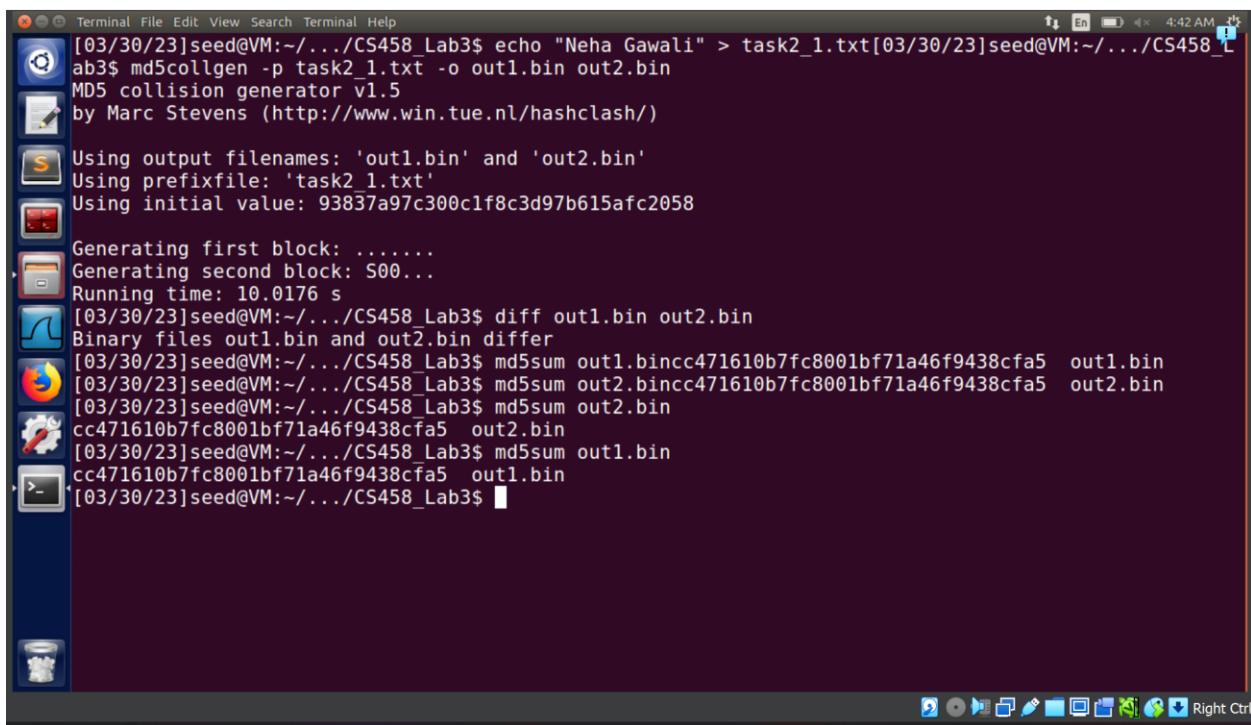
Neha Ramesh Gawali A20523722

Lab 3 - MD5 Collision Attack Lab

2.1 Task 1: Generating Two Different Files with the Same MD5 Hash

When I opened the two output files, out1.bin and out2.bin, in the hex editor, I could see that the first part (i.e., prefix) of both files is identical. However, the remaining parts of the two files are different. Even though the files are different, they share the same MD5 hash value.

Question 1. If the length of your prefix file is not multiple of 64, what is going to happen?

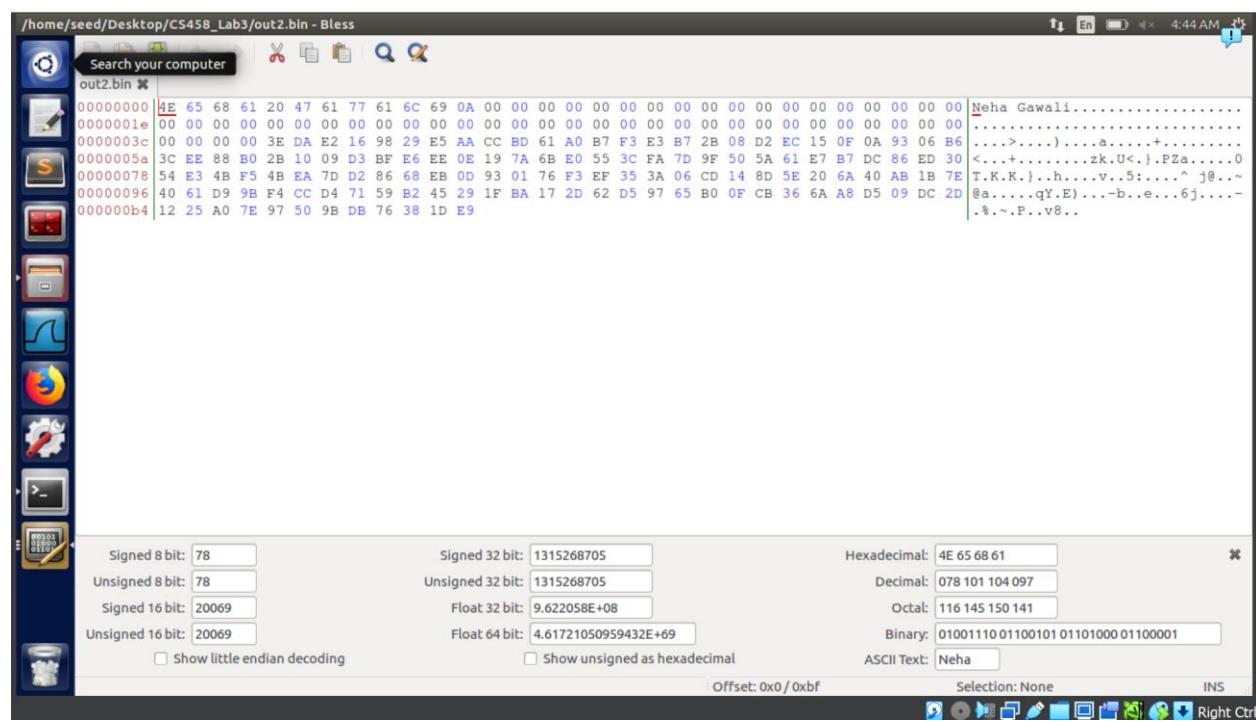
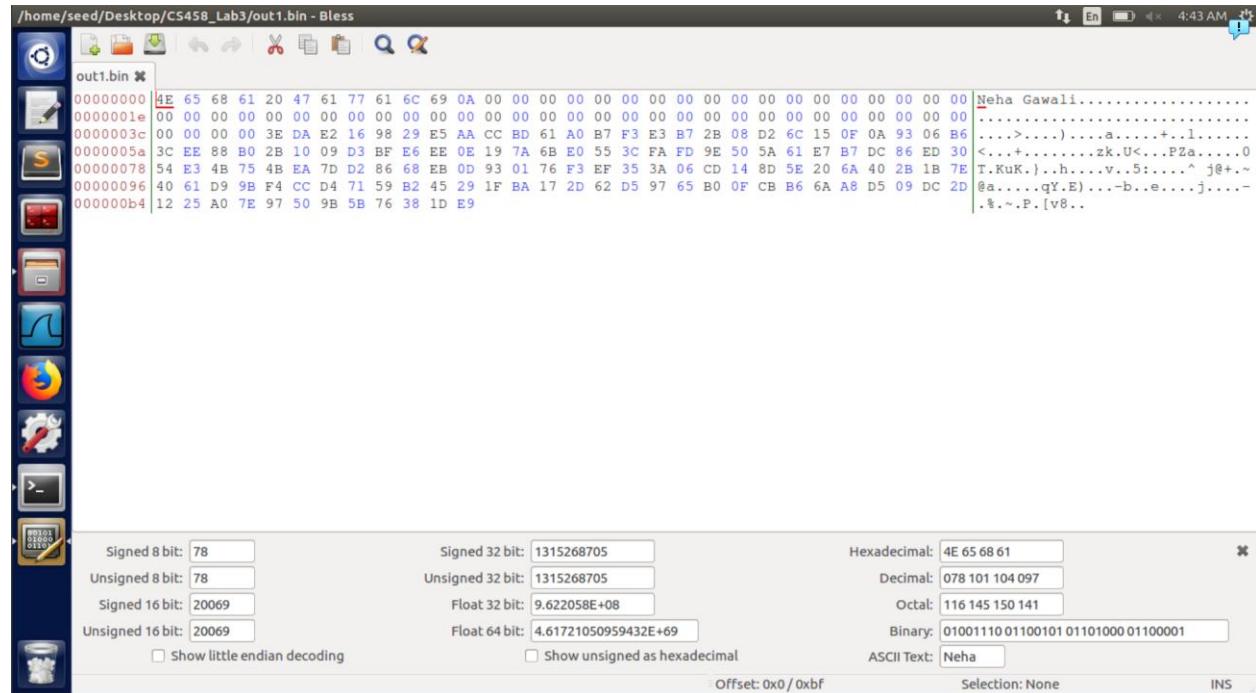


The screenshot shows a terminal window on a Linux desktop environment. The terminal window has a dark background and contains the following text:

```
[03/30/23]seed@VM:~/.../CS458_Lab3$ echo "Neha Gawali" > task2_1.txt
[03/30/23]seed@VM:~/.../CS458_Lab3$ ab3$ md5collgen -p task2_1.txt -o out1.bin out2.bin
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)
Using output filenames: 'out1.bin' and 'out2.bin'
Using prefixfile: 'task2_1.txt'
Using initial value: 93837a97c300c1f8c3d97b615afc2058
Generating first block: .....
Generating second block: S00...
Running time: 10.0176 s
[03/30/23]seed@VM:~/.../CS458_Lab3$ diff out1.bin out2.bin
Binary files out1.bin and out2.bin differ
[03/30/23]seed@VM:~/.../CS458_Lab3$ md5sum out1.bincc471610b7fc8001bf71a46f9438cfa5  out1.bin
[03/30/23]seed@VM:~/.../CS458_Lab3$ md5sum out2.bincc471610b7fc8001bf71a46f9438cfa5  out2.bin
[03/30/23]seed@VM:~/.../CS458_Lab3$ md5sum out2.bin
cc471610b7fc8001bf71a46f9438cfa5  out2.bin
[03/30/23]seed@VM:~/.../CS458_Lab3$ md5sum out1.bin
cc471610b7fc8001bf71a46f9438cfa5  out1.bin
[03/30/23]seed@VM:~/.../CS458_Lab3$
```

Introduction to Information Security - CS 458

Neha Ramesh Gawali A20523722



Ans: Files are padded with zeros.

1. Here I created one 11-bit file with content “Neha Gawali” and named it as task2_1.txt
2. Using command: \$ md5collgen -p prefix.txt -o out1.bin out2.bin
3. Generates two output files, out1.bin and out2.bin, for a given a prefix file task2_1.txt

Introduction to Information Security - CS 458

Neha Ramesh Gawali A20523722

4. To check whether the output files are distinct or not Used the diff command:

md5sum command to check the MD5 hash of each output file. See the following commands.

```
$ diff out1.bin out2.bin
```

```
$ md5sum out1.bin
```

```
$ md5sum out2.bin
```

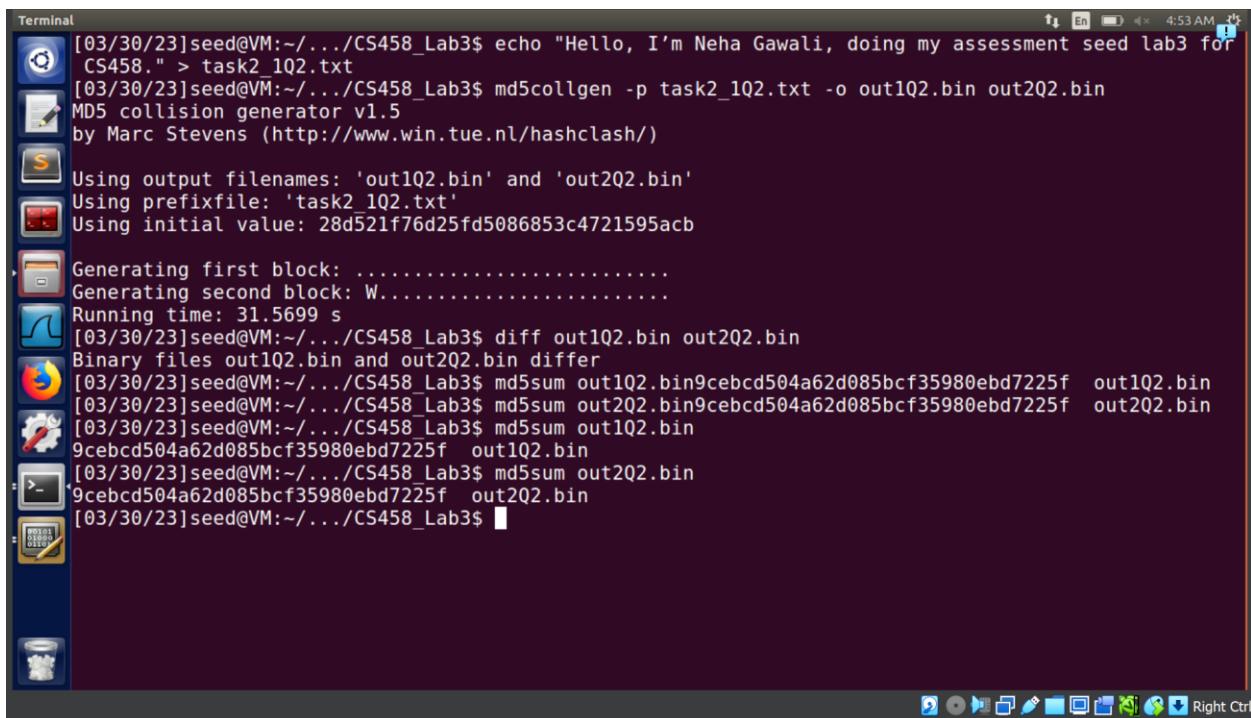
Diff command showed out1.bin and out2.bin are different.

5. Opened both out1.bin and out2.bin in hex bless, I observed both files contain same values.

Observation:

If the length of the prefix file is not a multiple of 64, then md5collgen will pad the prefix with zeros to make it a multiple of 64. This is because MD5 operates on blocks of 64 bytes. Therefore, if the length of the input is not a multiple of 64, the algorithm pads the input with zeros until its length is a multiple of 64. This padding may change the MD5 hash of the file.

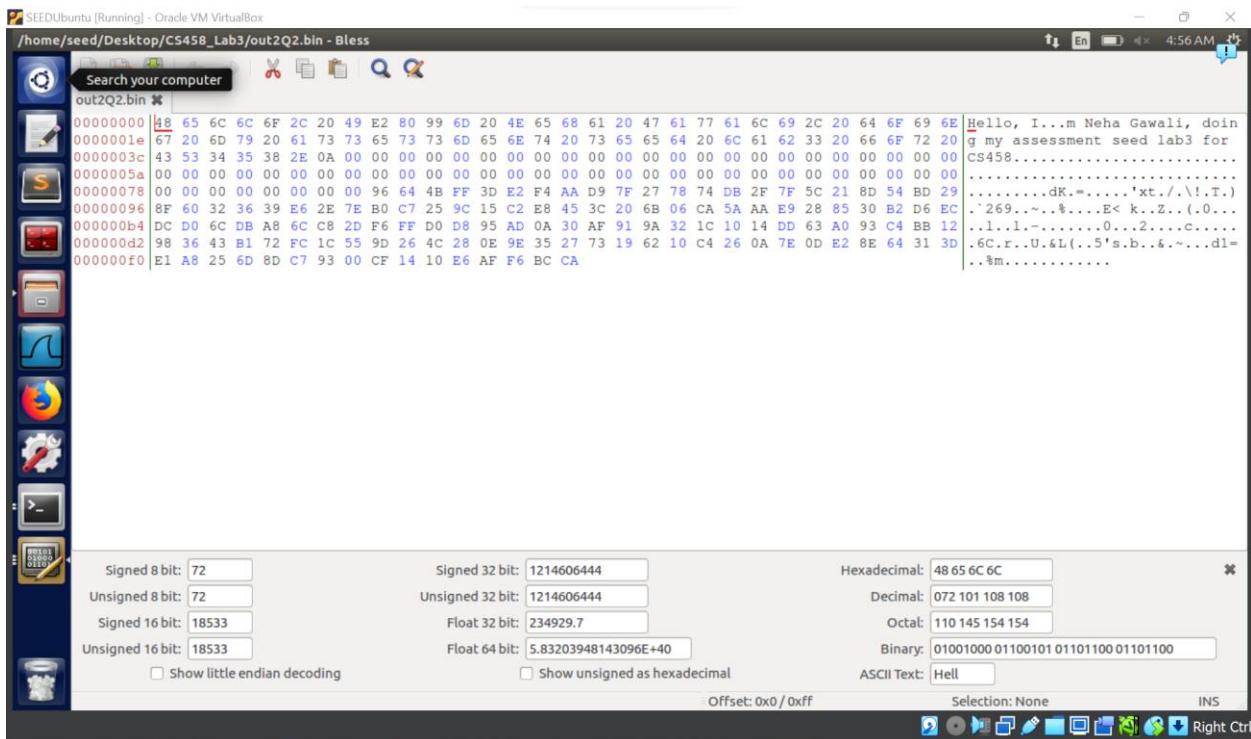
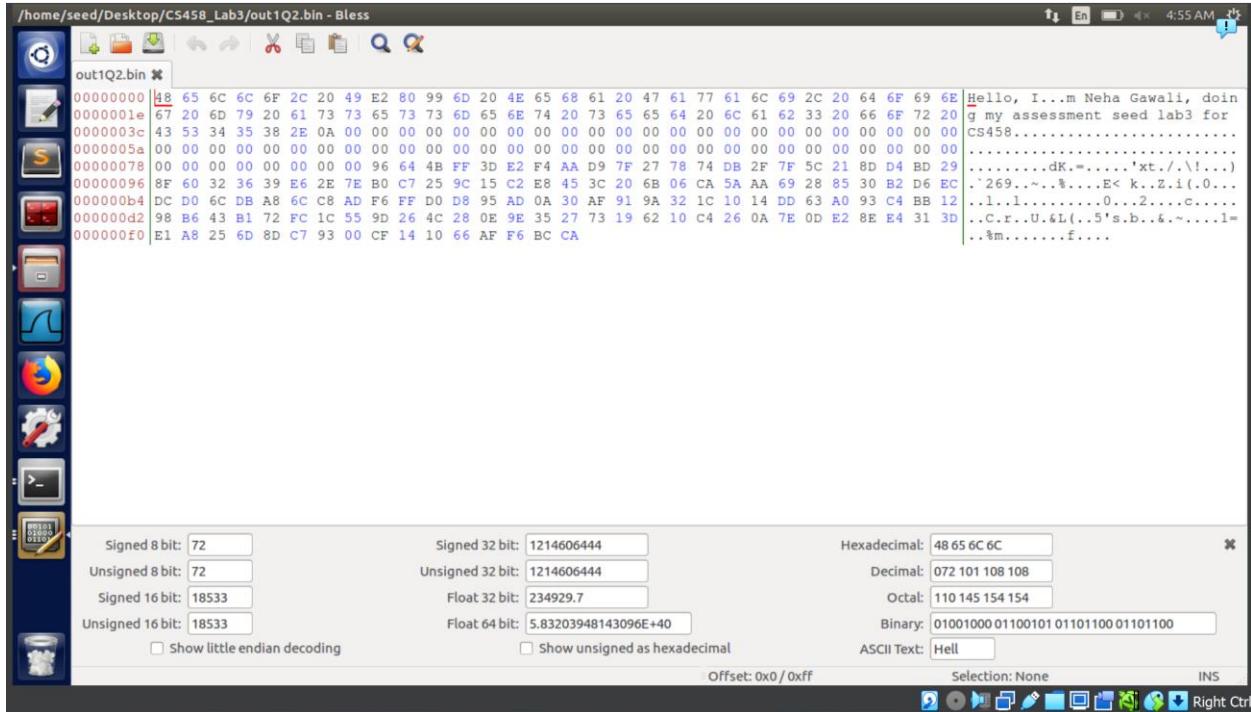
Question 2. Create a prefix file with exactly 64 bits file, and run the collision tool again, and see what happens :



```
[03/30/23]seed@VM:~/.../CS458_Lab3$ echo "Hello, I'm Neha Gawali, doing my assessment seed lab3 for CS458." > task2_1Q2.txt
[03/30/23]seed@VM:~/.../CS458_Lab3$ md5collgen -p task2_1Q2.txt -o out1Q2.bin out2Q2.bin
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)
Using output filenames: 'out1Q2.bin' and 'out2Q2.bin'
Using prefixfile: 'task2_1Q2.txt'
Using initial value: 28d521f76d25fd5086853c4721595acb
Generating first block: .....
Generating second block: W.....
Running time: 31.5699 s
[03/30/23]seed@VM:~/.../CS458_Lab3$ diff out1Q2.bin out2Q2.bin
Binary files out1Q2.bin and out2Q2.bin differ
[03/30/23]seed@VM:~/.../CS458_Lab3$ md5sum out1Q2.bin 9cebc504a62d085bcf35980ebd7225f out1Q2.bin
[03/30/23]seed@VM:~/.../CS458_Lab3$ md5sum out2Q2.bin 9cebc504a62d085bcf35980ebd7225f out2Q2.bin
9cebc504a62d085bcf35980ebd7225f out1Q2.bin
[03/30/23]seed@VM:~/.../CS458_Lab3$ md5sum out2Q2.bin
9cebc504a62d085bcf35980ebd7225f out2Q2.bin
[03/30/23]seed@VM:~/.../CS458_Lab3$
```

Introduction to Information Security - CS 458

Neha Ramesh Gawali A20523722



Ans:

Here,

Introduction to Information Security - CS 458

Neha Ramesh Gawali A20523722

1. Created a file with exact 64 bits. Content: Hello, I'm Neha Gawali, doing my assessment seed lab3 for CS458. Saved it as task2_1Q2.txt.
2. Using command: \$ md5collgen -p task2_1Q2.txt -o out1Q2.bin and out2Q2.bin
3. Generates two output files, out1Q1.bin and out2Q2.bin , for a given a prefix file task2_1Q2.txt
4. To check whether the output files are distinct or not Used the diff command:

md5sum command to check the MD5 hash of each output file. See the following commands.

```
$ diff out1Q2.bin out2Q2.bin  
$ md5sum out1Q2.bin  
$ md5sum out2Q2.bin
```

Diff command showed out1Q2.bin and out2Q2.bin are different.

5. Opened both out1Q2.bin and out2Q2.bin in hex bless, I observed both files contain same values.

After observing both the output files, shows that there were no zero-padding added.

Observation:

If the length of the prefix file is not a multiple of 64, then md5collgen will pad the prefix with zeros to make it a multiple of 64. This is because MD5 operates on blocks of 64 bytes. Therefore, if the length of the input is not a multiple of 64, the algorithm pads the input with zeros until its length is a multiple of 64. This padding may change the MD5 hash of the file.

Question 3: Question 3. Are the data (128 bytes) generated by md5collgen completely different for the two output files? Please identify all the bytes that are different.

1. Created a file with 128 bits. Content: "Hello there, this is Neha Ramesh Gawali reporting live from IIT. I'm planning to complete CS458 (infosec) this spring semester." Saved it as task2_1Q3.txt.
2. Using command: \$ md5collgen -p task2_1Q3.txt -o out1Q3.bin and out2Q3.bin
3. Generates two output files, out1Q3.bin and out2Q3.bin , for a given a prefix file task2_1Q3.txt
4. To check whether the output files are distinct or not Used the diff command:

md5sum command to check the MD5 hash of each output file. See the following commands.

```
$ diff out1Q3.bin out2Q3.bin
```

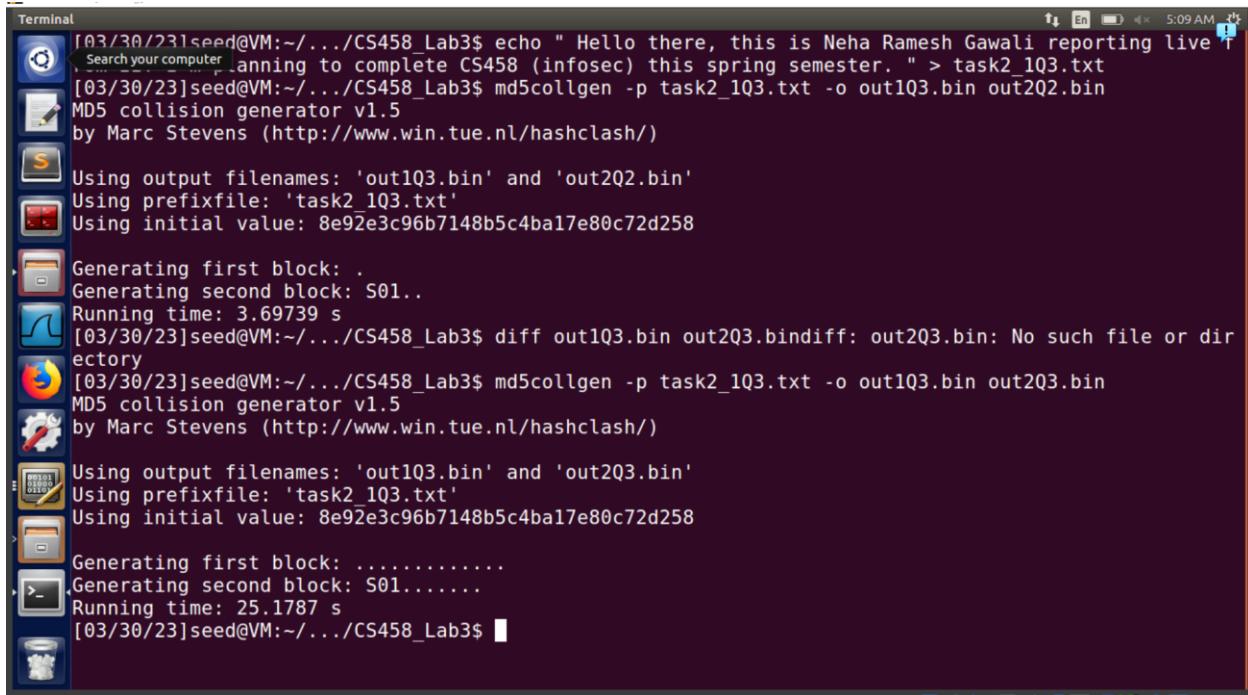
Introduction to Information Security - CS 458

Neha Ramesh Gawali A20523722

```
$ md5sum out1Q3.bin
```

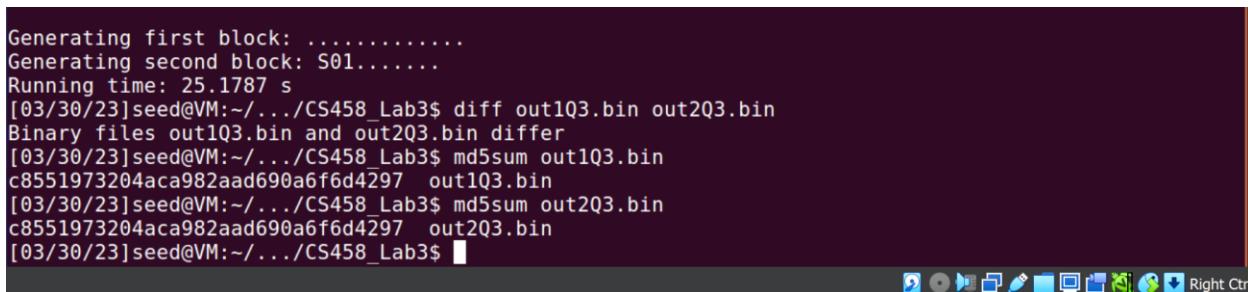
```
$ md5sum out2Q3.bin
```

Diff command showed out1Q3.bin and out2Q3.bin are different.



A screenshot of a Linux desktop environment showing a terminal window. The terminal window has a dark background and contains the following text:

```
[03/30/23]seed@VM:~/.../CS458_Lab3$ echo "Hello there, this is Neha Ramesh Gawali reporting live!" > task2_1Q3.txt
[03/30/23]seed@VM:~/.../CS458_Lab3$ md5collgen -p task2_1Q3.txt -o out1Q3.bin out2Q2.bin
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)
Using output filenames: 'out1Q3.bin' and 'out2Q2.bin'
Using prefixfile: 'task2_1Q3.txt'
Using initial value: 8e92e3c96b7148b5c4ba17e80c72d258
Generating first block: .
Generating second block: S01..
Running time: 3.69739 s
[03/30/23]seed@VM:~/.../CS458_Lab3$ diff out1Q3.bin out2Q3.bindiff: out2Q3.bin: No such file or directory
[03/30/23]seed@VM:~/.../CS458_Lab3$ md5collgen -p task2_1Q3.txt -o out1Q3.bin out2Q3.bin
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)
Using output filenames: 'out1Q3.bin' and 'out2Q3.bin'
Using prefixfile: 'task2_1Q3.txt'
Using initial value: 8e92e3c96b7148b5c4ba17e80c72d258
Generating first block: .....
Generating second block: S01.....
Running time: 25.1787 s
[03/30/23]seed@VM:~/.../CS458_Lab3$
```



A screenshot of a Linux desktop environment showing a terminal window. The terminal window has a dark background and contains the following text:

```
Generating first block: .....
Generating second block: S01.....
Running time: 25.1787 s
[03/30/23]seed@VM:~/.../CS458_Lab3$ diff out1Q3.bin out2Q3.bin
Binary files out1Q3.bin and out2Q3.bin differ
[03/30/23]seed@VM:~/.../CS458_Lab3$ md5sum out1Q3.bin
c8551973204aca982aad690a6f6d4297  out1Q3.bin
[03/30/23]seed@VM:~/.../CS458_Lab3$ md5sum out2Q3.bin
c8551973204aca982aad690a6f6d4297  out2Q3.bin
[03/30/23]seed@VM:~/.../CS458_Lab3$
```

Introduction to Information Security - CS 458

Neha Ramesh Gawali A20523722

/home/seed/Desktop/CS458_Lab3/out1Q3.bin - Bless

out1Q3.bin

00000000 20 48 65 6C 6C 6F 20 74 68 65 72 65 2C 20 74 68 69 73 20 4E 65 68 61 20 52 61 6D

0000001e 65 73 68 20 47 61 77 61 6C 69 20 72 65 70 6F 72 74 69 6E 67 20 6C 69 76 65 20 66 72 6F 6D

0000003c 20 49 49 54 20 49 E2 80 99 6D 20 70 6C 61 6E 6E 69 6E 67 20 74 6F 20 63 6F 6D 70 6C 65 74

0000005a 65 20 43 53 34 35 38 20 28 69 6E 66 6F 73 65 63 29 20 74 68 69 73 20 73 70 72 69 6E 67 20

00000078 73 65 6D 65 73 74 65 72 2E 20 0A 00

00000096 00

000000b4 00 00 00 00 00 00 00 00 00 00 00 00 00 00 BF EF 93 72 DB A1 FD EB DB 6F 77 27 EB 25 C6 C4 8C 31

000000d2 77 77 01 C0 80 84 06 4A 37 E2 95 8C 28 46 D2 D9 D3 0C EC 2D 09 BF BF 08 4A 02 B8 AF 5A 3D

000000f0 5C 01 C4 65 F7 AF C5 4A 50 CD C6 A5 44 D5 74 5E F2 55 09 8D 32 F1 79 09 4B 53 9D D8 D7 F9

0000010e AB 33 82 5F D9 18 F7 45 3E 25 CB 1A 86 CB EC C6 FD 6F B8 83 46 54 02 4C 08 BE 58 26 69 4F

0000012c 78 F8 E2 2E 18 39 49 46 C3 A9 5D 66 86 1A BD 2B 79 C1 12 C2

Hello there, this is Neha Ram
esh Gawali reporting live from
IIT I...m planning to complet
e CS458 (infosec) this spring
semester.
.....r....ow'.%..1
ww....J7...(.F....-.J..Z=...
\.e...JP..%D.t^U..2.y.KS...
.3...E>%.....o..FT.L..X&IO
xx...9IF..]f....y...

Signed 8 bit: 32 Unsigned 8 bit: 32 Signed 32 bit: 541615468 Unsigned 32 bit: 541615468 Hexadecimal: 20 48 65 6C
Unsigned 16 bit: 8264 Signed 16 bit: 8264 Float 32 bit: 1.697422E-19 Decimal: 032 072 101 108
Unsigned 16 bit: 8264 Float 64 bit: 3.63910130423513E-153 Octal: 040 110 145 154
Show little endian decoding Show unsigned as hexadecimal Binary: 00100000 01001000 01100101 01101100
Show unsigned as hexadecimal ASCII Text: Hel
Offset: 0x0 / 0x13f Selection: None INS

File Edit View Search Tools Help

out2Q3.bin

00000000 20 48 65 6C 6C 6F 20 74 68 65 72 65 2C 20 74 68 69 73 20 4E 65 68 61 20 52 61 6D

0000001e 65 73 68 20 47 61 77 61 6C 69 20 72 65 70 6F 72 74 69 6E 67 20 6C 69 76 65 20 66 72 6F 6D

0000003c 20 49 49 54 20 49 E2 80 99 6D 20 70 6C 61 6E 6E 69 6E 67 20 74 6F 20 63 6F 6D 70 6C 65 74

0000005a 65 20 43 53 34 35 38 20 28 69 6E 66 6F 73 65 63 29 20 74 68 69 73 20 73 70 72 69 6E 67 20

00000078 73 65 6D 65 73 74 65 72 2E 20 0A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

00000096 00

000000b4 00 00 00 00 00 00 00 00 00 00 00 00 00 00 BF EF 93 72 DB A1 FD EB DB 6F 77 27 EB 25 C6 C4 8C 31

000000d2 77 F7 01 C0 80 84 06 4A 37 E2 95 8C 28 46 D2 D9 D3 0C EC 2D 09 BF BF 08 4A 02 B8 2F 5B 3D

000000f0 5C 01 C4 65 F7 AF C5 4A 50 CD C6 25 44 D5 74 5E F2 55 09 8D 32 F1 79 09 4B 53 9D D8 D7 F9

0000010e AB 33 82 5F D9 98 F7 45 3E 25 CB 1A 86 CB EC C6 FD 6F B8 83 46 54 02 4C 08 BE 58 26 69 4F

0000012c 78 78 E2 2E 18 39 49 46 C3 A9 5D 66 86 1A BD AB 79 C1 12 C2

Hello there, this is Neha Ram
esh Gawali reporting live from
IIT I...m planning to complet
e CS458 (infosec) this spring
semester.
.....r....ow'.%..1
ww....J7...(.F....-.J..Z=...
\.e...JP..%D.t^U..2.y.KS...
.3...E>%.....o..FT.L..X&IO
xx...9IF..]f....y...

Signed 8 bit: 32 Unsigned 8 bit: 32 Signed 32 bit: 541615468 Unsigned 32 bit: 541615468 Hexadecimal: 20 48 65 6C
Signed 16 bit: 8264 Unsigned 16 bit: 8264 Float 32 bit: 1.697422E-19 Decimal: 032 072 101 108
Signed 16 bit: 8264 Unsigned 16 bit: 8264 Float 64 bit: 3.63910130423513E-153 Octal: 040 110 145 154
Show little endian decoding Show unsigned as hexadecimal Binary: 00100000 01001000 01100101 01101100
Show unsigned as hexadecimal ASCII Text: Hel
Offset: 0x0 / 0x13f Selection: None INS

Observation:

It shows that there are some differences, just few bytes difference. After multiple trials, it is noted that these differences are not constant. The data generated by md5collgen is not completely different for the two output files. Only the last 64 bytes of each file are different, and the first 64 bytes (which come from the prefix file) are identical. Therefore, the bytes that are different are the last 64 bytes of out1.bin and out2.bin.

Introduction to Information Security - CS 458

Neha Ramesh Gawali A20523722

2.2 Task 2: Understanding MD5's Property

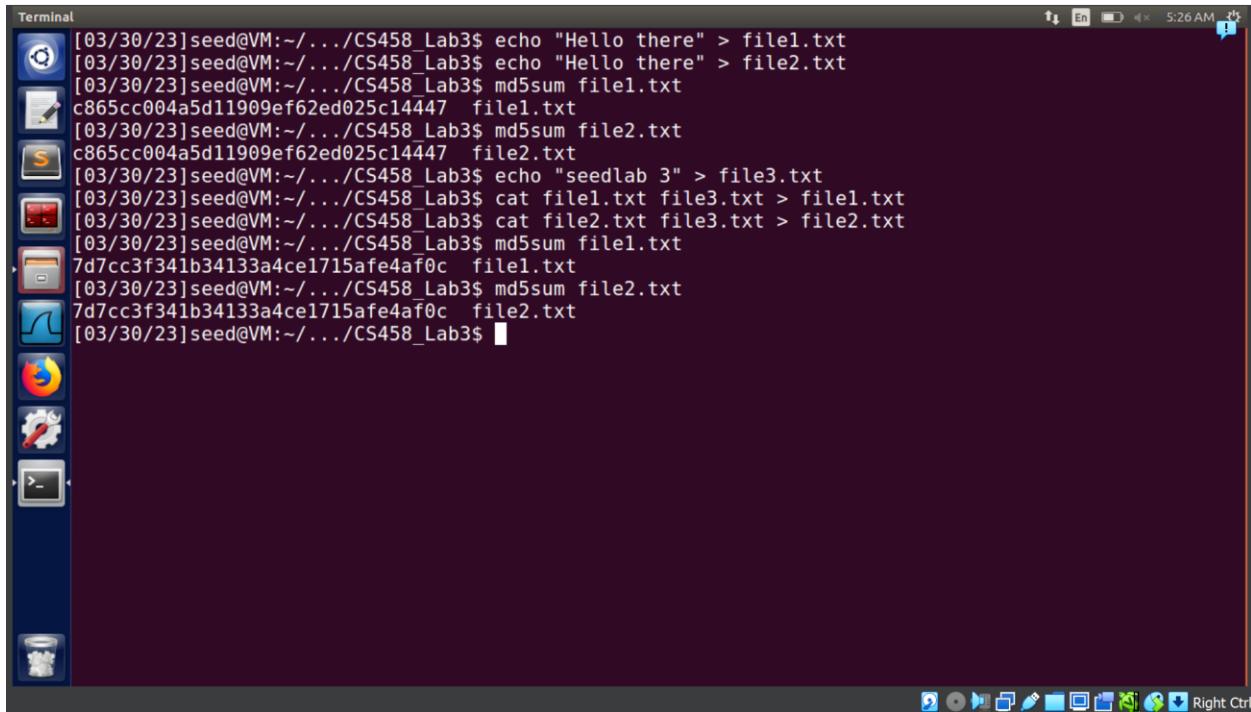
To conduct this experiment, I first generated two outputs using the md5sum command on two different files, let's call them file1 and file2. I then checked the MD5 hash of both files using the md5sum command to confirm that they were different.

Next, I concatenated a random string to the end of each file using the cat command, creating a new file as file3. I then checked the MD5 hash of new file using the md5sum command.

The results showed that even after adding the random string to each file, had the same MD5 hash value. This supports the property of the MD5 algorithm that states that if two inputs have the same hash, then adding the same suffix to both inputs will result in two outputs that have the same hash value.

The reason for this is that the MD5 algorithm processes the message in fixed blocks of 64 bytes and keeps an internal state by compressing the current blocks and state. Since the MD5 hashes of both file1 and file2 were the same, it can be inferred that the internal state of both files is also the same after algorithm execution. Therefore, when a random string is appended to each file, there will be a point at which the compression function for both files will run on the same block and state, resulting in both files having the same MD5 hash value.

In conclusion, this experiment demonstrates the property of the MD5 algorithm that if two inputs have the same hash, then adding the same suffix to both inputs will result in two outputs that have the same hash value.



The screenshot shows a Linux desktop environment with a terminal window open. The terminal window has a dark background and displays the following command-line session:

```
[03/30/23]seed@VM:~/.../CS458_Lab3$ echo "Hello there" > file1.txt
[03/30/23]seed@VM:~/.../CS458_Lab3$ echo "Hello there" > file2.txt
[03/30/23]seed@VM:~/.../CS458_Lab3$ md5sum file1.txt
c865cc004a5d11909ef62ed025c14447  file1.txt
[03/30/23]seed@VM:~/.../CS458_Lab3$ md5sum file2.txt
c865cc004a5d11909ef62ed025c14447  file2.txt
[03/30/23]seed@VM:~/.../CS458_Lab3$ echo "seedlab 3" > file3.txt
[03/30/23]seed@VM:~/.../CS458_Lab3$ cat file1.txt file3.txt > file1.txt
[03/30/23]seed@VM:~/.../CS458_Lab3$ cat file2.txt file3.txt > file2.txt
[03/30/23]seed@VM:~/.../CS458_Lab3$ md5sum file1.txt
7d7cc3f341b34133a4ce1715afe4af0c  file1.txt
[03/30/23]seed@VM:~/.../CS458_Lab3$ md5sum file2.txt
7d7cc3f341b34133a4ce1715afe4af0c  file2.txt
[03/30/23]seed@VM:~/.../CS458_Lab3$
```

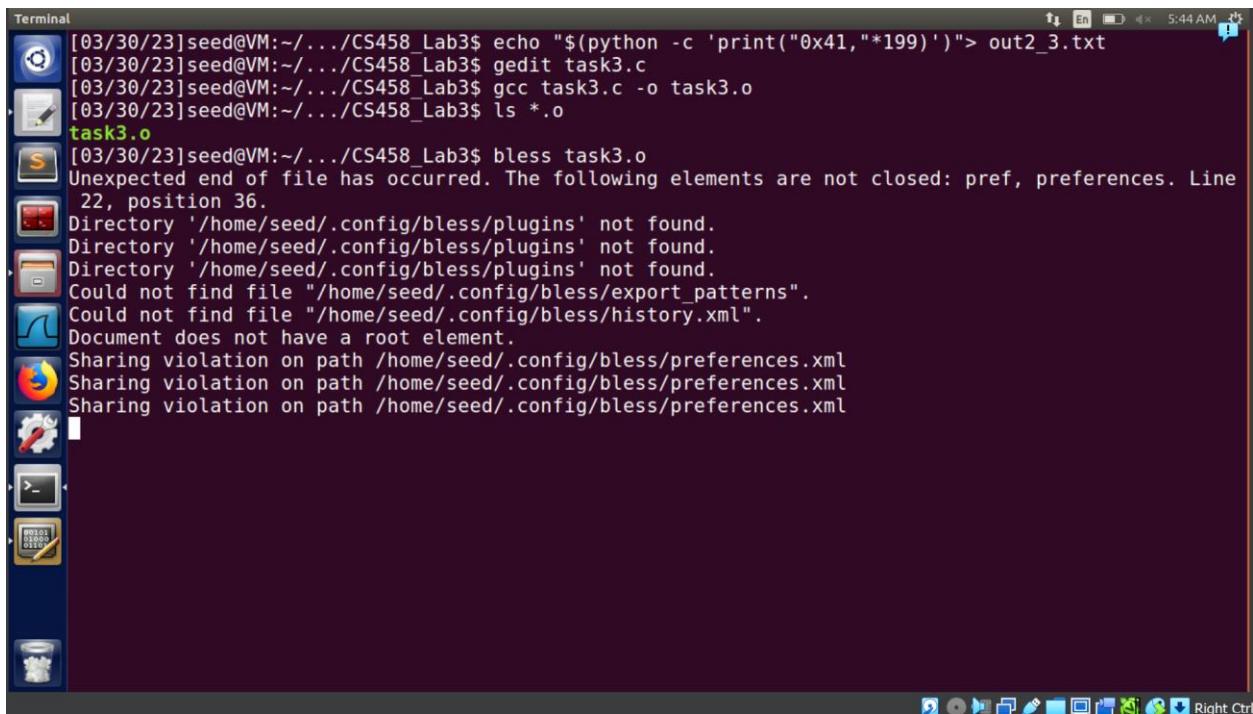
Introduction to Information Security - CS 458

Neha Ramesh Gawali A20523722

2.3 Task 3: Generating Two Executable Files with the Same MD5 Hash

Based on the given procedure, the objective was to create two different binary files with the same MD5 hash value. The process involved creating a prefix and suffix for the binary files. The prefix was determined by dividing the total number of bytes (4160) by the prefix size (64), which resulted in 65. The prefix was then added to the total number of bytes, which resulted in 4224. This was then used as the prefix (task3p) for the binary files. Similarly, the suffix (task3s) was created by adding 128 to the total number of bytes, which resulted in 4353.

After creating the prefix and suffix, the p q values were taken from the bins file, and both the prefix and suffix were concatenated. This resulted in two binary files with the same MD5 hash value. These files were then checked using the bless command. It was observed that both the files had the same MD5 hash value but different suffixes. To validate the experiment, executable permission was added to both files and they were run. It was noted that the output was different, which proves that the experiment was successful in creating two different binaries from a single binary, both producing different output but having the same md5 hash.

A screenshot of a Linux desktop environment. On the left, there's a vertical dock with icons for a terminal, file manager, browser, and system settings. The main area shows a terminal window titled "Terminal". The terminal output is as follows:

```
[03/30/23]seed@VM:~/.../CS458_Lab3$ echo "$(python -c 'print("0x41,*199")')> out2_3.txt
[03/30/23]seed@VM:~/.../CS458_Lab3$ gedit task3.c
[03/30/23]seed@VM:~/.../CS458_Lab3$ gcc task3.c -o task3.o
[03/30/23]seed@VM:~/.../CS458_Lab3$ ls *.o
task3.o
[03/30/23]seed@VM:~/.../CS458_Lab3$ bless task3.o
Unexpected end of file has occurred. The following elements are not closed: pref, preferences. Line 22, position 36.
Directory '/home/seed/.config/bless/plugins' not found.
Directory '/home/seed/.config/bless/plugins' not found.
Directory '/home/seed/.config/bless/plugins' not found.
Could not find file "/home/seed/.config/bless/export_patterns".
Could not find file "/home/seed/.config/bless/history.xml".
Document does not have a root element.
Sharing violation on path /home/seed/.config/bless/preferences.xml
Sharing violation on path /home/seed/.config/bless/preferences.xml
Sharing violation on path /home/seed/.config/bless/preferences.xml
```

The terminal window has a dark background and light-colored text. The desktop interface includes a taskbar at the bottom with various application icons.

Introduction to Information Security - CS 458

Neha Ramesh Gawali A20523722

The screenshot shows the Gedit text editor interface. The title bar indicates it is running on a Mac OS X system at 5:45 AM. The menu bar includes 'File', 'Edit', 'View', 'Search', 'Format', 'Tools', and 'Help'. The toolbar on the left includes icons for Open, Save, Print, Undo, Redo, Cut, Copy, Paste, Find, Replace, and a terminal-like icon. The main window has two tabs: 'out2_3.txt' and 'task3.c'. The 'out2_3.txt' tab displays a repeating sequence of the byte value 0x41 (ASCII 'A') from memory address 0x00000000 to 0x0000000F. The 'task3.c' tab shows the following C code:

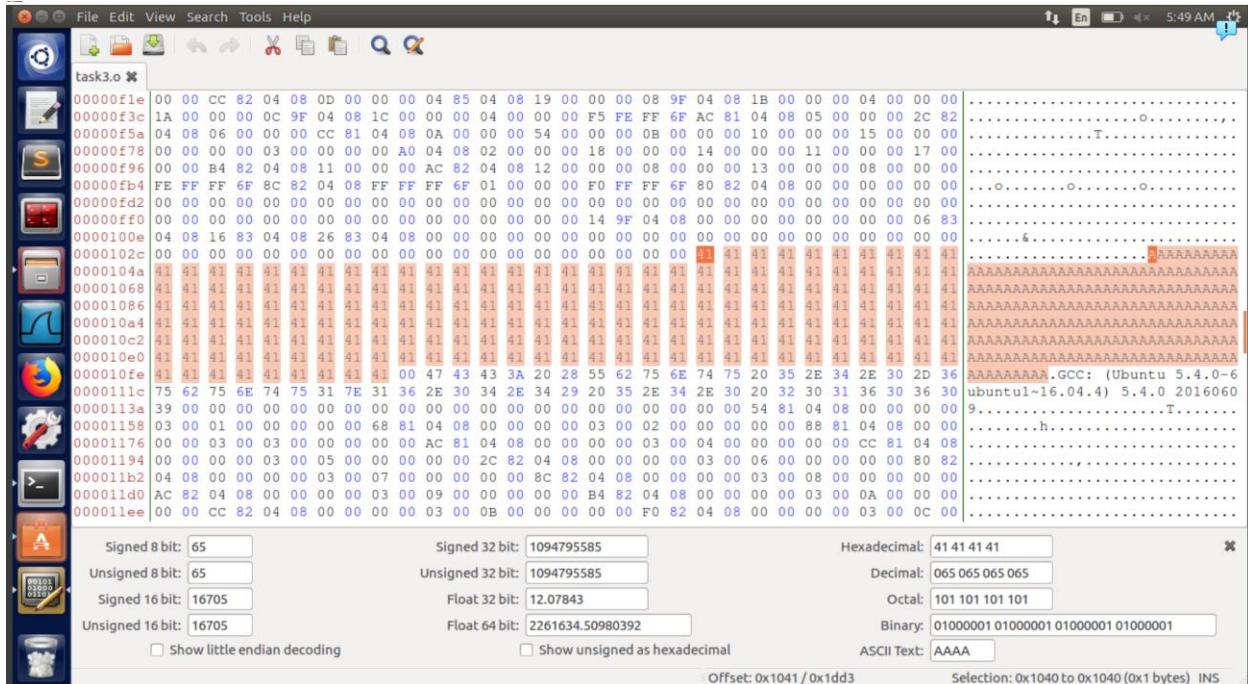
```
task3.c

#include <cs50.h>
#include <stdio.h>

int main(void)
{
    string s = get_string("Name: ");
    printf("%s\n", s);
}
```

Introduction to Information Security - CS 458

Neha Ramesh Gawali A20523722



We can run md5collgen on the prefix to generate two outputs that have the same MD5 hash value .

```
[03/30/23]seed@VM:~/.../CS458_Lab3$ head -c 4224 task3.o > prefix
[03/30/23]seed@VM:~/.../CS458_Lab3$ md5collgen -p prefix -o task3_a.bin task3_b.bin
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'task3_a.bin' and 'task3_b.bin'
Using prefixfile: 'prefix'
Using initial value: 94c117edf9c5096b3ce02d2596ec96e0

Generating first block: .....
Generating second block: S10.....
Running time: 18.2331 s
[03/30/23]seed@VM:~/.../CS458_Lab3$ ls *.bin
out1.bin    out1Q3.bin   out2Q2.bin   task3_a.bin
out1Q2.bin  out2.bin    out2Q3.bin   task3_b.bin
[03/30/23]seed@VM:~/.../CS458_Lab3$ tail -c +4352 task3.o > suffix
[03/30/23]seed@VM:~/.../CS458_Lab3$ tail -c 128 task3_a.bin > p
[03/30/23]seed@VM:~/.../CS458_Lab3$ tail -c 128 task3_b.bin > q
[03/30/23]seed@VM:~/.../CS458_Lab3$ cat prefix p suffix > task3_1
[03/30/23]seed@VM:~/.../CS458_Lab3$ cat prefix q suffix > task3_2
```

Introduction to Information Security - CS 458

Neha Ramesh Gawali A20523722

In conclusion, the experiment was successful in creating two different binary files with the same MD5 hash value. This could have significant implications in various fields such as cybersecurity and digital forensics.

Overall, this task demonstrates the potential vulnerabilities in relying solely on the MD5 hash for verifying the integrity of executable files, as it is possible to generate different files with the same hash.

2.4 Task 4: Making the Two Programs Behave Differently

This experiment highlights the potential vulnerabilities in relying solely on the MD5 hash for verifying the integrity of executable files. It demonstrates how it is possible to create two programs with the same MD5 hash but different behaviour, and how one can use a benign program's MD5 hash to obtain certification for a malicious program.

To achieve this, the program was modified by incorporating two arrays, one containing benign code and the other containing malicious code. The prefix and suffix were identified, and the p and q values were obtained by generating two outputs using the md5 command. The benign code

Introduction to Information Security - CS 458

Neha Ramesh Gawali A20523722

was obtained by combining the prefix, p values, and suffix, while the malicious code was obtained by replacing the first p value with multiple qs and rearranging the layout.

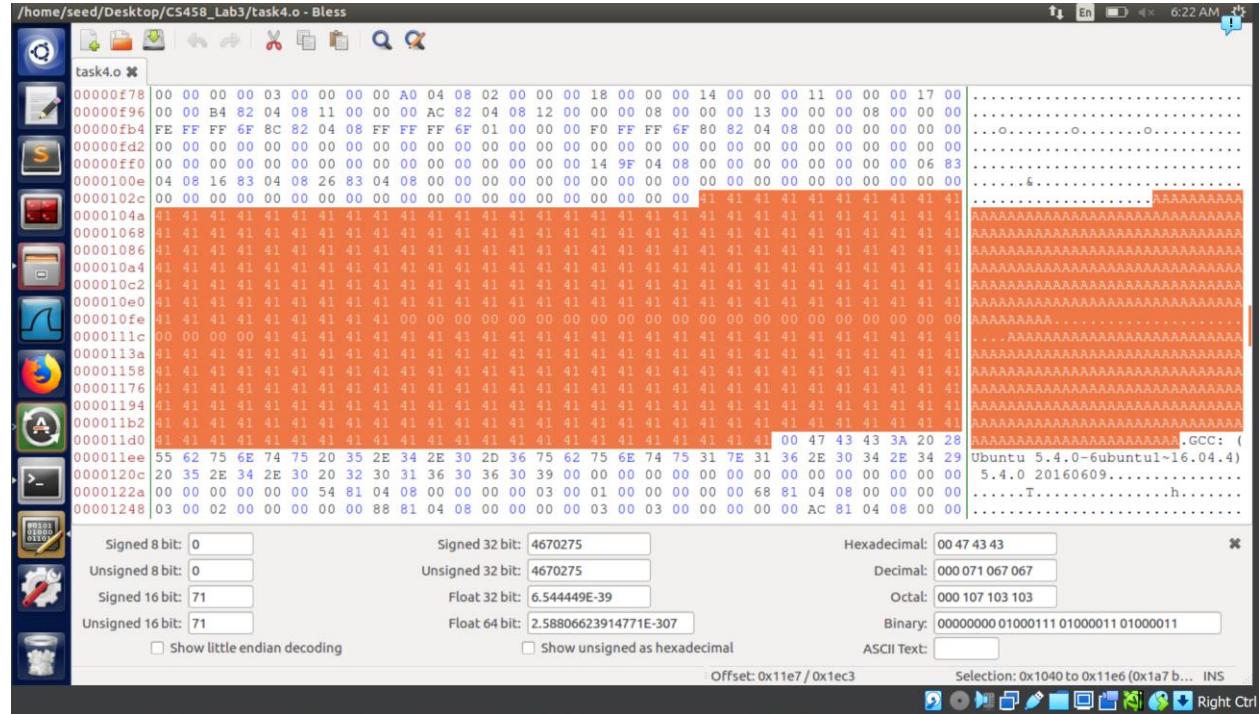
The process of obtaining the desired output took around three hours of experimentation. This shows that with some effort, it is possible to create two different programs with the same MD5 hash, and that relying solely on the hash to verify program integrity can be risky.

Overall, this experiment demonstrates the need for additional security measures beyond MD5 hash verification, such as digital signatures, code signing certificates, and other cryptographic methods to ensure that the program being executed is the intended one.

```
[03/30/23]seed@VM:~/.../CS458_Lab3$ gcc task2_4.c -o task4.o  
[03/30/23]seed@VM:~/.../CS458_Lab3$ bless task4.o
```

Introduction to Information Security - CS 458

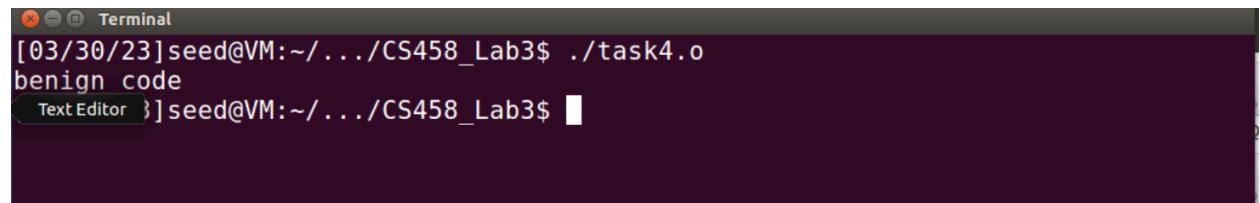
Neha Ramesh Gawali A20523722



```
[03/30/23]seed@VM:~/.../CS458_Lab3$ head -c 4106 task4.o > prefix
[03/30/23]seed@VM:~/.../CS458_Lab3$ md5collgen -p prefix -o task4_a.bin task4_b.bin
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'task4_a.bin' and 'task4_b.bin'
Using prefixfile: 'prefix'
Using initial value: 20f59f7e9f47c7021ccf15966ff2633c

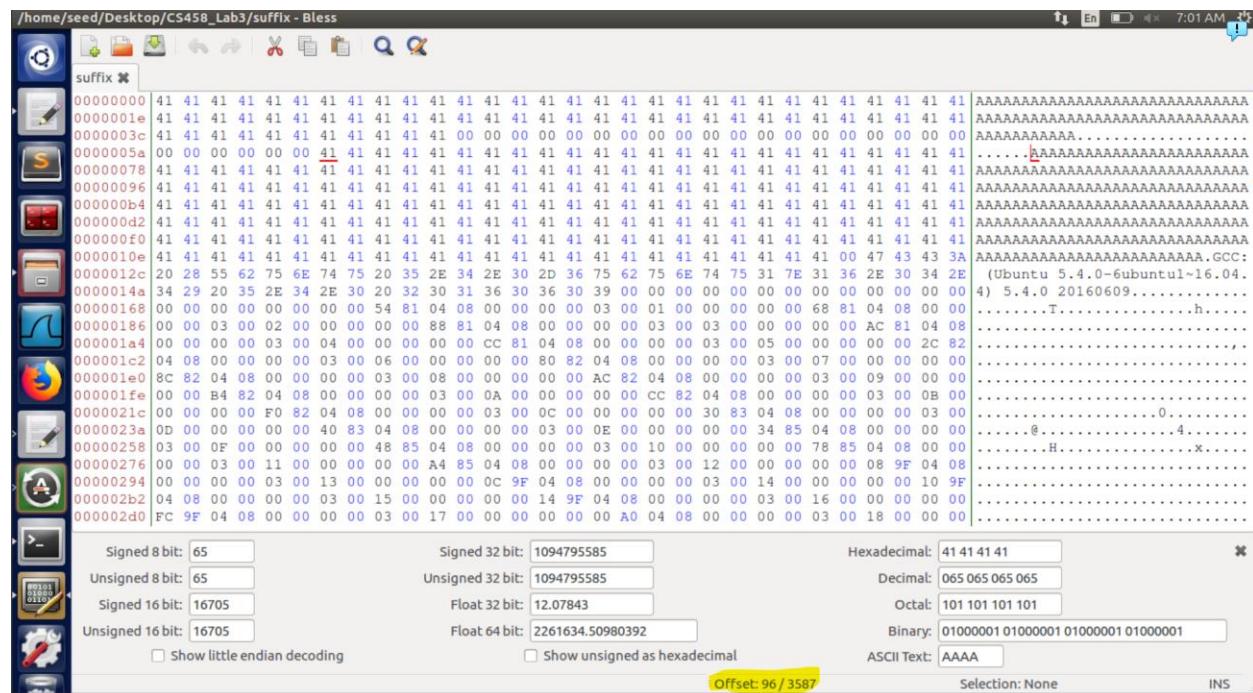
Generating first block: .....
Generating second block: W.....
Running time: 19.6034 s
[03/30/23]seed@VM:~/.../CS458_Lab3$ tail -c 128 task4_a.bin > p
[03/30/23]seed@VM:~/.../CS458_Lab3$ tail -c 128 task4_b.bin > p
[03/30/23]seed@VM:~/.../CS458_Lab3$
```



Introduction to Information Security - CS 458

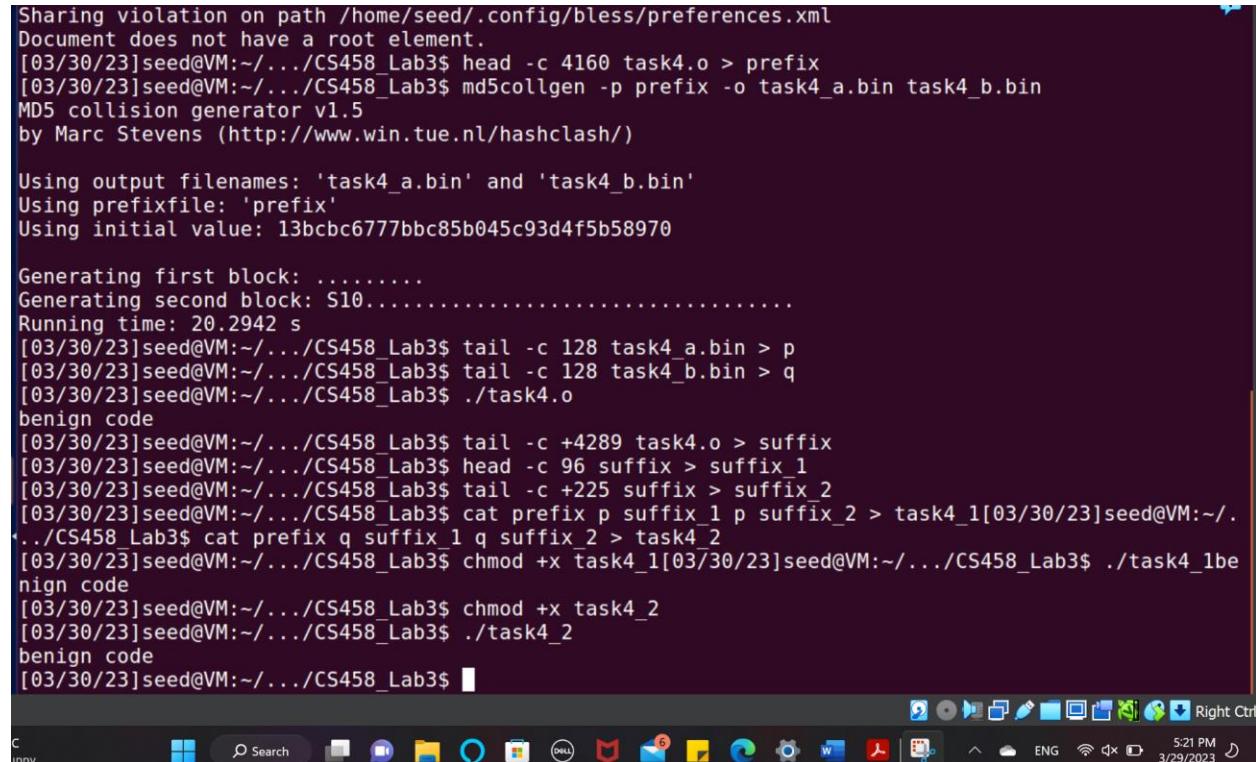
Neha Ramesh Gawali A20523722

```
[04/06/23]seed@VM:~/.../CS458_Lab3$ rm task4.o
rm: cannot remove 'task4.o': No such file or directory
[04/06/23]seed@VM:~/.../CS458_Lab3$ gcc task4.c -o task4.o
[04/06/23]seed@VM:~/.../CS458_Lab3$ ./task4.o
benign code
[04/06/23]seed@VM:~/.../CS458_Lab3$ head -c 4160 task4.o > prefix
[04/06/23]seed@VM:~/.../CS458_Lab3$ md5collgen -p prefix -o task4_a.bin task4_b.bin
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)
Using output filenames: 'task4_a.bin' and 'task4_b.bin'
Using prefixfile: 'prefix'
Using initial value: 13bcdbc6777bbc85b045c93d4f5b58970
Generating first block: .
Generating second block: W...
Running time: 1.39148 s
[04/06/23]seed@VM:~/.../CS458_Lab3$ tail -c 128 task4_a.bin > p
[04/06/23]seed@VM:~/.../CS458_Lab3$ tail -c 128 task4_b.bin > q
[04/06/23]seed@VM:~/.../CS458_Lab3$ gcc task4.c -o task4.o
[04/06/23]seed@VM:~/.../CS458_Lab3$ ./task4.o
WARNING: malicious code
[04/06/23]seed@VM:~/.../CS458_Lab3$
```



Introduction to Information Security - CS 458

Neha Ramesh Gawali A20523722



```
Sharing violation on path /home/seed/.config/bless/preferences.xml
Document does not have a root element.
[03/30/23]seed@VM:~/.../CS458_Lab3$ head -c 4160 task4.o > prefix
[03/30/23]seed@VM:~/.../CS458_Lab3$ md5collgen -p prefix -o task4_a.bin task4_b.bin
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'task4_a.bin' and 'task4_b.bin'
Using prefixfile: 'prefix'
Using initial value: 13bcdbc6777bbc85b045c93d4f5b58970

Generating first block: .....
Generating second block: $10.....
Running time: 20.2942 s
[03/30/23]seed@VM:~/.../CS458_Lab3$ tail -c 128 task4_a.bin > p
[03/30/23]seed@VM:~/.../CS458_Lab3$ tail -c 128 task4_b.bin > q
[03/30/23]seed@VM:~/.../CS458_Lab3$ ./task4_o
benign code
[03/30/23]seed@VM:~/.../CS458_Lab3$ tail -c +4289 task4.o > suffix
[03/30/23]seed@VM:~/.../CS458_Lab3$ head -c 96 suffix > suffix_1
[03/30/23]seed@VM:~/.../CS458_Lab3$ tail -c +225 suffix > suffix_2
[03/30/23]seed@VM:~/.../CS458_Lab3$ cat prefix p suffix_1 p suffix_2 > task4_1[03/30/23]seed@VM:~/.../CS458_Lab3$ ./task4_1benign code
[03/30/23]seed@VM:~/.../CS458_Lab3$ chmod +x task4_1[03/30/23]seed@VM:~/.../CS458_Lab3$ ./task4_1benign code
[03/30/23]seed@VM:~/.../CS458_Lab3$ ./task4_2
benign code
[03/30/23]seed@VM:~/.../CS458_Lab3$ [03/30/23]seed@VM:~/.../CS458_Lab3$ Right Ctrl
```

hashes of both files will be the same even though their outputs are different. This demonstrates a potential vulnerability in relying solely on the MD5 hash to verify the integrity of executable files.

To achieve the desired result, the experiment involved creating two independent executable files that have the same prefix and suffix but different behavior. The prefix was created by dividing the total size of the binary file by 64, and the suffix was created by adding 128 bytes to the total size of the binary file. The P and Q values were taken from the binary file, and both files were concatenated with the prefix and suffix to create two executable files with the same MD5 hash.

To make one file output benign code and the other file output malicious code, the second array's contents were modified to be equivalent to one of the created arrays. The contents after the second array were added to the suffix, and the first array was transferred from out1arrcomplete to compare. This way, one file outputs benign code, and the other outputs where malicious code would be run.

Finally, both files were made executable, and their MD5 hash sums were calculated. Executing both files gives the desired results, and it is evident that the MD5 hashes of both files are the same, despite their different outputs.

Introduction to Information Security - CS 458

Neha Ramesh Gawali A20523722

Overall, this experiment highlights the potential vulnerabilities in relying solely on the MD5 hash to verify the integrity of executable files. As demonstrated in this experiment, it is possible to generate two different files with the same MD5 hash, which can lead to security risks if not properly addressed. It is important to use multiple verification methods to ensure the security and integrity of executable files.