Lab 3: MD5 Collision

Sahil Sheikh A20518693

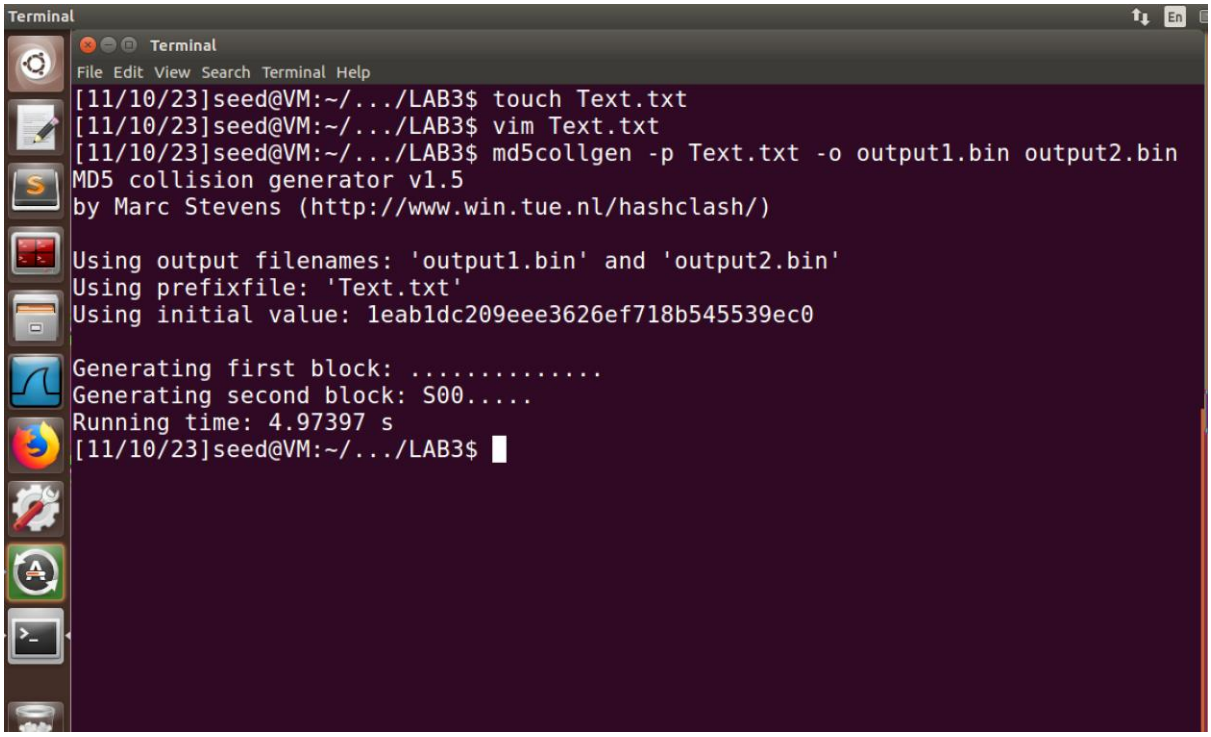
Introduction:

A collision in the context of cryptographic hash function MD5 (Message Digest Algorithm 5), occurs when two different inputs produce the same hash output. Ideally, a cryptographic hash function should produce a unique hash value for each unique input, making it computationally infeasible for two different inputs to generate the same hash value.

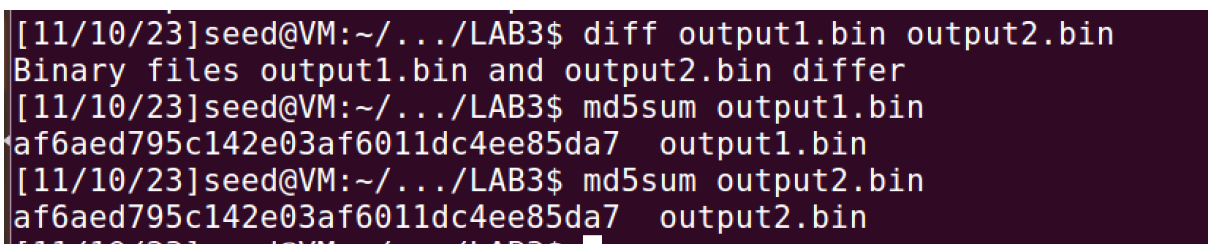
However, MD5 is considered broken for cryptographic purposes because researchers have demonstrated the ability to find collisions efficiently. Because of such vulnerabilities, MD5 is no longer considered secure for cryptographic purposes, and more secure hash functions like SHA-256 are recommended for applications requiring collision resistance and cryptographic strength. It's important to note that cryptographic hash functions should ideally be resistant to collisions, meaning that it should be computationally infeasible to find two different inputs that produce the same hash value.

Task I: Generating two files with same hash

Creating a file “Text.txt” then creating two md5 file with same hash values



Using diff and md5sum command to check if the files are unique or not



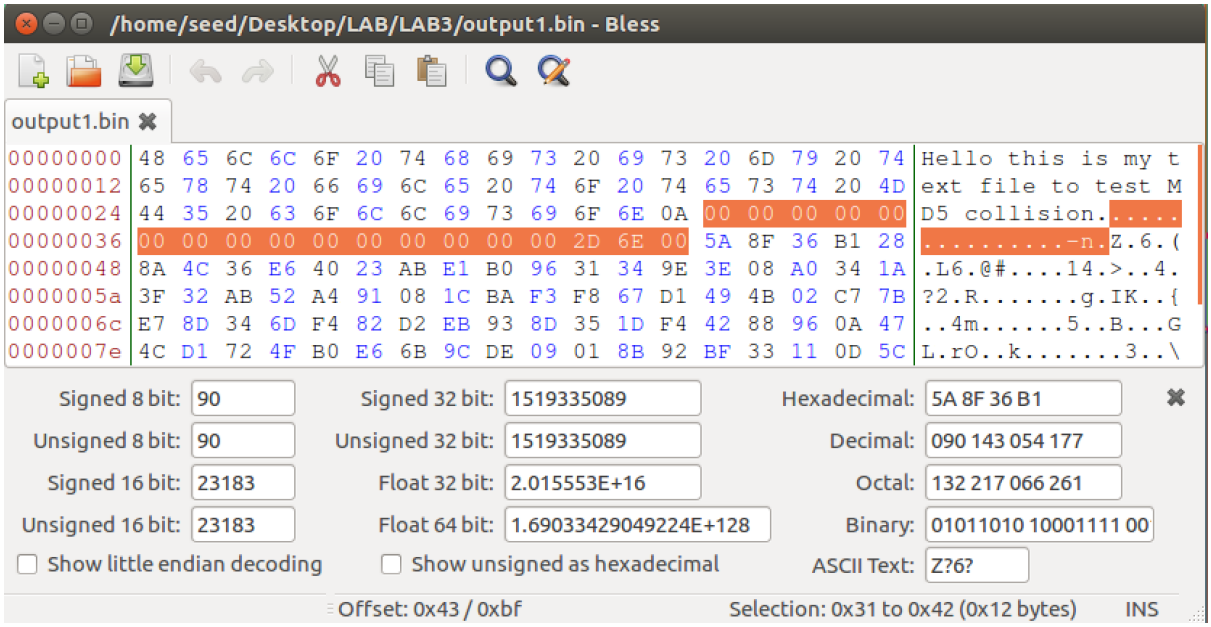
Q1.

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

In the above case the size of my file, “Text.txt” was 49 bytes(refer below for size related information)

```
[11/10/23]seed@VM:~/.../LAB3$ ls -l
total 12
-rw-rw-r-- 1 seed seed 192 Nov 10 10:58 output1.bin
-rw-rw-r-- 1 seed seed 192 Nov 10 10:58 output2.bin
-rw-rw-r-- 1 seed seed  49 Nov 10 10:56 Text.txt
[11/10/23]seed@VM:~/.../LAB3$ ls -lh
total 12K
-rw-rw-r-- 1 seed seed 192 Nov 10 10:58 output1.bin
-rw-rw-r-- 1 seed seed 192 Nov 10 10:58 output2.bin
-rw-rw-r-- 1 seed seed  49 Nov 10 10:56 Text.txt
[11/10/23]seed@VM:~/.../LAB3$ ls -sh
total 12K
4.0K output1.bin  4.0K output2.bin  4.0K Text.txt
```

The file will be padded with zeros . I’ll view the output1.bin file in hex editor to verify this

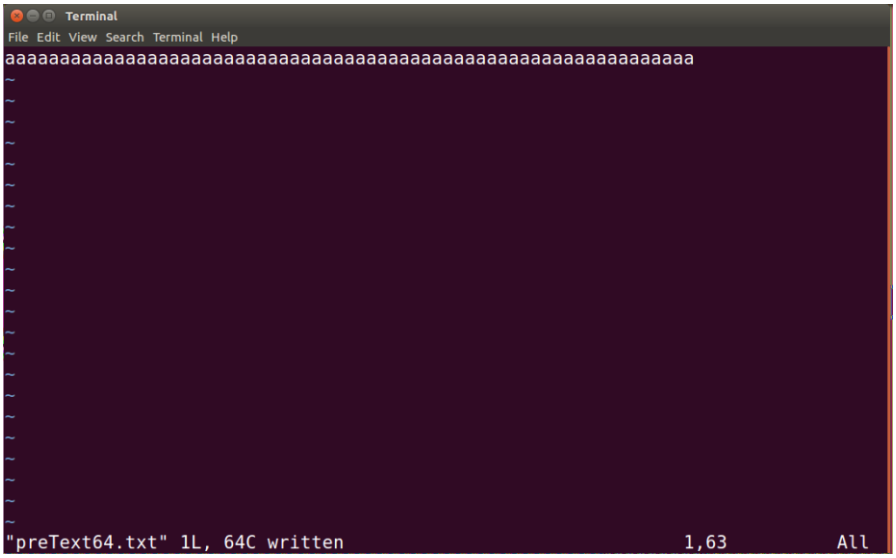


Q2.

Create a prefix file with exactly 64 bytes, and run the collision tool again, and see what happens.

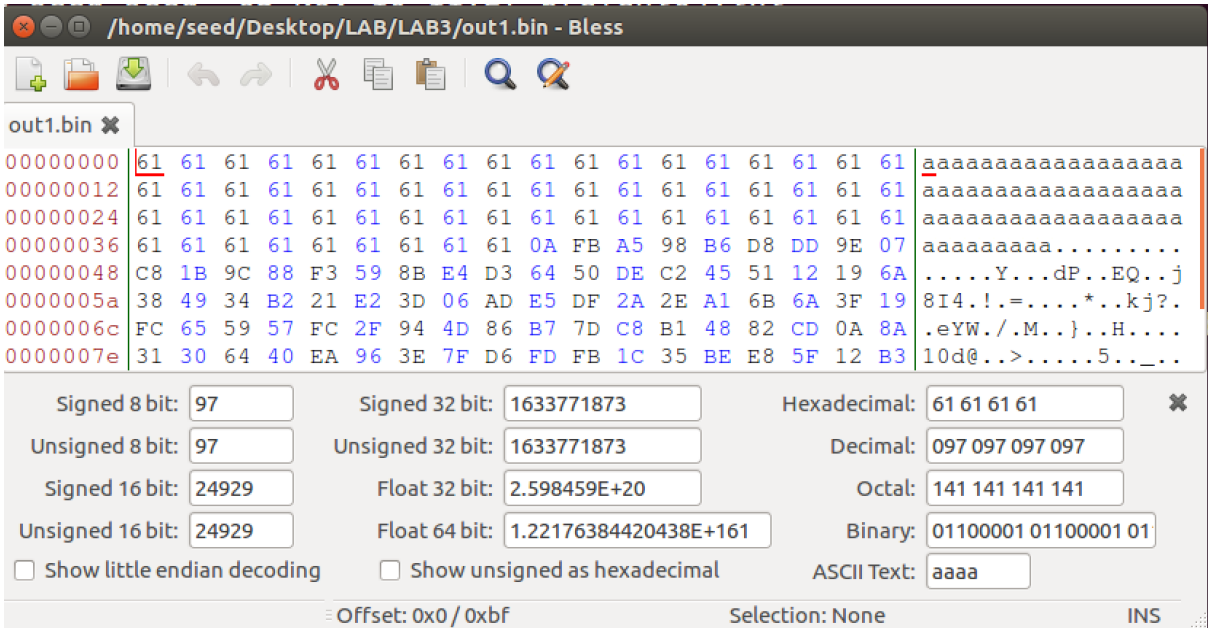
Creating a new pretext file “preText.txt” of 64 bytes in size then creating two md5 file with same hash values. Then using diff and md5sum command to check if the files are unique or not

```
[11/10/23]seed@VM:~/.../LAB3$ touch preText64.txt
[11/10/23]seed@VM:~/.../LAB3$ vim preText64.txt
[11/10/23]seed@VM:~/.../LAB3$ ls -l
total 16
-rw-rw-r-- 1 seed seed 192 Nov 10 10:58 output1.bin
-rw-rw-r-- 1 seed seed 192 Nov 10 10:58 output2.bin
-rw-rw-r-- 1 seed seed  64 Nov 10 11:28 preText64.txt
-rw-rw-r-- 1 seed seed  49 Nov 10 10:56 Text.txt
[11/10/23]seed@VM:~/.../LAB3$
```



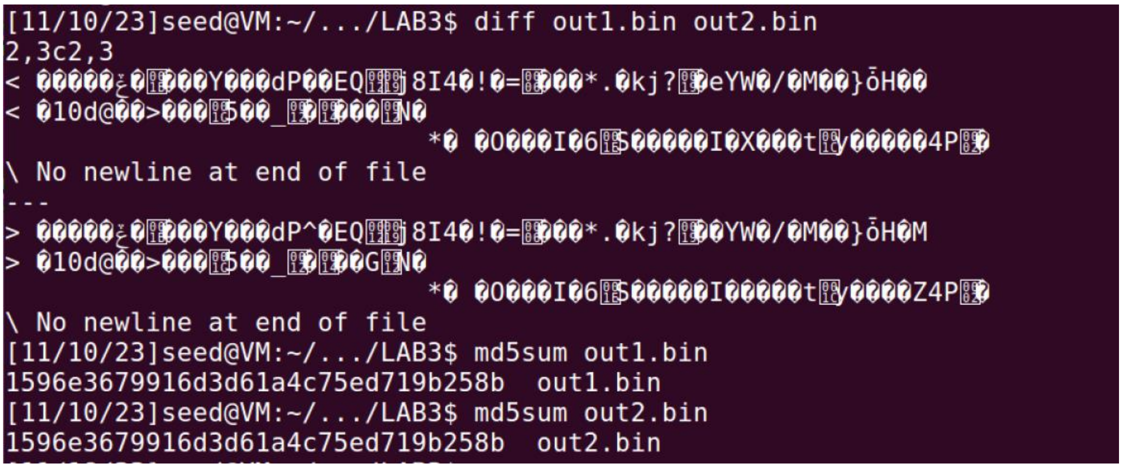

```
[11/10/23]seed@VM:~/.../LAB3$ md5collgen -p preText64.txt -o out1.bin out2.bin
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'out1.bin' and 'out2.bin'
Using prefixfile: 'preText64.txt'
Using initial value: acc715499c2063df53c1c98cb2187a3a

Generating first block: ......
Generating second block: W.
Running time: 2.45873 s
```

I can observe this file of size 64 bytes has no padding added to when the hash file is generated



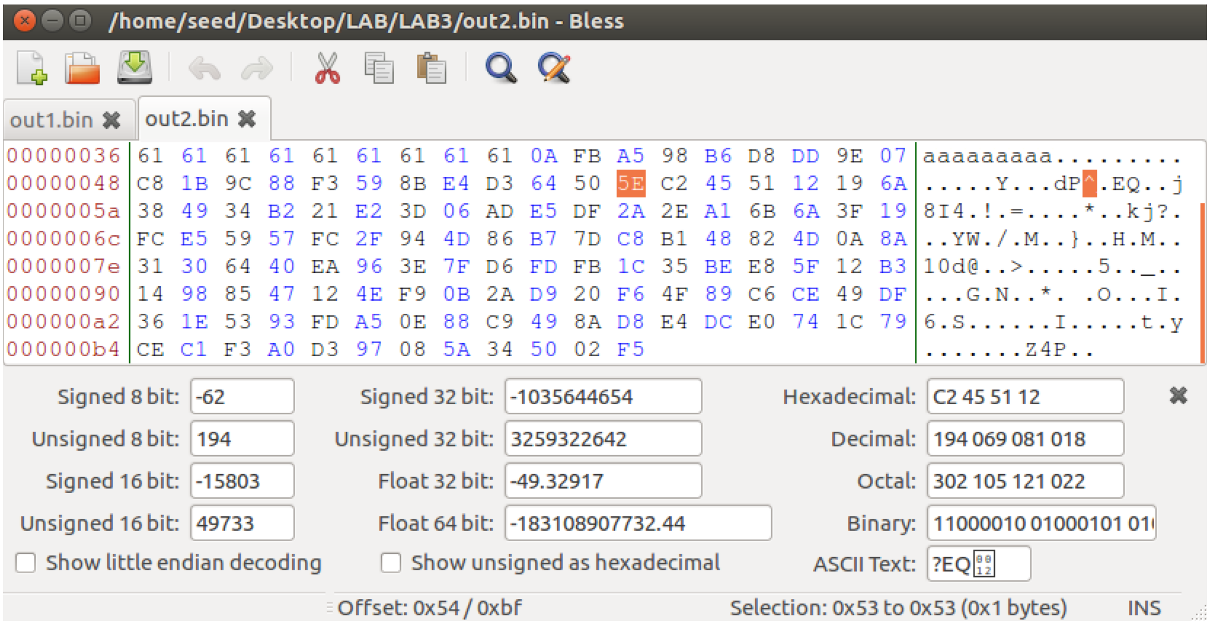
Q3.

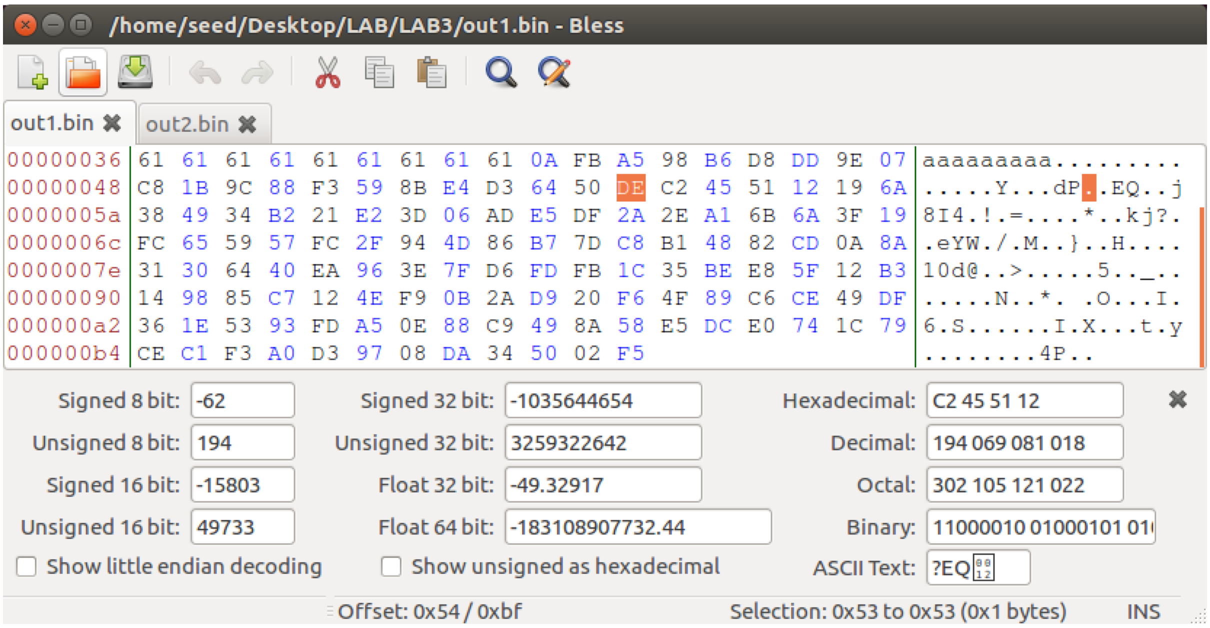
Are the data (128 bytes) generated by md5collgen completely different for the two output files? Please identify all the bytes that are different.

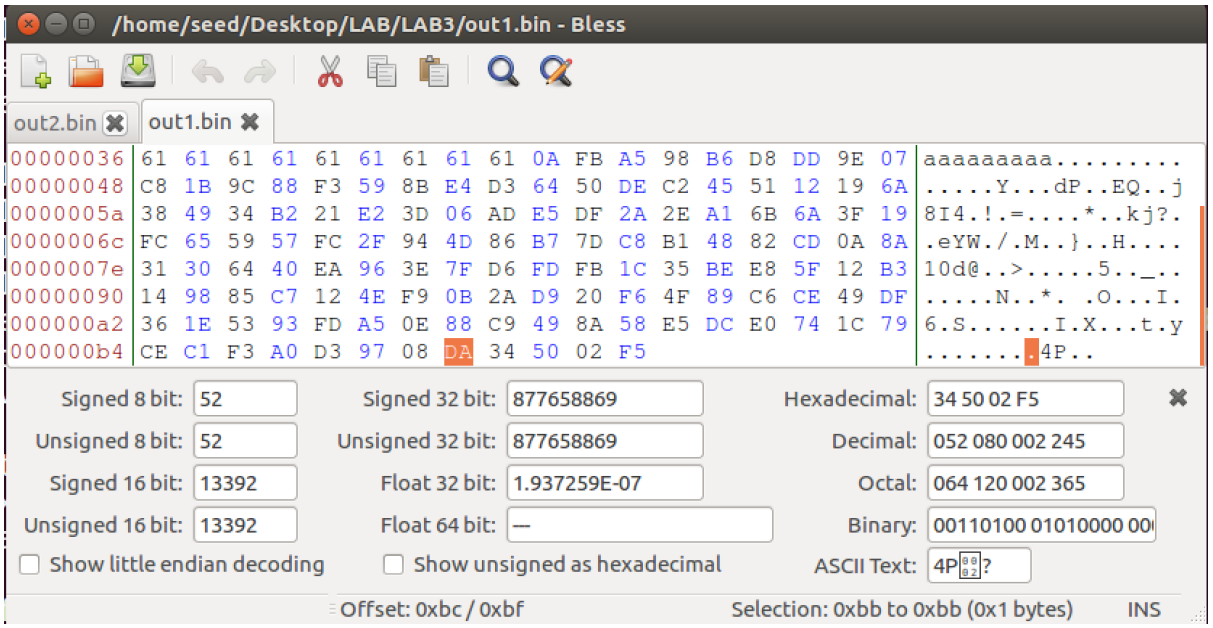


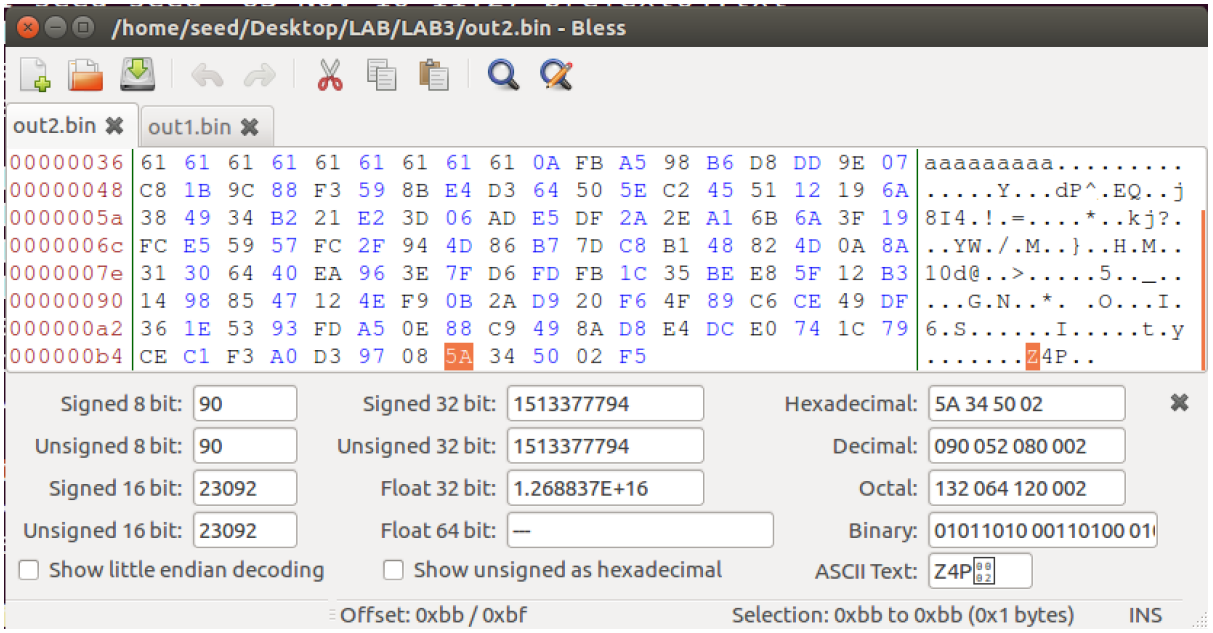
Both the output files out1.bin and out2.bin have the same hash values. (refer above pic)

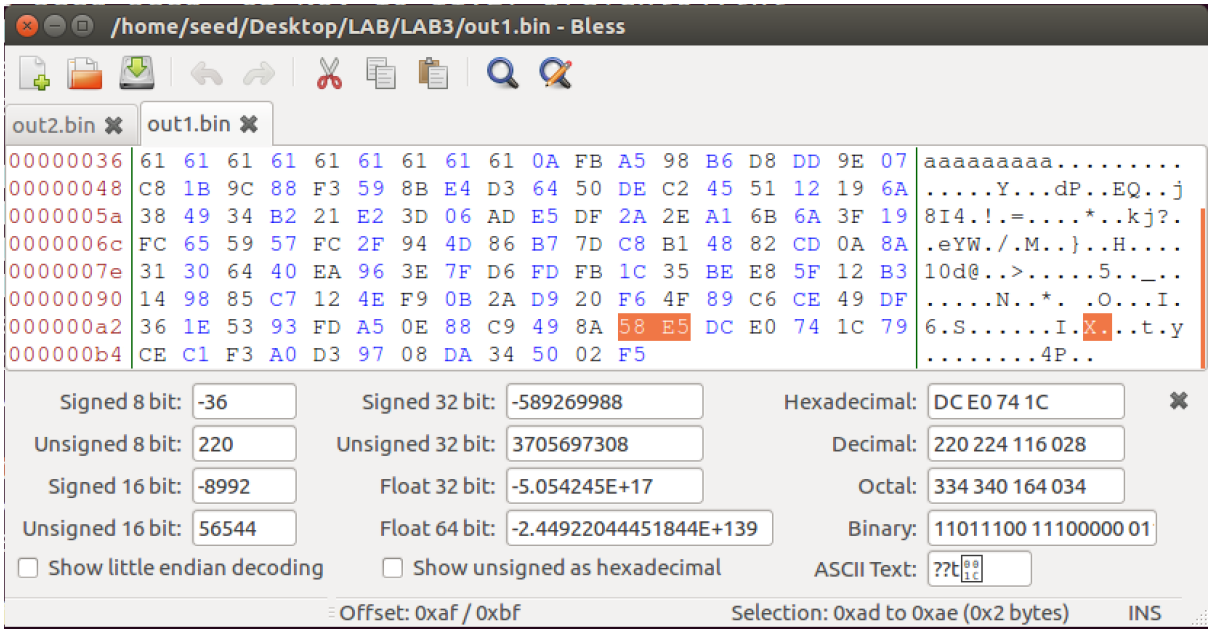
Out of all the 128 bytes generated by the md5collgen few of the bytes are different. Here are those bits:

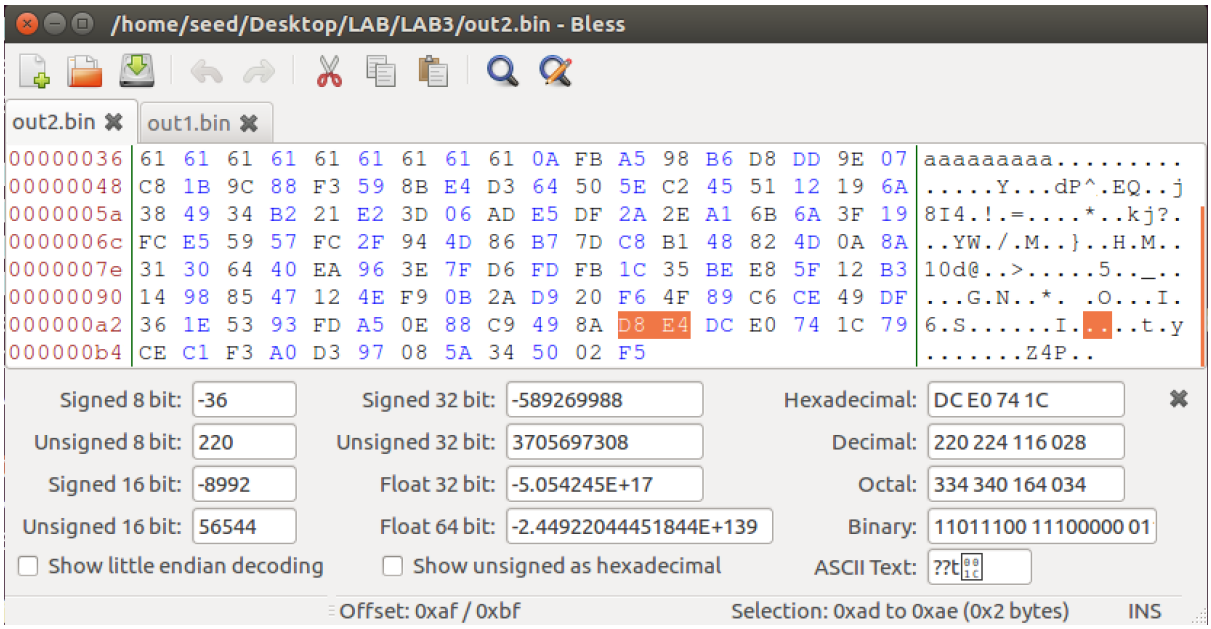


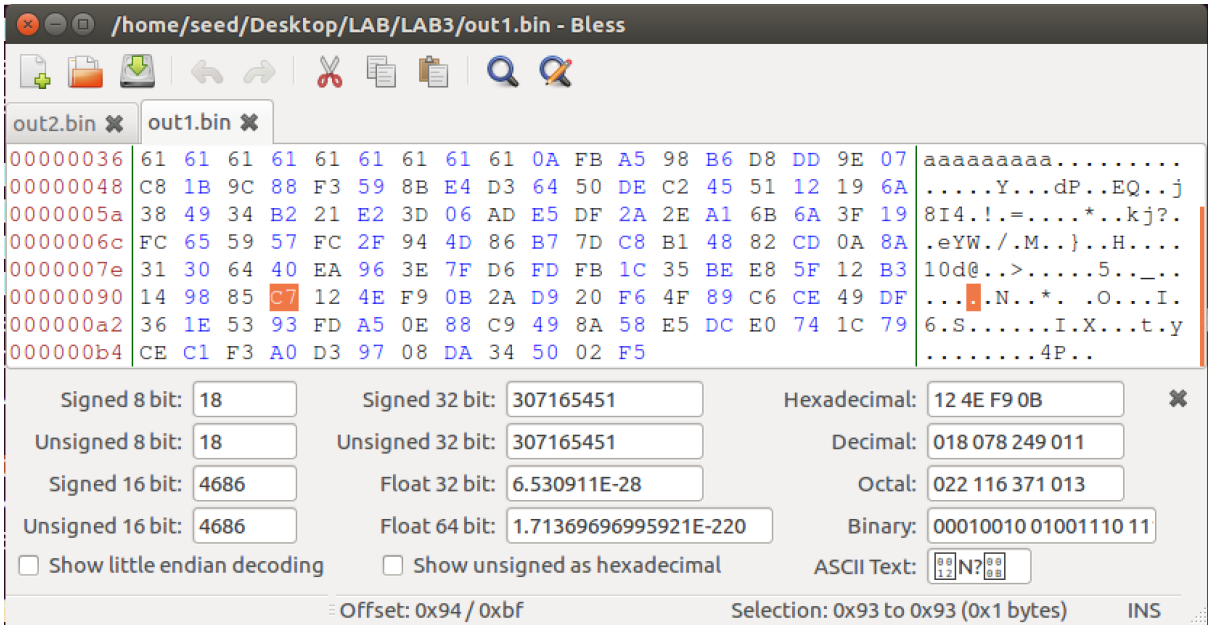


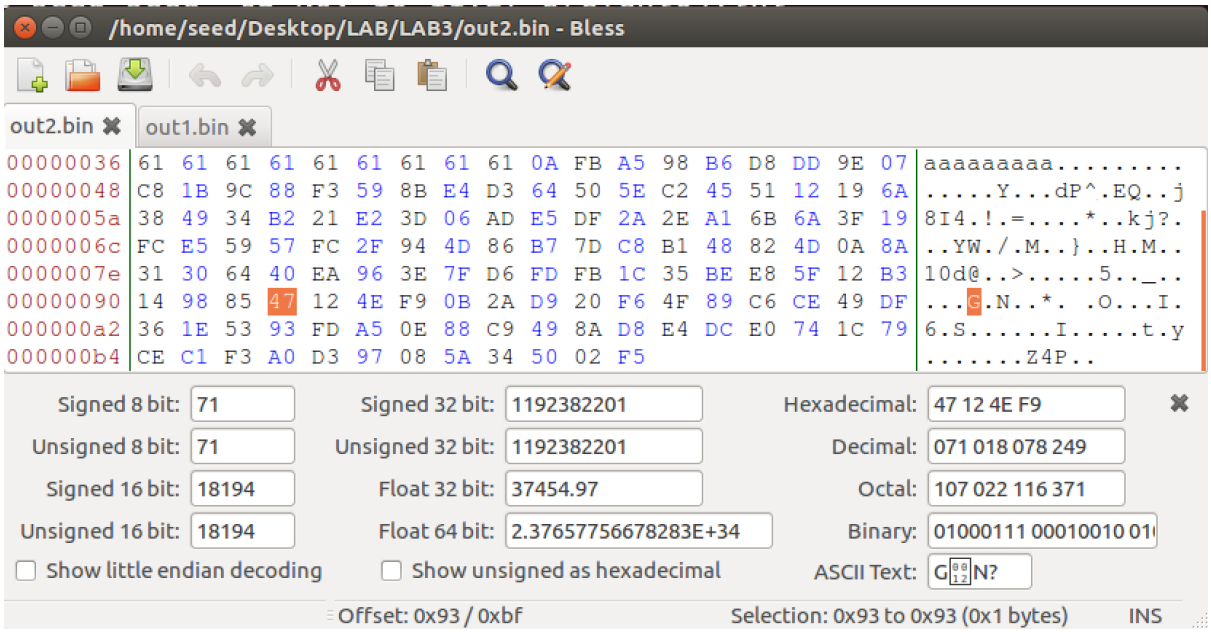

/home/seed/Desktop/LAB/LAB3/out1.bin - Bless
out2.bin
out1.bin
00000036 61 61 61 61 61 61 61 61 61 0A FB A5 98 B6 D8 DD 9E 07 aaaaaaaaa.........
00000048 C8 1B 9C 88 F3 59 8B E4 D3 64 50 DE C2 45 51 12 19 6A .....Y...dP..EQ..j
0000005a 38 49 34 B2 21 E2 3D 06 AD E5 DF 2A 2E A1 6B 6A 3F 19 8I4.!.=....*..kj?.
0000006c FC 65 59 57 FC 2F 94 4D 86 B7 7D C8 B1 48 82 CD 0A 8A .eYW./.M..}..H....
0000007e 31 30 64 40 EA 96 3E 7F D6 FD FB 1C 35 BE E8 5F 12 B3 10d@..>.....5._..
00000090 14 98 85 C7 12 4E F9 0B 2A D9 20 F6 4F 89 C6 CE 49 DF .....N..*. .O...I.
000000a2 36 1E 53 93 FD A5 0E 88 C9 49 8A 58 E5 DC E0 74 1C 79 6.S......I.X...t.y
000000b4 CE C1 F3 A0 D3 97 08 DA 34 50 02 F5 ....4P..
Signed 8 bit: 52
Signed 32 bit: 877658869
Hexadecimal: 34 50 02 F5
Unsigned 8 bit: 52
Unsigned 32 bit: 877658869
Decimal: 052 080 002 245
Signed 16 bit: 13392
Float 32 bit: 1.937259E-07
Octal: 064 120 002 365
Unsigned 16 bit: 13392
Float 64 bit: —
Binary: 00110100 01010000 00
Show little endian decoding
Show unsigned as hexadecimal
ASCII Text: 4P?
Offset: 0xbc / 0xbf
Selection: 0xbb to 0xbb (0x1 bytes)
INS

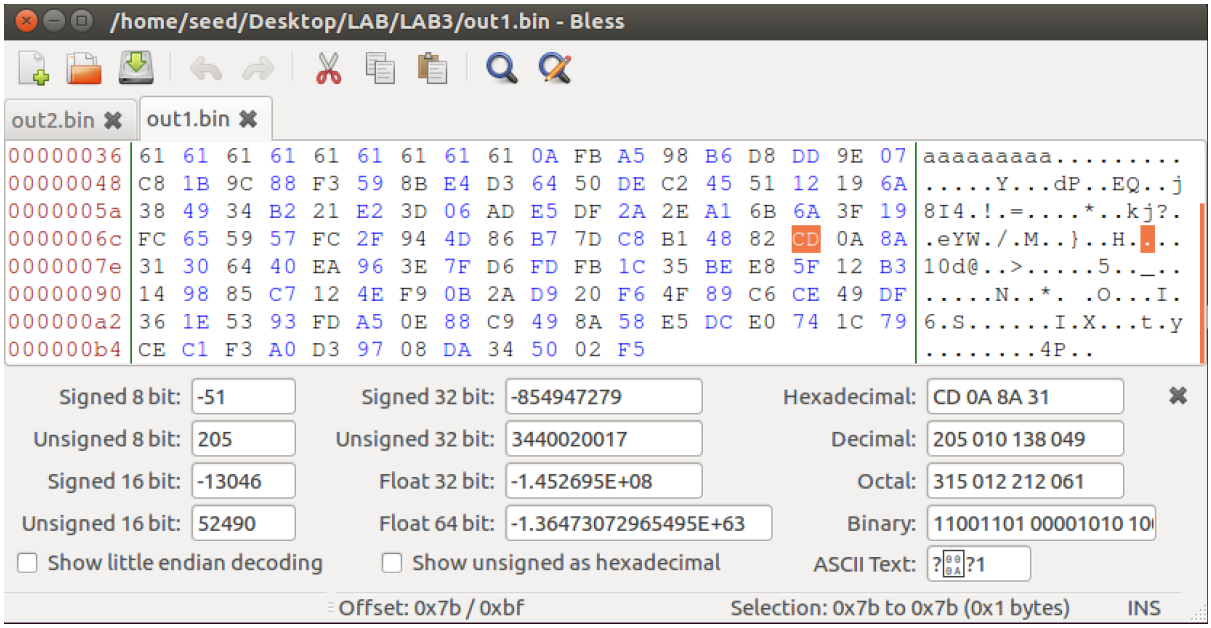

/home/seed/Desktop/LAB/LAB3/out2.bin - Bless
out2.bin
out1.bin
00000036 61 61 61 61 61 61 61 61 61 0A FB A5 98 B6 D8 DD 9E 07 aaaaaaaaa.........
00000048 C8 1B 9C 88 F3 59 8B E4 D3 64 50 5E C2 45 51 12 19 6A .....Y...dP^.EQ..j
0000005a 38 49 34 B2 21 E2 3D 06 AD E5 DF 2A 2E A1 6B 6A 3F 19 8I4.!.=....*..kj?.
0000006c FC E5 59 57 FC 2F 94 4D 86 B7 7D C8 B1 48 82 4D 0A 8A ..YW./.M..}..H.M..
0000007e 31 30 64 40 EA 96 3E 7F D6 FD FB 1C 35 BE E8 5F 12 B3 10d@..>.....5._..
00000090 14 98 85 47 12 4E F9 0B 2A D9 20 F6 4F 89 C6 CE 49 DF ...G.N..*. .O...I.
000000a2 36 1E 53 93 FD A5 0E 88 C9 49 8A D8 E4 DC E0 74 1C 79 6.S......I.....t.y
000000b4 CE C1 F3 A0 D3 97 08 5A 34 50 02 F5 ....Z4P..
Signed 8 bit: 90
Signed 32 bit: 1513377794
Hexadecimal: 5A 34 50 02
Unsigned 8 bit: 90
Unsigned 32 bit: 1513377794
Decimal: 090 052 080 002
Signed 16 bit: 23092
Float 32 bit: 1.268837E+16
Octal: 132 064 120 002
Unsigned 16 bit: 23092
Float 64 bit: —
Binary: 01011010 00110100 01
Show little endian decoding
Show unsigned as hexadecimal
ASCII Text: Z4P
Offset: 0xbb / 0xbf
Selection: 0xbb to 0xbb (0x1 bytes)
INS

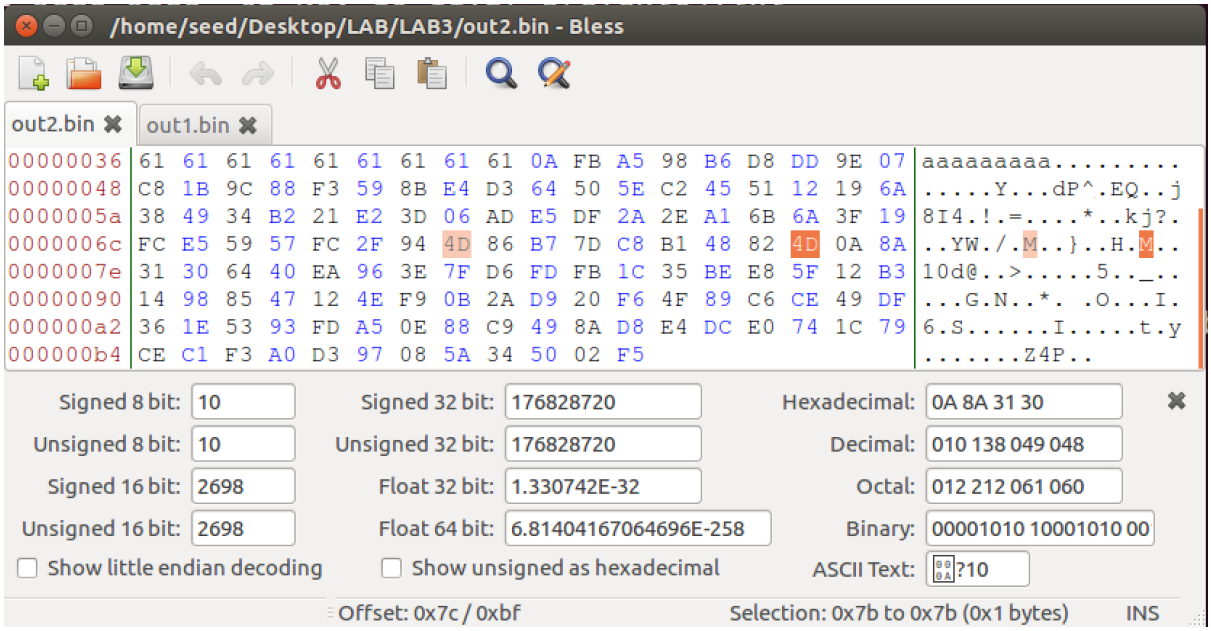

/home/seed/Desktop/LAB/LAB3/out1.bin - Bless
out2.bin
out1.bin
00000036 61 61 61 61 61 61 61 61 61 0A FB A5 98 B6 D8 DD 9E 07 aaaaaaaaa.........
00000048 C8 1B 9C 88 F3 59 8B E4 D3 64 50 DE C2 45 51 12 19 6A .....Y...dP..EQ..j
0000005a 38 49 34 B2 21 E2 3D 06 AD E5 DF 2A 2E A1 6B 6A 3F 19 8I4.!.=....*..kj?.
0000006c FC 65 59 57 FC 2F 94 4D 86 B7 7D C8 B1 48 82 CD 0A 8A .eYW./.M..}..H....
0000007e 31 30 64 40 EA 96 3E 7F D6 FD FB 1C 35 BE E8 5F 12 B3 10d@..>.....5..._..
00000090 14 98 85 C7 12 4E F9 0B 2A D9 20 F6 4F 89 C6 CE 49 DF .....N..*. .O...I.
000000a2 36 1E 53 93 FD A5 0E 88 C9 49 8A 58 E5 DC E0 74 1C 79 6.S......I.X..t.y
000000b4 CE C1 F3 A0 D3 97 08 DA 34 50 02 F5 ....4P..
Signed 8 bit: -36
Signed 32 bit: -589269988
Hexadecimal: DC E0 74 1C
Unsigned 8 bit: 220
Unsigned 32 bit: 3705697308
Decimal: 220 224 116 028
Signed 16 bit: -8992
Float 32 bit: -5.054245E+17
Octal: 334 340 164 034
Unsigned 16 bit: 56544
Float 64 bit: -2.44922044451844E+139
Binary: 11011100 11100000 01
Show little endian decoding
Show unsigned as hexadecimal
ASCII Text: ??t
Offset: 0xaf / 0xbf
Selection: 0xad to 0xae (0x2 bytes)
INS

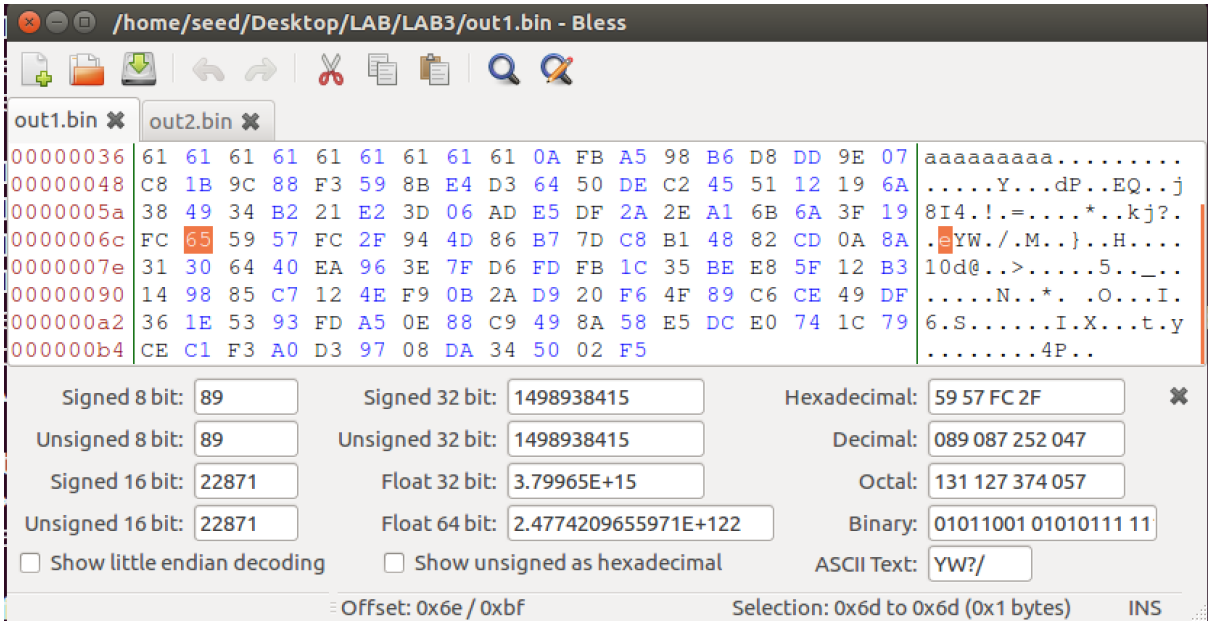

/home/seed/Desktop/LAB/LAB3/out2.bin - Bless
out2.bin
out1.bin
00000036 61 61 61 61 61 61 61 61 61 0A FB A5 98 B6 D8 DD 9E 07 aaaaaaaaa.........
00000048 C8 1B 9C 88 F3 59 8B E4 D3 64 50 5E C2 45 51 12 19 6A .....Y...dP^.EQ..j
0000005a 38 49 34 B2 21 E2 3D 06 AD E5 DF 2A 2E A1 6B 6A 3F 19 8I4.!.=....*..kj?.
0000006c FC E5 59 57 FC 2F 94 4D 86 B7 7D C8 B1 48 82 4D 0A 8A ..YW./.M..}..H.M..
0000007e 31 30 64 40 EA 96 3E 7F D6 FD FB 1C 35 BE E8 5F 12 B3 10d@..>.....5..._..
00000090 14 98 85 47 12 4E F9 0B 2A D9 20 F6 4F 89 C6 CE 49 DF ...G.N..*. .O...I.
000000a2 36 1E 53 93 FD A5 0E 88 C9 49 8A D8 E4 DC E0 74 1C 79 6.S......I.....t.y
000000b4 CE C1 F3 A0 D3 97 08 5A 34 50 02 F5 ....Z4P..
Signed 8 bit: -36
Signed 32 bit: -589269988
Hexadecimal: DC E0 74 1C
Unsigned 8 bit: 220
Unsigned 32 bit: 3705697308
Decimal: 220 224 116 028
Signed 16 bit: -8992
Float 32 bit: -5.054245E+17
Octal: 334 340 164 034
Unsigned 16 bit: 56544
Float 64 bit: -2.44922044451844E+139
Binary: 11011100 11100000 01
Show little endian decoding
Show unsigned as hexadecimal
ASCII Text: ??t
Offset: 0xaf / 0xbf
Selection: 0xad to 0xae (0x2 bytes)
INS

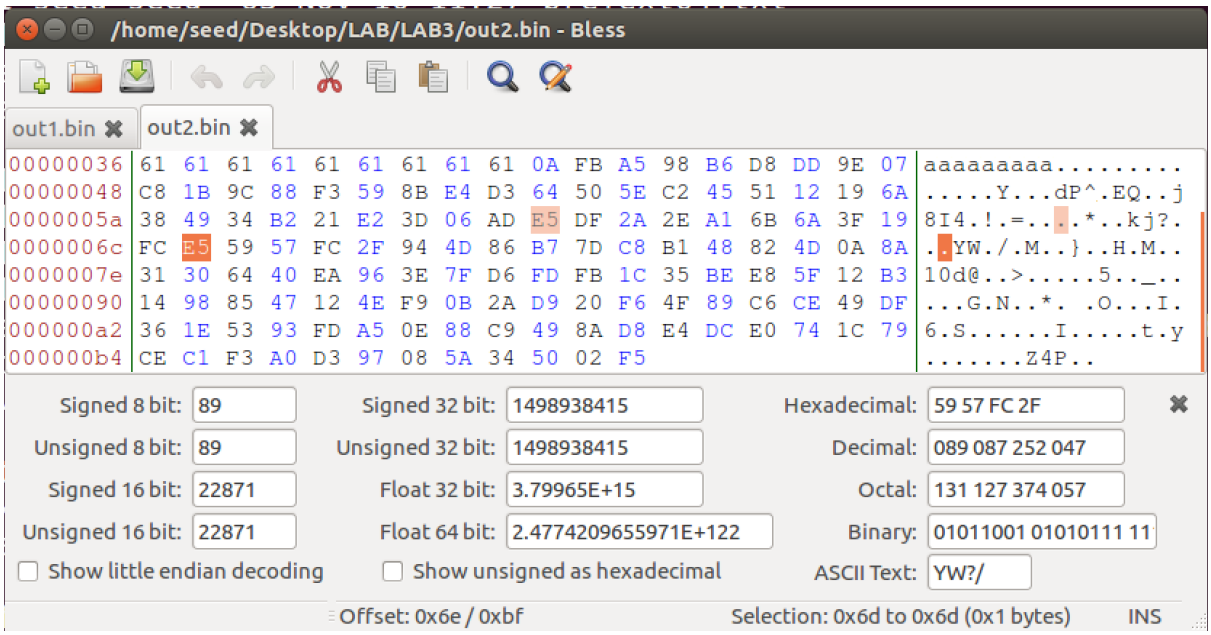

/home/seed/Desktop/LAB/LAB3/out1.bin - Bless
out2.bin
out1.bin
00000036 61 61 61 61 61 61 61 61 61 0A FB A5 98 B6 D8 DD 9E 07 aaaaaaaaa.........
00000048 C8 1B 9C 88 F3 59 8B E4 D3 64 50 DE C2 45 51 12 19 6A .....Y...dP..EQ..j
0000005a 38 49 34 B2 21 E2 3D 06 AD E5 DF 2A 2E A1 6B 6A 3F 19 8I4.!.=....*..kj?.
0000006c FC 65 59 57 FC 2F 94 4D 86 B7 7D C8 B1 48 82 CD 0A 8A .eYW./.M..}..H....
0000007e 31 30 64 40 EA 96 3E 7F D6 FD FB 1C 35 BE E8 5F 12 B3 10d@..>.....5.._..
00000090 14 98 85 C7 12 4E F9 0B 2A D9 20 F6 4F 89 C6 CE 49 DF ...G.N..*. .O...I.
000000a2 36 1E 53 93 FD A5 0E 88 C9 49 8A 58 E5 DC E0 74 1C 79 6.S......I.X...t.y
000000b4 CE C1 F3 A0 D3 97 08 DA 34 50 02 F5 ....4P..
Signed 8 bit: 18
Signed 32 bit: 307165451
Hexadecimal: 12 4E F9 0B
Unsigned 8 bit: 18
Unsigned 32 bit: 307165451
Decimal: 018 078 249 011
Signed 16 bit: 4686
Float 32 bit: 6.530911E-28
Octal: 022 116 371 013
Unsigned 16 bit: 4686
Float 64 bit: 1.71369696995921E-220
Binary: 00010010 01001110 11
Show little endian decoding
Show unsigned as hexadecimal
ASCII Text: N?
Offset: 0x94 / 0xbf
Selection: 0x93 to 0x93 (0x1 bytes)
INS


/home/seed/Desktop/LAB/LAB3/out2.bin - Bless
out2.bin
out1.bin
00000036 61 61 61 61 61 61 61 61 61 0A FB A5 98 B6 D8 DD 9E 07 aaaaaaaaa.........
00000048 C8 1B 9C 88 F3 59 8B E4 D3 64 50 5E C2 45 51 12 19 6A .....Y...dP^.EQ..j
0000005a 38 49 34 B2 21 E2 3D 06 AD E5 DF 2A 2E A1 6B 6A 3F 19 8I4.!.=....*..kj?.
0000006c FC E5 59 57 FC 2F 94 4D 86 B7 7D C8 B1 48 82 4D 0A 8A ..YW./.M..}..H.M..
0000007e 31 30 64 40 EA 96 3E 7F D6 FD FB 1C 35 BE E8 5F 12 B3 10d@..>.....5.._..
00000090 14 98 85 47 12 4E F9 0B 2A D9 20 F6 4F 89 C6 CE 49 DF ...G.N..*. .O...I.
000000a2 36 1E 53 93 FD A5 0E 88 C9 49 8A D8 E4 DC E0 74 1C 79 6.S......I.....t.y
000000b4 CE C1 F3 A0 D3 97 08 5A 34 50 02 F5 ....Z4P..
Signed 8 bit: 71
Signed 32 bit: 1192382201
Hexadecimal: 47 12 4E F9
Unsigned 8 bit: 71
Unsigned 32 bit: 1192382201
Decimal: 071 018 078 249
Signed 16 bit: 18194
Float 32 bit: 37454.97
Octal: 107 022 116 371
Unsigned 16 bit: 18194
Float 64 bit: 2.37657756678283E+34
Binary: 01000111 00010010 01
Show little endian decoding
Show unsigned as hexadecimal
ASCII Text: GN?
Offset: 0x93 / 0xbf
Selection: 0x93 to 0x93 (0x1 bytes)
INS


/home/seed/Desktop/LAB/LAB3/out1.bin - Bless
out2.bin
out1.bin
00000036 61 61 61 61 61 61 61 61 61 0A FB A5 98 B6 D8 DD 9E 07 aaaaaaaaa.........
00000048 C8 1B 9C 88 F3 59 8B E4 D3 64 50 DE C2 45 51 12 19 6A .....Y...dP..EQ..j
0000005a 38 49 34 B2 21 E2 3D 06 AD E5 DF 2A 2E A1 6B 6A 3F 19 8I4.!.=....*..kj?.
0000006c FC 65 59 57 FC 2F 94 4D 86 B7 7D C8 B1 48 82 CD 0A 8A .eYW./.M..}..H..
0000007e 31 30 64 40 EA 96 3E 7F D6 FD FB 1C 35 BE E8 5F 12 B3 10d@..>.....5.._..
00000090 14 98 85 C7 12 4E F9 0B 2A D9 20 F6 4F 89 C6 CE 49 DF .....N..*. .O...I.
000000a2 36 1E 53 93 FD A5 0E 88 C9 49 8A 58 E5 DC E0 74 1C 79 6.S......I.X...t.y
000000b4 CE C1 F3 A0 D3 97 08 DA 34 50 02 F5 ....4P..
Signed 8 bit: -51
Signed 32 bit: -854947279
Hexadecimal: CD 0A 8A 31
Unsigned 8 bit: 205
Unsigned 32 bit: 3440020017
Decimal: 205 010 138 049
Signed 16 bit: -13046
Float 32 bit: -1.452695E+08
Octal: 315 012 212 061
Unsigned 16 bit: 52490
Float 64 bit: -1.36473072965495E+63
Binary: 11001101 00001010 10
Show little endian decoding
Show unsigned as hexadecimal
ASCII Text: ??1
Offset: 0x7b / 0xbf
Selection: 0x7b to 0x7b (0x1 bytes)
INS


/home/seed/Desktop/LAB/LAB3/out2.bin - Bless
out2.bin
out1.bin
00000036 61 61 61 61 61 61 61 61 61 0A FB A5 98 B6 D8 DD 9E 07 aaaaaaaaa.........
00000048 C8 1B 9C 88 F3 59 8B E4 D3 64 50 5E C2 45 51 12 19 6A .....Y...dP^.EQ..j
0000005a 38 49 34 B2 21 E2 3D 06 AD E5 DF 2A 2E A1 6B 6A 3F 19 8I4.!.=....*..kj?.
0000006c FC E5 59 57 FC 2F 94 4D 86 B7 7D C8 B1 48 82 4D 0A 8A ..YW./.M..}..H.M..
0000007e 31 30 64 40 EA 96 3E 7F D6 FD FB 1C 35 BE E8 5F 12 B3 10d@..>.....5.._..
00000090 14 98 85 47 12 4E F9 0B 2A D9 20 F6 4F 89 C6 CE 49 DF ...G.N..*. .O...I.
000000a2 36 1E 53 93 FD A5 0E 88 C9 49 8A D8 E4 DC E0 74 1C 79 6.S......I.....t.y
000000b4 CE C1 F3 A0 D3 97 08 5A 34 50 02 F5 ....Z4P..
Signed 8 bit: 10
Signed 32 bit: 176828720
Hexadecimal: 0A 8A 31 30
Unsigned 8 bit: 10
Unsigned 32 bit: 176828720
Decimal: 010 138 049 048
Signed 16 bit: 2698
Float 32 bit: 1.330742E-32
Octal: 012 212 061 060
Unsigned 16 bit: 2698
Float 64 bit: 6.81404167064696E-258
Binary: 00001010 10001010 00
Show little endian decoding
Show unsigned as hexadecimal
ASCII Text: ?10
Offset: 0x7c / 0xbf
Selection: 0x7b to 0x7b (0x1 bytes)
INS


/home/seed/Desktop/LAB/LAB3/out1.bin - Bless
out1.bin
out2.bin
00000036 61 61 61 61 61 61 61 61 61 0A FB A5 98 B6 D8 DD 9E 07 aaaaaaaaa.........
00000048 C8 1B 9C 88 F3 59 8B E4 D3 64 50 DE C2 45 51 12 19 6A .....Y...dP..EQ..j
0000005a 38 49 34 B2 21 E2 3D 06 AD E5 DF 2A 2E A1 6B 6A 3F 19 8I4.!.=....*..kj?.
0000006c FC 65 59 57 FC 2F 94 4D 86 B7 7D C8 B1 48 82 CD 0A 8A .eYW./.M..}..H....
0000007e 31 30 64 40 EA 96 3E 7F D6 FD FB 1C 35 BE E8 5F 12 B3 10d@..>.....5.._..
00000090 14 98 85 C7 12 4E F9 0B 2A D9 20 F6 4F 89 C6 CE 49 DF .....N..*. .O...I.
000000a2 36 1E 53 93 FD A5 0E 88 C9 49 8A 58 E5 DC E0 74 1C 79 6.S......I.X...t.y
000000b4 CE C1 F3 A0 D3 97 08 DA 34 50 02 F5 ....4P..
Signed 8 bit: 89
Signed 32 bit: 1498938415
Hexadecimal: 59 57 FC 2F
Unsigned 8 bit: 89
Unsigned 32 bit: 1498938415
Decimal: 089 087 252 047
Signed 16 bit: 22871
Float 32 bit: 3.79965E+15
Octal: 131 127 374 057
Unsigned 16 bit: 22871
Float 64 bit: 2.4774209655971E+122
Binary: 01011001 01010111 11
Show little endian decoding
Show unsigned as hexadecimal
ASCII Text: YW?/
Offset: 0x6e / 0xbf
Selection: 0x6d to 0x6d (0x1 bytes)
INS

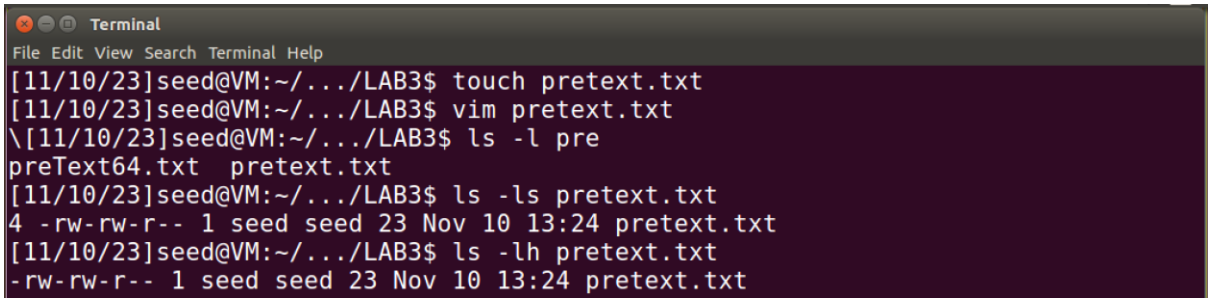

/home/seed/Desktop/LAB/LAB3/out2.bin - Bless
out1.bin
out2.bin
00000036 61 61 61 61 61 61 61 61 61 0A FB A5 98 B6 D8 DD 9E 07 aaaaaaaaa.........
00000048 C8 1B 9C 88 F3 59 8B E4 D3 64 50 5E C2 45 51 12 19 6A .....Y...dP^.EQ..j
0000005a 38 49 34 B2 21 E2 3D 06 AD E5 DF 2A 2E A1 6B 6A 3F 19 8I4.!.=....*..kj?.
0000006c FC E5 59 57 FC 2F 94 4D 86 B7 7D C8 B1 48 82 4D 0A 8A .YW./.M..}..H.M..
0000007e 31 30 64 40 EA 96 3E 7F D6 FD FB 1C 35 BE E8 5F 12 B3 10d@..>.....5.._..
00000090 14 98 85 47 12 4E F9 0B 2A D9 20 F6 4F 89 C6 CE 49 DF ...G.N..*. .O...I.
000000a2 36 1E 53 93 FD A5 0E 88 C9 49 8A D8 E4 DC E0 74 1C 79 6.S......I.....t.y
000000b4 CE C1 F3 A0 D3 97 08 5A 34 50 02 F5 ....Z4P..
Signed 8 bit: 89
Signed 32 bit: 1498938415
Hexadecimal: 59 57 FC 2F
Unsigned 8 bit: 89
Unsigned 32 bit: 1498938415
Decimal: 089 087 252 047
Signed 16 bit: 22871
Float 32 bit: 3.79965E+15
Octal: 131 127 374 057
Unsigned 16 bit: 22871
Float 64 bit: 2.4774209655971E+122
Binary: 01011001 01010111 11
Show little endian decoding
Show unsigned as hexadecimal
ASCII Text: YW?/
Offset: 0x6e / 0xbf
Selection: 0x6d to 0x6d (0x1 bytes)
INS

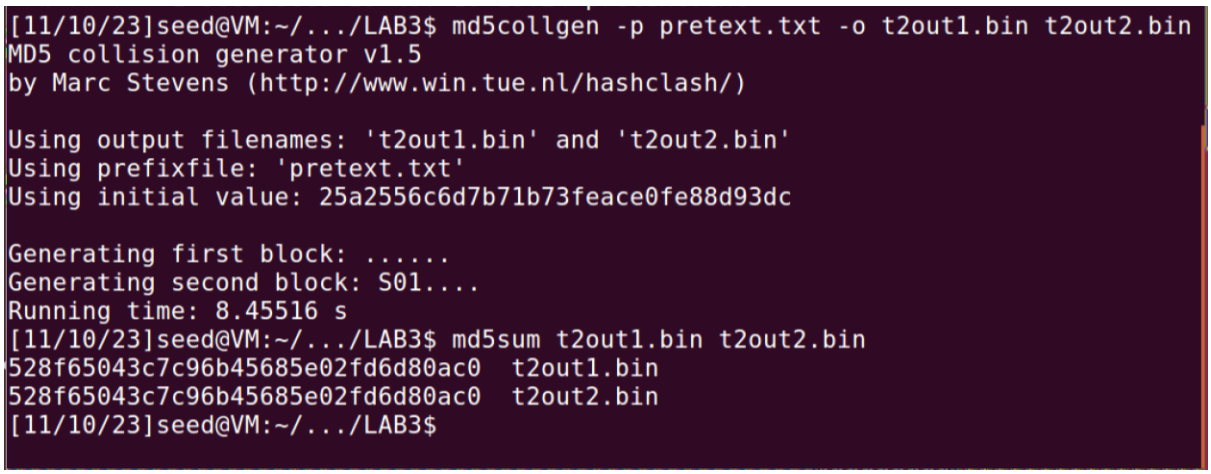
Task II: Understanding MD5 Properties

> Given two inputs M and N , if MD5(M) = MD5(N) , i.e., the MD5 hashes of M and N are the same, then for any input T , MD5(M || T) = MD5(N || T) , where || represents concatenation.
> That is, if inputs M and N have the same hash, adding the same suffix T to them will result in two outputs that have the same hash value.

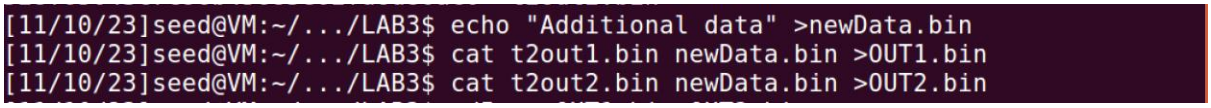
To demonstrate the above property, I performed this experiment –

Creating a new pretext file “pretext.txt” then creating two md5 file with same hash values. Then using **md5sum** command to check the hash values of the two output files.

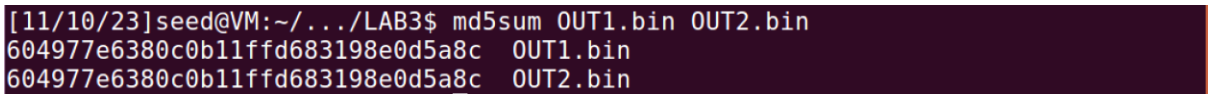




Now I’ll create a new .bin file, namely “newData.bin” with help of **echo** command and concatenate this new file to previous output files using **cat** command.



Now I’ll check the hash values of the modified files

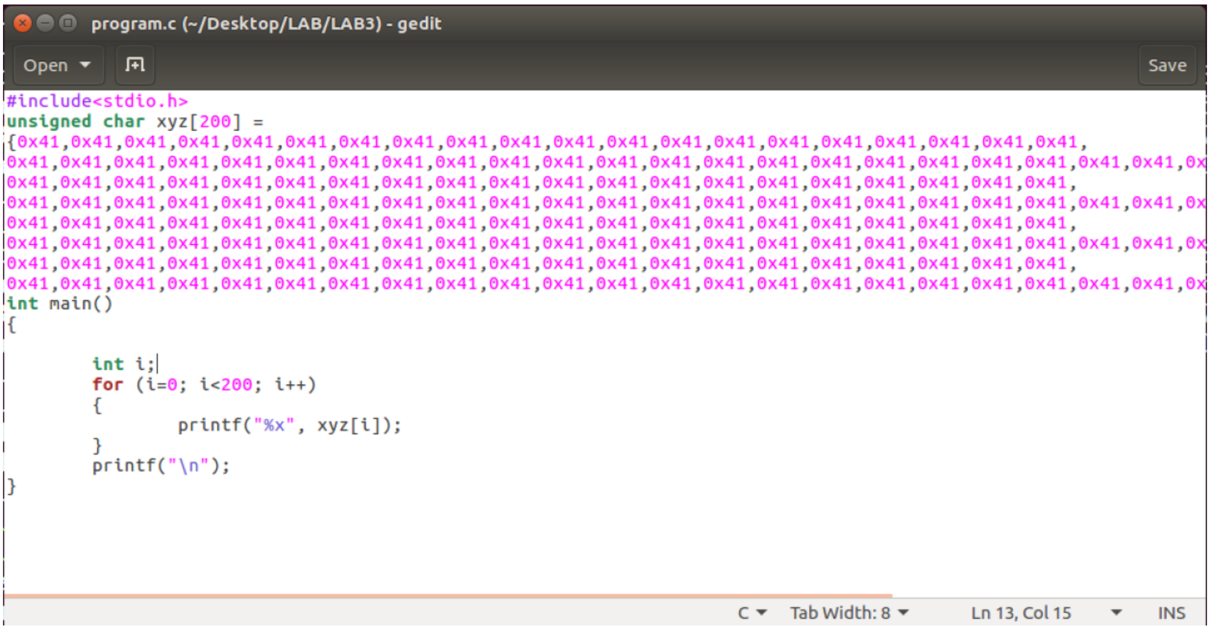


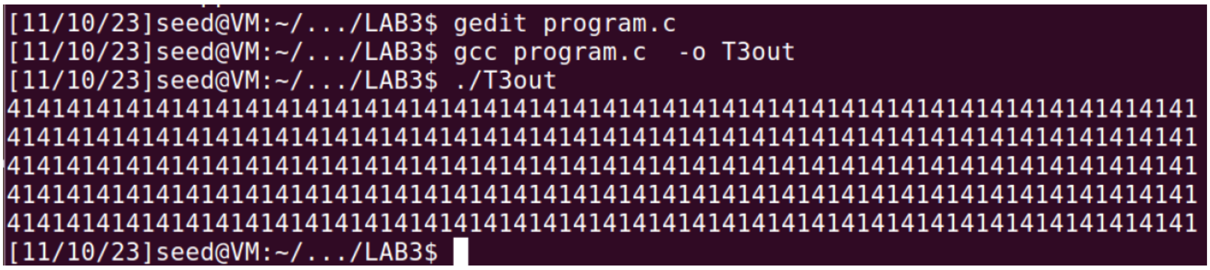
I concur that the there was a slight change in hash value due to the modification but these changes are consistent and can be seen in both the files. In the end the hash vales between two files end up being same.

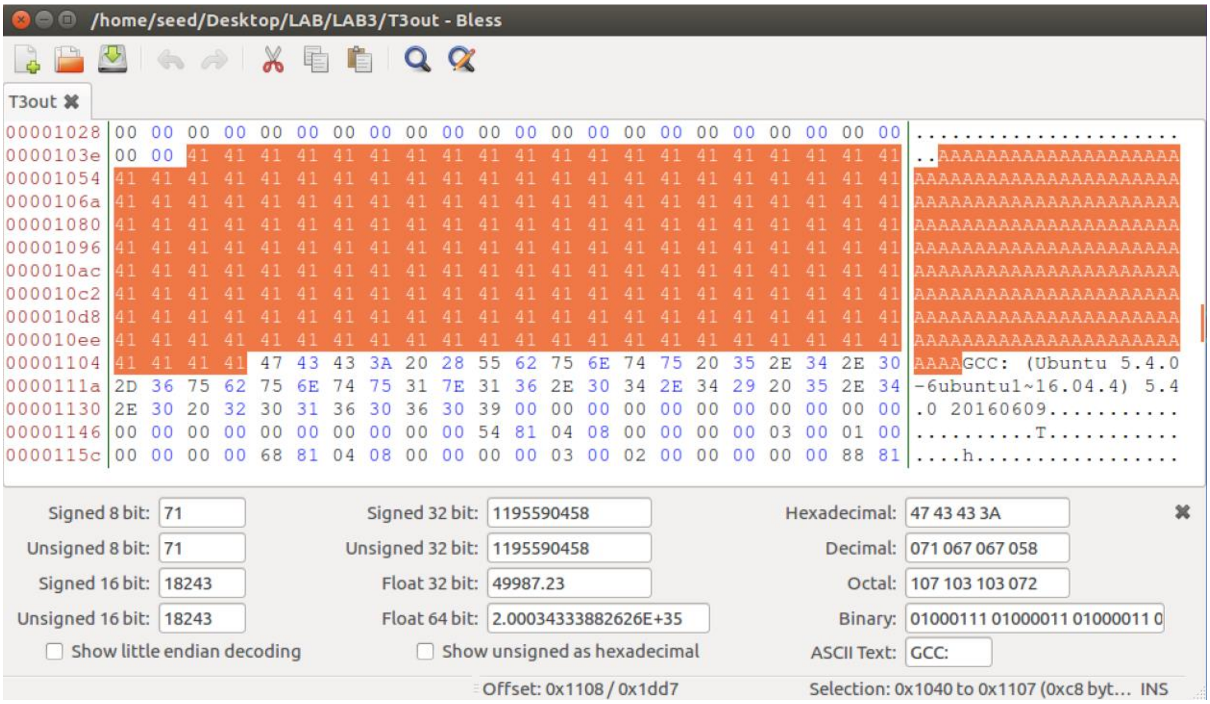
I have successfully demonstrated said property of MD5.

Task III: Generating Two executable Files with the Same MD5 Hash

Here's the code which I prepared







The A,A,A.. starts from 1040 location I the file, which equates to 4160th byte.

All the bytes before byte will be in prefix text

For suffix, so it will contain contents after 4160 + 128 = 4288 byte till end.

```
[11/10/23]seed@VM:~/.../LAB3$ head -c 4160 T3out > T3prefix
```

```
[11/11/23]seed@VM:~/.../LAB3$ tail -c +4289 T3out > T3Suffix
```

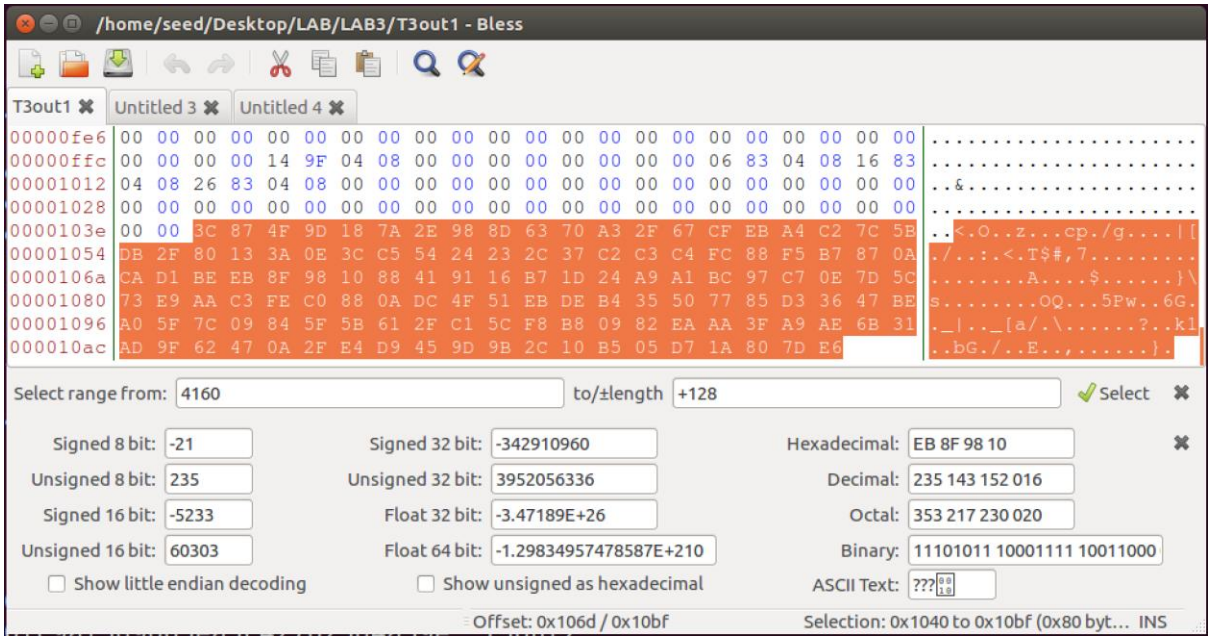
Now I'll apply md5collgen on this T3prefix file to get two files with same hash values

```
[11/11/23]seed@VM:~/.../LAB3$ md5collgen -p T3prefix -o T3out1 T3out2
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'T3out1' and 'T3out2'
Using prefixfile: 'T3prefix'
Using initial value: af0310fc20c32a33c9710c228be83872

Generating first block: ....................................
Generating second block: S11.....
Running time: 16.2968 s
```

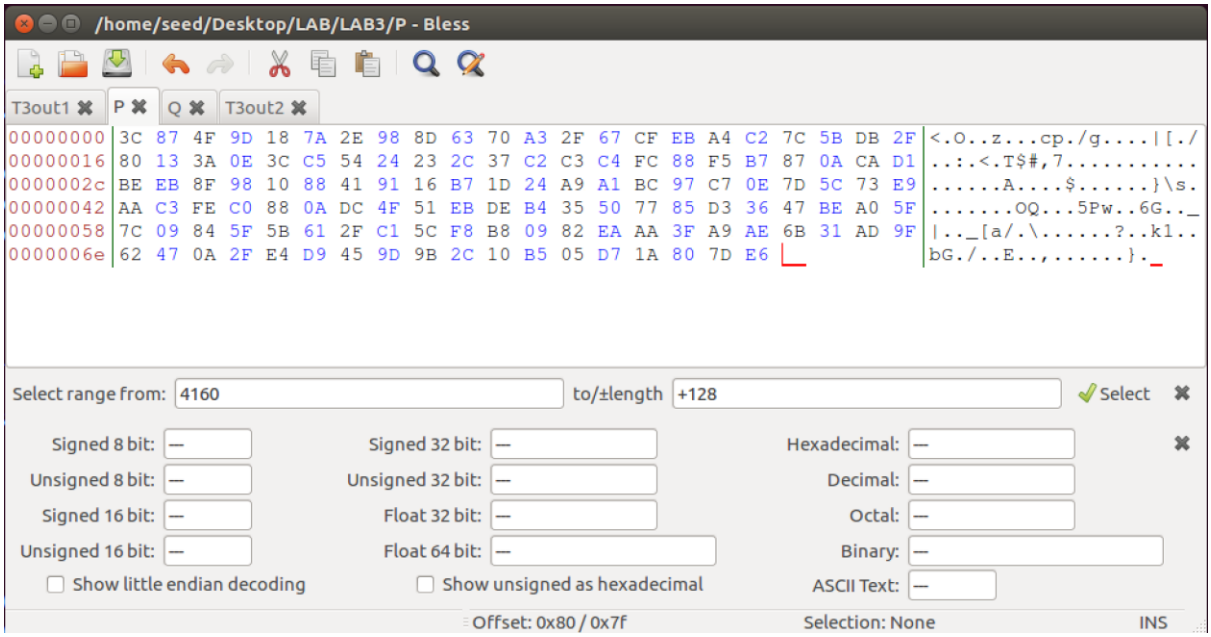
```
[11/11/23]seed@VM:~/.../LAB3$ md5sum T3out1 T3out2
cf2c0fc9af5d9665ea5c427625d4a59e  T3out1
cf2c0fc9af5d9665ea5c427625d4a59e  T3out2
```

After getting these 2 outputs, the 128 bytes after the prefix's length will be our P and Q for both outputs.

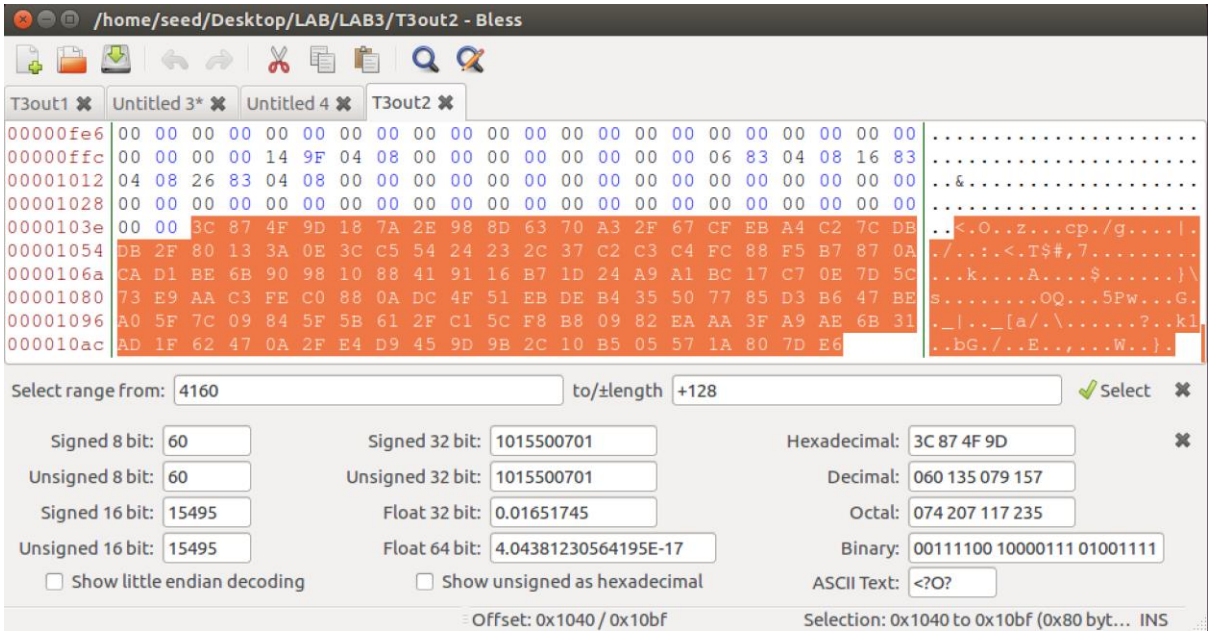
Making P file:



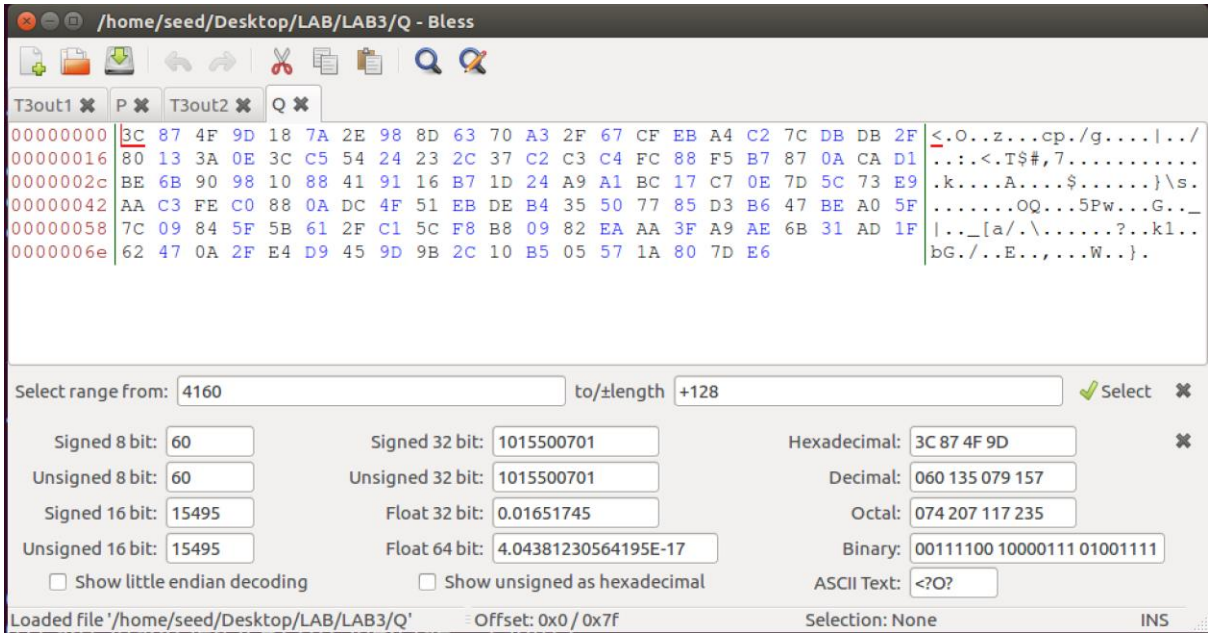
The above bytes are copies from T3out1 and pasted in P



Making Q file:



The above bytes are copies from T3out2 and pasted in Q



Now I'll concatenate crate 2 file such as File1 :prefix||P||suffix

and File2: prefix||Q||suffix

```
[11/11/23]seed@VM:~/.../LAB3$ cat T3prefix P > temp1
[11/11/23]seed@VM:~/.../LAB3$ cat temp1 T3Suffix > File1
[11/11/23]seed@VM:~/.../LAB3$ cat T3prefix Q > temp2
[11/11/23]seed@VM:~/.../LAB3$ cat temp2 T3Suffix > File2
```

```
[11/11/23]seed@VM:~/.../LAB3$ ls
File1         out2.bin     prefix         t2out1.bin  T3out2      T3suffix
File2         OUT2.bin     preText64.txt  t2out2.bin  T3out2.bin  T3Suffix
newData.bin   output1.bin  pretext.txt    T3out       T3P         temp1
out1.bin      output2.bin  program.c      T3out1      T3prefix    temp2
OUT1.bin      P            Q              T3out1.bin  T3Q         Text.txt
```

Now I can see these files are not executable, I'll make these executable by using **chmod +x** command.

```
[11/11/23]seed@VM:~/.../LAB3$ chmod +x File1
[11/11/23]seed@VM:~/.../LAB3$ chmod +x File2
```

```
[11/11/23]seed@VM:~/.../LAB3$ ls
File1         out2.bin     prefix         t2out1.bin  T3out2      T3suffix
File2         OUT2.bin     preText64.txt  t2out2.bin  T3out2.bin  T3Suffix
newData.bin   output1.bin  pretext.txt    T3out       T3P         temp1
out1.bin      output2.bin  program.c      T3out1      T3prefix    temp2
OUT1.bin      P            Q              T3out1.bin  T3Q         Text.txt
```

I'll execute these file and comapre there harsh values

```
[11/11/23]seed@VM:~/.../LAB3$ File1
3c874f9d187a2e988d6370a32f67cfeba4c27c5bdb2f80133ae3cc55424232c37c2c3c4fc88f5b78
7acad1beeb8f981088419116b71d24a9a1bc97c7e7d5c73e9aac3fec088adc4f51ebdeb435507785
d33647bea05f7c9845f5b612fc15cf8b8982eaaa3fa9ae6b31ad9f6247a2fe4d9459d9b2c10b55d7
1a807de641414141414141414141414141414141414141414141414141414141414141414141
414141414141414141414141414141414141414141414141414141414141414141414141414141
[11/11/23]seed@VM:~/.../LAB3$ File2
3c874f9d187a2e988d6370a32f67cfeba4c27cdbdb2f80133ae3cc55424232c37c2c3c4fc88f5b78
7acad1be6b90981088419116b71d24a9a1bc17c7e7d5c73e9aac3fec088adc4f51ebdeb435507785
d3b647bea05f7c9845f5b612fc15cf8b8982eaaa3fa9ae6b31ad1f6247a2fe4d9459d9b2c10b5557
1a807de641414141414141414141414141414141414141414141414141414141414141414141
414141414141414141414141414141414141414141414141414141414141414141414141414141
[11/11/23]seed@VM:~/.../LAB3$ md5sum File1 File2
8d5084c82a9d18351c5b6d11fb9ad3ef  File1
8d5084c82a9d18351c5b6d11fb9ad3ef  File2
```

I can see both the file shave same hash values but different data values.

P ✖ Q ✖

```
00000000  3C 87 4F 9D 18 7A 2E 98  8D 63 70 A3 2F 67 CF EB  A4 C2 7C 5B DB 2F  |<.O..z...cp./g....|[./|
00000016  80 13 3A 0E 3C C5 54 24  23 2C 37 C2 C3 C4 FC 88  F5 B7 87 0A CA D1  |..:.<.T$#,7.........|
0000002c  BE EB 8F 98 10 88 41 91  16 B7 1D 24 A9 A1 BC 97  C7 0E 7D 5C 73 E9  |......A....$......}\s.|
00000042  AA C3 FE C0 88 0A DC 4F  51 EB DE B4 35 50 77 85  D3 36 47 BE A0 5F  |.......OQ...5Pw..6G.._|
00000058  7C 09 84 5F 5B 61 2F C1  5C F8 B8 09 82 EA AA 3F  A9 AE 6B 31 AD 9F  |..._[a/.\......?..k1..|
0000006e  62 47 0A 2F E4 D9 45 9D  9B 2C 10 B5 05 D7 1A 80  7D E6              |bG./..E..,......}.|
```

P ✖ Q ✖

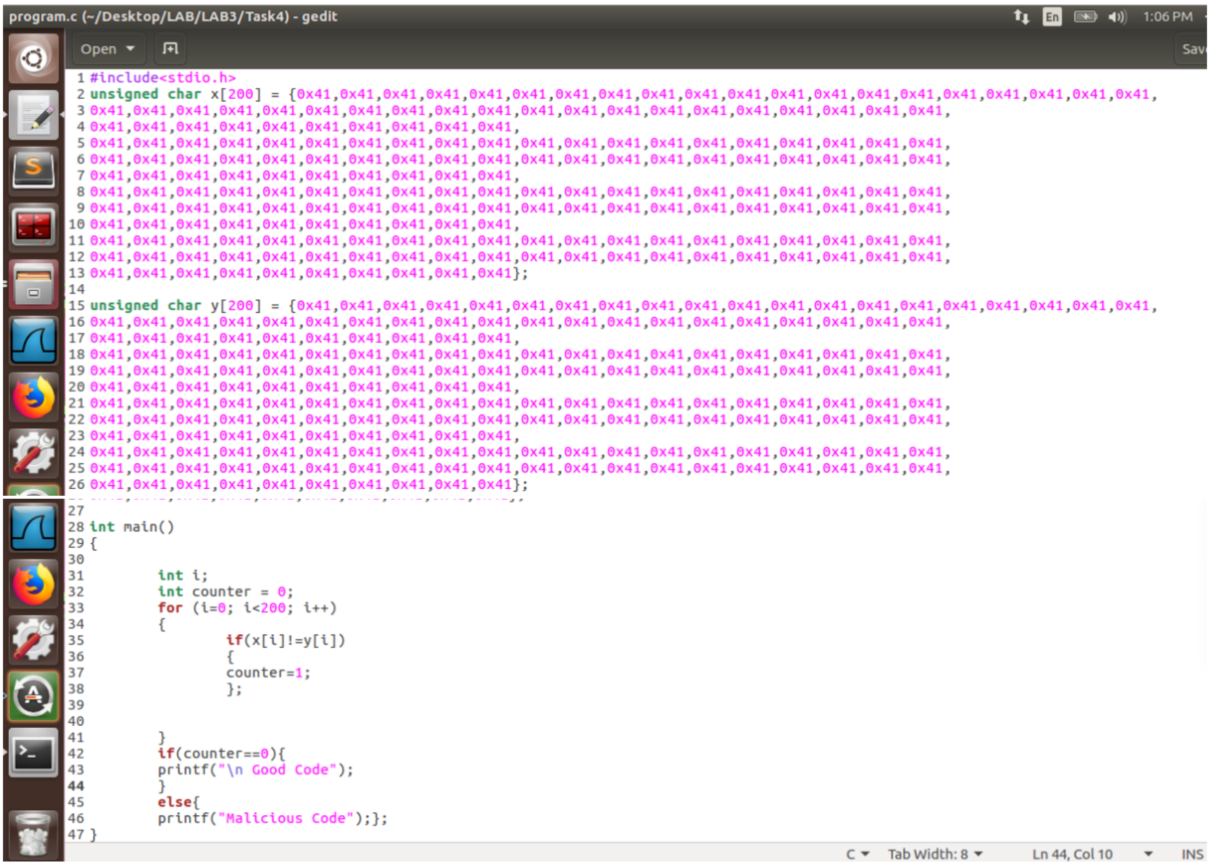
```
00000000  3C 87 4F 9D 18 7A 2E 98  8D 63 70 A3 2F 67 CF EB  A4 C2 7C DB DB 2F  |<.O..z...cp./g....|../|
00000016  80 13 3A 0E 3C C5 54 24  23 2C 37 C2 C3 C4 FC 88  F5 B7 87 0A CA D1  |..:.<.T$#,7.........|
0000002c  BE 6B 90 98 10 88 41 91  16 B7 1D 24 A9 A1 BC 17  C7 0E 7D 5C 73 E9  |.k....A....$......}\s.|
00000042  AA C3 FE C0 88 0A DC 4F  51 EB DE B4 35 50 77 85  D3 B6 47 BE A0 5F  |.......OQ...5Pw...G.._|
00000058  7C 09 84 5F 5B 61 2F C1  5C F8 B8 09 82 EA AA 3F  A9 AE 6B 31 AD 1F  |..._[a/.\......?..k1..|
0000006e  62 47 0A 2F E4 D9 45 9D  9B 2C 10 B5 05 57 1A 80  7D E6              |bG./..E..,...W..}.|
```

P and Q are derived from T3out1 and T3out2 respectively, which were obtained after performing md5collgen on T3prefix file. Even though T3out1 and T3out2 had same hash value but difference in their data sections. This was also observed in the first task and is consistent to MD5 properties.

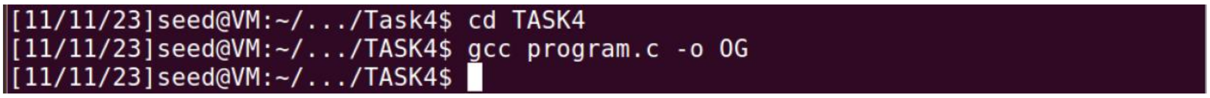
I was successfully able to create two executable files with different data and having the same hash values.

Task IV: Making the Two Programs Behave Differently

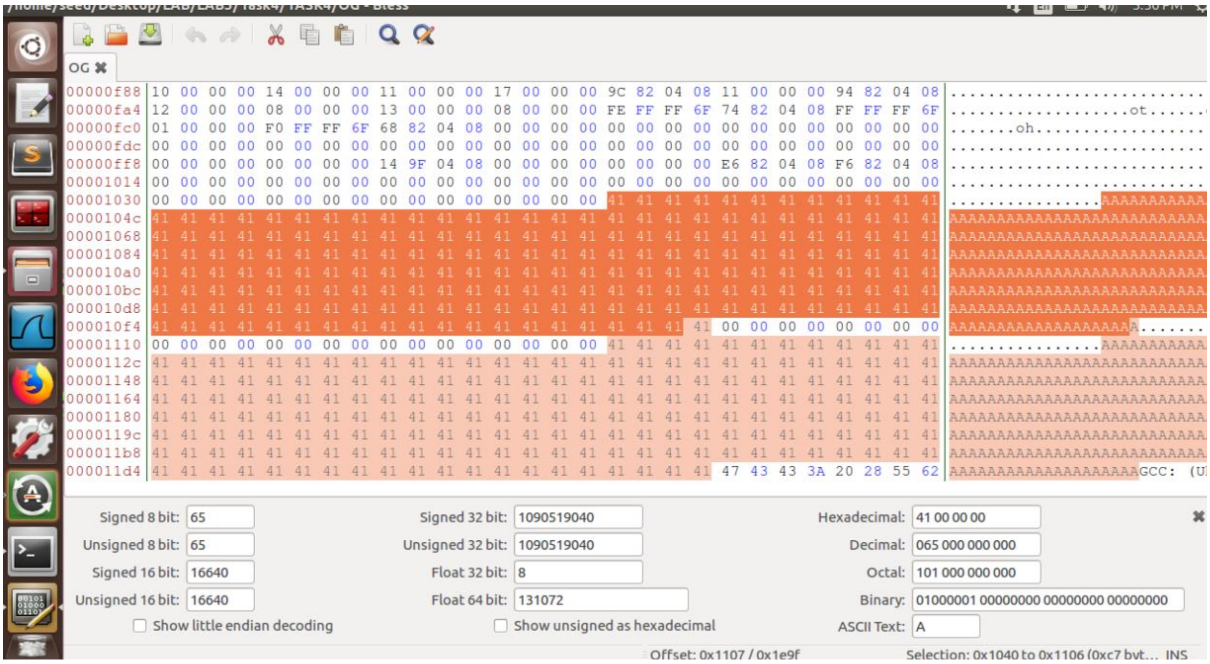
Making the code :



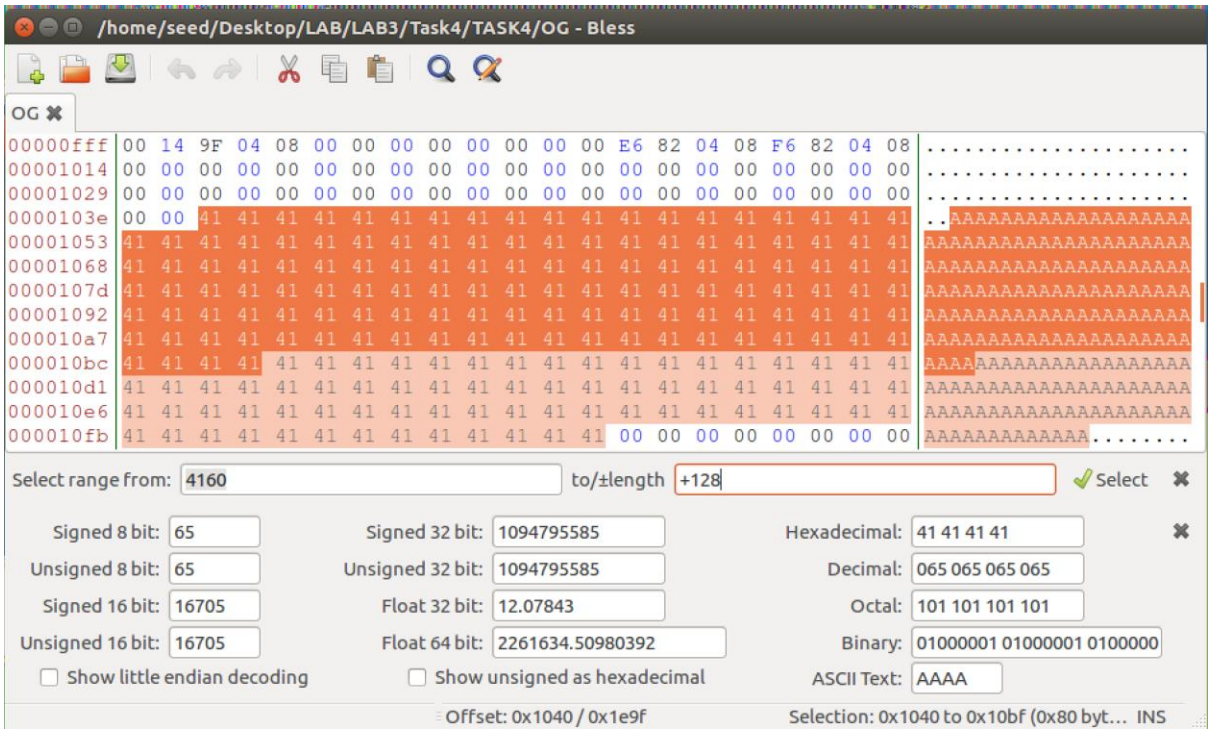
Initial output after running the code and saving it as an executable file



Now viewing this OG file on hex



Now I find the location from where the Array X starts



So all the bytes before the highlighted region(Array X) will be in our prefix.

```
[11/11/23]seed@VM:~/.../TASK4$ head -c 4160 OG > prefix
```

And this highlighted region will the 128-byte part. And the remaining part will be S

```
[11/11/23]seed@VM:~/.../TASK4$ tail -c +4289 OG > S
```

Performing MD5 hash on prefix

```
[11/11/23]seed@VM:~/.../TASK4$ md5collgen -p prefix -o p1 p2
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'p1' and 'p2'
Using prefixfile: 'prefix'
Using initial value: bbed8ff794b0428e480ea2cbc97b3762

Generating first block: ......
Generating second block: S01....
Running time: 3.57935 s
```

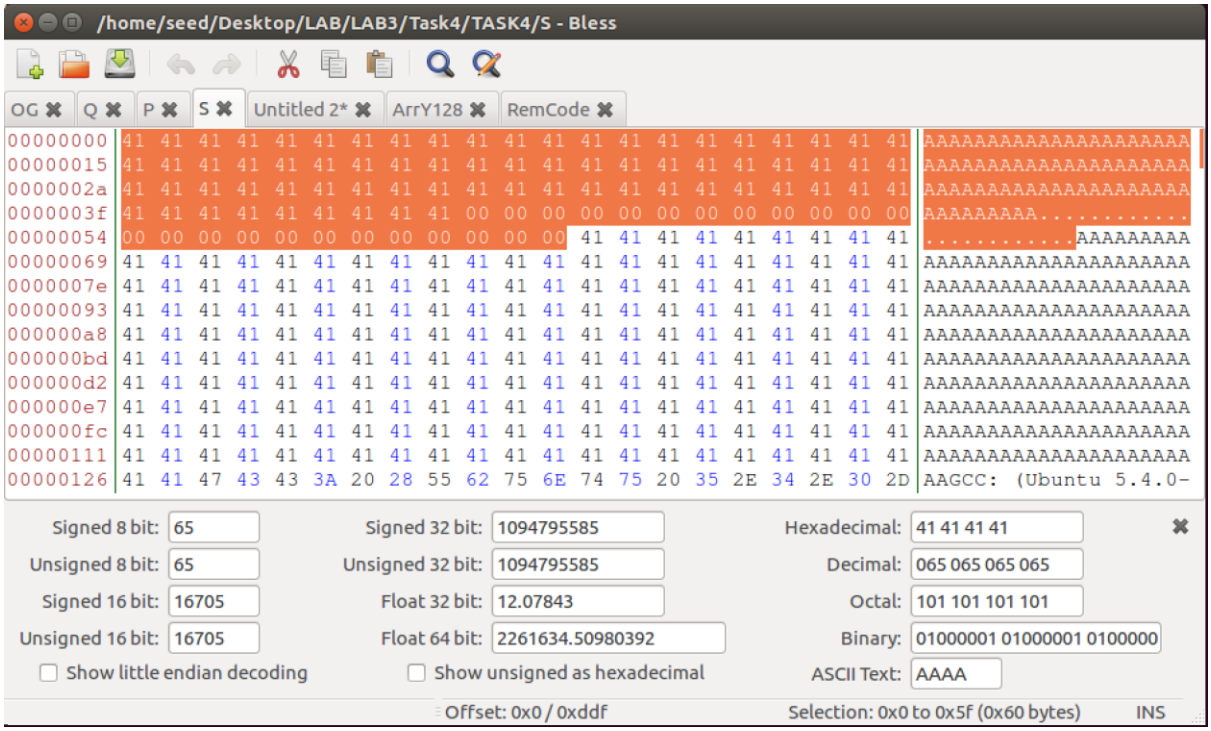
Now I save the last 128 byte from P1 and P2 as P and Q.

These 128 bytes consist of first 128 byte of array X.

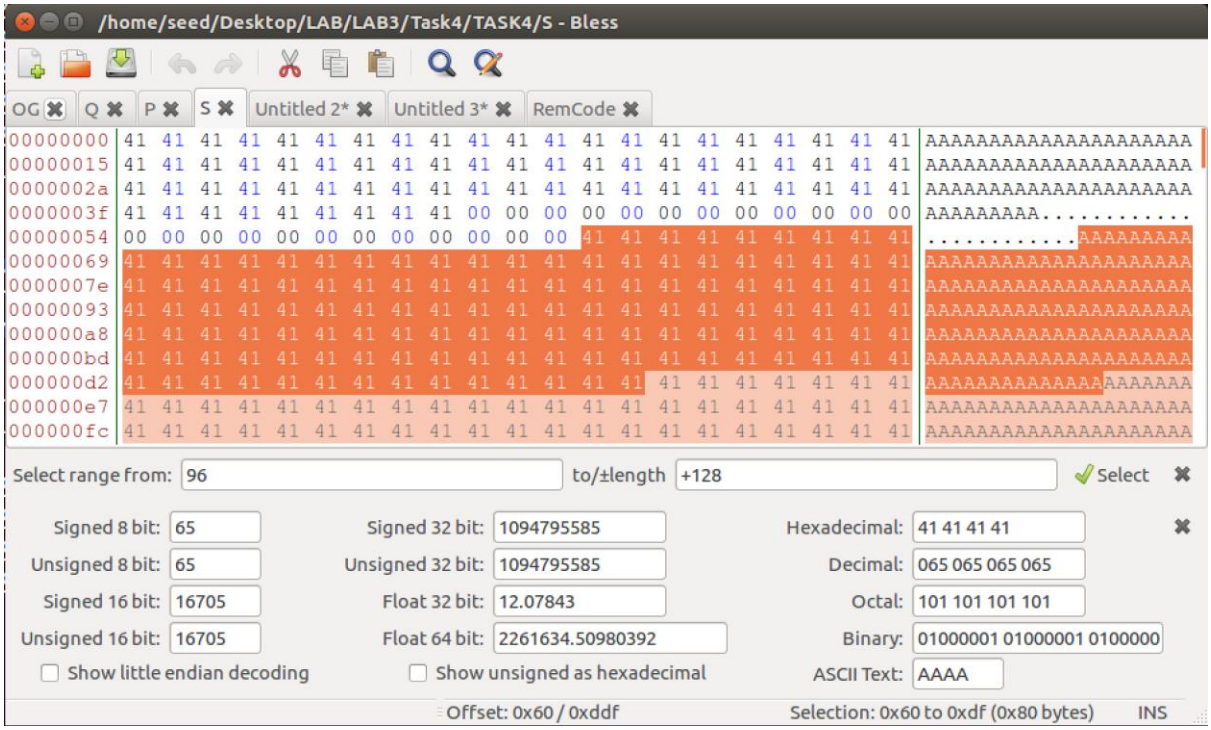
Now I'll divide the suffix into the following parts

Remaining arrX | first 128 bytes of arry | reaming code

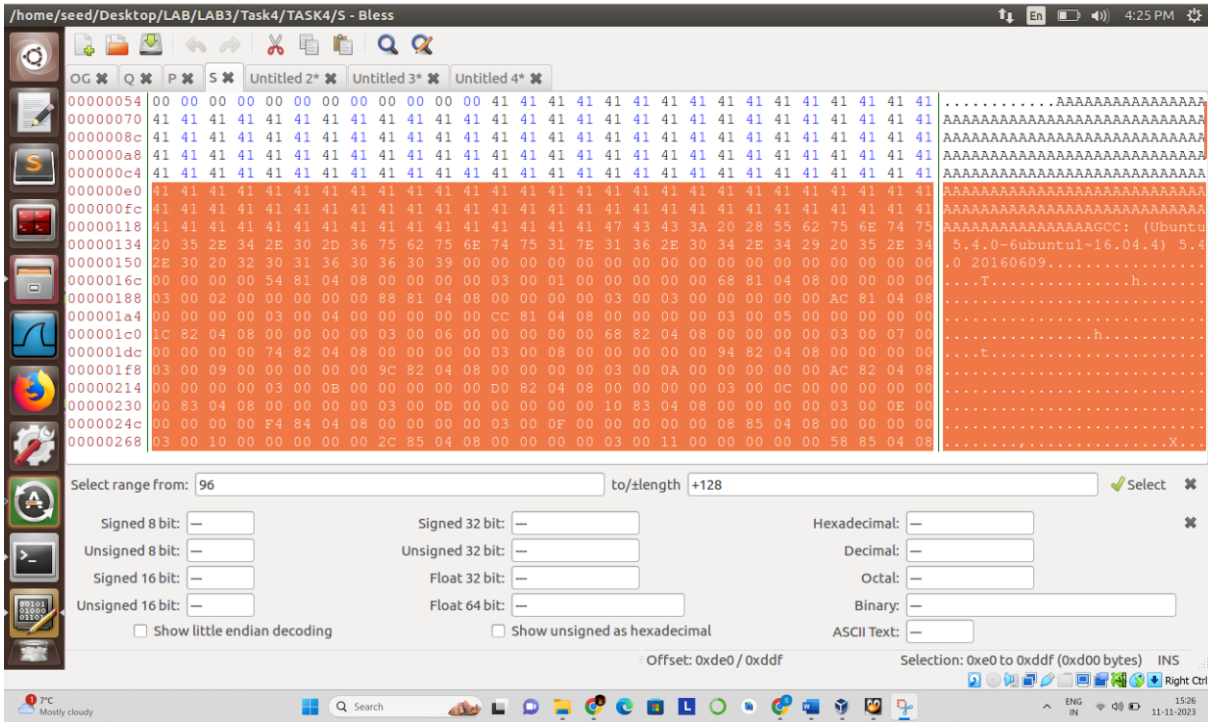
Remaining part of arr X

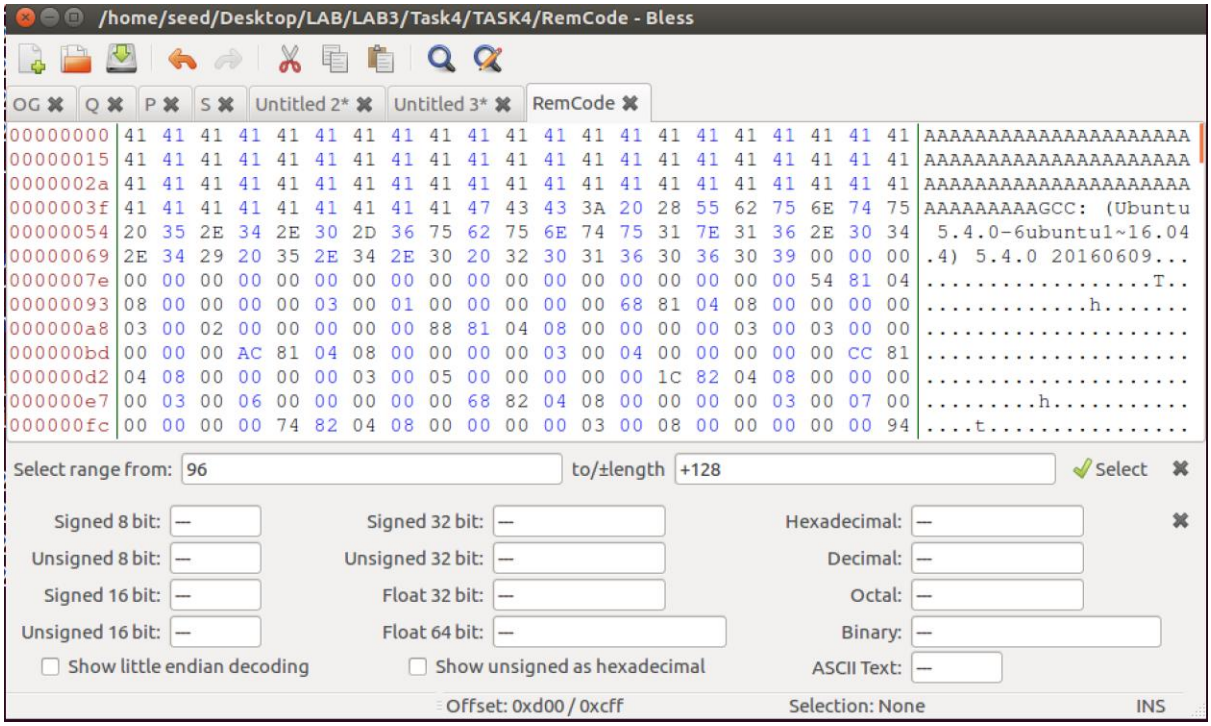


First 128 bytes of array Y



Remaining code part:





Now the Good code will consist of:

Prefix || P || Arr x Remaining || P || Remaining Code

Bad Code:

Prefix || Q || Arr x Remaining || P || Remaining Code

After creating the Good Code and Bad Code file, I'll make them executable

After doing so, I'll execute them and check their hash values

```
[11/11/23]seed@VM:~/.../TASK4$ cat prefix P ArrXrem P RemCode > GoodCode
[11/11/23]seed@VM:~/.../TASK4$ cat prefix Q ArrXrem P RemCode > BadCode
[11/11/23]seed@VM:~/.../TASK4$ chmod +x GoodCode BadCode
[11/11/23]seed@VM:~/.../TASK4$ ./GoodCode

 Good Code[11/11/23]seed@VM:~/.../TASK4$ ./BadCode
Malicious Code[11/11/23]seed@VM:~/.../TASK4$ md5sum GoodCode BadCode
76b815a29ba996295365bf59a03d9cd1  GoodCode
76b815a29ba996295365bf59a03d9cd1  BadCode
[11/11/23]seed@VM:~/.../TASK4$
```

As I can see, Both Good and Bad codes have the same hash values but execute different codes.

I was successfully able to create two programs with same hash values while they perform different functions. The good code performs the intended/ benign code where as the Bad code performs malicious code.

Conclusion:

> *After performing the various experiments on MD5 hashing function, I can concur that MD5 hash is not effective at handling collisions. Because of this MD5 is considered insecure for cryptographic purposes. Even though MD5 is a fast-hashing function but due to its vulnerability to attacks it is no longer acceptable for digital certificates.*
> *SHA-256 and SHA3 are considered to be its modern counterparts as they are more secure*