CS 201P Project #9— MD5 Collision Attack Lab

Name: Shikhir Goel

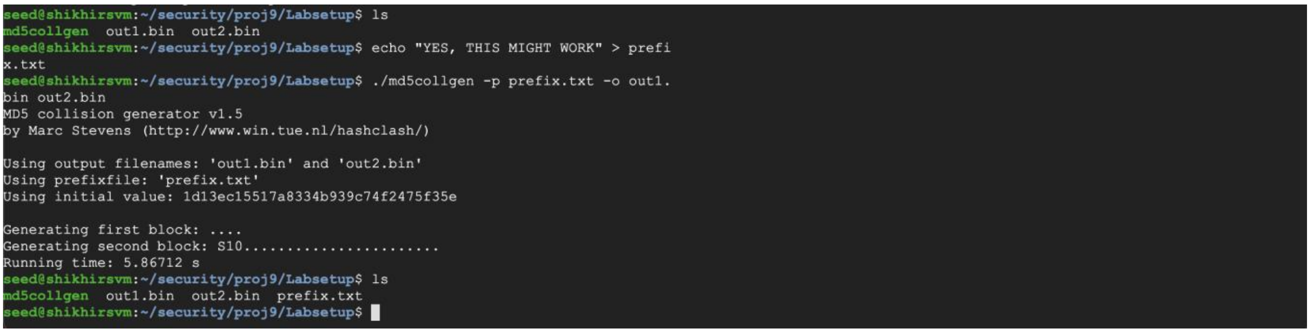
UCI ID: shikhirg@uci.edu

Computing/Cloud Platform Chosen: Google Cloud platform

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

Here we are trying to generate two different files with same prefix and same Hash Value. To do this, we create two different files with same beginning part or prefix. Then we use the md5collgen tool which allows us to create an arbitrary file, the contents of which will be used as prefix to generate two files out1.bin and out2.bin which will have the same MD5 hash.

Using md5collgen to produce two different output files out1.bin, out2.bin having same prefix stored in prefix.txt.

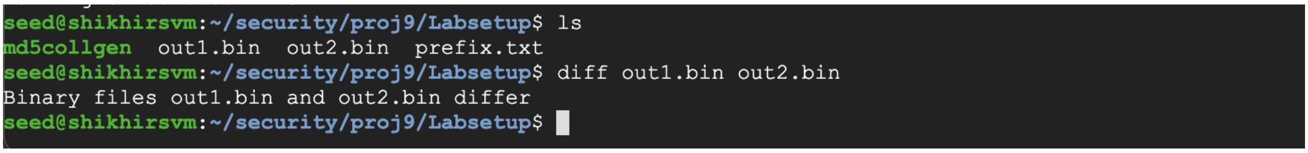


```
seed@shikhirsvm:~/security/proj9/Labsetup$ ls
md5collgen  out1.bin  out2.bin
seed@shikhirsvm:~/security/proj9/Labsetup$ echo "YES, THIS MIGHT WORK" > prefi
x.txt
seed@shikhirsvm:~/security/proj9/Labsetup$ ./md5collgen -p prefix.txt -o out1.
bin out2.bin
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'out1.bin' and 'out2.bin'
Using prefixfile: 'prefix.txt'
Using initial value: 1d13ec15517a8334b939c74f2475f35e

Generating first block: ....
Generating second block: S10.......................
Running time: 5.86712 s
seed@shikhirsvm:~/security/proj9/Labsetup$ ls
md5collgen  out1.bin  out2.bin  prefix.txt
seed@shikhirsvm:~/security/proj9/Labsetup$
```

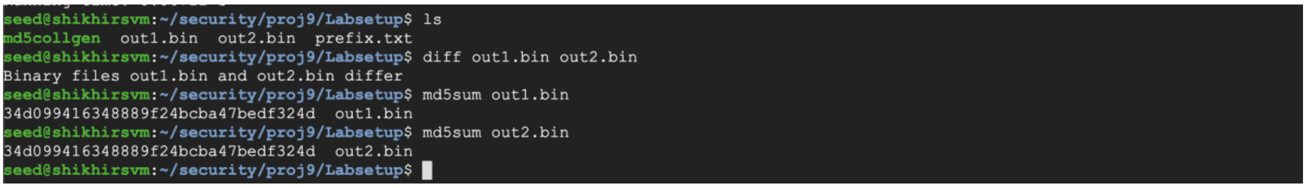
By using diff command, we observe that the two output files are different.



```
seed@shikhirsvm:~/security/proj9/Labsetup$ ls
md5collgen  out1.bin  out2.bin  prefix.txt
seed@shikhirsvm:~/security/proj9/Labsetup$ diff out1.bin out2.bin
Binary files out1.bin and out2.bin differ
seed@shikhirsvm:~/security/proj9/Labsetup$
```

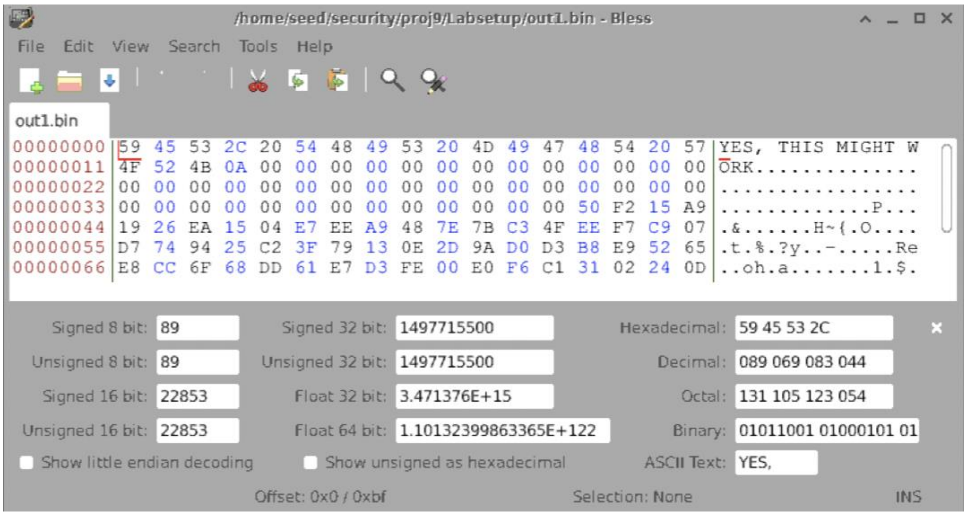
These two files are different, but they have same hash value.

The tool generates parts P and Q for given prefix text such that hash (prefix + P) = hash (prefix + Q). We can confirm this using md5sum to check Hash value of each output file.



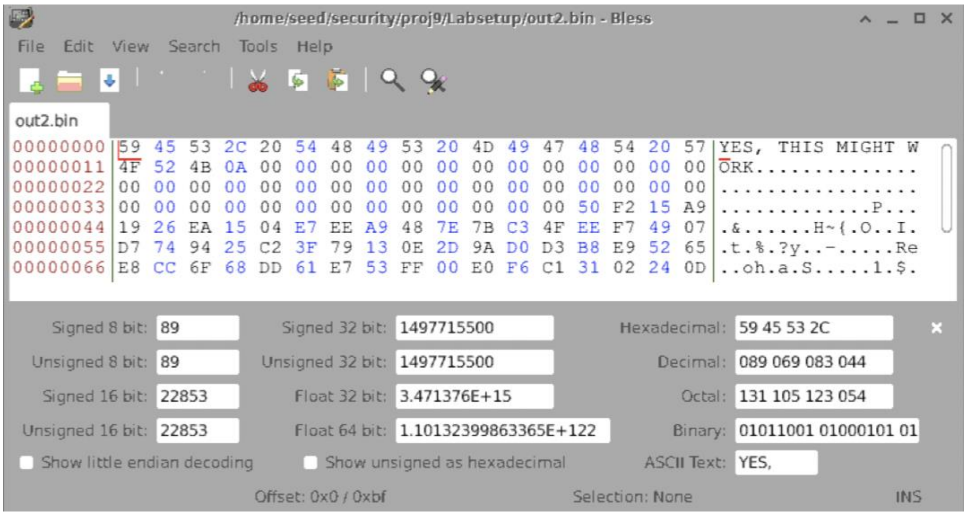
```
seed@shikhirsvm:~/security/proj9/Labsetup$ ls
md5collgen  out1.bin  out2.bin  prefix.txt
seed@shikhirsvm:~/security/proj9/Labsetup$ diff out1.bin out2.bin
Binary files out1.bin and out2.bin differ
seed@shikhirsvm:~/security/proj9/Labsetup$ md5sum out1.bin
34d099416348889f24bcba47bedf324d  out1.bin
seed@shikhirsvm:~/security/proj9/Labsetup$ md5sum out2.bin
34d099416348889f24bcba47bedf324d  out2.bin
seed@shikhirsvm:~/security/proj9/Labsetup$
```

Using Hex we can check the binary files:


/home/seed/security/proj9/Labsetup/out1.bin - Bless
File Edit View Search Tools Help
out1.bin

```
00000000 59 45 53 2C 20 54 48 49 53 20 4D 49 47 48 54 20 57 YES, THIS MIGHT W
00000011 4F 52 4B 0A 00 00 00 00 00 00 00 00 00 00 00 00 00 ORK..............
00000022 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .................
00000033 00 00 00 00 00 00 00 00 00 00 00 00 00 50 F2 15 A9 .............P...
00000044 19 26 EA 15 04 E7 EE A9 48 7E 7B C3 4F EE F7 C9 07 .&......H~{.O....
00000055 D7 74 94 25 C2 3F 79 13 0E 2D 9A D0 D3 B8 E9 52 65 .t.%.?y..-.....Re
00000066 E8 CC 6F 68 DD 61 E7 D3 FE 00 E0 F6 C1 31 02 24 0D ..oh.a.......1.$.
```

Signed 8 bit: 89
Unsigned 8 bit: 89
Signed 16 bit: 22853
Unsigned 16 bit: 22853
Signed 32 bit: 1497715500
Unsigned 32 bit: 1497715500
Float 32 bit: 3.471376E+15
Float 64 bit: 1.10132399863365E+122
Hexadecimal: 59 45 53 2C
Decimal: 089 069 083 044
Octal: 131 105 123 054
Binary: 01011001 01000101 01
ASCII Text: YES,
□ Show little endian decoding
□ Show unsigned as hexadecimal
Offset: 0x0 / 0xbf
Selection: None
INS


/home/seed/security/proj9/Labsetup/out2.bin - Bless
File Edit View Search Tools Help
out2.bin

```
00000000 59 45 53 2C 20 54 48 49 53 20 4D 49 47 48 54 20 57 YES, THIS MIGHT W
00000011 4F 52 4B 0A 00 00 00 00 00 00 00 00 00 00 00 00 00 ORK..............
00000022 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .................
00000033 00 00 00 00 00 00 00 00 00 00 00 00 00 50 F2 15 A9 .............P...
00000044 19 26 EA 15 04 E7 EE A9 48 7E 7B C3 4F EE F7 49 07 .&......H~{.O..I.
00000055 D7 74 94 25 C2 3F 79 13 0E 2D 9A D0 D3 B8 E9 52 65 .t.%.?y..-.....Re
00000066 E8 CC 6F 68 DD 61 E7 53 FF 00 E0 F6 C1 31 02 24 0D ..oh.a.S.....1.$.
```

Signed 8 bit: 89
Unsigned 8 bit: 89
Signed 16 bit: 22853
Unsigned 16 bit: 22853
Signed 32 bit: 1497715500
Unsigned 32 bit: 1497715500
Float 32 bit: 3.471376E+15
Float 64 bit: 1.10132399863365E+122
Hexadecimal: 59 45 53 2C
Decimal: 089 069 083 044
Octal: 131 105 123 054
Binary: 01011001 01000101 01
ASCII Text: YES,
□ Show little endian decoding
□ Show unsigned as hexadecimal
Offset: 0x0 / 0xbf
Selection: None
INS

Using the hex dump, we observe that the prefix.txt data was first padded with zeros and brought up to 64 bytes, and then random data was appended at the end. The out2.bin file has the same initial 64 bytes (the prefix and the padding) but differs very slightly in the remaining bytes.

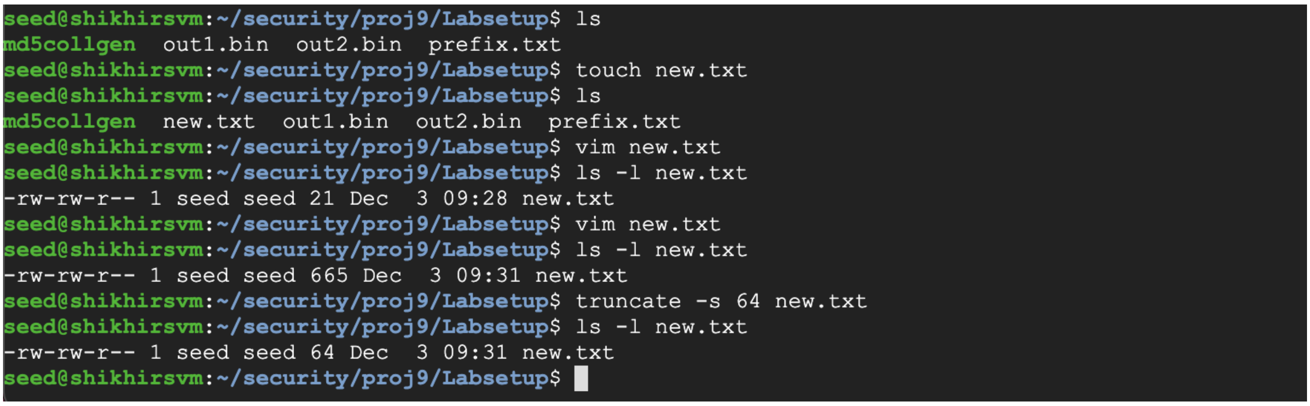
Question 1. If the length of your prefix file is not multiple of 64, what is going to happen?

As seen in the above example, I added some text to prefix file, it was lesser than 64 bytes, to make it a multiple of 64-bytes zeroes are appended as it can be observed in the above hex dump of output files.

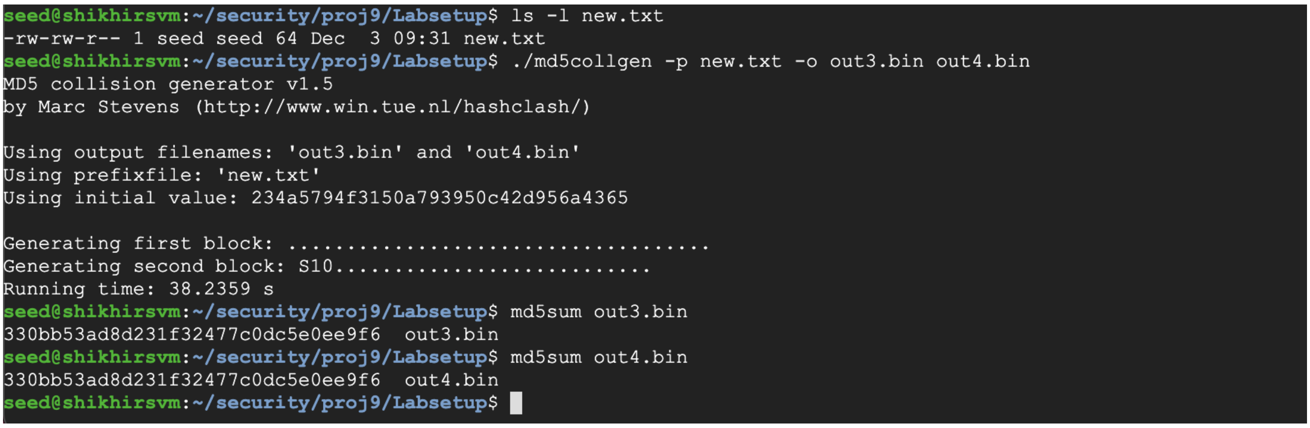
Question 2. Create a prefix file with exactly 64 bytes, and run the collision tool again, and see what happens?

For creating a file of size exactly 64 Bytes, I added random text to new.txt and then truncated it.

We can observe that its size is equal to 64 Bytes.



```
seed@shikhirsvm:~/security/proj9/Labsetup$ ls
md5collgen  out1.bin  out2.bin  prefix.txt
seed@shikhirsvm:~/security/proj9/Labsetup$ touch new.txt
seed@shikhirsvm:~/security/proj9/Labsetup$ ls
md5collgen  new.txt  out1.bin  out2.bin  prefix.txt
seed@shikhirsvm:~/security/proj9/Labsetup$ vim new.txt
seed@shikhirsvm:~/security/proj9/Labsetup$ ls -l new.txt
-rw-rw-r-- 1 seed seed 21 Dec  3 09:28 new.txt
seed@shikhirsvm:~/security/proj9/Labsetup$ vim new.txt
seed@shikhirsvm:~/security/proj9/Labsetup$ ls -l new.txt
-rw-rw-r-- 1 seed seed 665 Dec  3 09:31 new.txt
seed@shikhirsvm:~/security/proj9/Labsetup$ truncate -s 64 new.txt
seed@shikhirsvm:~/security/proj9/Labsetup$ ls -l new.txt
-rw-rw-r-- 1 seed seed 64 Dec  3 09:31 new.txt
seed@shikhirsvm:~/security/proj9/Labsetup$ 
```



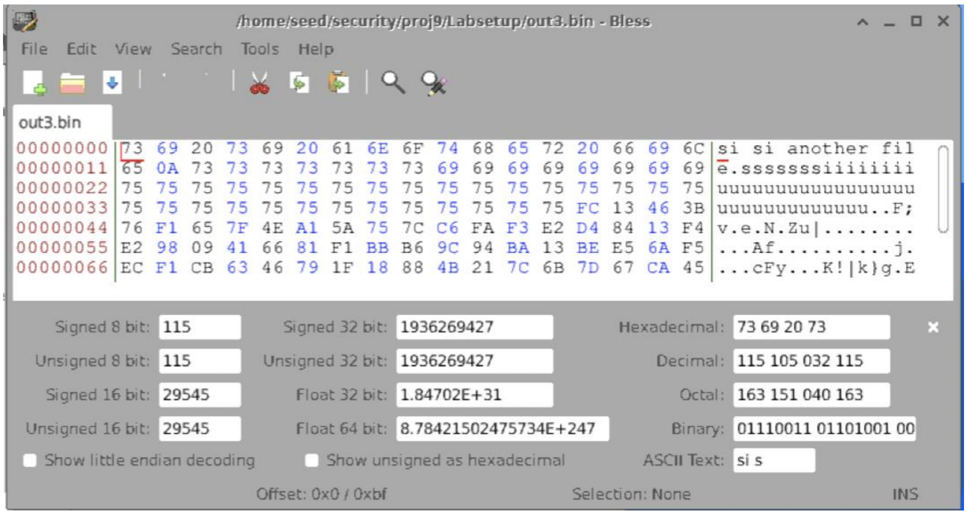
```
seed@shikhirsvm:~/security/proj9/Labsetup$ ls -l new.txt
-rw-rw-r-- 1 seed seed 64 Dec  3 09:31 new.txt
seed@shikhirsvm:~/security/proj9/Labsetup$ ./md5collgen -p new.txt -o out3.bin out4.bin
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'out3.bin' and 'out4.bin'
Using prefixfile: 'new.txt'
Using initial value: 234a5794f3150a793950c42d956a4365

Generating first block: ..................................
Generating second block: S10..........................
Running time: 38.2359 s
seed@shikhirsvm:~/security/proj9/Labsetup$ md5sum out3.bin
330bb53ad8d231f32477c0dc5e0ee9f6  out3.bin
seed@shikhirsvm:~/security/proj9/Labsetup$ md5sum out4.bin
330bb53ad8d231f32477c0dc5e0ee9f6  out4.bin
seed@shikhirsvm:~/security/proj9/Labsetup$ 
```

Using md5collgen tool I create two output files out3.bin and out4.bin using the new.txt as prefix. With the 64-byte prefix, no zero-bytes are added for padding.

The prefix file is followed by the random data, created for the collision as can be seen in following screenshots.

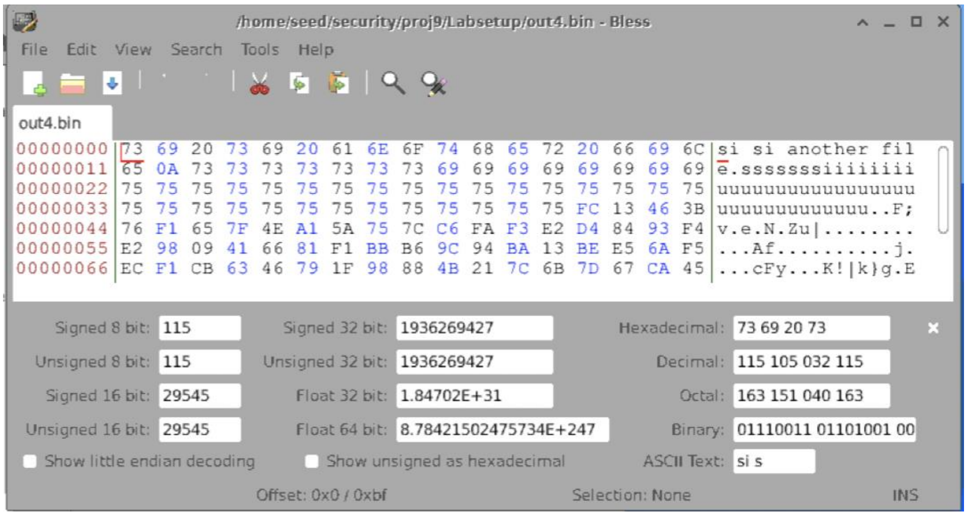


/home/seed/security/proj9/Labsetup/out3.bin - Bless
File Edit View Search Tools Help
out3.bin

```
00000000 73 69 20 73 69 20 61 6E 6F 74 68 65 72 20 66 69 6C si si another fil
00000011 65 0A 73 73 73 73 73 73 73 69 69 69 69 69 69 69 69 e.sssssssiiiiiiii
00000022 75 75 75 75 75 75 75 75 75 75 75 75 75 75 75 75 75 uuuuuuuuuuuuuuuuu
00000033 75 75 75 75 75 75 75 75 75 75 75 75 75 FC 13 46 3B uuuuuuuuuuuuu..F;
00000044 76 F1 65 7F 4E A1 5A 75 7C C6 FA F3 E2 D4 84 13 F4 v.e.N.Zu|........
00000055 E2 98 09 41 66 81 F1 BB B6 9C 94 BA 13 BE E5 6A F5 ...Af.........j.
00000066 EC F1 CB 63 46 79 1F 18 88 4B 21 7C 6B 7D 67 CA 45 ...cFy...K!|k}g.E
```

Signed 8 bit: 115
Unsigned 8 bit: 115
Signed 16 bit: 29545
Unsigned 16 bit: 29545
Signed 32 bit: 1936269427
Unsigned 32 bit: 1936269427
Float 32 bit: 1.84702E+31
Float 64 bit: 8.78421502475734E+247
Hexadecimal: 73 69 20 73
Decimal: 115 105 032 115
Octal: 163 151 040 163
Binary: 01110011 01101001 00
ASCII Text: si s
□ Show little endian decoding
□ Show unsigned as hexadecimal

Offset: 0x0 / 0xbf
Selection: None
INS



/home/seed/security/proj9/Labsetup/out4.bin - Bless
File Edit View Search Tools Help
out4.bin

```
00000000 73 69 20 73 69 20 61 6E 6F 74 68 65 72 20 66 69 6C si si another fil
00000011 65 0A 73 73 73 73 73 73 73 69 69 69 69 69 69 69 69 e.sssssssiiiiiiii
00000022 75 75 75 75 75 75 75 75 75 75 75 75 75 75 75 75 75 uuuuuuuuuuuuuuuuu
00000033 75 75 75 75 75 75 75 75 75 75 75 75 75 FC 13 46 3B uuuuuuuuuuuuu..F;
00000044 76 F1 65 7F 4E A1 5A 75 7C C6 FA F3 E2 D4 84 93 F4 v.e.N.Zu|........
00000055 E2 98 09 41 66 81 F1 BB B6 9C 94 BA 13 BE E5 6A F5 ...Af..........j.
00000066 EC F1 CB 63 46 79 1F 98 88 4B 21 7C 6B 7D 67 CA 45 ...cFy...K!|k}g.E
```

Signed 8 bit: 115 Signed 32 bit: 1936269427 Hexadecimal: 73 69 20 73

Unsigned 8 bit: 115 Unsigned 32 bit: 1936269427 Decimal: 115 105 032 115

Signed 16 bit: 29545 Float 32 bit: 1.84702E+31 Octal: 163 151 040 163

Unsigned 16 bit: 29545 Float 64 bit: 8.78421502475734E+247 Binary: 01110011 01101001 00

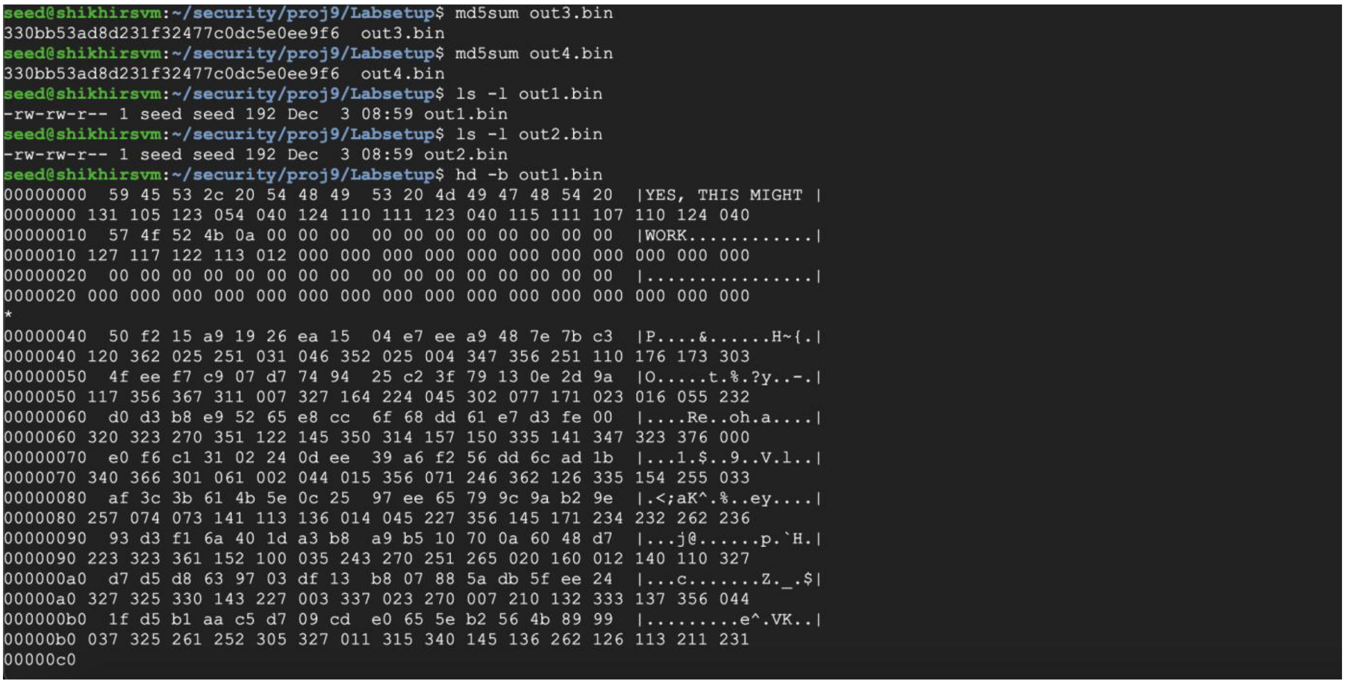
□ Show little endian decoding □ Show unsigned as hexadecimal ASCII Text: si s

Offset: 0x0 / 0xbf Selection: None INS

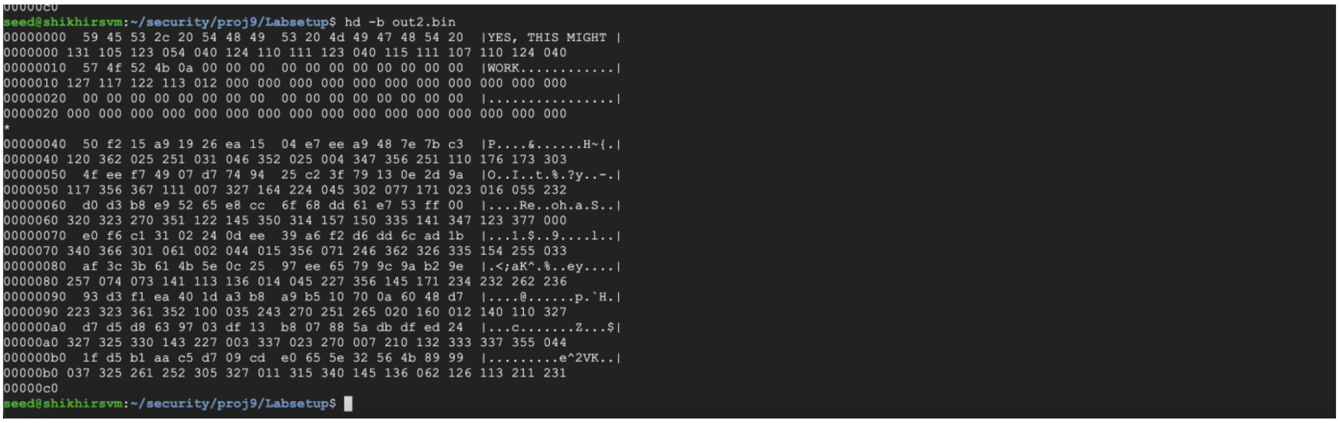
Question 3. Are the data (128 bytes) generated by md5collgen completely different for the two output files?

Please identify all the bytes that are different.

We look at the hex dumps of the last two output files (out3.bin and out4.bin) shows that only a few bytes are different between the files

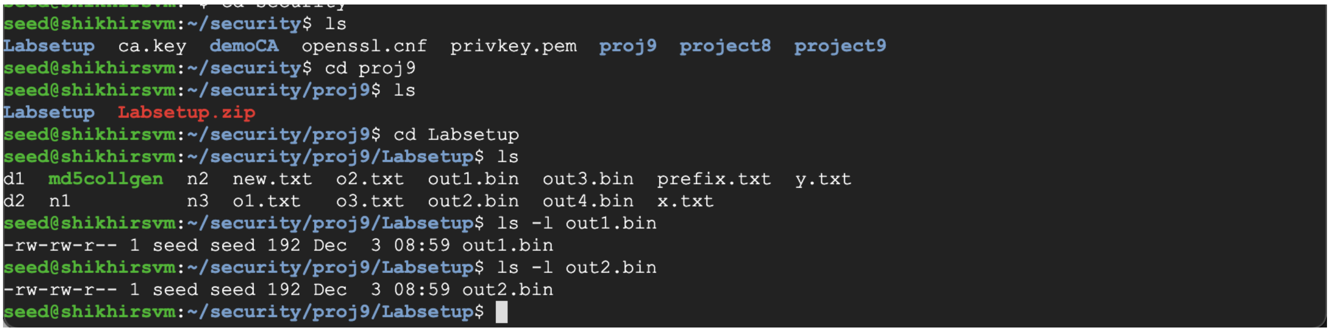


```
seed@shikhirsvm:~/security/proj9/Labsetup$ md5sum out3.bin
330bb53ad8d231f32477c0dc5e0ee9f6  out3.bin
seed@shikhirsvm:~/security/proj9/Labsetup$ md5sum out4.bin
330bb53ad8d231f32477c0dc5e0ee9f6  out4.bin
seed@shikhirsvm:~/security/proj9/Labsetup$ ls -l out1.bin
-rw-rw-r-- 1 seed seed 192 Dec  3 08:59 out1.bin
seed@shikhirsvm:~/security/proj9/Labsetup$ ls -l out2.bin
-rw-rw-r-- 1 seed seed 192 Dec  3 08:59 out2.bin
seed@shikhirsvm:~/security/proj9/Labsetup$ hd -b out1.bin
00000000  59 45 53 2c 20 54 48 49  53 20 4d 49 47 48 54 20  |YES, THIS MIGHT |
0000000 131 105 123 054 040 124 110 111 123 040 115 111 107 110 124 040
00000010  57 4f 52 4b 0a 00 00 00  00 00 00 00 00 00 00 00  |WORK............|
0000010 127 117 122 113 012 000 000 000 000 000 000 000 000 000 000 000
00000020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
0000020 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000
*
00000040  50 f2 15 a9 19 26 ea 15  04 e7 ee a9 48 7e 7b c3  |P....&......H~{.|
0000040 120 362 025 251 031 046 352 025 004 347 356 251 110 176 173 303
00000050  4f ee f7 c9 07 d7 74 94  25 c2 3f 79 13 0e 2d 9a  |O.....t.%.?y..-.|
0000050 117 356 367 311 007 327 164 224 045 302 077 171 023 016 055 232
00000060  d0 d3 b8 e9 52 65 e8 cc  6f 68 dd 61 e7 d3 fe 00  |....Re..oh.a....|
0000060 320 323 270 351 122 145 350 314 157 150 335 141 347 323 376 000
00000070  e0 f6 c1 31 02 24 0d ee  39 a6 f2 56 dd 6c ad 1b  |...1.$..9..V.l..|
0000070 340 366 301 061 002 044 015 356 071 246 362 126 335 154 255 033
00000080  af 3c 3b 61 4b 5e 0c 25  97 ee 65 79 9c 9a b2 9e  |.<;aK^.%..ey....|
0000080 257 074 073 141 113 136 014 045 227 356 145 171 234 232 262 236
00000090  93 d3 f1 6a 40 1d a3 b8  a9 b5 10 70 0a 60 48 d7  |...j@......p.`H.|
0000090 223 323 361 152 100 035 243 270 251 265 020 160 012 140 110 327
000000a0  d7 d5 d8 63 97 03 df 13  b8 07 88 5a db 5f ee 24  |...c.......Z._.$|
00000a0 327 325 330 143 227 003 337 023 270 007 210 132 333 137 356 044
000000b0  1f d5 b1 aa c5 d7 09 cd  e0 65 5e b2 56 4b 89 99  |.........e^.VK..|
00000b0 037 325 261 252 305 327 011 315 340 145 136 262 126 113 211 231
00000c0
```



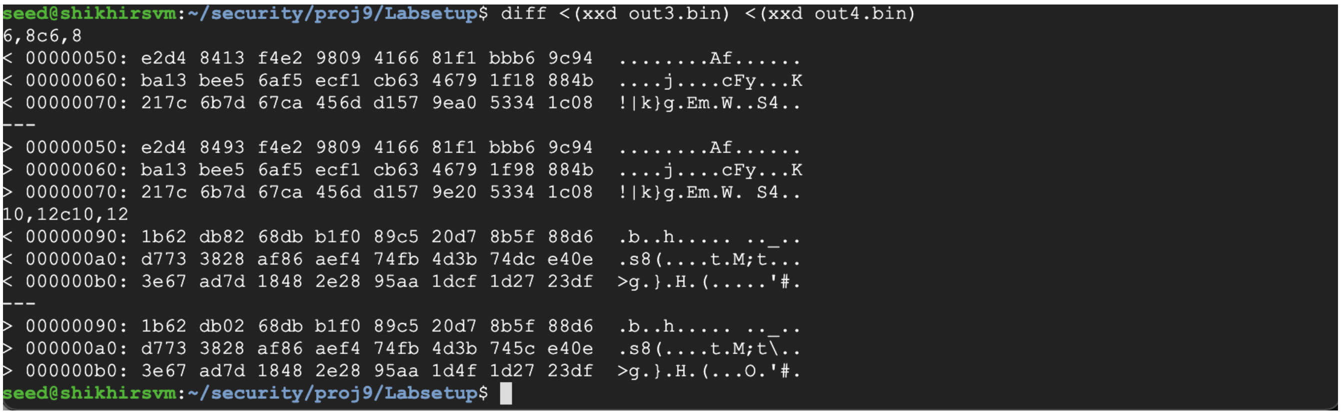
```
000000c0
seed@shikhirsvm:~/security/proj9/Labsetup$ hd -b out2.bin
00000000  59 45 53 2c 20 54 48 49  53 20 4d 49 47 48 54 20  |YES, THIS MIGHT |
0000000 131 105 123 054 040 124 110 111 123 040 115 111 107 110 124 040
00000010  57 4f 52 4b 0a 00 00 00  00 00 00 00 00 00 00 00  |WORK............|
0000010 127 117 122 113 012 000 000 000 000 000 000 000 000 000 000 000
00000020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
0000020 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000
*
00000040  50 f2 15 a9 19 26 ea 15  04 e7 ee a9 48 7e 7b c3  |P....&......H~{.|
0000040 120 362 025 251 031 046 352 025 004 347 356 251 110 176 173 303
00000050  4f ee f7 49 07 d7 74 94  25 c2 3f 79 13 0e 2d 9a  |O..I..t.%.?y..-.|
0000050 117 356 367 111 007 327 164 224 045 302 077 171 023 016 055 232
00000060  d0 d3 b8 e9 52 65 e8 cc  6f 68 dd 61 e7 53 ff 00  |....Re..oh.a.S..|
0000060 320 323 270 351 122 145 350 314 157 150 335 141 347 123 377 000
00000070  e0 f6 c1 31 02 24 0d ee  39 a6 f2 d6 dd 6c ad 1b  |...1.$..9....l..|
0000070 340 366 301 061 002 044 015 356 071 246 362 326 335 154 255 033
00000080  af 3c 3b 61 4b 5e 0c 25  97 ee 65 79 9c 9a b2 9e  |.<;aK^.%..ey....|
0000080 257 074 073 141 113 136 014 045 227 356 145 171 234 232 262 236
00000090  93 d3 f1 ea 40 1d a3 b8  a9 b5 10 70 0a 60 48 d7  |....@......p.`H.|
0000090 223 323 361 352 100 035 243 270 251 265 020 160 012 140 110 327
000000a0  d7 d5 d8 63 97 03 df 13  b8 07 88 5a db df ed 24  |...c.......Z...$|
00000a0 327 325 330 143 227 003 337 023 270 007 210 132 333 337 355 044
000000b0  1f d5 b1 aa c5 d7 09 cd  e0 65 5e 32 56 4b 89 99  |.........e^2VK..|
00000b0 037 325 261 252 305 327 011 315 340 145 136 062 126 113 211 231
00000c0
seed@shikhirsvm:~/security/proj9/Labsetup$
```

Also, the output files are of 192 bytes - 128 bytes of data + 64 bytes of padding



```
seed@shikhirsvm:~/security$ ls
Labsetup  ca.key  demoCA  openssl.cnf  privkey.pem  proj9  project8  project9
seed@shikhirsvm:~/security$ cd proj9
seed@shikhirsvm:~/security/proj9$ ls
Labsetup  Labsetup.zip
seed@shikhirsvm:~/security/proj9$ cd Labsetup
seed@shikhirsvm:~/security/proj9/Labsetup$ ls
d1  md5collgen  n2  new.txt  o2.txt  out1.bin  out3.bin  prefix.txt  y.txt
d2  n1          n3  o1.txt   o3.txt  out2.bin  out4.bin  x.txt
seed@shikhirsvm:~/security/proj9/Labsetup$ ls -l out1.bin
-rw-rw-r-- 1 seed seed 192 Dec  3 08:59 out1.bin
seed@shikhirsvm:~/security/proj9/Labsetup$ ls -l out2.bin
-rw-rw-r-- 1 seed seed 192 Dec  3 08:59 out2.bin
seed@shikhirsvm:~/security/proj9/Labsetup$ 
```

Using command diff < (xxd out1.bin) <(xxd out2.bin) we can see the difference in the bytes of the two binary files.



```
seed@shikhirsvm:~/security/proj9/Labsetup$ diff <(xxd out3.bin) <(xxd out4.bin)
6,8c6,8
< 00000050: e2d4 8413 f4e2 9809 4166 81f1 bbb6 9c94  ........Af......
< 00000060: ba13 bee5 6af5 ecf1 cb63 4679 1f18 884b  ....j....cFy...K
< 00000070: 217c 6b7d 67ca 456d d157 9ea0 5334 1c08  !|k}g.Em.W..S4..
---
> 00000050: e2d4 8493 f4e2 9809 4166 81f1 bbb6 9c94  ........Af......
> 00000060: ba13 bee5 6af5 ecf1 cb63 4679 1f98 884b  ....j....cFy...K
> 00000070: 217c 6b7d 67ca 456d d157 9e20 5334 1c08  !|k}g.Em.W. S4..
10,12c10,12
< 00000090: 1b62 db82 68db b1f0 89c5 20d7 8b5f 88d6  .b..h..... .._..
< 000000a0: d773 3828 af86 aef4 74fb 4d3b 74dc e40e  .s8(....t.M;t...
< 000000b0: 3e67 ad7d 1848 2e28 95aa 1dcf 1d27 23df  >g.}.H.(.....'#.
---
> 00000090: 1b62 db02 68db b1f0 89c5 20d7 8b5f 88d6  .b..h..... .._..
> 000000a0: d773 3828 af86 aef4 74fb 4d3b 745c e40e  .s8(....t.M;t\..
> 000000b0: 3e67 ad7d 1848 2e28 95aa 1d4f 1d27 23df  >g.}.H.(...O.'#.
seed@shikhirsvm:~/security/proj9/Labsetup$ 
```

Task 2: Understanding MD5's Property

Here we try to prove the property of MD5 Algorithm:

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 k T) = MD5(N k T), where k represents concatenation. That is, if inputs M and N have the same hash,

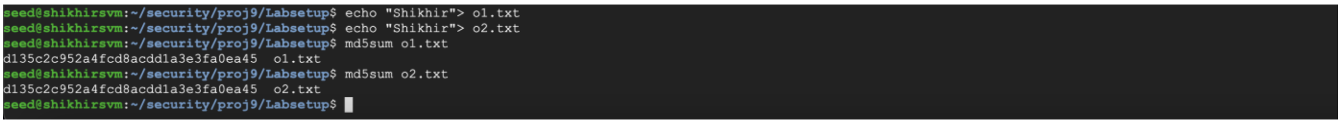
adding the same suffix T to them will result in two outputs that have the same hash value.

I use two methods where the hash values of two files will be same and we can prove that after appending same text to both the files the resulting file has the same hash value.

At high level, MD5 divides its data into blocks of 64 bytes and then computes the hash iteratively on these blocks. The core of MD5 is a compression function which produces a 128-bit IHV or intermediate hash value.

Method 1:

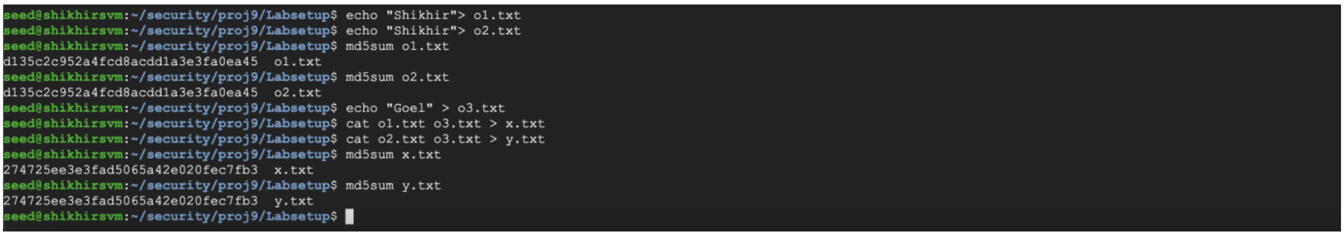
I create two files o1 and o2 which have same hash value, by storing same content in it. Using md5sum we can see that they have same hash value generated.



```
seed@shikhirsvm:~/security/proj9/Labsetup$ echo "Shikhir"> o1.txt
seed@shikhirsvm:~/security/proj9/Labsetup$ echo "Shikhir"> o2.txt
seed@shikhirsvm:~/security/proj9/Labsetup$ md5sum o1.txt
d135c2c952a4fcd8acdd1a3e3fa0ea45  o1.txt
seed@shikhirsvm:~/security/proj9/Labsetup$ md5sum o2.txt
d135c2c952a4fcd8acdd1a3e3fa0ea45  o2.txt
seed@shikhirsvm:~/security/proj9/Labsetup$ 
```

Creating third file o3.txt which will be used append to these two files and then we will be checking their hash values. We append o3 to o1 and o3 to o2 using cat command and then store the results in x and y respectively.

We can see that after adding same suffix to o1 and o2 the resultant output files have same generated hash value (By using md5sum).

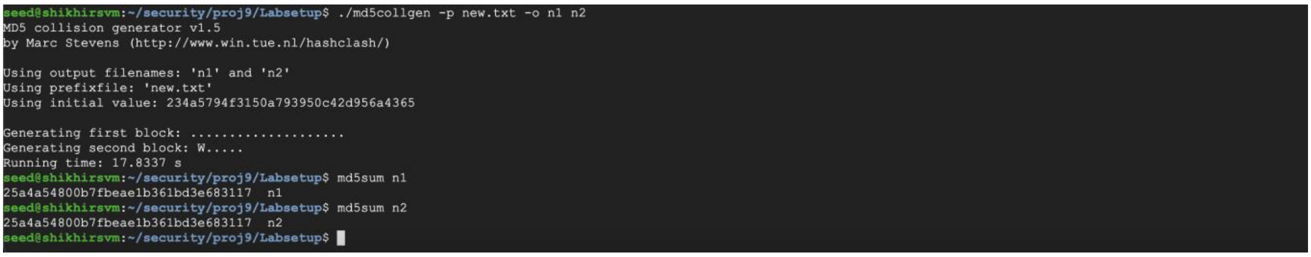


```
seed@shikhirsvm:~/security/proj9/Labsetup$ echo "Shikhir"> o1.txt
seed@shikhirsvm:~/security/proj9/Labsetup$ echo "Shikhir"> o2.txt
seed@shikhirsvm:~/security/proj9/Labsetup$ md5sum o1.txt
d135c2c952a4fcd8acdd1a3e3fa0ea45  o1.txt
seed@shikhirsvm:~/security/proj9/Labsetup$ md5sum o2.txt
d135c2c952a4fcd8acdd1a3e3fa0ea45  o2.txt
seed@shikhirsvm:~/security/proj9/Labsetup$ echo "Goel" > o3.txt
seed@shikhirsvm:~/security/proj9/Labsetup$ cat o1.txt o3.txt > x.txt
seed@shikhirsvm:~/security/proj9/Labsetup$ cat o2.txt o3.txt > y.txt
seed@shikhirsvm:~/security/proj9/Labsetup$ md5sum x.txt
274725ee3e3fad5065a42e020fec7fb3  x.txt
seed@shikhirsvm:~/security/proj9/Labsetup$ md5sum y.txt
274725ee3e3fad5065a42e020fec7fb3  y.txt
seed@shikhirsvm:~/security/proj9/Labsetup$ 
```

Method 2:

We create two files having same hash value - generated from same prefix(new.txt).

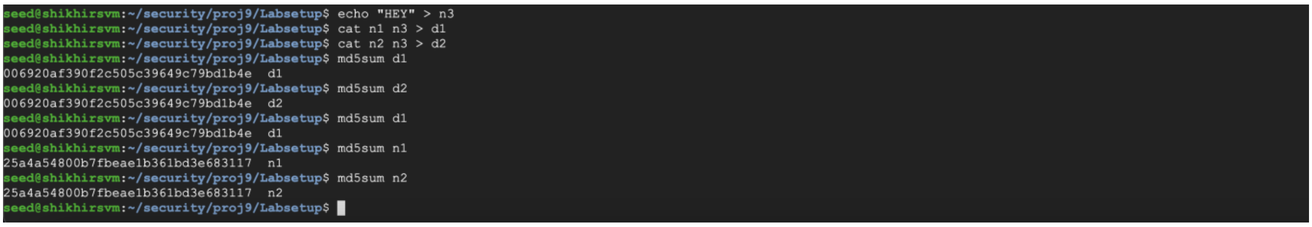
By following the same procedure, we add the same suffix and the output files generated also have same hash value, hence this proves the MD5 property.



```
seed@shikhirsvm:~/security/proj9/Labsetup$ ./md5collgen -p new.txt -o n1 n2
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'n1' and 'n2'
Using prefixfile: 'new.txt'
Using initial value: 234a5794f3150a793950c42d956a4365

Generating first block: ...................
Generating second block: W.....
Running time: 17.8337 s
seed@shikhirsvm:~/security/proj9/Labsetup$ md5sum n1
25a4a54800b7fbeae1b361bd3e683117  n1
seed@shikhirsvm:~/security/proj9/Labsetup$ md5sum n2
25a4a54800b7fbeae1b361bd3e683117  n2
seed@shikhirsvm:~/security/proj9/Labsetup$ 
```



```
seed@shikhirsvm:~/security/proj9/Labsetup$ echo "HEY" > n3
seed@shikhirsvm:~/security/proj9/Labsetup$ cat n1 n3 > d1
seed@shikhirsvm:~/security/proj9/Labsetup$ cat n2 n3 > d2
seed@shikhirsvm:~/security/proj9/Labsetup$ md5sum d1
006920af390f2c505c39649c79bd1b4e  d1
seed@shikhirsvm:~/security/proj9/Labsetup$ md5sum d2
006920af390f2c505c39649c79bd1b4e  d2
seed@shikhirsvm:~/security/proj9/Labsetup$ md5sum d1
006920af390f2c505c39649c79bd1b4e  d1
seed@shikhirsvm:~/security/proj9/Labsetup$ md5sum n1
25a4a54800b7fbeae1b361bd3e683117  n1
seed@shikhirsvm:~/security/proj9/Labsetup$ md5sum n2
25a4a54800b7fbeae1b361bd3e683117  n2
seed@shikhirsvm:~/security/proj9/Labsetup$ 
```