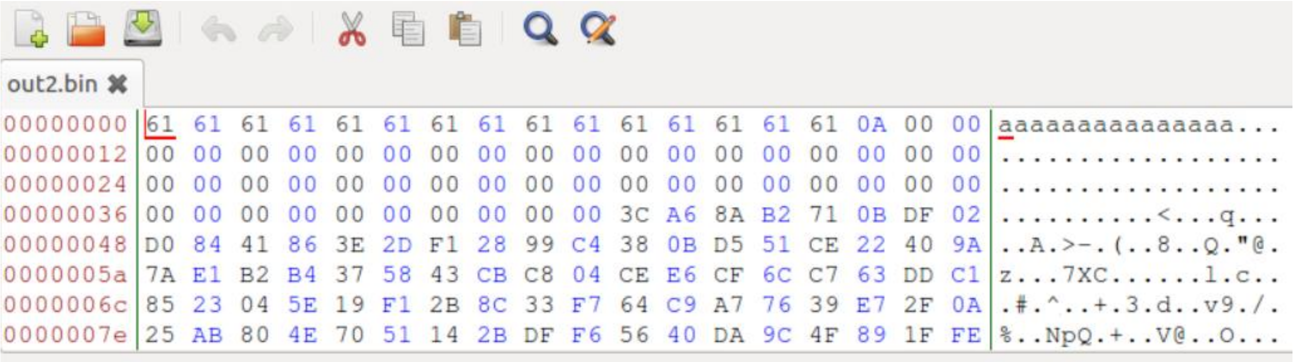
LAB 1 – MD5 Collision Attack Lab

Due: Wednesday Sep 29, 2:30pm (before the class starts)
Turn in: this lab report
Points: 60 pts

Please explain in detail, and make sure the screenshots are easy to read.

(20 pts) Task 1: Generating Two Different Files with the Same MD5 Hash

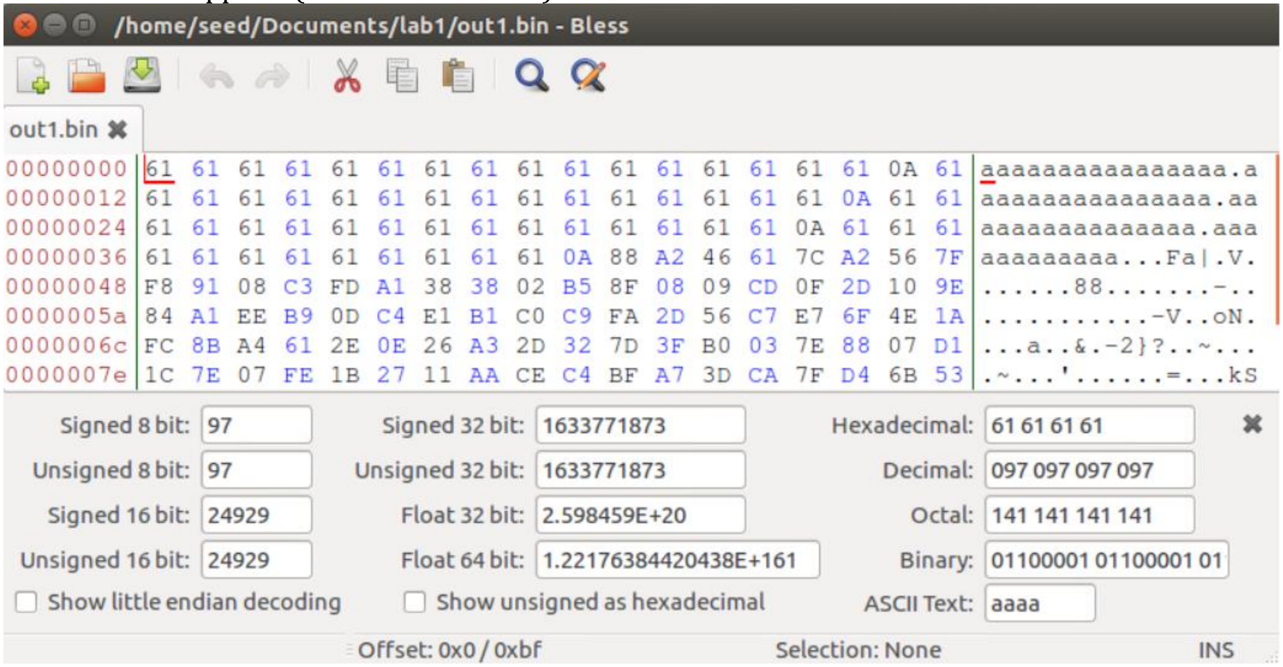
- (7 pts) Question 1. If the length of your prefix file is not multiple of 64, what is going to happen? (Screenshot needed)
  - md5collgen adds 00s onto the file until it reaches a multiple of 64.



out2.bin

```
00000000 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 0A 00 00 aaaaaaaaaaaaaaa...
00000012 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..................
00000024 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..................
00000036 00 00 00 00 00 00 00 00 00 00 3C A6 8A B2 71 0B DF 02 ..........<...q...
00000048 D0 84 41 86 3E 2D F1 28 99 C4 38 0B D5 51 CE 22 40 9A ..A.>-.(..8..Q."@.
0000005a 7A E1 B2 B4 37 58 43 CB C8 04 CE E6 CF 6C C7 63 DD C1 z...7XC......l.c..
0000006c 85 23 04 5E 19 F1 2B 8C 33 F7 64 C9 A7 76 39 E7 2F 0A .#.^..+.3.d..v9./.
0000007e 25 AB 80 4E 70 51 14 2B DF F6 56 40 DA 9C 4F 89 1F FE %..NpQ.+..V@..O...
```

- (7 pts) Question 2. Create a prefix file with exactly 64 bytes, and run the collision tool again, and see what happens. (Screenshot needed)



/home/seed/Documents/lab1/out1.bin - Bless

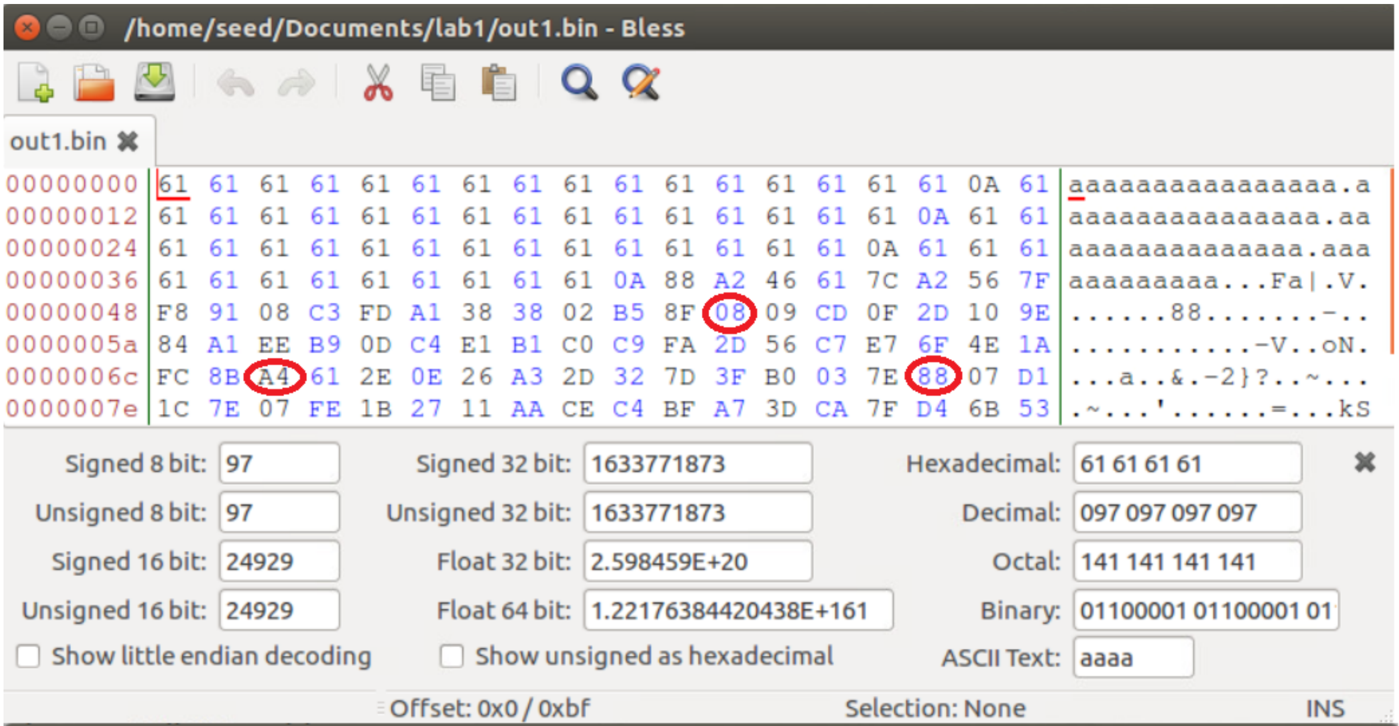
out1.bin

```
00000000 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 0A 61 aaaaaaaaaaaaaaaa.a
00000012 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 0A 61 61 aaaaaaaaaaaaaaa.aa
00000024 61 61 61 61 61 61 61 61 61 61 61 61 61 61 0A 61 61 61 aaaaaaaaaaaaaa.aaa
00000036 61 61 61 61 61 61 61 61 61 0A 88 A2 46 61 7C A2 56 7F aaaaaaaaa...Fa|.V.
00000048 F8 91 08 C3 FD A1 38 38 02 B5 8F 08 09 CD 0F 2D 10 9E ......88.......-..
0000005a 84 A1 EE B9 0D C4 E1 B1 C0 C9 FA 2D 56 C7 E7 6F 4E 1A ...........-V..oN.
0000006c FC 8B A4 61 2E 0E 26 A3 2D 32 7D 3F B0 03 7E 88 07 D1 ...a..&.-2}?..~...
0000007e 1C 7E 07 FE 1B 27 11 AA CE C4 BF A7 3D CA 7F D4 6B 53 .~...'......=...kS
```

Signed 8 bit:	97	Signed 32 bit:	1633771873	Hexadecimal:	61 61 61 61
Unsigned 8 bit:	97	Unsigned 32 bit:	1633771873	Decimal:	097 097 097 097
Signed 16 bit:	24929	Float 32 bit:	2.598459E+20	Octal:	141 141 141 141
Unsigned 16 bit:	24929	Float 64 bit:	1.22176384420438E+161	Binary:	01100001 01100001 01
☐ Show little endian decoding		☐ Show unsigned as hexadecimal		ASCII Text:	aaaa

Offset: 0x0 / 0xbf Selection: None INS

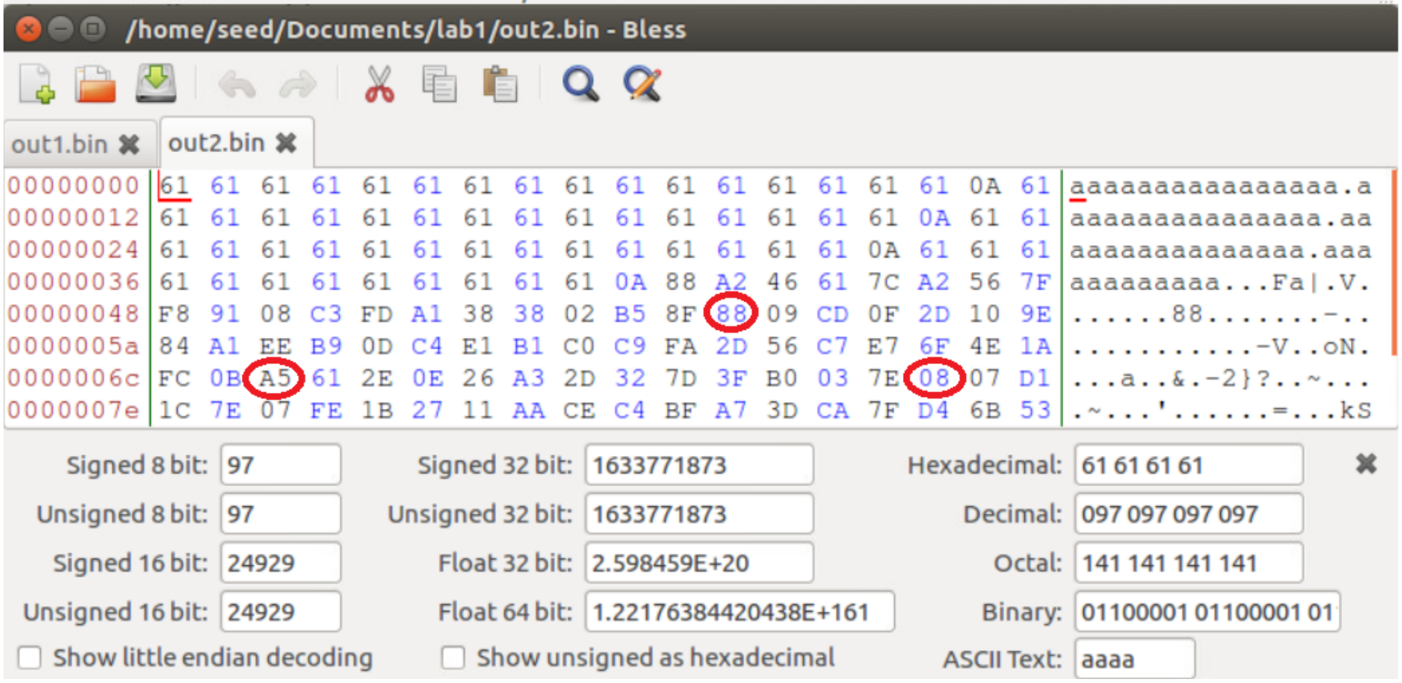
- (6 pts) Question 3. Are the data (128 bytes) generated by md5collgen completely different for the two output files? Please identify all the bytes that are different. (Screenshots needed, circle/highlight the different bytes)
  - The bytes are actually mostly the same. There are only 3 places that changed.



```
00000000 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 0A 61 aaaaaaaaaaaaaaaa.a
00000012 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 0A 61 61 aaaaaaaaaaaaaaa.aa
00000024 61 61 61 61 61 61 61 61 61 61 61 61 61 61 0A 61 61 61 aaaaaaaaaaaaaa.aaa
00000036 61 61 61 61 61 61 61 61 61 0A 88 A2 46 61 7C A2 56 7F aaaaaaaaa...Fa|.V.
00000048 F8 91 08 C3 FD A1 38 38 02 B5 8F 08 09 CD 0F 2D 10 9E ......88.......-..
0000005a 84 A1 EE B9 0D C4 E1 B1 C0 C9 FA 2D 56 C7 E7 6F 4E 1A ...........-V..oN.
0000006c FC 8B A4 61 2E 0E 26 A3 2D 32 7D 3F B0 03 7E 88 07 D1 ...a..&.-2}?..~...
0000007e 1C 7E 07 FE 1B 27 11 AA CE C4 BF A7 3D CA 7F D4 6B 53 .~...'......=...kS
```

Signed 8 bit: 97 | Signed 32 bit: 1633771873 | Hexadecimal: 61 61 61 61
Unsigned 8 bit: 97 | Unsigned 32 bit: 1633771873 | Decimal: 097 097 097 097
Signed 16 bit: 24929 | Float 32 bit: 2.598459E+20 | Octal: 141 141 141 141
Unsigned 16 bit: 24929 | Float 64 bit: 1.22176384420438E+161 | Binary: 01100001 01100001 01
☐ Show little endian decoding | ☐ Show unsigned as hexadecimal | ASCII Text: aaaa

Offset: 0x0 / 0xbf Selection: None INS



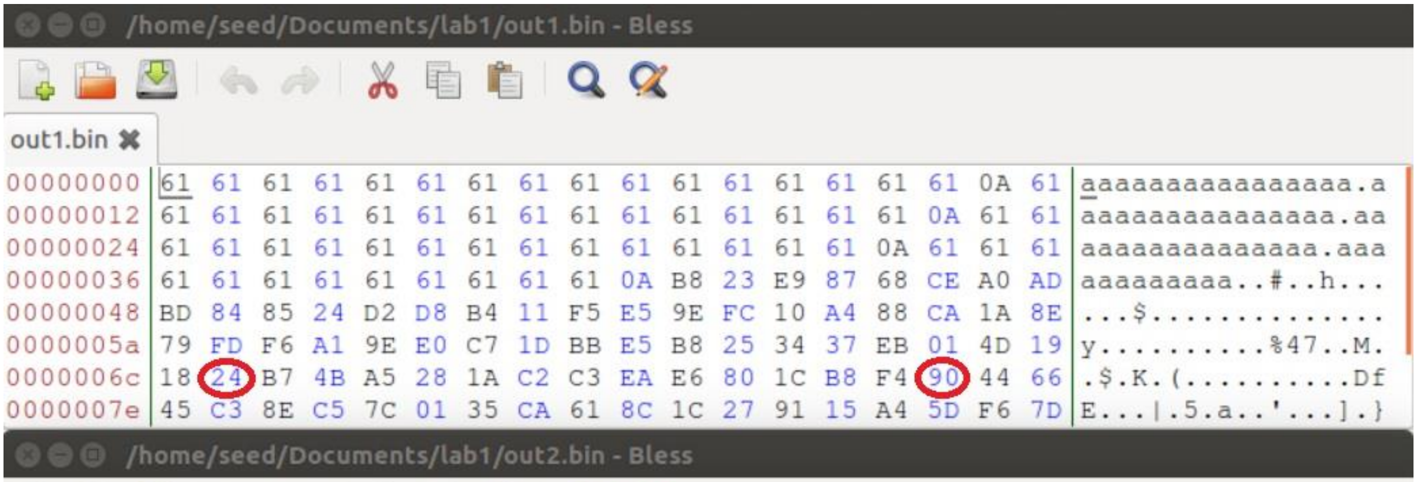
```
00000000 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 0A 61 aaaaaaaaaaaaaaaa.a
00000012 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 0A 61 61 aaaaaaaaaaaaaaa.aa
00000024 61 61 61 61 61 61 61 61 61 61 61 61 61 61 0A 61 61 61 aaaaaaaaaaaaaa.aaa
00000036 61 61 61 61 61 61 61 61 61 0A 88 A2 46 61 7C A2 56 7F aaaaaaaaa...Fa|.V.
00000048 F8 91 08 C3 FD A1 38 38 02 B5 8F 88 09 CD 0F 2D 10 9E ......88.......-..
0000005a 84 A1 EE B9 0D C4 E1 B1 C0 C9 FA 2D 56 C7 E7 6F 4E 1A ...........-V..oN.
0000006c FC 0B A5 61 2E 0E 26 A3 2D 32 7D 3F B0 03 7E 08 07 D1 ...a..&.-2}?..~...
0000007e 1C 7E 07 FE 1B 27 11 AA CE C4 BF A7 3D CA 7F D4 6B 53 .~...'......=...kS
```

Signed 8 bit: 97 | Signed 32 bit: 1633771873 | Hexadecimal: 61 61 61 61
Unsigned 8 bit: 97 | Unsigned 32 bit: 1633771873 | Decimal: 097 097 097 097
Signed 16 bit: 24929 | Float 32 bit: 2.598459E+20 | Octal: 141 141 141 141
Unsigned 16 bit: 24929 | Float 64 bit: 1.22176384420438E+161 | Binary: 01100001 01100001 01
☐ Show little endian decoding | ☐ Show unsigned as hexadecimal | ASCII Text: aaaa

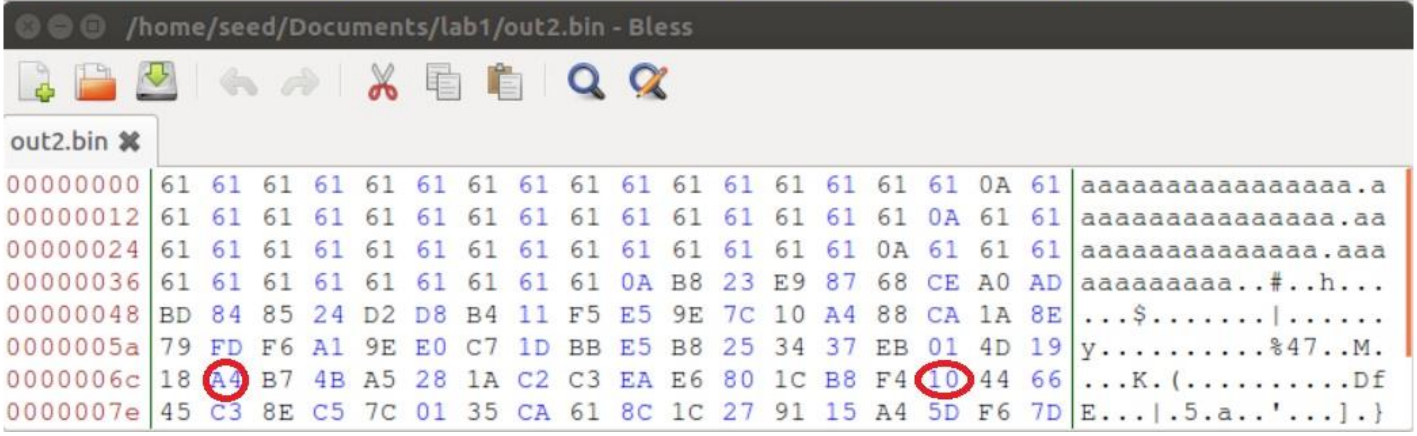
(10 pts) Task 2: Understanding MD5's Property

Explain in detail about how you finish Task2.

- (5 pts) Screenshots needed to show M and N are different, but their MD5 hash values are identical.
    - Different Files:

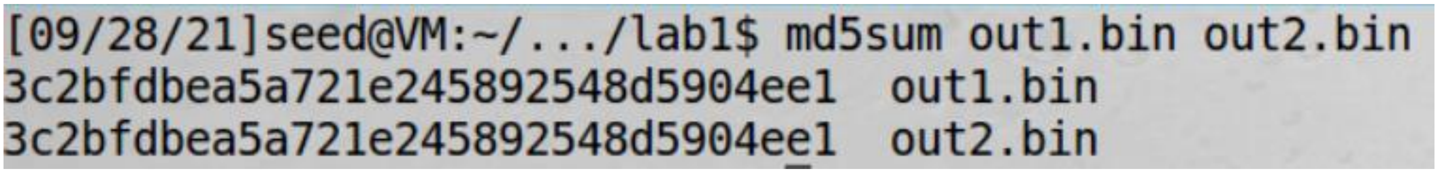


```
00000000 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 0A 61 aaaaaaaaaaaaaaaa.a
00000012 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 0A 61 61 aaaaaaaaaaaaaaa.aa
00000024 61 61 61 61 61 61 61 61 61 61 61 61 61 61 0A 61 61 61 aaaaaaaaaaaaaa.aaa
00000036 61 61 61 61 61 61 61 61 61 0A B8 23 E9 87 68 CE A0 AD aaaaaaaaa..#..h...
00000048 BD 84 85 24 D2 D8 B4 11 F5 E5 9E FC 10 A4 88 CA 1A 8E ...$..............
0000005a 79 FD F6 A1 9E E0 C7 1D BB E5 B8 25 34 37 EB 01 4D 19 y..........%47..M.
0000006c 18 24 B7 4B A5 28 1A C2 C3 EA E6 80 1C B8 F4 90 44 66 .$.K.(..........Df
0000007e 45 C3 8E C5 7C 01 35 CA 61 8C 1C 27 91 15 A4 5D F6 7D E...|.5.a..'...].}
```



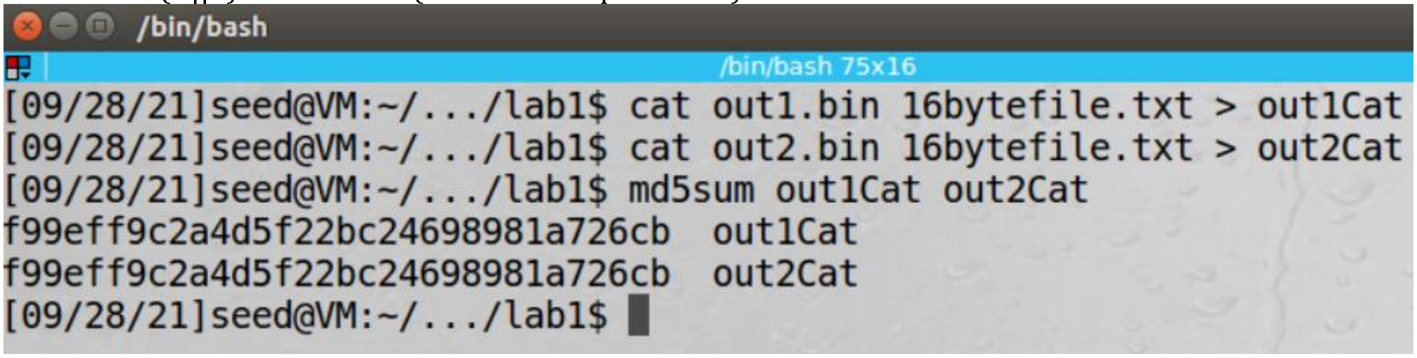
```
00000000 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 0A 61 aaaaaaaaaaaaaaaa.a
00000012 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 0A 61 61 aaaaaaaaaaaaaaa.aa
00000024 61 61 61 61 61 61 61 61 61 61 61 61 61 61 0A 61 61 61 aaaaaaaaaaaaaa.aaa
00000036 61 61 61 61 61 61 61 61 61 0A B8 23 E9 87 68 CE A0 AD aaaaaaaaa..#..h...
00000048 BD 84 85 24 D2 D8 B4 11 F5 E5 9E 7C 10 A4 88 CA 1A 8E ...$.......|......
0000005a 79 FD F6 A1 9E E0 C7 1D BB E5 B8 25 34 37 EB 01 4D 19 y..........%47..M.
0000006c 18 A4 B7 4B A5 28 1A C2 C3 EA E6 80 1C B8 F4 10 44 66 ...K.(..........Df
0000007e 45 C3 8E C5 7C 01 35 CA 61 8C 1C 27 91 15 A4 5D F6 7D E...|.5.a..'...].}
```

- Same Hash:



```
[09/28/21]seed@VM:~/.../lab1$ md5sum out1.bin out2.bin
3c2bfdbea5a721e245892548d5904ee1  out1.bin
3c2bfdbea5a721e245892548d5904ee1  out2.bin
```

- (5 pts) Have screenshots to show you concatenated the files and your MD5(M||T) and MD5(N||T) are identical. (T can be a simple .txt file)

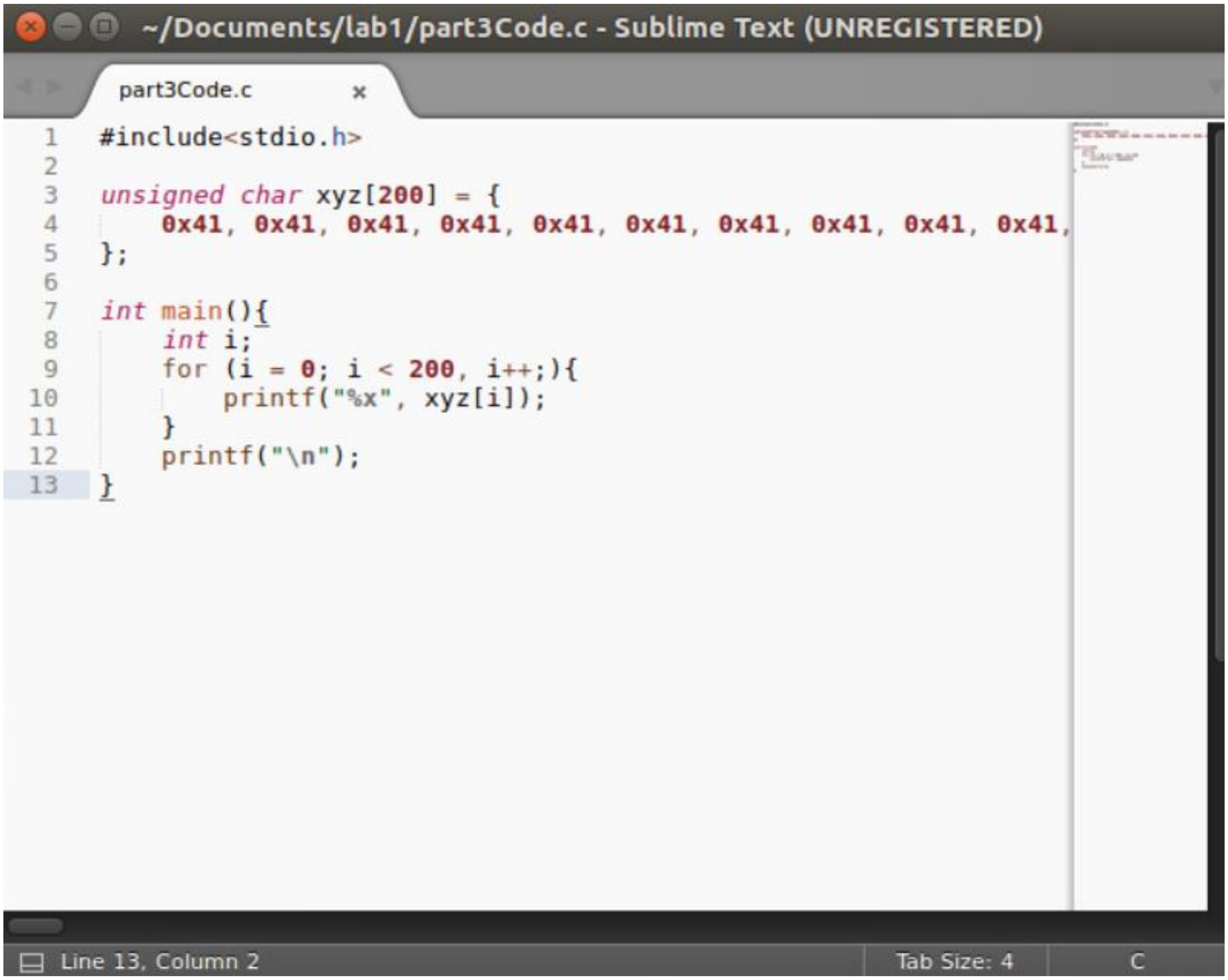


```
[09/28/21]seed@VM:~/.../lab1$ cat out1.bin 16bytefile.txt > out1Cat
[09/28/21]seed@VM:~/.../lab1$ cat out2.bin 16bytefile.txt > out2Cat
[09/28/21]seed@VM:~/.../lab1$ md5sum out1Cat out2Cat
f99eff9c2a4d5f22bc24698981a726cb  out1Cat
f99eff9c2a4d5f22bc24698981a726cb  out2Cat
[09/28/21]seed@VM:~/.../lab1$ 
```

  - The files start with the same hash, and after concatenating the same file to each of them, the outputs still have the same hash.

(20 pts) Task 3: Generating Two Executable Files with the Same MD5 Hash

- (5 pts) Take a screenshot of your C/C++ file used in this task.

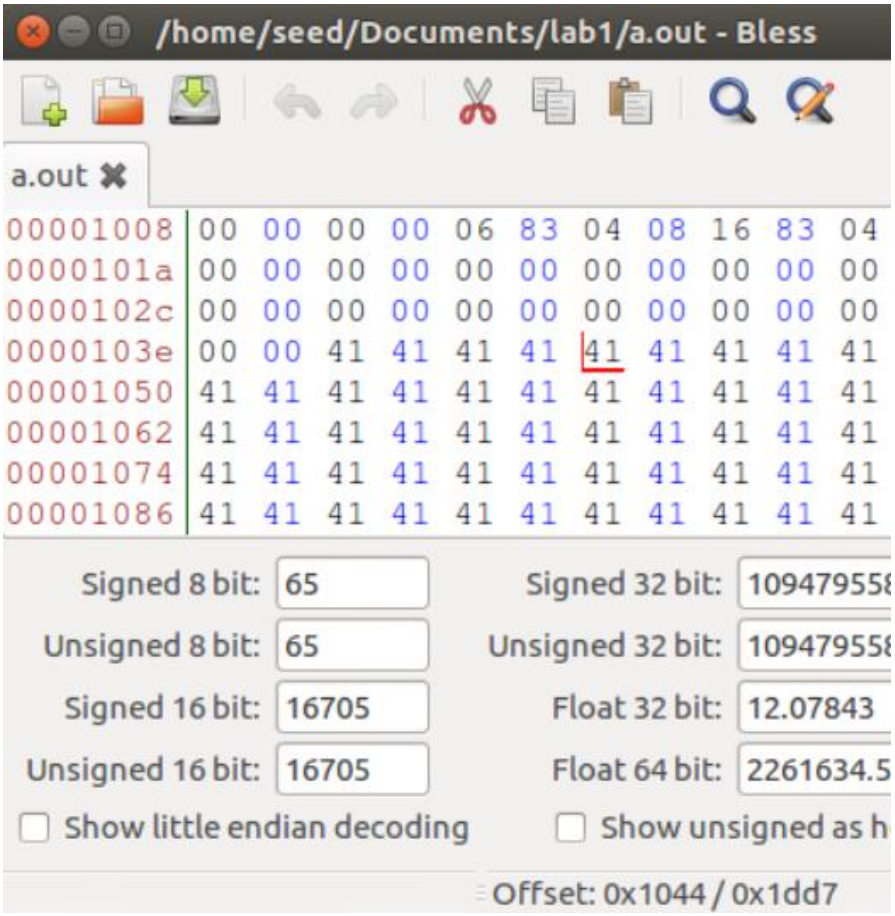


```c
#include<stdio.h>

unsigned char xyz[200] = {
    0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
};

int main(){
    int i;
    for (i = 0; i < 200, i++;){
        printf("%x", xyz[i]);
    }
    printf("\n");
}
```

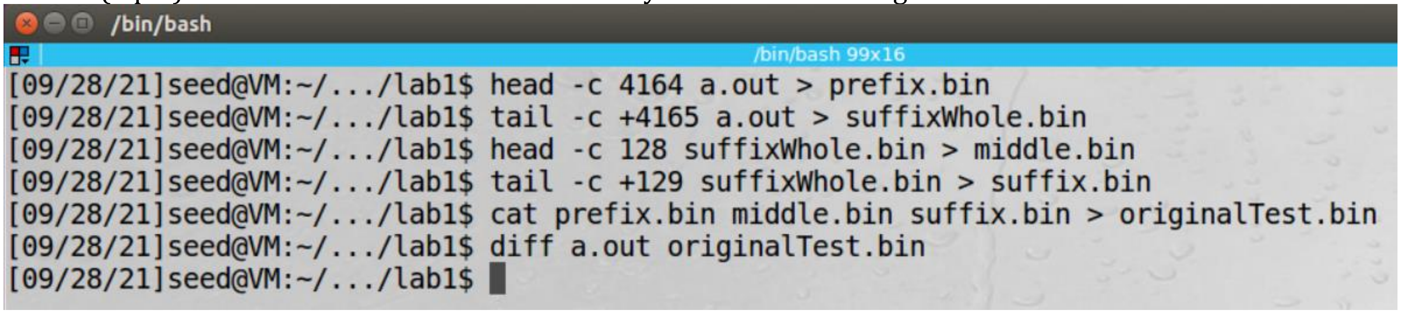
- (5 pts) How did you find the end position of the prefix? (Screenshot needed to explain)
  - I looked for the list of As, then I looked at the Offset at the bottom of the Bless windows to get the character location.
  - 0x1044 = 4164



```
00001008 00 00 00 00 06 83 04 08 16 83 04
0000101a 00 00 00 00 00 00 00 00 00 00 00
0000102c 00 00 00 00 00 00 00 00 00 00 00
0000103e 00 00 41 41 41 41 41 41 41 41 41
00001050 41 41 41 41 41 41 41 41 41 41 41
00001062 41 41 41 41 41 41 41 41 41 41 41
00001074 41 41 41 41 41 41 41 41 41 41 41
00001086 41 41 41 41 41 41 41 41 41 41 41
```

Signed 8 bit: 65
Unsigned 8 bit: 65
Signed 16 bit: 16705
Unsigned 16 bit: 16705
Signed 32 bit: 109479955
Unsigned 32 bit: 109479955
Float 32 bit: 12.07843
Float 64 bit: 2261634.5
□ Show little endian decoding
□ Show unsigned as h
Offset: 0x1044 / 0x1dd7

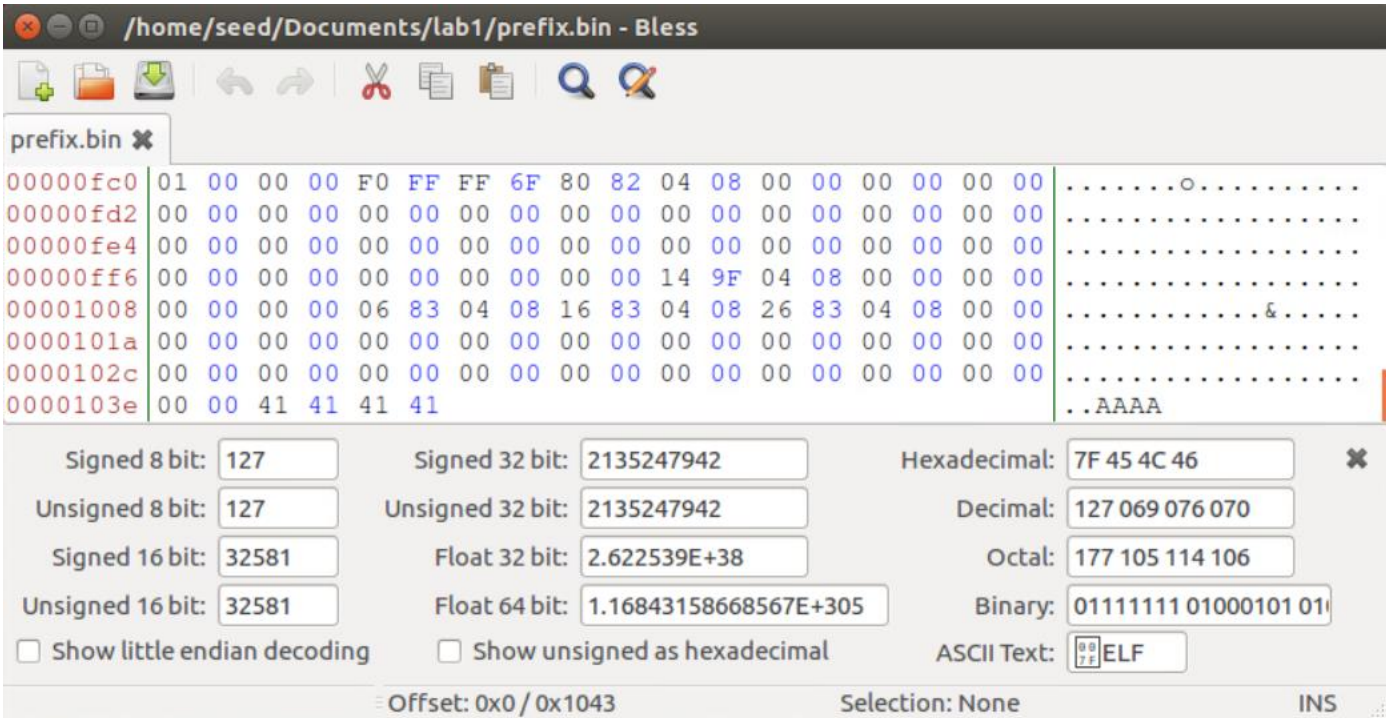
- (5 pts) Take a screenshot of the commands you used to cut and glue them.



```
[09/28/21]seed@VM:~/.../lab1$ head -c 4164 a.out > prefix.bin
[09/28/21]seed@VM:~/.../lab1$ tail -c +4165 a.out > suffixWhole.bin
[09/28/21]seed@VM:~/.../lab1$ head -c 128 suffixWhole.bin > middle.bin
[09/28/21]seed@VM:~/.../lab1$ tail -c +129 suffixWhole.bin > suffix.bin
[09/28/21]seed@VM:~/.../lab1$ cat prefix.bin middle.bin suffix.bin > originalTest.bin
[09/28/21]seed@VM:~/.../lab1$ diff a.out originalTest.bin
[09/28/21]seed@VM:~/.../lab1$ 
```

  - The first 4 commands split up the file, the last 2 check to make sure the first 4 worked right. In this case, there was no difference, so the commands worked.

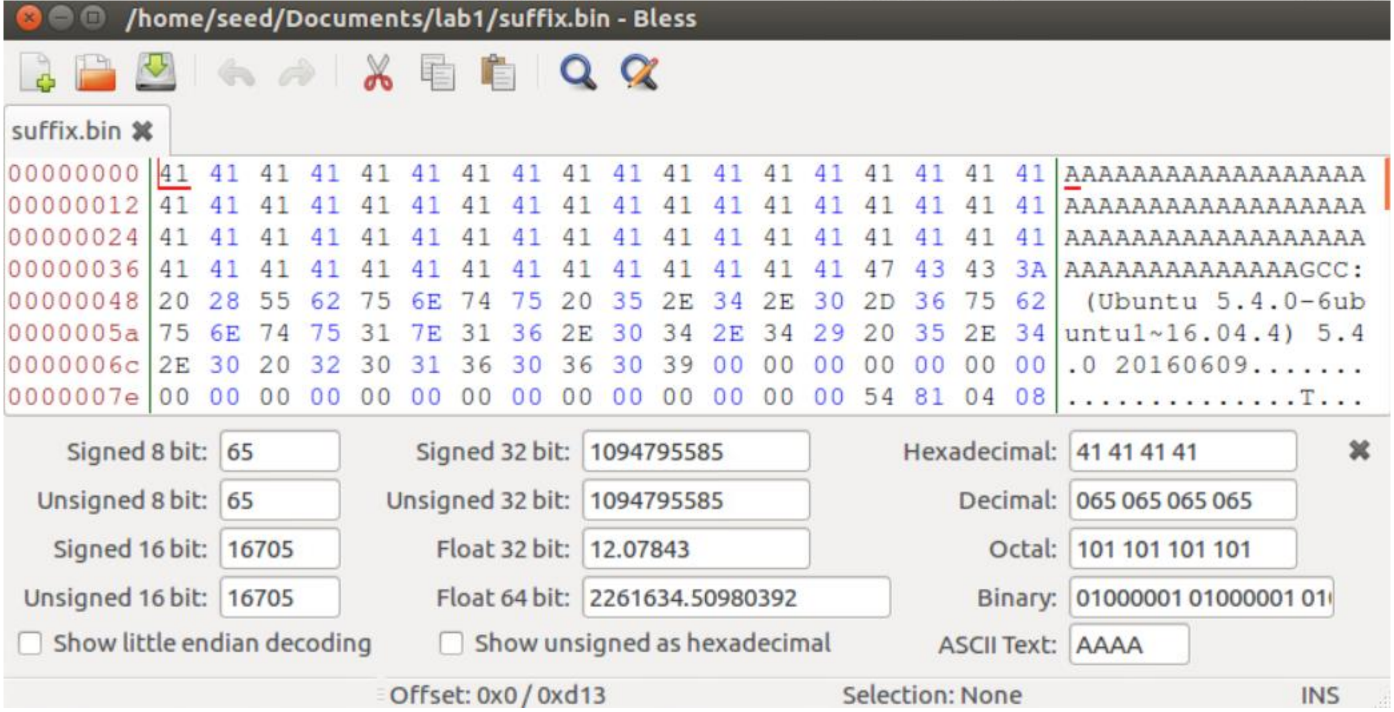
- (5 pts) After you have the final two files, take two screenshots to show the contents of each of them (run **`bless`** and just screenshot the part around the "128 bytes" which glued to the prefix and suffix, I don't need other part of the executable).



```
00000fc0 01 00 00 00 F0 FF FF 6F 80 82 04 08 00 00 00 00 00 00 .......o..........
00000fd2 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..................
00000fe4 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..................
00000ff6 00 00 00 00 00 00 00 00 00 00 14 9F 04 08 00 00 00 00 ..................
00001008 00 00 00 00 06 83 04 08 16 83 04 08 26 83 04 08 00 00 ............&.....
0000101a 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..................
0000102c 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..................
0000103e 00 00 41 41 41 41                                     ..AAAA
```

Signed 8 bit: 127 | Signed 32 bit: 2135247942 | Hexadecimal: 7F 45 4C 46
Unsigned 8 bit: 127 | Unsigned 32 bit: 2135247942 | Decimal: 127 069 076 070
Signed 16 bit: 32581 | Float 32 bit: 2.622539E+38 | Octal: 177 105 114 106
Unsigned 16 bit: 32581 | Float 64 bit: 1.16843158668567E+305 | Binary: 01111111 01000101 01
□ Show little endian decoding | □ Show unsigned as hexadecimal | ASCII Text: ELF

Offset: 0x0 / 0x1043 Selection: None INS



```
00000000 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAA
00000012 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAA
00000024 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAA
00000036 41 41 41 41 41 41 41 41 41 41 41 41 41 41 47 43 43 3A AAAAAAAAAAAAAAGCC:
00000048 20 28 55 62 75 6E 74 75 20 35 2E 34 2E 30 2D 36 75 62  (Ubuntu 5.4.0-6ub
0000005a 75 6E 74 75 31 7E 31 36 2E 30 34 2E 34 29 20 35 2E 34 untu1~16.04.4) 5.4
0000006c 2E 30 20 32 30 31 36 30 36 30 39 00 00 00 00 00 00 00 .0 20160609.......
0000007e 00 00 00 00 00 00 00 00 00 00 00 00 00 00 54 81 04 08 ..............T...
```

Signed 8 bit: 65 | Signed 32 bit: 1094795585 | Hexadecimal: 41 41 41 41
Unsigned 8 bit: 65 | Unsigned 32 bit: 1094795585 | Decimal: 065 065 065 065
Signed 16 bit: 16705 | Float 32 bit: 12.07843 | Octal: 101 101 101 101
Unsigned 16 bit: 16705 | Float 64 bit: 2261634.50980392 | Binary: 01000001 01000001 01
□ Show little endian decoding | □ Show unsigned as hexadecimal | ASCII Text: AAAA

Offset: 0x0 / 0xd13 Selection: None INS

(10 pts) Conclusion:

A summary of the lab and What have you learned in this lab?

- In this lab we learned why the MD5 hash function is not the most secure. I've learned that the MD5 function is iterative, so you can append data with the same hash to two files with the same hash, and their new hashes will still match. One thing that surprised me was how little you may have to change; for one of the in Task 2 only 2 characters where changed between the output binaries to create a file with the same hash.

Also answer these questions:

What is a one-way hash function?

- A one-way hash function takes in a file of variable length as an input then outputs a fixed-length seemingly random string of characters. Ideally, it will be very hard to find a hash collision, and it is impossible to calculate the original input from any given output.

What is a collision?

- A collision is when a file with different content hashes to the same string as another file. This is a problem for security, as you may *think* that the file has not been changed when in fact it has.