CSE 433S LAB1 - Secret Key

Zihan Chen

October 2024

Contents

1 Task1

Use md5collgen to generate two files.

```
[10/22/24]seed@VM:Steven$ echo "abcdefg" > prefix.txt
[10/22/24]seed@VM:Steven$ ./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: 50e4c772b0c9dd2be73ac4ca4c1d43a9

Generating first block: .......
Generating second block: W.................
Running time: 4.08442 s
```

Check the md5 value of the two files and difference between them. They are diff files but they have the same md5 value.

```
[10/22/24]seed@VM:Steven$ diff out1.bin  out2.bin
Binary files out1.bin and out2.bin differ
[10/22/24]seed@VM:Steven$ md5sum out1.bin
1edde10968ce2b80123b21015e191d17  out1.bin
[10/22/24]seed@VM:Steven$ md5sum out2.bin
1edde10968ce2b80123b21015e191d17  out2.bin
```

If length is not multiple of 64, it will add some padding to the end of the file.

For a 64 bytes, no need to add padding.

We can easily find the difference between 64 bytes and not multiple of 64 bytes.

```
[10/22/24]seed@VM:Steven$ hd fo1
00000000  61 61 61 61 61 61 61 61  61 61 61 61 61 61 61 61  |aaaaaaaaaaaaaaaa|
*
00000040  62 36 16 3c 2a 50 55 a0  ad 50 fb d5 4a 61 c5 7f  |b6.<*PU..P..Ja..|
00000050  b3 21 29 6d 28 10 a6 1b  0a a6 3a a6 12 06 2a af  |.!)m(.....:...*.|
00000060  09 fa 14 c2 fa 83 79 af  87 28 4a 5a b5 19 67 43  |......y..(JZ..gC|
00000070  5e 7c 50 ad 2a 73 7b dc  3e 04 d1 83 ac f6 29 50  |^|P.*s{.>.....)P|
00000080  96 33 c9 1f 1c bf eb 74  71 f1 45 9f 1d 2b 2d 3a  |.3.....tq.E..+-:|
00000090  ae 18 5a eb 06 76 a1 19  d1 57 f5 c4 55 62 59 22  |..Z..v...W..UbY"|
000000a0  3f 67 d7 d5 29 57 a3 02  79 64 c0 cf 7d 82 82 ec  |?g..)W..yd..}...|
000000b0  ae f8 e4 a3 d8 36 ce ef  ae e2 36 91 f6 bf 9c bb  |.....6....6.....|
000000c0
[10/22/24]seed@VM:Steven$ hd out1.bin
00000000  61 62 63 64 65 66 67 0a  00 00 00 00 00 00 00 00  |abcdefg.........|
00000010  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000040  31 f4 42 76 ea 63 e4 6a  fc 60 06 3a ec c1 89 af  |1.Bv.c.j.`.:....|
00000050  0e 8f 1d bb d7 fa dc 49  48 b2 7a 49 d9 d0 41 1b  |.......IH.zI..A.|
00000060  54 25 cb fc be 4a 3b 71  eb 21 49 c1 c3 11 7e 40  |T%...J;q.!I...~@|
00000070  18 77 b7 d5 3e 91 bc 50  bf de ad a1 f4 54 81 00  |.w..>..P.....T..|
00000080  0a da b9 61 b8 bc b4 a2  f3 d1 b6 bc 8e 7a aa dc  |...a.........z..|
00000090  f4 f4 78 c6 ba 1d d3 0c  0a 45 a9 06 cd d4 0d 9e  |..x......E......|
000000a0  95 c7 73 6e 38 4f 74 67  1a 34 fd 39 94 34 42 9d  |..sn8Otg.4.9.4B.|
```

We can easily find that most of the bytes are the same for 128 bytes.

```
00000040  31 f4 42 76 ea 63 e4 6a  fc 60 06 3a ec c1 89 af  |1.Bv.c.j.`.:....|
00000050  0e 8f 1d bb d7 fa dc 49  48 b2 7a 49 d9 d0 41 1b  |.......IH.zI..A.|
00000060  54 25 cb fc be 4a 3b 71  eb 21 49 c1 c3 11 7e 40  |T%...J;q.!I...~@|
00000070  18 77 b7 d5 3e 91 bc 50  bf de ad a1 f4 54 81 00  |.w..>..P.....T..|
00000080  0a da b9 61 b8 bc b4 a2  f3 d1 b6 bc 8e 7a aa dc  |...a.........z..|
00000090  f4 f4 78 c6 ba 1d d3 0c  0a 45 a9 06 cd d4 0d 9e  |..x......E......|
000000a0  95 c7 73 6e 38 4f 74 67  1a 34 fd 39 94 34 42 9d  |..sn8Otg.4.9.4B.|
000000b0  9f fc 1c 3f 79 d5 c3 83  79 24 3c 6f 85 07 8b 27  |...?y...y$<o...'|
000000c0
[10/22/24]seed@VM:Steven$ hd out2.bin
00000000  61 62 63 64 65 66 67 0a  00 00 00 00 00 00 00 00  |abcdefg.........|
00000010  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000040  31 f4 42 76 ea 63 e4 6a  fc 60 06 3a ec c1 89 af  |1.Bv.c.j.`.:....|
00000050  0e 8f 1d 3b d7 fa dc 49  48 b2 7a 49 d9 d0 41 1b  |...;...IH.zI..A.|
00000060  54 25 cb fc be 4a 3b 71  eb 21 49 c1 c3 91 7e 40  |T%...J;q.!I...~@|
00000070  18 77 b7 d5 3e 91 bc 50  bf de ad 21 f4 54 81 00  |.w..>..P...!.T..|
00000080  0a da b9 61 b8 bc b4 a2  f3 d1 b6 bc 8e 7a aa dc  |...a.........z..|
00000090  f4 f4 78 46 ba 1d d3 0c  0a 45 a9 06 cd d4 0d 9e  |..xF.....E......|
000000a0  95 c7 73 6e 38 4f 74 67  1a 34 fd 39 94 b4 41 9d  |..sn8Otg.4.9..A.|
000000b0  9f fc 1c 3f 79 d5 c3 83  79 24 3c ef 85 07 8b 27  |...?y...y$<....'|
```

Only little difference we can find.

```
[10/22/24]seed@VM:Steven$ diff hex1 hex2
5,7c5,7
< 0000050 8f0e bb1d fad7 49dc b248 497a d0d9 1b41
< 0000060 2554 fccb 4abe 713b 21eb c149 11c3 407e
< 0000070 7718 d5b7 913e 50bc debf a1ad 54f4 0081
---
> 0000050 8f0e 3b1d fad7 49dc b248 497a d0d9 1b41
> 0000060 2554 fccb 4abe 713b 21eb c149 91c3 407e
> 0000070 7718 d5b7 913e 50bc debf 21ad 54f4 0081
9,11c9,11
< 0000090 f4f4 c678 1dba 0cd3 450a 06a9 d4cd 9e0d
< 00000a0 c795 6e73 4f38 6774 341a 39fd 3494 9d42
< 00000b0 fc9f 3f1c d579 83c3 2479 6f3c 0785 278b
---
> 0000090 f4f4 4678 1dba 0cd3 450a 06a9 d4cd 9e0d
> 00000a0 c795 6e73 4f38 6774 341a 39fd b494 9d41
> 00000b0 fc9f 3f1c d579 83c3 2479 ef3c 0785 278b
```

2 Task2

out1.bin and out2.bin are the two files generated by md5collgen. They have the same md5 value. Now I generate a new file called test by perl -e 'print "b"x100;' ¿ test.

Then I combine test with out1.bin and out2.bin and generate test1 and test2. I can find that test1 and test2 have the same md5 value.

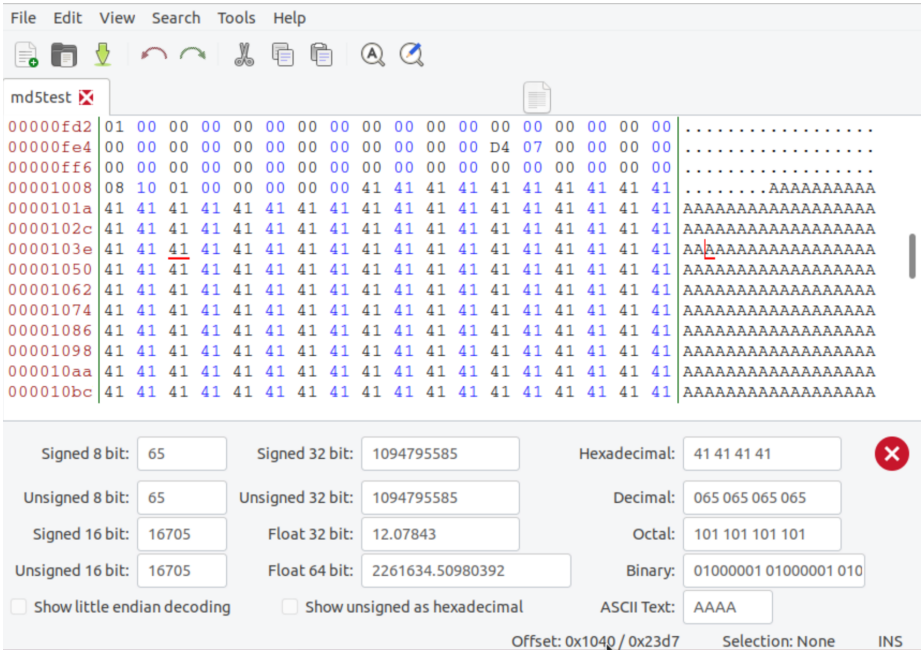
```
[10/23/24]seed@VM:Steven$ md5sum out1.bin
1edde10968ce2b80123b21015e191d17  out1.bin
[10/23/24]seed@VM:Steven$ md5sum out2.bin
1edde10968ce2b80123b21015e191d17  out2.bin
[10/23/24]seed@VM:Steven$ perl -e 'print "b"x100;' > test
[10/23/24]seed@VM:Steven$ md5sum test
d84a935724eac27d7c9676679b6cdbaf  test
[10/23/24]seed@VM:Steven$ cat out1.bin test > test1
[10/23/24]seed@VM:Steven$ cat out2.bin test > test2
[10/23/24]seed@VM:Steven$ md5sum test1
2039f9ee2826621841decb5f037d0309  test1
[10/23/24]seed@VM:Steven$ md5sum test2
2039f9ee2826621841decb5f037d0309  test2
```

3 Task3

Write a program with source code as follows.

```
#include <stdio.h>
unsigned char xyz[200] = {
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
};
int main()
{
int i;
for (i=0; i<200; i++){
printf("%x", xyz[i]);
}
printf("\n");
}
```

The ”A” is that can divide prefix and 128bytes A is at 0x1040 which is the 4160th byte.



So we cat the prefix and the suffix. And use md5collgen to generate two files with same md5 value from the prefix. Then we combine the prefix and the suffix to generate the final file.

```
[10/23/24]seed@VM:Steven$ head -c 4160 md5test > prefix
[10/23/24]seed@VM:Steven$ tail -c +4288 md5test > suffix
[10/23/24]seed@VM:Steven$ ./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: d507fe734a37c6860a719b794d4d721b

Generating first block: ..................
Generating second block: W......
Running time: 14.5613 s
[10/23/24]seed@VM:Steven$ cat p1 suffix > m1
[10/23/24]seed@VM:Steven$ cat p2 suffix > m2
[10/23/24]seed@VM:Steven$ md5sum m1
841e7d525d010168be222413640ad34c  m1
[10/23/24]seed@VM:Steven$ md5sum m2
841e7d525d010168be222413640ad34c  m2
```

4 Task4

Write a benign program.

```c
#include <stdio.h>
#define PADDING 256

unsigned char X[PADDING] = {
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
};

unsigned char Y[PADDING] = {
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
};

int main() {
    int i;
    int ma = 0;

    for(i = 0; i < PADDING; i++) {
      if(X[i] != Y[i]) {
        printf("Malicious Code!\n");
        ma = 1;
        break;
      }
    }

    if(ma == 0) {
      printf("Benign Code!\n");
    }
    return 0;
}
```

Compile the program and get the prefix and suffix of the program.

```
[10/23/24]seed@VM:Steven$ head -c 4160 benign > prefix
[10/23/24]seed@VM:Steven$ tail -c +4288 benign > suffix
```

Use md5collgen to generate two files p1 and p2 with same md5 value from the prefix.

```
[10/23/24]seed@VM:Steven$ ./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: 7067ec14f4aa949e339379271a4f8e6d

Generating first block: ..............................
Generating second block: W.......
Running time: 17.1324 s
```

Get the prefix and suffix of the previous suffix.

Get the P from generated file p1.

Generate o1 by combining the prefix, prefix of the suffix, P, suffix of the suffix.

Generate o2 by combining the prefix, prefix of the suffix, P, suffix of the suffix.

Check the md5 value of o1 and o2, they are the same.

```
[10/23/24]seed@VM:Steven$ head -c 128 suffix > presuf
[10/23/24]seed@VM:Steven$ tail -c +256 suffix > sufsuf
[10/23/24]seed@VM:Steven$ tail -c 128 p1 > P
[10/23/24]seed@VM:Steven$ cat p1 presuf P sufsuf > o1
[10/23/24]seed@VM:Steven$ cat p2 presuf p sufsuf > o2
cat: p: No such file or directory
[10/23/24]seed@VM:Steven$ cat p2 presuf P sufsuf > o2
[10/23/24]seed@VM:Steven$ md5sum o1
85874c42cfb63d30c846642b71666d1b  o1
[10/23/24]seed@VM:Steven$ md5sum o2
85874c42cfb63d30c846642b71666d1b  o2
```

Run o1 and o2, then the program have different output.

```
[10/23/24]seed@VM:Steven$ chmod 777 o1
[10/23/24]seed@VM:Steven$ chmod 777 o2
[10/23/24]seed@VM:Steven$ ./o1
Benign Code!
[10/23/24]seed@VM:Steven$ ./o2
Malicious Code!
```