

One-Way Hash Function and MAC

Sareh Jalalizad

2.1 Generating Message Digest and MAC

Q1:

For the first task, we should try out three different one-way hash algorithms using the openssl dgst command to generate the hash value for a file. I generated message digests using my email address as a text file (sarehj@kth.se).

The difference between the digests generated by the algorithms MD5, SHA1, and SHA256 seems to be the length of them. For MD5, got a digest consisting of 32 hexadecimal signs. SHA1 is represented by 40 and SHA256 by 64 hexadecimal signs. The algorithm is more secure the longer the generated hash value is.

MD5: 128 bits (32*4)

SHA1: 160 bits (40*4)

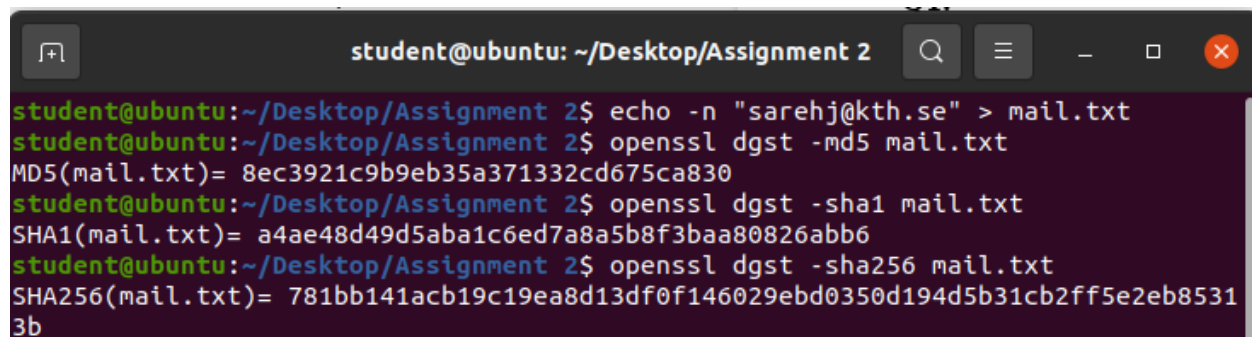
SHA256: 256 bits (64*4)

Q2:

MD5= 8ec3921c9b9eb35a371332cd675ca830

SHA1= a4ae48d49d5aba1c6ed7a8a5b8f3baa80826abb6

SHA256= 781bb141acb19c19ea8d13df0f146029ebd0350d194d5b31cb2ff5e2eb85313b



```
student@ubuntu: ~/Desktop/Assignment 2
student@ubuntu:~/Desktop/Assignment 2$ echo -n "sarehj@kth.se" > mail.txt
student@ubuntu:~/Desktop/Assignment 2$ openssl dgst -md5 mail.txt
MD5(mail.txt)= 8ec3921c9b9eb35a371332cd675ca830
student@ubuntu:~/Desktop/Assignment 2$ openssl dgst -sha1 mail.txt
SHA1(mail.txt)= a4ae48d49d5aba1c6ed7a8a5b8f3baa80826abb6
student@ubuntu:~/Desktop/Assignment 2$ openssl dgst -sha256 mail.txt
SHA256(mail.txt)= 781bb141acb19c19ea8d13df0f146029ebd0350d194d5b31cb2ff5e2eb85313b
```

2.2 Keyed Hash and HMAC

Q3:

For this task, we tested out the three different algorithms for the file created in Section 2.1, to generate a keyed hash for the file using the HMAC with the different keys of different lengths. I have tested with several keys of different sizes and it generates a result and a new hash value every time. As a result, the size does not affect the result and we do not need to use a key with a fixed size in HMAC. However, if the key length is shorter than the message, padding will be used in order to make it work, and if the key is too long it will be shortened.

Q4:

HMAC-MD5= 777e03e1cdd5f696ae9410166b848288

HMAC-SHA1= 0b21e33037077eb5645b96afc3bcd0f0ad08d8

HMAC-SHA256= 93d04e7c5489a9f0e479cf84952859aad0692b5ce4aa891636463650dd4f51ac

```
student@ubuntu:~/Desktop/Assignment 2$ openssl dgst -md5 -hmac "IV1013-key" mail.txt
HMAC-MD5(mail.txt)= 777e03e1cdd5f696ae9410166b848288
student@ubuntu:~/Desktop/Assignment 2$ openssl dgst -sha1 -hmac "IV1013-key" mail.txt
HMAC-SHA1(mail.txt)= 0b21e33037077eb5645b96afc3bcd0f0ad08d8
student@ubuntu:~/Desktop/Assignment 2$ openssl dgst -sha256 -hmac "IV1013-key" mail.txt
HMAC-SHA256(mail.txt)= 93d04e7c5489a9f0e479cf84952859aad0692b5ce4aa891636463650dd4f51ac
student@ubuntu:~/Desktop/Assignment 2$
```

2.3 The Randomness of One-way Hash

Q5:

First, I generated the hash value H1 for the file created in Section 2.1 then flipped the first bit of the file by using ghex binary editor in Linux and generated the hash value H2 for the modified file by the MD5 and SHA256 algorithms.

H1: MD5= 8ec3921c9b9eb35a371332cd675ca830

H2: MD5= 8a18d6da56cecf677a1f8452174467b2

H1: SHA256=781bb141acb19c19ea8d13df0f146029ebd0350d194d5b31cb2ff5e2eb85313b

H2: SHA256=896fc0ea300f5ef7dd450f7fa67867688e90ac874a5a549077cbf3c75fa38b40

```

student@ubuntu:~/Desktop/Assignment 2$ openssl dgst -md5 mail.txt
MD5(mail.txt)= 8ec3921c9b9eb35a371332cd675ca830
student@ubuntu:~/Desktop/Assignment 2$ openssl dgst -md5 ma1.txt
MD5(ma1.txt)= 8a18d6da56cecf677a1f8452174467b2

student@ubuntu:~/Desktop/Assignment 2$ openssl dgst -sha256 mail.txt
SHA256(mail.txt)= 781bb141acb19c19ea8d13df0f146029ebd0350d194d5b31cb2ff5e2eb85313b
student@ubuntu:~/Desktop/Assignment 2$ openssl dgst -sha256 ma1.txt
SHA256(ma1.txt)= 896fc0ea300f5ef7dd450f7fa67867688e90ac874a5a549077cbf3c75fa38b40
student@ubuntu:~/Desktop/Assignment 2$

```

I wrote a short java program that counts the same bits. The result given from the program was:

MD5: 68 similar bits

SHA-256: 134 similar bits

```

student@ubuntu:~/Desktop/Assignment 2$ javac CountSB.java
student@ubuntu:~/Desktop/Assignment 2$ java CountSB
MD5: 68
SHA256: 134

```

So, by analyzing the percentage of resembling the hashed values each other, I got approximately 53 percent for MD5 ($68/128 = 0.531$) and approximately 52 percent for SHA-256 ($134/256 = 0.523$) which means only about half of the bits are similar. Because each bit can only take two different values (0 or 1), so for every random digest around 50% of the bits will be similar to the original digest.

2.4 Collision Resistance

- IV1013 security 28872840
- Security is fun 10015372
- Yes, indeed 9777773
- Secure IV1013 23746118
- No way 1139565

