One-way Hash Functions

Lab report

Patrik Zhong

Question 1. Describe your observations. What differences do you see between the algorithms?

The most obvious observation is that they differ in their hash lengths. We see that MD5 has a word length of 32. SHA1 has a length of 40 and SHA256 has a length of 64. Since the hash is given in hexadecimal and a hexadecimal symbol is a nibble long, we multiply it with 4 to get their bitlength. MD5 has a bitlength of 128, SHA1 has a bitlength of 160 and SHA256 has a bitlength of 256.

Question 2. Write down the digests generated using the three algorithms.

Running the commands for every hash algorithm outputs the following:

MD5(lab2)= 0d9a676a2a4b9322036097877b7747f7

SHA1(lab2)= a4a971a8e9c7cb1c9256a06e27f1356c1bbd0f59

SHA256(lab2)=cee965ae715ad545e4290c4926188b5e7093e578e446fdf770adbf1a522304ae

Question 3. Do we have to use a key with a fixed size in HMAC? If so, what is the key size? If not, why?

When creating the keyed hash we use 3 types of strings.

- “abcdefg” with a length of 7.
- “abcdef” with a length of 6.
- “abcde” with a length of 5.

To answer the question if we have to use a key with a fixed size, the answer would simply be that we do not. The hmac can handle keys with different lengths. The way it does this is that if the key is too short relative to the text, it simply pads it out. If its too long its hashed down and then padded additionally if needed.

The following output is presented below:

MD5

HMAC-MD5(lab2)= 6b5840bff829840e65fde17d030e08fc
HMAC-MD5(lab2)= de7265886f868ef52e2887c94425ef3b
HMAC-MD5(lab2)= 4ec9af7873c298c237c016d45fab0e98

Sha256

HMAC-SHA256(lab2)=
1b1c08783829f6ced104681c78c5f15c92ae17e1f987067fc9ed39bce0dcaa5b

HMAC-SHA256(lab2)=
0dec5ecdfab6284263b1554b11d7e43a5e21bb68cbace22bced40014edc888c2

HMAC-SHA256(lab2)=
7cb5a9a05a27986ee41c666b48278f985a7467f118736c0f59db0a65a9818adf

Sha-1

HMAC-SHA1(lab2)= 4c41b7221b1d7b982a5156ce0ab5e96edeb6d498

HMAC-SHA1(lab2)= 17ade598c6f63cff43fd628a9cdb11e9cc97e96d

HMAC-SHA1(lab2)= 61e3bb0882fe6a1a7bad7d32a7ee9c5245e5d6c0

Question 4. Now use the string IV1013-key as the secret key and write down the keyed hashes generated using the three algorithms.

HMAC-SHA1(lab2)= 302c8551da2e39c51490b2cacd1519af5f6e0344

HMAC-SHA256(lab2)=
4b4ee407a025d0d0613cbbca79eb22cd9286f1b924565fdbbeaa6573d91fcb83

HMAC-MD5(lab2)= 43a426499407c3e4b4655c068980631d

Question 5. Describe your observations. Count how many bits are the same between H1 and H2 fo MD5 and SHA256. Specify how many are the same.

The content of the file to be encrypted is my mail “pzhong@kth.se”
The modifications with changing the first bit changes it into “.zhong@kth.se”

SHA256

Table 5.1 Overview of SHA256 results

	SHA256 Hash	Binary
Modified File	ee40eb0cd837842bf3785aadd655ec6796fcf4faf9aa294fa714517bda00bec2	1110111001000000011101011000011001101100000110111100001000010101111110011011110000101101010101101110101100101010111101100011001111001011011111100111101001111101011111001101010100010100101001111101001110001010001010001011110111101101000000000101111101100001 0
Unmodified File	cee965ae715ad545e4290c4926188b5e7093e578e446fdf770adbf1a522304ae	11001110111010010110010110101110011100010101101011010101010001011110010000010100100001100010010010010011000011000100010110101011110011100001001001111100101011110001110010001000110111111011111011100001010110110111110001101001010010001000110000010010101110

Using a very simple java program that simply compares each digit and if they’re the same, we come to the conclusion that 129/256 bits is the same, which is around half.

MD5

	MD5 Hash	Binary
Modified File	19c2e3108a1e768e281be090b3e9b90e	00011001110000101110001100010000100010101000011110011101101000111000101000000110111110000010010000101100111110100110111100100001110
Unmodified File	0d9a676a2a4b9322036097877b7747f7	00001101100110100110011101101010000101010010010111001001100100010000000011011000001001011110000111011011101101110111010001111111110111

Using the same java program we find that 60/128 bits are the same, which is around half of them.

I find it interesting that just changing one bit creates such wildly different hashes. The coursebook (and wikipedia) explains this through some kind of “butterfly effect”, where the encrypting algorithms make so operations on the plaintext that it basically becomes something wildly different in the end. I also noted that around half of the bits were the same, and I figure this is pure chance. The way I see it, each bit can be either a 0 and 1, making it a 50/50 whether they will share the same bits. It seems to be pure chance.

“