Alex Lindström

One Way Hash Lab

Task 2.1 Generating message digest and MAC

1. **Describe your observations, what differences do you see between the algorithms?**

   For this first task I tried out three different one-way hash algorithms using the *openssl dgst* command. The different algorithms are MD5, SHA1 and SHA256. Using these three I generated message digests using my email address as text (alexli@kth.se).

   For MD5 I got a digest consisting of 32 hexadecimal signs which all represents 4 bits each meaning it is a value consisting of 32*4, 128 bits. SHA1 is represented by 40 (160 bits) and SHA256 by 64 (256 bits) hexadecimal signs. The longer the generated hash value, the more secure the algorithm is.

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

   MD5: aaf4d2d0e824f0ea8005054bc7f01e84

   SHA1: 9c18fb71896fefe59ec5408f0cd929f9790e0bd7

   SHA256: 9e3e66fba5d87bc9290a07d199bbe40433e637e40855c7bcbe69c8cd4ebf915e

Task 2.2 Keyed Hash and HMAC

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

   By testing out the three different algorithms with the text alexli@kth.se, but this time using HMAC combined with the following different passwords:

   - **1**
   - **1234**
   - **12345678910**
   - **012391212191249123123412343**

   I can see that it can generate hashes of the same length with keys of different length. So therefore, I can safely say that we do not need to enter keys of a fixed size using HMAC. This is because HMAC helps us with that by padding the key to the necessary internal blocksize.

2. **Now use the string IV-1013-key as the secret key and write down the keyed hashes generated using the three algorithms.**

   When using the text alexli@kth.se and the IV1013-key as key I receive the following hashes using the three different algorithms MD5, SHA1 and SHA256:

   HMAC-MD5 = 51de6b5e668403925c327d9952206849

   HMAC-SHA1 = 25f98ac9093aa6c2d61c3edd7bcbecaabb710127

   HMAC-SHA256 = 8b2056fa89d37331b551a1cbce97efb073f95daaabde3e3f0de93f83f89b7191

Alex Lindström

Task 2.3 The Randomness of One-Way Hash

1. **Describe your observations. Count how many bits are the same between H_1 and H_2 for MD5 and SHA256 (writing a short program to count the same bits might help you). In the report, specify how many bits are the same.**

For this task I took my original text from task 2.1 but I swapped the first bit of the first character in the text using the ghex tool in linux. I then compared the old text with the new one with the changed bit and compared the different hashes generated by the MD5 and SHA256 algorithms. From just looking at the two different hashed values:

(MD5 with bit changed)   07cf6ef674fe522f97920e574315d8fe
(MD5 with bit unchanged) aaf4d2d0e824f0ea8005054bc7f01e84

(SHA256 with bit changed)
a343151c6ace567135859039edfa327dc87167b269188c14a531985910449cde

(SHA256 with bit unchanged)
9e3e66fba5d87bc9290a07d199bbe40433e637e40855c7bcbe69c8cd4ebf915e

I can't say that I see any real resemblance between the two hashed values despite only changing the one bit in the original text alexli@kth.se. The outcome of changing the one bit resulted in the text álexli@kth.se.

By comparing the hashed values bit for bit using my java code which can be found in the same zipped folder as this document, I received a total of 63 bits being the same for MD5. For SHA256 I received a total of 124 bits being the same.

By analyzing the percentile in which the hashed values resemble one another we get 63/128 (0.4921875, approx. 49%) and 124/256 (0,484375, approx. 48%). And since we are dealing with bits which can only take 2 different values, I would say a result of approximately 50% is really good in terms of randomness.