Assignment 4: Cryptographic Hash Functions

Deadline: 11:59 pm Thursday, 15th October 2015

Reading: 2.3 Cryptographic Hash Functions

We will be using the same VM as in the previous assignment.
Steps:
1) Download http://www.cis.syr.edu/~wedu/seed/Labs_12.04/Crypto/Crypto_Hash/Crypto_Hash.pdf
2) Perform Tasks 1, 2 and 3
3) Perform Task 4 below

Task 4:
In this task, we will investigate the difference between two important properties of hash functions: the one-way property versus the collision-free property. We will use the brute-force method to see how long it takes to break each of these properties. Instead of using openssl's command-line tools, you are required to write our own C programs to invoke the message digest functions in openssl's crypto library. A sample code can be found here: http://www.openssl.org/docs/crypto/EVP_DigestInit.html. It has also been reproduced below (sample.c). Please get familiar with this sample code. Since most of the hash functions are quite strong against the brute-force attack on those two properties, it would take us years to break them using the brute-force method. To make the task feasible, we reduce the length of the hash value to 24 bits. We can use any one-way hash function, but we only use the first 24 bits of the hash value in this task. That is, we are using a modified, weaker, one-way hash function.

Before beginning the next task please check the sample code given at the end.

4.1) Explain the one-way and collision-free properties of hash functions.

4.2) In this task you will be breaking the one-way property using the brute-force method. Find an input which would give same first 24 bits in MD5 hash as "Hello world". You are required to write your own C program for this task. Submit it as GradeId_FirstName_LastName_Task4_2.c. Report your observations and attach screenshots.

4.3) In this task you will be breaking the collision-free property using the brute-force method. Generate two random numbers and compute their first 24 bits of MD5 hash, repeat until both gives same first 24 bits. You should do this 10 times and report the average number of trials it took to get same first 24 bits. You are required to write your own C program for this task. Submit it as GradeId_FirstName_LastName_Task4_3.c. Report your observations and attach screenshots.

4.4) Based on your findings, which of the two properties do you think is easiest to break?

Sample code: sample.c

```c
#include <stdio.h>
#include<string.h>
#include <openssl/evp.h>

#include <stdio.h>
#include<string.h>
#include <openssl/evp.h>

main(int argc, char *argv[])
{
    EVP_MD_CTX *mdctx;
    const EVP_MD *md;
    char mess1[] ="so code many debug much logic very program wow";
    unsigned char md_value[EVP_MAX_MD_SIZE];
    int md_len, i;

    OpenSSL_add_all_digests();

    if(!argv[1]) {
        printf("Usage: mdtest digestname\n");
        exit(1);
     }

     md = EVP_get_digestbyname(argv[1]) ;

    if(!md) {
        printf("Unknown message digest %s\n", argv[1]);
        exit(1);
     }

     mdctx = EVP_MD_CTX_create();
    EVP_DigestInit_ex(mdctx, md, NULL);
    EVP_DigestUpdate(mdctx, mess1, strlen(mess1));
    EVP_DigestFinal_ex(mdctx, md_value, &md_len);
    EVP_MD_CTX_destroy(mdctx);

    printf("Input: %s\n",mess1);
    printf("Digest: ");
    for(i = 0; i < md_len; i++)
    printf("%02x", md_value[i]);
    printf("\n");

    /* Call this once before exit. */
    EVP_cleanup();
    exit(0);
}
```

To compile and run:

```
gcc -I/home/seed/openssl-1.0.1/include -o sample sample.c -
L/home/seed/openssl-1.0.1/ -lcrypto -ldl && ./sample md5
```

md5