ECE 471 Lab 3

MD5 Collision Attack Lab

Mitchell Dzurick

3/9/2020

Github with all documentation - `https://www.github.com/mitchdz/ECE471`

Contents

1 Overview **2**

2 Lab Tasks **3**

3 Task 1: Generating Two Different Files with the Same MD5 Hash **3**

3.1 Task 1 Solution 4

3.1.1 Task 1 Question 1 Solution 4

3.1.2 Task 1 Question 2 Solution 4

3.1.3 Task 1 Question 3 Solution 4

3.2 Task 2: Understanding MD5's Property 7

3.2.1 Task 2: Solution 7

Secret Key Encryption Lab

1 Overview

A secure one-way hash function needs to satisfy two properties: the one-way property and the collision- resistance property. The one-way property ensures that given a hash value h, it is computationally infeasible to find an input M , such that hash(M) = h. The collision-resistance property ensures that it is compu- tationally infeasible to find two different inputs M 1 and M 2 , such that hash(M 1) = hash(M 2). Several widely-used one-way hash functions have trouble maintaining the collision-resistance prop- erty. At the rump session of CRYPTO 2004, Xiaoyun Wang and co-authors demonstrated a collision attack against MD5 [1]. In February 2017, CWI Amsterdam and Google Research announced the SHAttered attack, which breaks the collision-resistance property of SHA-1 [3]. While many students do not have trouble understanding the importance of the one-way property, they cannot easily grasp why the collision-resistance property is necessary, and what impact these attacks can cause. The learning objective of this lab is for students to really understand the impact of collision attacks, and see in first hand what damages can be caused if a widely-used one-way hash function's collision-resistance property is broken. To achieve this goal, students need to launch actual collision attacks against the MD5 hash function. Using the attacks, students should be able to create two different programs that share the same MD5 hash but have completely different behaviors. This lab covers a number of topics described in the following:

- One-way hash function
- The collision-resistance property
- Collision attacks
- MD5

Lab Environment. This lab has been tested on our pre-built Ubuntu 12.04 VM and Ubuntu 16.04 VM, both of which can be downloaded from the SEED website.

2 Lab Tasks

3 Task 1: Generating Two Different Files with the Same MD5 Hash

In this task, we will generate two different files with the same MD5 hash values. The beginning parts of these two files need to be the same, i.e., they share the same prefix. We can achieve this using the md5collgen program, which allows us to provide a prefix file with any arbitrary content. The way how the program works is illustrated in Figure 1. The following command generates two output files, out1.bin and out2.bin, for a given a prefix file *prefix.txt*:

```
$ md5collgen -p prefix.txt -o out1.bin out2.bin
```

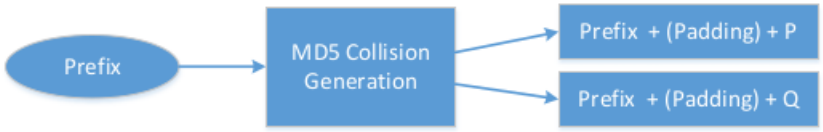

Figure 1: Running the key generation program without srand multiple times

We can check whether the output files are distinct or not using the diff command. We can also use the md5sum command to check the MD5 hash of each output file. See the following commands.

```
$ diff out1.bin out2.bin
$ md5sum out1.bin
$ md5sum out2.bin
```

Since out1.bin and out2.bin are binary, we cannot view them using a text-viewer program, such as cat or more; we need to use a binary editor to view (and edit) them. We have already installed a hex editor software called bless in our VM. Please use such an editor to view these two output files, and describe your observations. In addition, you should answer the following questions:

- Question 1. If the length of your prefix file is not multiple of 64, what is going to happen?
- Question 2. Create a prefix file with exactly 64 bytes, and run the collision tool again, and see what happens.
- Question 3. Are the data (128 bytes) generated by md5collgen completely different for the two output files? Please identify all the bytes that are different.

3.1 Task 1 Solution

3.1.1 Task 1 Question 1 Solution

the prefix was padded with zero bytes until the size is a multiple of 64.

3.1.2 Task 1 Question 2 Solution

With 64-byte prefix, no bytes are needed for padding. The prefix then has extra data added on for the collision.

3.1.3 Task 1 Question 3 Solution

The md5collgen program is used for the first time in this task, so some initial observations are made.

```
[02/26/20]seed@VM:~/.../part0$ 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: bc7a2c389bfa80dae23108b4ad86f79a

Generating first block: ........................................
Generating second block: S11.........................
Running time: 26.1328 s
[02/26/20]seed@VM:~/.../part0$ xxd out1.bin
00000000: 4865 6c6c 6f20 576f 726c 6421 0a00 0000  Hello World!....
00000010: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000020: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000030: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000040: 9c99 badf 739c 9392 922a a47f d324 1a43  ....s....*...$.C
00000050: 5344 bc2d b0ee 1fa8 30d2 39d5 b415 b39d  SD.-....0.9.....
00000060: 4d09 efe9 f0cb 39e9 4fc6 c1f9 a973 2f3a  M.....9.O....s/:
00000070: c921 75c6 13af ce4d 593f b97e 7435 14bc  .!u....MY?.~t5..
00000080: a948 b920 9926 fb22 7af3 d132 b160 b2d0  .H. .&."z..2.`..
00000090: 7f6f 706c 33f8 703e 6e79 a3cb 4b61 cf41  .opl3.p>ny..Ka.A
000000a0: a9e0 b06d 0689 7a57 199d 6f59 6d17 f268  ...m..zW..oYm..h
000000b0: a17c a243 be52 6d66 15c2 4ecb ae01 3309  .|.C.Rmf..N...3.
```

Figure 2: Running the key generation program without srand multiple times

Figure 2 shows the results of md5collgen being ran on a file called prefix.txt that has the

contents ‘Hello World¡ and the prefix is seen to be padded to 64 bytes with bytes of 0 bits.

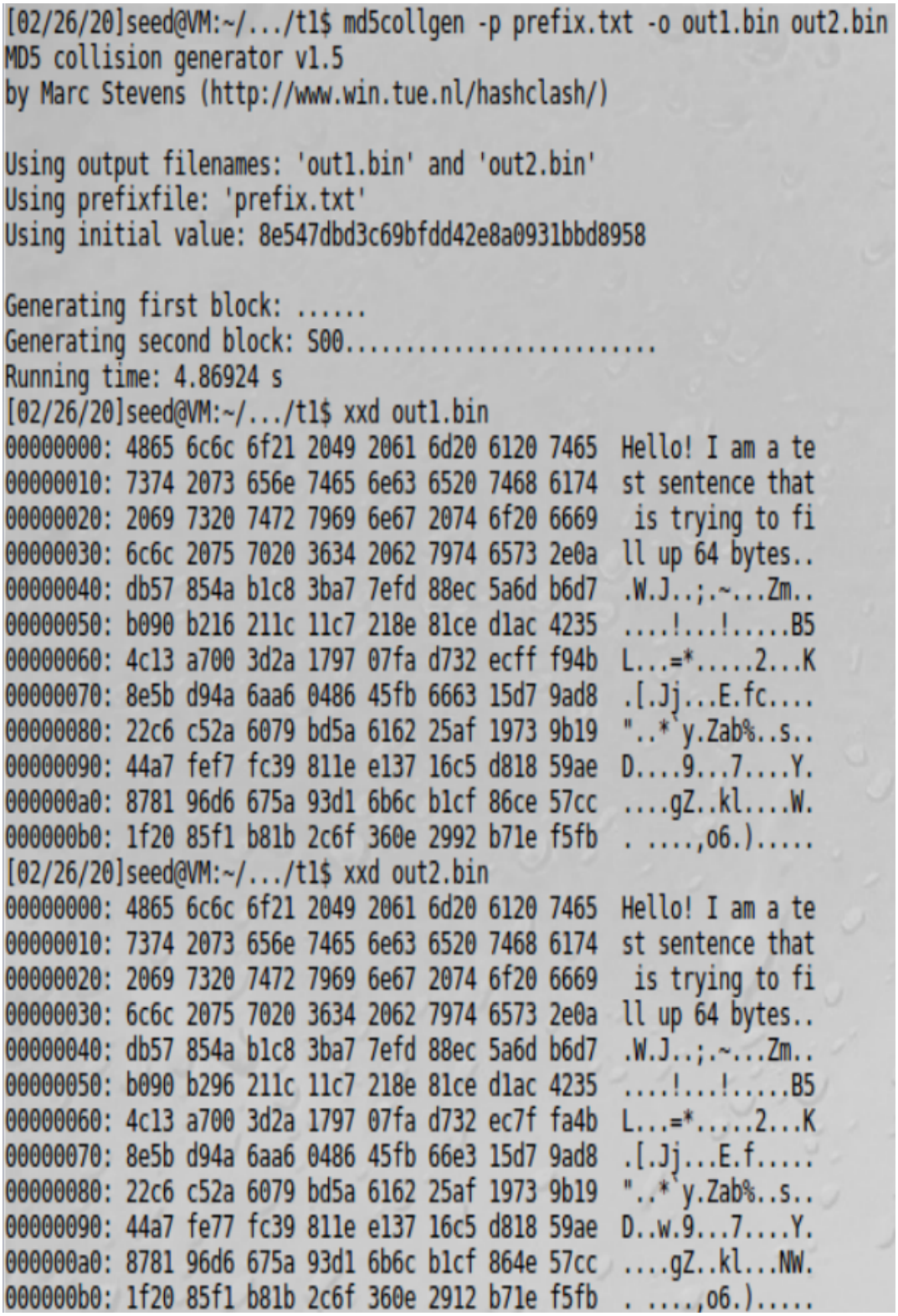

Figure 3: 64 bytes of prefix being put into the md5collgen

Figure 3 shows that the files don't completely differ, but they do differ in certain areas. It's

actually apparent that only 8 bytes are different out of the 192 bytes that are output. The bytes that changed only changed a little bit as well. They only differ in one bit.

3.2 Task 2: Understanding MD5's Property

3.2.1 Task 2: Solution