

Patrik Kozak, Gabe Hendrickson
Professor Wang
CIS 458 - 01
12 September 2019

Lab 2

Question 01: Give at least three hash functions supported by the openssl dgst.

1. **md5**
2. **sha1**
3. **sha256**

Question 02: What are the hash functions you used to generate hash values. Did they work as expected?

1. **The hash functions I used to generate hash values include md5, sha1, and sha256. Yes, they worked as expected.**
2. **The commands I used to generate the hash values are as follows:**
 - **openssl dgst -md5 -hmac "MyPassword" test1.txt**
 - **openssl dgst -sha1 -hmac "GrandVstudent" test1.txt**
 - **openssl dgst -sha256 -hmac "CIS458" test1.txt**

Question 03: What are the hash functions you used to generate the keyed hash values? What is/are the key(s) you used?

1. **The first hash function I used was md5.**
 - **The key I used for this function is "MyPassword"**
2. **The second hash function I used was sha1.**
 - **The key I used for this function is "GrandVstudent"**
3. **The third hash function I used was sha256.**
 - **The key I used for this hash function is "CIS458"**

Question 04: Please describe your observations briefly. Please attach a screenshot to support your observations.

Any slight alteration in the file will result in a completely different hash value.

```
ashley@ashley-Aspire-E5-521:~/gvsu/cis458/lab/2$ openssl dgst -md5 lab02_hash.docx
MD5(lab02_hash.docx)= e2521adc1f7df564113075e97c2a422d
ashley@ashley-Aspire-E5-521:~/gvsu/cis458/lab/2$ openssl dgst -md5 lab02_hash_final.docx
MD5(lab02_hash_final.docx)= 78711dfc3c28928e545c02c21f8c99b5
```

Question 05: Attach your script of the sender's part.

The file is the original lab

```
1 #!/bin/bash
2 openssl dgst -md5 -hmac $1 $2 > .authenticate
3 echo $1 >> .authenticate
4
```

Question 06: Attach your script of the receiver's part.

```
1 #!/bin/bash
2
3 # hashfunction  hashvalue the message and the password
4
5 key=$(sed -n '2p' .authenticate)
6 openssl dgst -md5 -hmac $key $1 >> .authenticate
7
8
9 thereHash=$( sed -n '1p' .authenticate | cut -d' ' -f2 )
10 yourHash=$(sed -n '3p' .authenticate | cut -d' ' -f2)
11
12
13 echo comparing the following two hashes
14 echo $thereHash
15 echo $yourHash
16
17 sed -i '3d' .authenticate
18
19 if [ $thereHash == $yourHash ]; then
20     echo files are the same
21 else
22     echo files are different
23 fi
```

Question 07: Briefly describe how the script at the receiver's side works and how to test your script.

The sender must email the receiver with the original file and .authenticate file

To run the sender Type

./sender keyword filename

To run the receiver type

./receiver filename

The sender gets a hash and saves to the .authenticate file then the user must email the .authenticate file. The receiver must put the file in the same directory as the test file the receiver should then run the script with the file name if they are the same it will return files are the same if they are different it will return files are different

Question 08: How many trials it will take you to break the one-way property with the brute-force method? To make your answer reliable, you should repeat your experiment for multiple times

(e.g. 5-10 times) and report the average number of trials. Use screenshot to demonstrate your work if you have.

- The way we implemented the one-way property involves reading through a 'combination' file that reads through every possible combination of strings and hashes the string until the correct hash is found that matches the string we input into the terminal. Therefore, there is no average but the program shows how many trials it took to break the one-way property.
- Here are screenshots showing an input of a string and then the amount of trials it takes to break:

```
[kozakp@eos01 CIS458lab2]$ python2.7 oneWay.py
This is a brute force cracker that can crack 3 character words
[0] enter a 3 character stringing to be cracked
[1] enter a hash to find the stringing
[2] enter a string to find the hashes collision
[3] enter a hash to find a collision

[> 0
enter in a 3 character word all lower case
pat

8dc648c4693d05e3969e5ebbfbd8055 7852341745c93238222a65a910d1dcc5
ad3a9fdcc61bf5a3af69d7b581236f55 7852341745c93238222a65a910d1dcc5
fef6b6dbe04fc3213e4f5555a05a406d 7852341745c93238222a65a910d1dcc5
2fde6b770d99c2db1cbb669960f8318c 7852341745c93238222a65a910d1dcc5
3aaf018941127dc788c49a5ef2554a44 7852341745c93238222a65a910d1dcc5
46868402329ead93f08c779481a35b08 7852341745c93238222a65a910d1dcc5
5bbaa32d1ffe54b3350bf0349f558ecd 7852341745c93238222a65a910d1dcc5
182a15b93cd323556be21fd4fe8f3a8a 7852341745c93238222a65a910d1dcc5
4dca00da67c692296690e90c50c96b79 7852341745c93238222a65a910d1dcc5
6d2d25cac6ce5b576c4509e535e4d3d4 7852341745c93238222a65a910d1dcc5
b90a64580ab4862d303597e6491f4380 7852341745c93238222a65a910d1dcc5
77c345c88d5abc96dff43b05f067805d 7852341745c93238222a65a910d1dcc5
ab7136ba08704e4eb4a19169e1a26695 7852341745c93238222a65a910d1dcc5
d828a5b9b09b334ce76bf241ca16c4eb 7852341745c93238222a65a910d1dcc5
0b438dd454bc6a17de239ebf0a46b91b 7852341745c93238222a65a910d1dcc5
8d569333abbce9e26646dc6a398891324 7852341745c93238222a65a910d1dcc5
46d85374284e4224cbad52d4d7882e08 7852341745c93238222a65a910d1dcc5
91c0f7100bde719c44790e7df757a1a6 7852341745c93238222a65a910d1dcc5
96ac0342a3ccf9553e3d4c9da9b821b0 7852341745c93238222a65a910d1dcc5
655ae040aec1d7f78d790580c5ac37ab 7852341745c93238222a65a910d1dcc5
4e951936957c783062a399c629ce9a95 7852341745c93238222a65a910d1dcc5
981e5fff06f26c103c199601d06a3749 7852341745c93238222a65a910d1dcc5
d018268506e2868537a478629b59e7c1 7852341745c93238222a65a910d1dcc5
cd0acfe085eeb0f874391fb9b8009bed 7852341745c93238222a65a910d1dcc5
hash cracked !!!
the hash 7852341745c93238222a65a910d1dcc5 is equal to pat

the process took 10862 trials
[kozakp@eos01 CIS458lab2]$
```

Question 09: How many trials it will take you to break the collision resistance property with the brute force method? Similarly, to make your answer reliable, you should repeat your experiment for multiple times (e.g. 5-10 times) and report the average number of trials. Use screenshot to demonstrate your work if you have.

- The way we implemented the collection resistance there is no average because it takes the hash and then hashes that hash repeating this process until the first three characters of the hash match. In other words, each hash output is the previous hash, but hashed again. The number of trials needed to break the inputted string is shown in the screenshot below:

```

5eb63bbe01eed093cb22bb8f5acd3 5a3650c6e49c7ab6277b8f2059ffcfd
5eb63bbe01eed093cb22bb8f5acd3 23f053b07a317f2ce247fcb9b1f1f92f
5eb63bbe01eed093cb22bb8f5acd3 9ed58e3175790df0f765a046c05cc0b7
5eb63bbe01eed093cb22bb8f5acd3 feacf6f447a50f3357ac8369c3a2213e
5eb63bbe01eed093cb22bb8f5acd3 6e37b75d40a5a87a30a53daf965435f0
5eb63bbe01eed093cb22bb8f5acd3 1bfd29e2ffbe70486057981041848513
5eb63bbe01eed093cb22bb8f5acd3 f3aeabb97a5a238a05a7df4780a51630
5eb63bbe01eed093cb22bb8f5acd3 8203dd4920979c92b7b14c951d16f1c7
5eb63bbe01eed093cb22bb8f5acd3 59a0eb597cd3549c5d2d82b45985f5b4
5eb63bbe01eed093cb22bb8f5acd3 6e7053639ee0e8d3a1f538b8b2082e2a
5eb63bbe01eed093cb22bb8f5acd3 d79bc54cec9eb9ded96673f634357f97
5eb63bbe01eed093cb22bb8f5acd3 0cef252750357ed779283ee558886841
5eb63bbe01eed093cb22bb8f5acd3 0a7b23dc06426075771354e64a497fc6
5eb63bbe01eed093cb22bb8f5acd3 29455382c83009fc4a72e4e012bc1d0
5eb63bbe01eed093cb22bb8f5acd3 2064205bc3af2614e9aa86fd11ae1f91
5eb63bbe01eed093cb22bb8f5acd3 f1a3b5bb48924eb350ed062245a06846
5eb63bbe01eed093cb22bb8f5acd3 53dd5fa48886011cf87a37d8ef821765
5eb63bbe01eed093cb22bb8f5acd3 086c1359c1f494c69b5b8c0c60227426
5eb63bbe01eed093cb22bb8f5acd3 ff097215903340b4e98a3b7fe24b5a83
5eb63bbe01eed093cb22bb8f5acd3 26a5edb0e3fe04d47dfe16e840df54c4
5eb63bbe01eed093cb22bb8f5acd3 57718a40ad4c75b1ffd9075e5cf38ad1
5eb63bbe01eed093cb22bb8f5acd3 085506ee88517de51f75527d405dae89
5eb63bbe01eed093cb22bb8f5acd3 c8e299a03e6b7902c767b618fe5ca17b
5eb63bbe01eed093cb22bb8f5acd3 620769ad9091e7b19dad2ce488437452
5eb63bbe01eed093cb22bb8f5acd3 e376c74b9931971f8b7ebd81e04b7c5d
5eb63bbe01eed093cb22bb8f5acd3 d62896abeb1635c662edf2cafa31233f
5eb63bbe01eed093cb22bb8f5acd3 5b70d4d151123f0e9593d660e1fa6501
5eb63bbe01eed093cb22bb8f5acd3 58c003d680e660cb51a9b4ce483193ce
5eb63bbe01eed093cb22bb8f5acd3 004ea6fe363d53c0786242b5504ad951
5eb63bbe01eed093cb22bb8f5acd3 c52b305bd3572c432ec0da9161cd29cd
5eb63bbe01eed093cb22bb8f5acd3 54e864b4dfc505548cf1b958ce9ede54
5eb63bbe01eed093cb22bb8f5acd3 4d93277d851cf786fdcd70962ea5c0e5
5eb63bbe01eed093cb22bb8f5acd3 db79bc0bd64e04d51ef1471e5f287b60
5eb63bbe01eed093cb22bb8f5acd3 71b878e73272a861d51e8a87496dbf73
5eb63bbe01eed093cb22bb8f5acd3 5ebb0ee392fa55a1c8e9df76bc6710c4
found a collision from the first three hash bits 5eb
the process took 2590 trials
ashley@ashley-Aspire-E5-521:~/gvsu/cis458/lab/2$

```

-

Question 10: Based on your observation, which property is easier to break by using the brute-force method? Please justify your answer. Use screenshot to demonstrate your work if you have.

- It's easier to break a hash with the collision property rather than a bruteforce match with the one-way property because you must go through every single possibility in a one way property. With the collision method there's a higher probability that you will find a collision in the hash. The number of trails are in the picture

Question 11: (20 points) Please turn in your program code to Blackboard.