Homework 5 Lab Tasks

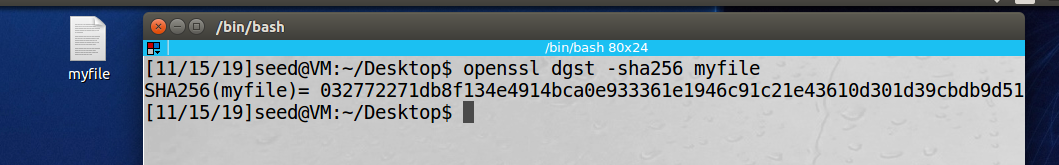
3.1 Task 1: Generating Message Digest and MAC

A screenshot of a social media post

Description automatically generated Using md5

A screenshot of a computer screen

Description automatically generated Using sha1

 Using sha256

3.2 Task 2: Keyed Hash and HMAC

A screenshot of a cell phone

Description automatically generated Using HMAC-MD5, key “hello”

A screenshot of a cell phone

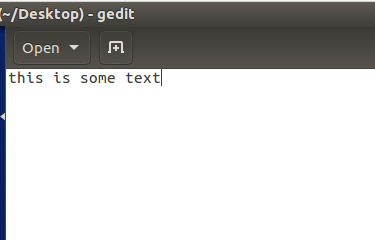
Description automatically generated Using HMAC-SHA256, key “oakland”

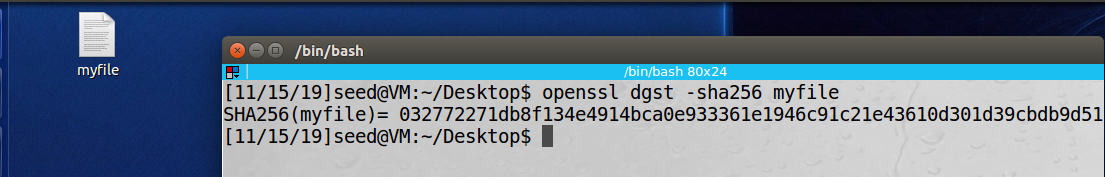
A screenshot of a cell phone

Description automatically generated Using HMAC-SHA1, key “thedogranreallyfast123”

When using the HMAC function, a key of any size can be used because HMAC because it is a cryptographic hash function and allows for the mapping of data **arbitrary** in size to a bit string that is fixed in size.

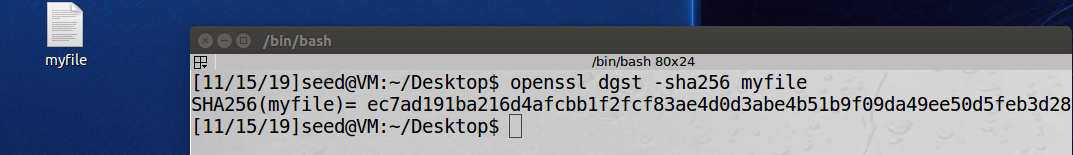
3.3 Task 3: The Randomness of One-way Hash [9 pts]

 Create a text file of any length

 Generate a hash h1 for the file using a specific algorithm, sha-256 used

A screenshot of a cell phone

Description automatically generated Flip one bit of the input file using ghex

 Generate the hash value for the modified file

The 2 generated hashes are incredibly different. Here is a short program to count the number of same bits between h1 and h2

A screenshot of a cell phone screen with text

Description automatically generatedThe program found that there were only 3 corresponding values in the 2 hashes.

3.4 Task 4: Hash Collision-Free Property [20+10 bonus pts]

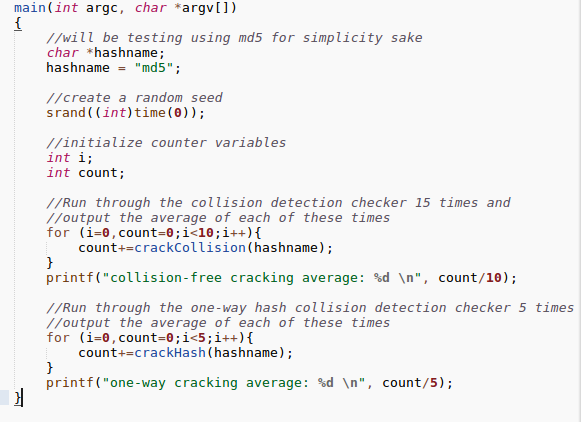
Components of program

1. randomMessage
   1. A screenshot of a cell phone

      Description automatically generateduses a random seed to create a randomly generated message for hash collision checking
2. getHash function
   1. A screenshot of a cell phone

      Description automatically generatedgiven a message and a digest name, this function produces the hash for the given message according to the given digest
3. crackHash
   1. A screenshot of a cell phone

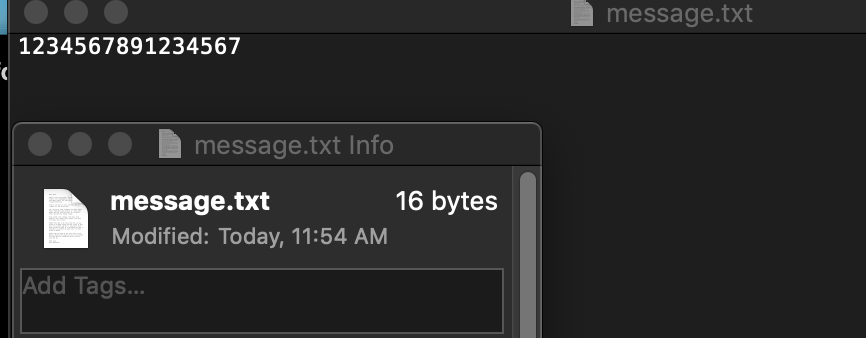
      Description automatically generatedgiven a specified hash, this function will generate random messages and their corresponding hash values until the produced hash matches the given hash
4. crackCollision
   1. A screenshot of text

      Description automatically generatedthis function generates two random messages and their corresponding hash values and checks to see if the produced hash values are equivalent.
5. Main
   1. This function calls crackCollision 10 times and outputs the average number of tries the function utilizes to find the matching hashes
   2. This function also calls crackHash 5 times and outputs the average number of tries the function utilizes to find the matching corresponding hash for the given hash value
6. How many trials it will take you to find two messages with the same hash values using the brute-force method? You should repeat your experiment for multiple times, and report your average number of trials.
   1. A close up of a newspaper

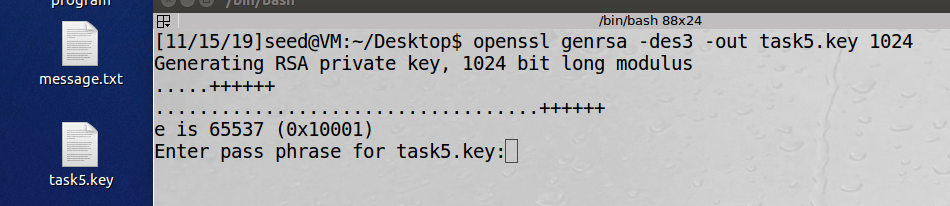
      Description automatically generatedAfter 10 trials of hash collision detecting, the average number of tries was about 30000
7. How many trials will it take you to find a message that has the same hash value as a given/known message’s hash value using the brute-force method? Similarly, you should report the average.
   1. A close up of a sign

      Description automatically generatedAfter 5 trials of one-way hash cracking , the average number of trails was about 11500000
8. Based on your observation, which case is easier to break using the brute-force method?
   1. It is clearly easier to break the collision-free property using the brute force method

3.5 Task 5: Performance Comparison: RSA versus AES [8 pts]

 Prepare a 16 byte-message called message.txt

generate a 1024-bit RSA public/private key pair



Encrypt message.txt using the public key; save the output in message enc.txt.

A close up of a screen

Description automatically generated

A screenshot of a computer screen

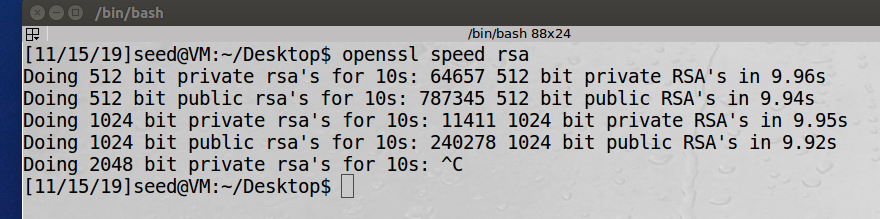
Description automatically generatedDecrypt message enc.txt using the private key.

Encrypt message.txt using a 128-bit AES key.

A screenshot of a cell phone

Description automatically generated

On my machine, the operations occurred too fast to notice a significant difference. The times appeared to be similar

Measuring speed using openssl speed rsa

A screenshot of a computer screen

Description automatically generatedMeasuring speed using openssl speed aes

After these tests, it appears that the aes function runs much quicker than the rsa function

3.6 Task 6: Create Digital Signature

A screenshot of a cell phone

Description automatically generated Prepare example.txt of any size

A screenshot of a computer

Description automatically generatedAlso prepare an RSA public/private key pair

A screenshot of a computer

Description automatically generated

Creating a digital signature and with SHA256 hash of example.txt; save the output in example.sha256.

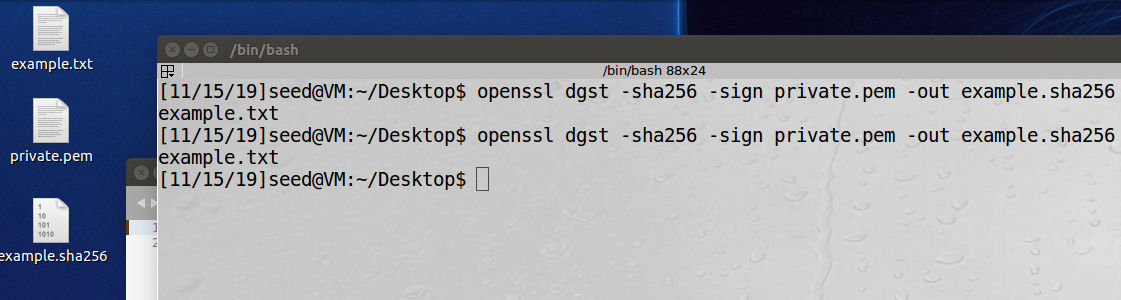
A screenshot of a computer

Description automatically generated Verifying the digital signature (the below command is just a combination of steps 1&2 in the assignment guideline)

A screenshot of a cell phone

Description automatically generatedModifying example.txt

reverifying signature



The commands I used for the previous actions

Preparing an rsa key: openssl genrsa -out private.pem 1024

Creating and verifying signature: openssl dgst -sha256 -sign private.pem -out example.sha256 example.txt

Digital signatures are useful because they act as a virtual fingerprint that is unique to a person or communicating entity. Additionally, digital signatures are used to identify users and protect the legitimacy of digital messages or documents.