

1. Name three applications for cryptographic hash functions.

One use is for hash functions is hashing passwords for storage in servers which is far more secure than storing in plaintext. Another use is to verify the integrity of data or a file by comparing the hashes at different stages of transmission to check for errors. Hash functions are also used for digital signatures.

2. Describe the problem with SHA-1 hash.

The problem with the SHA-1 hash is that it is outdated, so modern computing power is able to brute force collisions (two hashes of different data end up the same,) which means an attacker could fake digital certificates or signatures.

3. Describe the problem with MD5 hash.

MD5 has a similar problem as SHA-1 except magnified, since collisions can be forced much faster and on hardware much older. MD5 should never be used to store passwords, but it is still commonly used to check file integrity.

4. Since SHA-1 and MD5 are both defective, what would you use as a hash algorithm?

Newer hash algorithms like SHA-256, SHA-512, or Bcrypt should be used instead.

5. What is the difference between a hash function and encryption?

The primary difference between a hash function and encryption is that hash functions are “one-way.” With encryption the data can always be decrypted using the proper key, but a proper hash function would provide no way to reveal the original data.

6. Name the three types of tokens.

Tokens are a cryptographic key or password for authentication. The three types of tokens are “something you have,” “something you know,” and “something you are.” “Something you have” refers to a token that you have that an attacker could steal, like a software license or a two-factor authentication device. “Something you know” refers to a token that is only in your head that an attacker could steal, like a secret PIN or password that isn’t written anywhere else. “Something you are” refers to biometric tokens like a fingerprint, facial recognition, etc.

7. What is password entropy, and why is it important?

Password entropy is the measure of difficulty or uncertainty an attacker faces to determine the value of a secret, usually stated in bits. It is a statistical parameter which measures how much information is produced on the average for each letter of a text in the language. A high entropy is important because it can deter attackers from even trying. Increased length and more acceptable characters increase entropy the most.

8. NIST secret token requires a user generated string of 8 or more characters chosen from an alphabet of 90 or more characters. If I have a user generated string of 20 characters chosen from an alphabet of 64 characters, will it still meet NIST level 2 requirements?

$$\text{Entropy} = H = \log_2(b^l)$$

$$\text{Minimum } H = \log_2(90^8) = 51.93$$

$$\text{Calculated } H = \log_2(64^{20}) = 120$$

Yes, the string meets NIST level 2 requirements.

9. NIST secret token requires a user generated string of 8 or more characters chosen from an alphabet of 90 or more characters. If I have a user generated string of 200 characters chosen from an alphabet of 32 characters, will it still meet NIST level 2 requirements?

$$\text{Minimum } H = \log_2(90^8) = 51.93$$

$$\text{Calculated } H = \log_2(32^{200}) = 1,000$$

Yes, the string meets NIST level 2 requirements.

10. What are password rules? Are they effective against attacks?

Password rules are requirements passwords must meet in order to be acceptable for use. Some rules are effective and some not so much, which means they are constantly changing and being updated. Current rules that are effective include minimum length requirements, preventing common passwords with a dictionary, and allowing all characters to be used. Ineffective rules include hints, routine expiration, SMS two factor authentication, and more.

11. What is the problem with SMS two factor authentication?

The problem with SMS two factor authentication is that SMS is not secure. Attackers can easily socially engineer a customer service rep. to change your service over to a new SIM card in a different phone. Once this is complete, the attacker has unhindered access to everything given they also know the passwords. The attacker could also spoof a message to the victim and trick them into replying with a legitimate 2FA code.

12. What is the benefit of two-factor authentication?

Two factor authentication is still a great tool for security, just not using SMS. 2FA is beneficial because even if the attacker has a password, if the account is secured with 2FA they would also need access to your 2FA method, like a phone app that generates unique codes.

Name: Zak Rowland
Embedded Security
CST 466
Assignment 3

13. Using Python's built in sha1 and md5 hash functions, implement a program that will take a plaintext file called infile.txt and output the sha1 and md5 hash into another text file called outfile.txt. For Python2, use md5 and sha module. For Python3, use hashlib

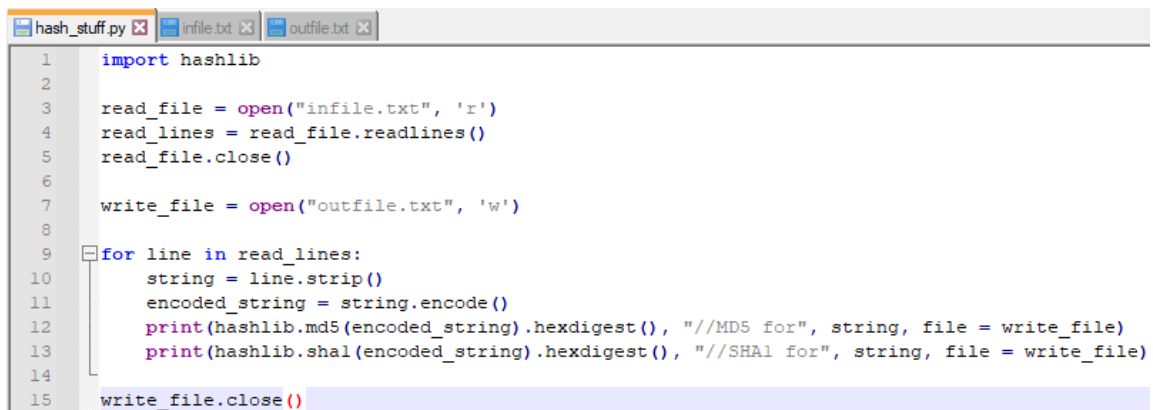
```
input from infile.txt  
kevinisasmoothposer  
lmao
```

```
output to outfile.txt  
e93d484863e01f1dd8f44ae7beb0c075 //md5 for kevinisasmoothposer  
234a0f6e8a64f9a496ab310e209e1cca8bed6dbc //sha1 for  
kevinisasmoothposer  
0f18fd4cf40bfb1dec646807c7fa5522 //md5 for lmao  
23d22c4e7fc62edfdea86a0e9a94c57a2c640e26 //sha1 for lmao
```

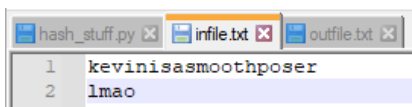
<https://docs.python.org/3/library/hashlib.html>

<https://docs.python.org/2/library/md5.html>

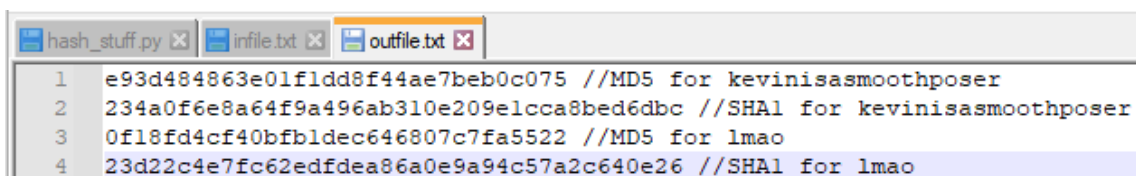
<https://docs.python.org/2/library/sha.html#module-sha>



```
1 import hashlib  
2  
3 read_file = open("infile.txt", 'r')  
4 read_lines = read_file.readlines()  
5 read_file.close()  
6  
7 write_file = open("outfile.txt", 'w')  
8  
9 for line in read_lines:  
10     string = line.strip()  
11     encoded_string = string.encode()  
12     print(hashlib.md5(encoded_string).hexdigest(), "//MD5 for", string, file = write_file)  
13     print(hashlib.sha1(encoded_string).hexdigest(), "//SHA1 for", string, file = write_file)  
14  
15 write_file.close()
```



```
1 kevinisasmoothposer  
2 lmao
```



```
1 e93d484863e01f1dd8f44ae7beb0c075 //MD5 for kevinisasmoothposer  
2 234a0f6e8a64f9a496ab310e209e1cca8bed6dbc //SHA1 for kevinisasmoothposer  
3 0f18fd4cf40bfb1dec646807c7fa5522 //MD5 for lmao  
4 23d22c4e7fc62edfdea86a0e9a94c57a2c640e26 //SHA1 for lmao
```

Name: Zak Rowland
Embedded Security
CST 466
Assignment 3

```
import hashlib

read_file = open("infile.txt", 'r')
read_lines = read_file.readlines()
read_file.close()

write_file = open("outfile.txt", 'w')

for line in read_lines:
    string = line.strip()
    encoded_string = string.encode()
    print(hashlib.md5(encoded_string).hexdigest(), "//MD5 for", string, file = write_file)
    print(hashlib.sha1(encoded_string).hexdigest(), "//SHA1 for", string, file = write_file)

write_file.close()
```