Lab 9 Template

Part 01

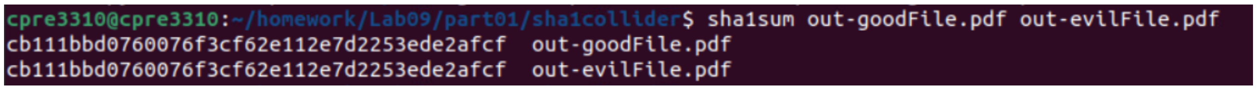
1) **“outfile-Good text” pdf and “out-file Bad text” pdf (upload the files in canvas submission)**

(10 points)

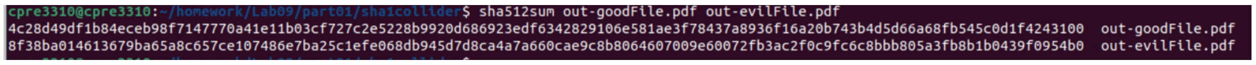
Uploaded.

2) **Screenshot of the hashes of the text PDFs colliding for SHA1 and differing with SHA512**

(10 points)



```
cpre3310@cpre3310:~/homework/Lab09/part01/sha1collider$ sha1sum out-goodFile.pdf out-evilFile.pdf
cb111bbd0760076f3cf62e112e7d2253ede2afcf  out-goodFile.pdf
cb111bbd0760076f3cf62e112e7d2253ede2afcf  out-evilFile.pdf
```



```
cpre3310@cpre3310:~/homework/Lab09/part01/sha1collider$ sha512sum out-goodFile.pdf out-evilFile.pdf
4c28d49df1b84eceb98f7147770a41e11b03cf727c2e5228b9920d686923edf6342829106e581ae3f78437a8936f16a20b743b4d5d66a68fb545c0d1f4243100  out-goodFile.pdf
8f38ba014613679ba65a8c657ce107486e7ba25c1efe068db945d7d8ca4a7a660cae9c8b8064607009e60072fb3ac2f0c9fc6c8bbb805a3fb8b1b0439f0954b0  out-evilFile.pdf
```

3) **Explain why the SHA1 hashes are the same for different files, but the SHA512 hashes are different. Please be specific about how the near collision/collision works.**

(10 points)

The hashes are the same due to vulnerabilities present in SHA1. Near collisions/ collisions can be generated more easily. It is easily possible in SHA1 because of the hash function’s relatively small output size and the weakness that have been discovered in its design. SHA512 is resistant to collisions, making it more secure and less likely to suffer from this issue. SHA512 produces different hashes for different files because of its larger hash space and stronger resistance to collisions, making it more secure and less likely to produce the same hash for different inputs.

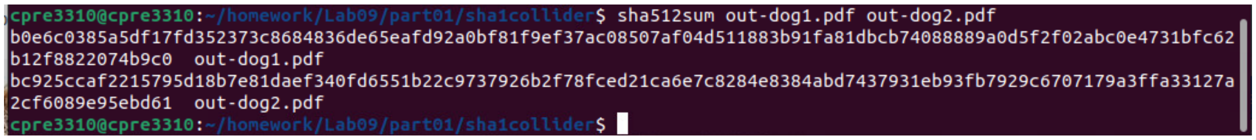
4) **Image out-1.pdf and Image out-2.pdf (Upload the files in canvas submission)**

(10 points)

Uploaded.

5) **Screenshot of the hashes of the image PDFs colliding for SHA1 and differing with SHA512**

(10 points)



```
cpre3310@cpre3310:~/homework/Lab09/part01/sha1collider$ sha512sum out-dog1.pdf out-dog2.pdf
b0e6c0385a5df17fd352373c8684836de65eafd92a0bf81f9ef37ac08507af04d511883b91fa81dbcb74088889a0d5f2f02abc0e4731bfc62
b12f8822074b9c0  out-dog1.pdf
bc925ccaf2215795d18b7e81daef340fd6551b22c9737926b2f78fced21ca6e7c8284e8384abd7437931eb93fb7929c6707179a3ffa33127a
2cf6089e95ebd61  out-dog2.pdf
cpre3310@cpre3310:~/homework/Lab09/part01/sha1collider$ 
```

6) **How likely is this to be used to carry out an attack in the wild?**

(10 total points, 5 points likelihood, 5 points example)

Digital signatures and certificates: If an attacker creates a collision for digital signature, they could replace a legitimate file with a malicious one that has the same hash. If a company uses SHA1 to sign an update package, an attacker could generate a malicious update that results in the same SA1 hash as the correct one. It works as SHA1 is vulnerable to collisions and the attacker is just exploiting the weakness of SHA1. Collision attacks on weak hash functions like SHA1 can be used in cases involving digital signatures, file integrity verification, and version control systems.

Part 02

1) **Upload your code to canvas as a <u>separate .py file</u>**

(10 points)

Uploaded.

2) **What is the winning lottery number for November 12th, 2024?**

(5 points)

8-0-4-1-8-6-1-5-8-6 is the winning lottery of November 12th, 2024.

3) **In your own words, what is the seed for this random number generator? You must provide the correct values for both pieces used in the November 11th pull and explain the calculation made to earn full points. Why is this a good or bad thing? Justify your answer.**

(15 total points, 5 points seed values, 5 points explanation, 5 points good/bad reasoning/justification)

The seed values: date is 11/11/2024 which is 11112024 in integer. The time in seconds is 52790 therefore the final seed will be the sum of these two values which is 11164814. This is bad as it allows anyone to recreate the exact sequence of random numbers if they know or can guess the time. Since there are only 86400 possible seconds in a day, it's easy to brute force.

4) **What would be an alternate way to seed this random number generator such that it couldn't be predicted either forward or backward? Justify your answer.**

(10 total points, 5 points alternate idea for seeding, 5 points reasoning/justification)

We can use dev/urandom that reads 8 bytes of randomness from the system's /dev/urandom device. Then it converts the bytes into an integer that can be used as the seed. Its secure and is unpredictable both forward and backward. This is much more secure than relying on the current time or date.