Lab10 - The Downfall of SHA1 and Bad Random Numbers

Part 01

1) **“Good” text pdf and “Bad” text pdf (upload the files in canvas submission)**
   (10 points)

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

```
cpre331@cpre331:~/labs/lab10/sha1collider$ sha1sum out-badletter.pdf out-goodlet
ter.pdf
ad57752d0d7c6a0e90a231259ced17e50a179989  out-badletter.pdf
ad57752d0d7c6a0e90a231259ced17e50a179989  out-goodletter.pdf
cpre331@cpre331:~/labs/lab10/sha1collider$ sha512sum out-badletter.pdf out-goodl
etter.pdf
7660e5586955df8bf3a1fab07d4dfd463877d95dc7d78ce55bfafe01abbf17bb320b279180fd2753
7fa4d94a5e07b265055a7d153c98bb0c6976200b0ae2881b  out-badletter.pdf
53eb9e91e31cb3403cff3b9fb57b8595ff4d068061dbce780c1bcfc5eafe490605e69f349e1ac27c
8105bb75c79f3dd67e4e36faeb4ef9838b5265a2c8662610  out-goodletter.pdf
```

3) **Explain why the SHA1 hashes are the same for different files, but the SHA512 hashes are different.**
   (10 points)
   - The reason why they can be the same is because SHA-1 generates 160-bit hash values, making it not so great in collision resistance (inputs produce the same hash). While SHA-512 produces 512-bit hash values which is a larger hash space and has greater collision resistance. This makes sense because of the restriction (1 page) that was given due to the bit length. Since SHA-512 has much more space, then it is significantly less likely for collisions to occur. Thus the name collider.py.

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

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

```
cpre331@cpre331:~/labs/lab10/sha1collider$ sha1sum out-cat.pdf out-cat2.pdf
9fe2e7409e4e4df9afb93426a9fef3fe720f4e9b  out-cat.pdf
9fe2e7409e4e4df9afb93426a9fef3fe720f4e9b  out-cat2.pdf
```

6) **Answer questions in part p**
    a. **How likely is this to be used to carry out an attack in the wild?**

(10 points)

- These hash collisions aren't very practically done in the wild. This is because most people have moved away from SHA-1 to SHA-512. Also we only do it with pdfs and "manually" making them with the same hash using exact steps and the attacker often would not have control over what data they would manipulate/collide. Even if the collisions were possible over SHA-1, the likely odds of this happening with the specific scenario we just demonstrated, is almost near 0 because of new cryptographic standards and security practices that have evolved over time.

Part 02

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

(10 points)

2) **What is the winning lottery number for November 12th, 2023?**

(10 points)

```
[tech180@TechTop-G14L:~/Documents/College/fall2023/cpre331/lab10]$ python3 template.py
Found matching timeInSeconds: 80000
Winning Lottery Ticket for November 12th, 2023: 2-3-6-0-0-4-1-2-0-3
```

3) **In your own words, what is the seed for this random number generator? Why is this a good or bad thing? Justify your answer.**

(20 total points, 10 points seed explanation, 10 points good/bad reasoning/justification)

- The seed is the initial value to start the generation of random (lottery) numbers. The seed is made by combining the date with the number of seconds in a day. This in turn influences the sequence of numbers that it can produce. This can be good because it makes sure that the numbers are generated every day and you might think it's random due to the nature of how it's giving out numbers. However, if this code was released and people knew how it was being generated, you could easily predict future lottery numbers. Overall, there is an illusion of randomness with a simple implementation, but something that can cause easy predictability if the solution is known.