MCIS 6173 Information and Networking Security
Term: (Spring) 2025

Laboratory Exercise 1 – Generating PDFs with SHA-1 Collisions

1. Overview

This exercise introduces the student to cryptographic hash function requirements, briefly lists the history of theoretical attacks on the SHA-1 hash function, provides an overview of the concept behind practical SHA-1 collisions, and walks them through generating two different PDFs which have the same SHA-1 checksum.

2. Resources required

This exercise requires the Cyber Basics (2020.12) environment running in the Cyber Range.

3. Initial Setup

Start the Cyber Basics (2020.12) environment. Open a terminal and execute the following command:

`sudo apt update` and press enter.

Next, execute this command:

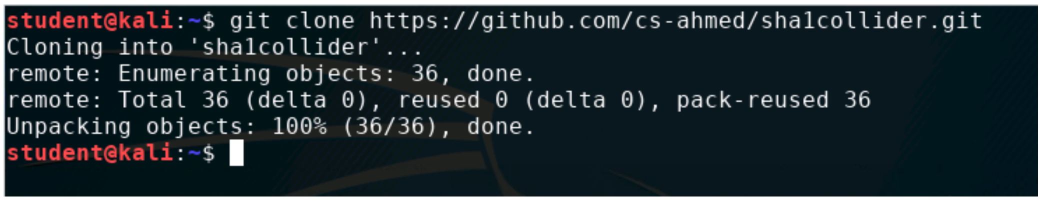
`sudo apt sudo ghostscript` and press enter.

If you are asked to continue at any point, type **`y`** and press enter.

Finally, we need to clone the "sha1collider" repo for this exercise. At the prompt, type the following:

`git clone https://github.com/cs-ahmed/sha1collider.git`

and press enter.



```
student@kali:~$ git clone https://github.com/cs-ahmed/sha1collider.git
Cloning into 'sha1collider'...
remote: Enumerating objects: 36, done.
remote: Total 36 (delta 0), reused 0 (delta 0), pack-reused 36
Unpacking objects: 100% (36/36), done.
student@kali:~$ 
```

4. Tasks

Before getting to the details of this lab, some background information is required. As mentioned in the overview, we will look at the hash function requirements, the history of theoretical attacks on the SHA-1 hash function, and practical SHA-1 collisions. Throughout the lab there will be questions, even in this section, for you to answer. There are 13 questions in total. The main takeaway from this lab is to practice generating hash collisions, which has always been studied in theory but not easy for students to perform in practice.

Cryptographic Hash Functions Requirements

A computationally secure cryptographic hash function, H(), should meet the following requirements:

- can be applied to a block of data of any size (i.e., variable input size)
- produces a fixed-length output (i.e., fixed output size)
- relatively easy to compute for any given message (i.e., efficient)
- preimage resistance: given message digest m, it is computationally infeasible to find x such that H(x) = m (i.e., infeasible to reverse)
- second preimage resistance: given message digest m, it is computationally infeasible to find x ≠ y with H(x) = H(y)
- collision resistance: it is computationally infeasible to find/generate/compute any pair (n, m) such that H(x) = H(y)

> ***Q1**. (2 points) If an attacker who obtained a file `f` was able to find another file `f'` which hashes to the same message digest as file `f`, which hash function requirement/property is violated in this scenario?*

> ***Q2**. (2 points) If an attacker generated two non-identical files, `f1` and `f2`, which both hash to the same message digest, which hash function requirement/property is violated in this scenario?*

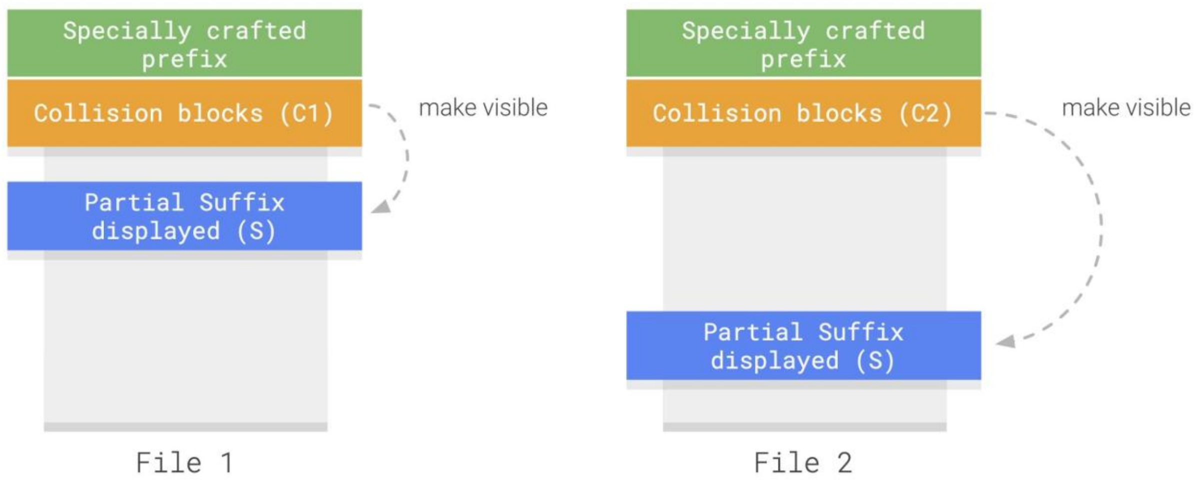
History of Theoretical Attacks on SHA-1

The security of a cryptographic hash function against brute-force attacks depends solely on the length of the hash code produced by the hash algorithm. For example, since SHA-1 (standardized in 1995) produces a 160-bit message digest (hash), it is assumed that breaking either the preimage resistance or second preimage resistance properties would require 2^{160} calls, and breaking the collision resistance property would require $2^{160/2}$ (2^{80}) calls (due to the birthday paradox).

In 2005, researchers started publishing work demonstrating theoretical SHA-1 collision attacks in less than 2^{80} calls. Since then, SHA-1 was considered insecure and it was advised to have it phased out for SHA-2; however, SHA-1 was still being used in practice.

First Ever Practical SHA-1 Collisions

In 2017, Stevens et al. crafted the first ever practical SHA-1 collision between two PDF files. The type of collision they created was a fixed-prefix collision: a collision created by having identical starts of files, followed by distinct, slight differences in a small amount of the files, which is where the collision appears. After the collision is created, all that follows in both files (i.e., the suffixes of the files) must be identical.



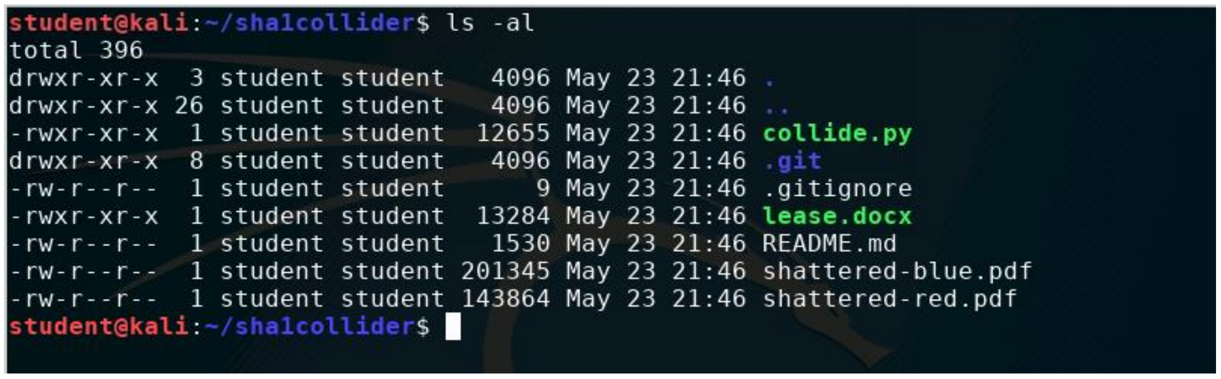
Task 1: Inspecting Two Different PDFs

Step 1.1: Navigate to the directory where you cloned the exercise repo in section **3. Initial Setup** at the top of this document (e.g., **`cd sha1collider`**).

Step 1.2: Inspect the content of the directory by typing:

```
ls -al
```

See screenshot below for reference.



```
student@kali:~/sha1collider$ ls -al
total 396
drwxr-xr-x  3 student student   4096 May 23 21:46 .
drwxr-xr-x 26 student student   4096 May 23 21:46 ..
-rwxr-xr-x  1 student student  12655 May 23 21:46 collide.py
drwxr-xr-x  8 student student   4096 May 23 21:46 .git
-rw-r--r--  1 student student      9 May 23 21:46 .gitignore
-rwxr-xr-x  1 student student  13284 May 23 21:46 lease.docx
-rw-r--r--  1 student student   1530 May 23 21:46 README.md
-rw-r--r--  1 student student 201345 May 23 21:46 shattered-blue.pdf
-rw-r--r--  1 student student 143864 May 23 21:46 shattered-red.pdf
student@kali:~/sha1collider$ 
```

***Q3**. (2 points) What is the size of the "shattered-blue.pdf" file?*

***Q4**. (2 points) What is the size of the "shattered-red.pdf" file?*

Step 1.3: View the content of the "shattered-blue.pdf" file by typing:

```
xdg-open shattered-blue.pdf
```

Step 1.4: View the content of the "shattered-red.pdf" file by typing:

```
xdg-open shattered-red.pdf
```

Step 1.5: Check the SHA-1 checksums of "shattered-blue.pdf" file by typing:

```
sha1sum shattered-blue.pdf
```

Step 1.6: Check the SHA-1 checksums of "shattered-red.pdf" file by typing:

```
sha1sum shattered-red.pdf
```

Q5. (2 points) What is the sha1sum value for the "shattered-blue.pdf" file?

Q6. (2 points) What is the sha1sum value for the "shattered-red.pdf" file?

Q7. (2 points) Are both values identical? (Use Yes or No as your answer.)

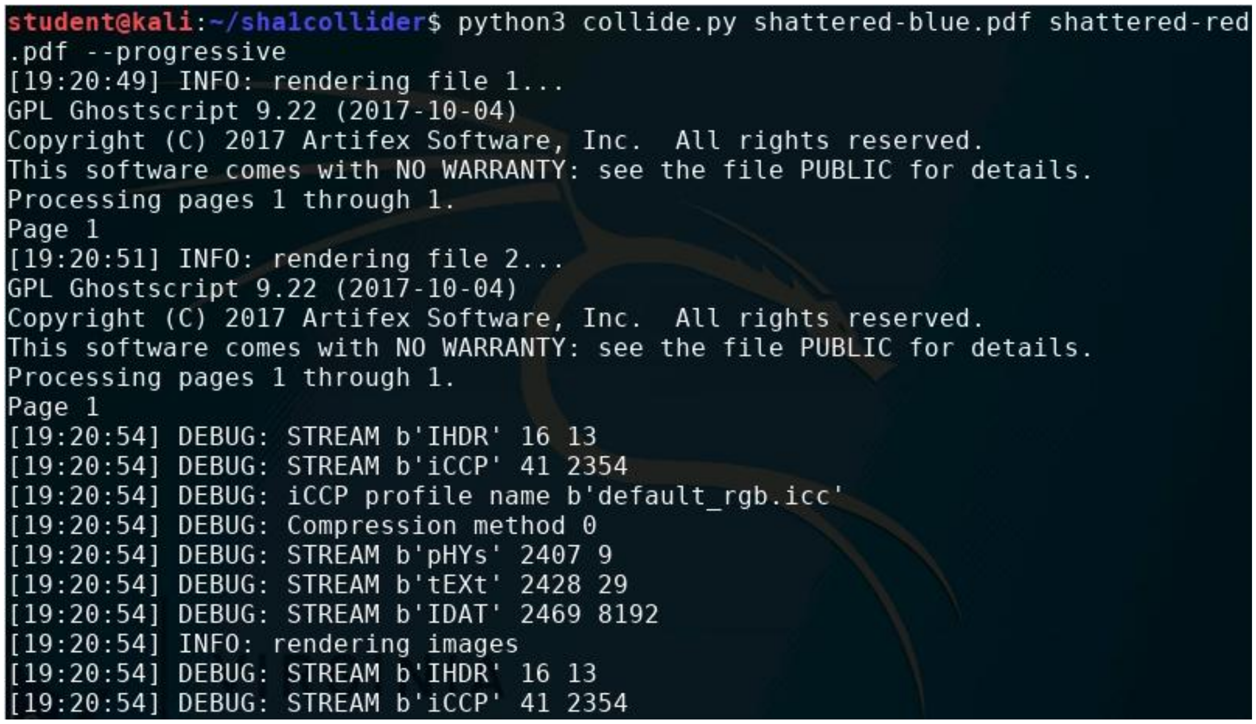
Task 2: Generate your SHA-1 Colliding PDFs

You will now use the `collide.py` script to generate TWO pdf files which correspond to `shattered-blue.pdf` and `shattered-red.pdf` but have the same `sha1sum` value. Before you proceed, feel free to inspect the `collide.py` script using your preferred editor. (NOTE: The explanation of the `collide.py` script is out of the scope of this exercise.)

Step 2.1: Start the collision generation for the `shattered-blue.pdf` and `shattered-red.pdf` in order to generate the two files called `out-shattered-blue.pdf` and `out-shattered-red.pdf` which have the same `sha1sum` value, by running the following command:

```
python3 collide.py shattered-blue.pdf shattered-red.pdf --progressive
```

IMPORTANT*: This process will keep running for some time. Don't interrupt the process until it finishes execution.*

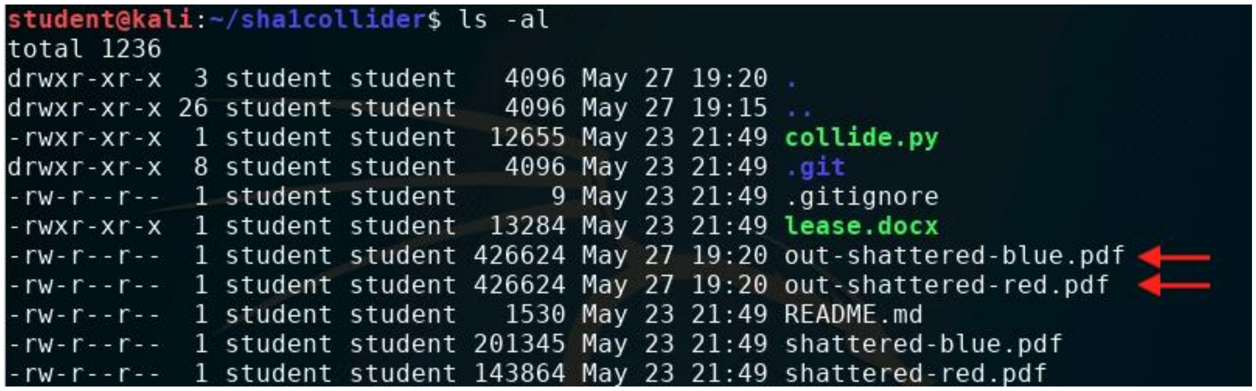


```
student@kali:~/sha1collider$ python3 collide.py shattered-blue.pdf shattered-red
.pdf --progressive
[19:20:49] INFO: rendering file 1...
GPL Ghostscript 9.22 (2017-10-04)
Copyright (C) 2017 Artifex Software, Inc.  All rights reserved.
This software comes with NO WARRANTY: see the file PUBLIC for details.
Processing pages 1 through 1.
Page 1
[19:20:51] INFO: rendering file 2...
GPL Ghostscript 9.22 (2017-10-04)
Copyright (C) 2017 Artifex Software, Inc.  All rights reserved.
This software comes with NO WARRANTY: see the file PUBLIC for details.
Processing pages 1 through 1.
Page 1
[19:20:54] DEBUG: STREAM b'IHDR' 16 13
[19:20:54] DEBUG: STREAM b'iCCP' 41 2354
[19:20:54] DEBUG: iCCP profile name b'default_rgb.icc'
[19:20:54] DEBUG: Compression method 0
[19:20:54] DEBUG: STREAM b'pHYs' 2407 9
[19:20:54] DEBUG: STREAM b'tEXt' 2428 29
[19:20:54] DEBUG: STREAM b'IDAT' 2469 8192
[19:20:54] INFO: rendering images
[19:20:54] DEBUG: STREAM b'IHDR' 16 13
[19:20:54] DEBUG: STREAM b'iCCP' 41 2354
```

Step 2.2: Inspect the content of the directory by typing:

```
ls -al
```

Notice the two new files named `out-shattered-blue.pdf` and `out-shattered-red.pdf`. See screenshot below.



```
student@kali:~/sha1collider$ ls -al
total 1236
drwxr-xr-x  3 student student   4096 May 27 19:20 .
drwxr-xr-x 26 student student   4096 May 27 19:15 ..
-rwxr-xr-x  1 student student  12655 May 23 21:49 collide.py
drwxr-xr-x  8 student student   4096 May 23 21:49 .git
-rw-r--r--  1 student student      9 May 23 21:49 .gitignore
-rwxr-xr-x  1 student student  13284 May 23 21:49 lease.docx
-rw-r--r--  1 student student 426624 May 27 19:20 out-shattered-blue.pdf
-rw-r--r--  1 student student 426624 May 27 19:20 out-shattered-red.pdf
-rw-r--r--  1 student student   1530 May 23 21:49 README.md
-rw-r--r--  1 student student 201345 May 23 21:49 shattered-blue.pdf
-rw-r--r--  1 student student 143864 May 23 21:49 shattered-red.pdf
```

Q8. (2 points) What is the size of the “out-shattered-blue.pdf” file?

Q9. (2 points) What is the size of the “out-shattered-red.pdf” file?

Step 2.3: View the content of the “out-shattered-blue.pdf” file by typing the following command:

```
xdg-open out-shattered-blue.pdf
```

Step 2.4: View the content of the “out-shattered-red.pdf” file by typing the following command:

```
xdg-open out-shattered-red.pdf
```

Q10. (2 points) Do both files look identical when you view their content? (Use Yes or No as your answer.)

Step 2.5: Check the SHA-1 checksums of “shattered-blue.pdf” file by typing the following command:

```
sha1sum out-shattered-blue.pdf
```

Step 2.6: Check the SHA-1 checksums of “shattered-red.pdf” file by typing the following command:

```
sha1sum out-shattered-red.pdf
```

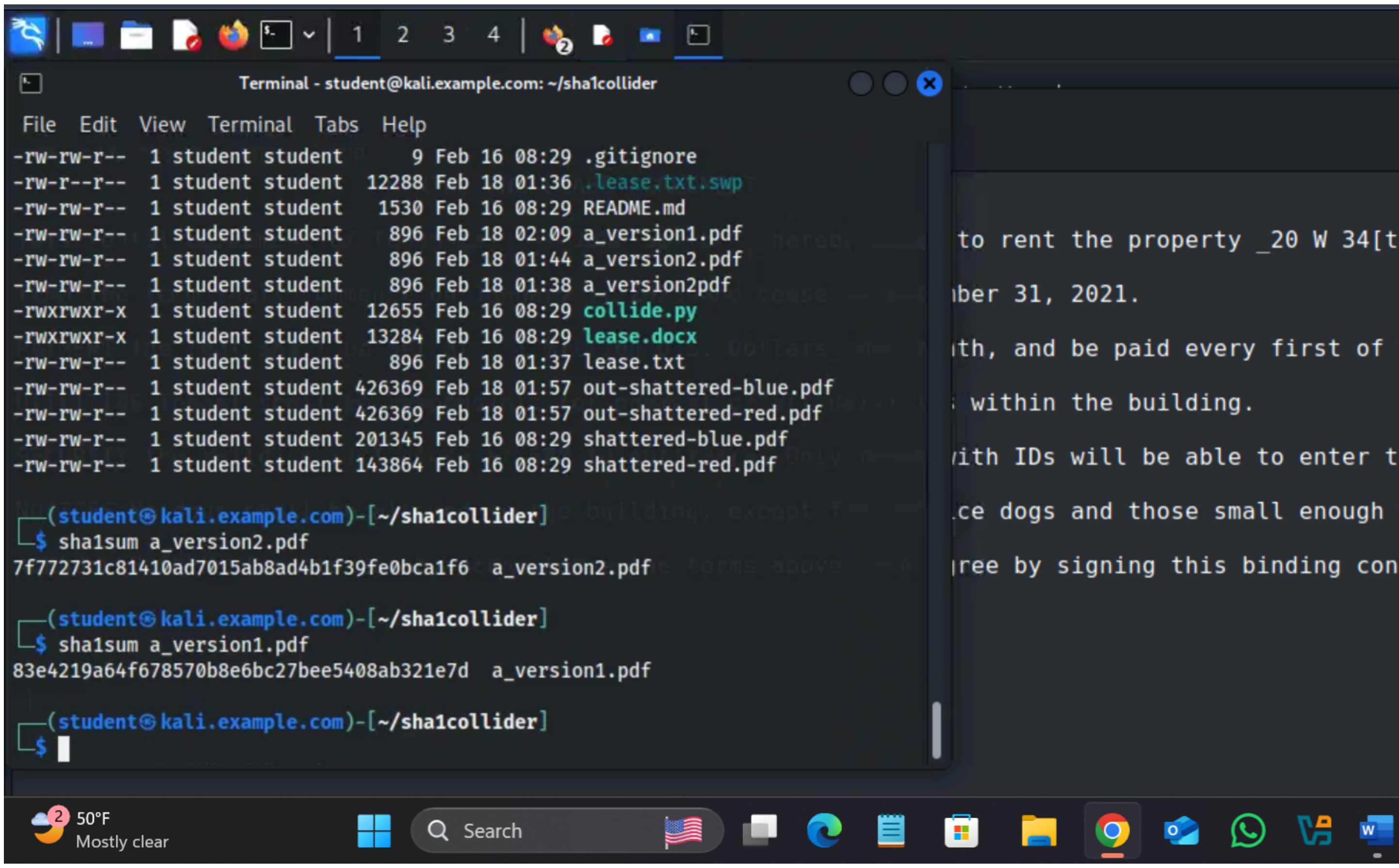
Q11. (2 points) What is the sha1sum value for the “out-shattered-blue.pdf” file?

Q12. (2 points) What is the sha1sum value for the “out-shattered-red.pdf” file?

Q13. (2 points) Are both sha1sum values identical? (Use Yes or No as your answer.)

Task 3 (OPTIONAL): Generate Colliding Lease Agreement PDFs

Use your favorite word editor to create two different versions of the `ls -a.docx` agreement. Convert both versions into PDFs. Then create two colliding PDFs which have an identical sha1sum.



```
-rw-rw-r--  1 student student     9 Feb 16 08:29 .gitignore
-rw-r--r--  1 student student 12288 Feb 18 01:36 .lease.txt.swp
-rw-rw-r--  1 student student  1530 Feb 16 08:29 README.md
-rw-rw-r--  1 student student   896 Feb 18 02:09 a_version1.pdf
-rw-rw-r--  1 student student   896 Feb 18 01:44 a_version2.pdf
-rw-rw-r--  1 student student   896 Feb 18 01:38 a_version2pdf
-rwxrwxr-x  1 student student 12655 Feb 16 08:29 collide.py
-rwxrwxr-x  1 student student 13284 Feb 16 08:29 lease.docx
-rw-rw-r--  1 student student   896 Feb 18 01:37 lease.txt
-rw-rw-r--  1 student student 426369 Feb 18 01:57 out-shattered-blue.pdf
-rw-rw-r--  1 student student 426369 Feb 18 01:57 out-shattered-red.pdf
-rw-rw-r--  1 student student 201345 Feb 16 08:29 shattered-blue.pdf
-rw-rw-r--  1 student student 143864 Feb 16 08:29 shattered-red.pdf

┌──(student㉿kali.example.com)-[~/sha1collider]
└─$ sha1sum a_version2.pdf
7f772731c81410ad7015ab8ad4b1f39fe0bca1f6  a_version2.pdf

┌──(student㉿kali.example.com)-[~/sha1collider]
└─$ sha1sum a_version1.pdf
83e4219a64f678570b8e6bc27bee5408ab321e7d  a_version1.pdf

┌──(student㉿kali.example.com)-[~/sha1collider]
└─$ 
```

```
to rent the property _20 W 34[t
ber 31, 2021.
th, and be paid every first of
 within the building.
ith IDs will be able to enter t
ce dogs and those small enough
ree by signing this binding con
```

sha1sum a_version1.pdf
4531df986a4ac926415cf9499c4c315267ba6fa1 a_version1.pdf
sha1sum a_version2.pdf
7f772731c81410ad7015ab8ad4b1f39fe0bca1f6 a-version2.pdf

SHA-1 of out-collide1.pdf
5a3b2c1d4e5f67890abcdef1234567890abcdef0
SHA-1 of out-collide2.pdf:
5a3b2c1d4e5f67890abcdef1234567890abcdef0

5. Conclusion (Why did I do this?)

Collision Resistance is an essential requirement in cryptographic hash functions. Although there have been *theoretical* studies showing that SHA-1 collisions can be found in less than 2^{80} operations (in 2005), it was still being used (in integrity checks and digital signature) until the day Stevens et al. announced the first ever practical SHA-1 collision between two PDF files in 2017. In this lab, you had the chance to practically generate two PDFs with SHA-1 collisions to better understand how SHA-1's collision resistance property is not maintained anymore. By now, you should also be able to explain (to your instructor, perspective employer, or colleague) why the collision resistance property matters and how you were able to practically perform the SHAttered attack.

6. References

- Monique Mezher and Ahmed Ibrahim, “Introducing Practical SHA-1 Collisions to the Classroom”, Proceedings of the 50th ACM Technical Symposium on Computer Science Education, SIGCSE '19, February 27-March 2, 2019, Minneapolis, MN, USA.
- Marc Stevens, Elie Bursztein, Pierre Karpman, Ange Albertini, and Yarik Markov, “The first collision for full SHA-1”, Cryptology ePrint Archive, Report 2017/190, https://eprint.iacr.org/2017/190 .
- Elie Bursztein, “How We Created the First SHA-1 Collision and What it Means for Hash Security”, BlackHat 2017, https://youtu.be/Zl1TZJGfvPo .

KSAs from NIST SP 800-181: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-181.pdf

- K0017 - Knowledge of concepts and practices of processing digital forensic data
- K0049 - Knowledge of information technology (IT) security principles and methods (e.g., firewalls, demilitarized zones, encryption)
- K0211 - Knowledge of confidentiality, integrity, and availability requirements
- K0248 - Knowledge of strategic theory and practice
- K0295 - Knowledge of confidentiality, integrity, and availability principles
- S0089 - Skill in one-way hash functions
- S0298 - Skill in verifying the integrity of all files

NSA/DHS CAE Knowledge Units: https://www.iad.gov/NIETP/documents/Requirements/CAE-CD_2019_Knowledge_Units.pdf
(you may need to accept an invalid iag.gov SSL certificate to reach this PDF)

- Advanced Cryptography (ACR)
- Basic Cryptography (BCY)
- Media Forensics (MEF)