Lab

04

LAB REPORT 04

Hash Functions and Digital Certificates

Subject: Computer Network Security LAB
Class: NT101.011.MMCL

Instructor	Nguyễn Xuân Hà
Students	Hoàng Trí Tường (21521654)
	Đỗ Thế Danh (21520685)
Progress	Done
Duration	12/11/2023 – 06/11/2023
Self-grade	9.5/10

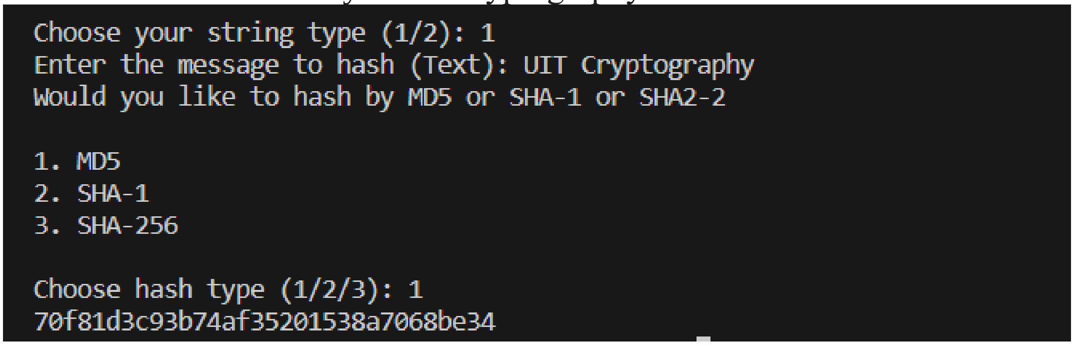
DETAILS REPORT

1. Generating message digests (hash values) and HMAC

1.1 Exercise

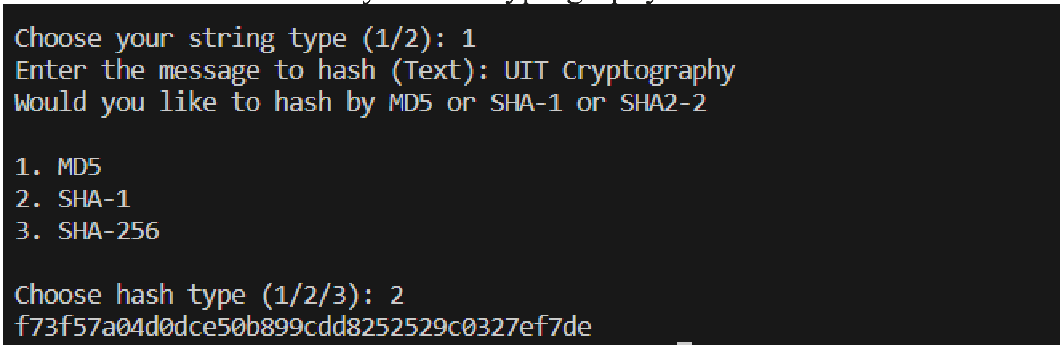
Text string

This is the hash value of **MD5** by “UIT Cryptography”



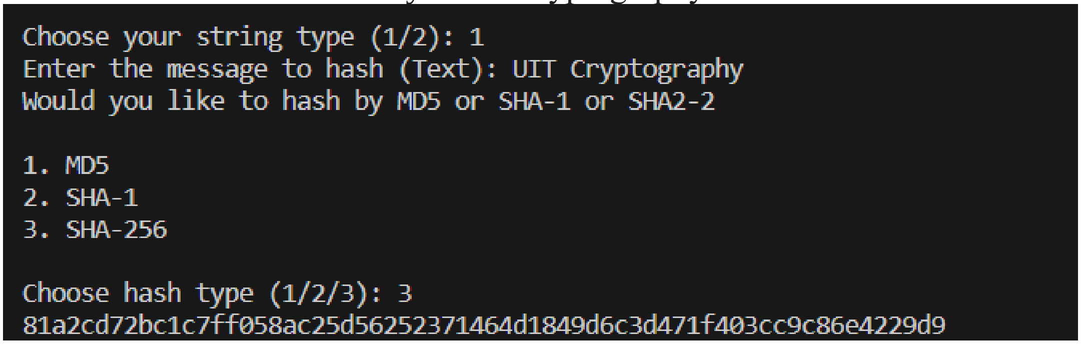
```
Choose your string type (1/2): 1
Enter the message to hash (Text): UIT Cryptography
Would you like to hash by MD5 or SHA-1 or SHA2-2

1. MD5
2. SHA-1
3. SHA-256

Choose hash type (1/2/3): 1
70f81d3c93b74af35201538a7068be34
```

This is the hash value of **SHA-1** by “UIT Cryptography”



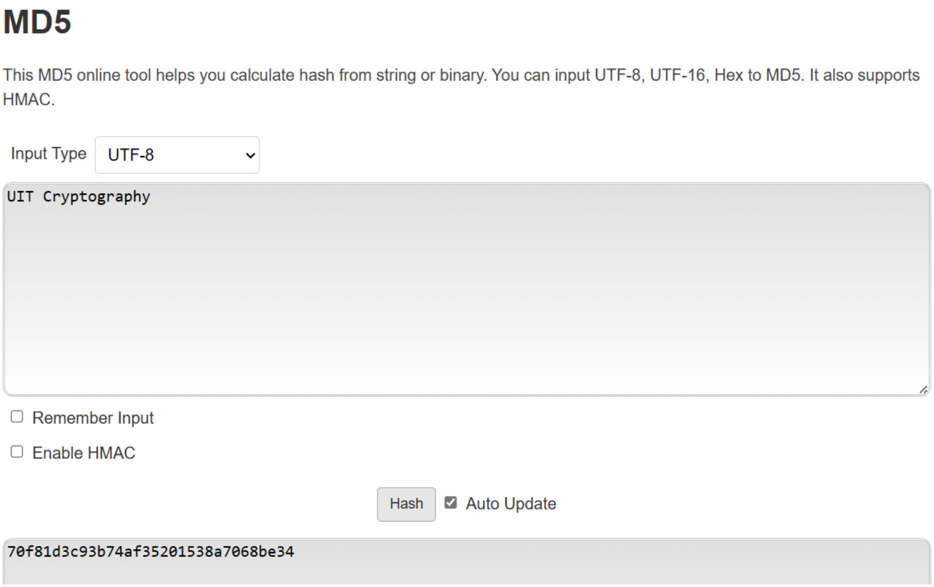
```
Choose your string type (1/2): 1
Enter the message to hash (Text): UIT Cryptography
Would you like to hash by MD5 or SHA-1 or SHA2-2

1. MD5
2. SHA-1
3. SHA-256

Choose hash type (1/2/3): 2
f73f57a04d0dce50b899cdd8252529c0327ef7de
```

This is the hash value of **SHA-256** by “UIT Cryptography”



```
Choose your string type (1/2): 1
Enter the message to hash (Text): UIT Cryptography
Would you like to hash by MD5 or SHA-1 or SHA2-2

1. MD5
2. SHA-1
3. SHA-256

Choose hash type (1/2/3): 3
81a2cd72bc1c7ff058ac25d56252371464d1849d6c3d471f403cc9c86e4229d9
```

Let’s compare the result of out application with others tools online to verify. We gonna use Online Tools (emn178.github.io) to compare.



MD5

This MD5 online tool helps you calculate hash from string or binary. You can input UTF-8, UTF-16, Hex to MD5. It also supports HMAC.

Input Type UTF-8

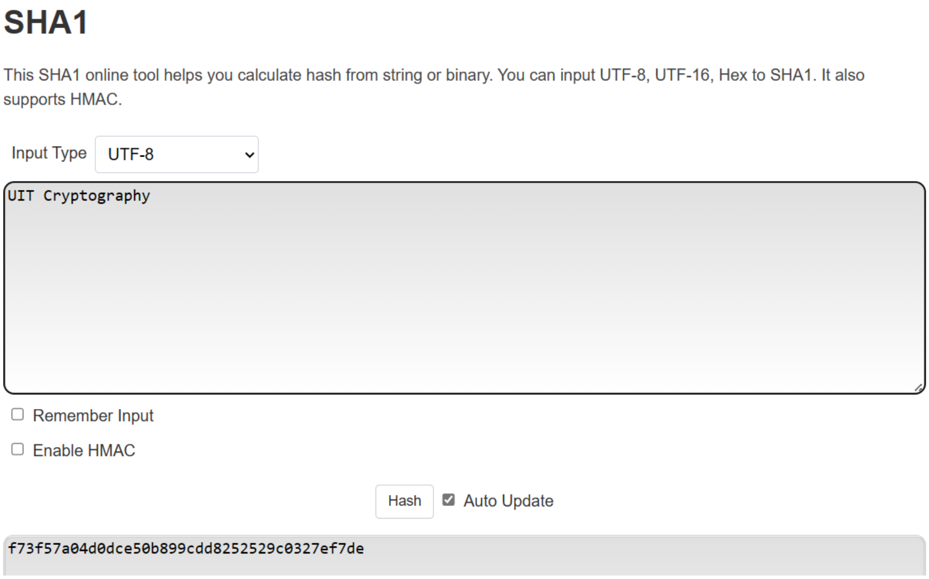
```
UIT Cryptography
```

□ Remember Input

□ Enable HMAC

☑ Auto Update

```
70f81d3c93b74af35201538a7068be34
```



SHA1

This SHA1 online tool helps you calculate hash from string or binary. You can input UTF-8, UTF-16, Hex to SHA1. It also supports HMAC.

Input Type UTF-8

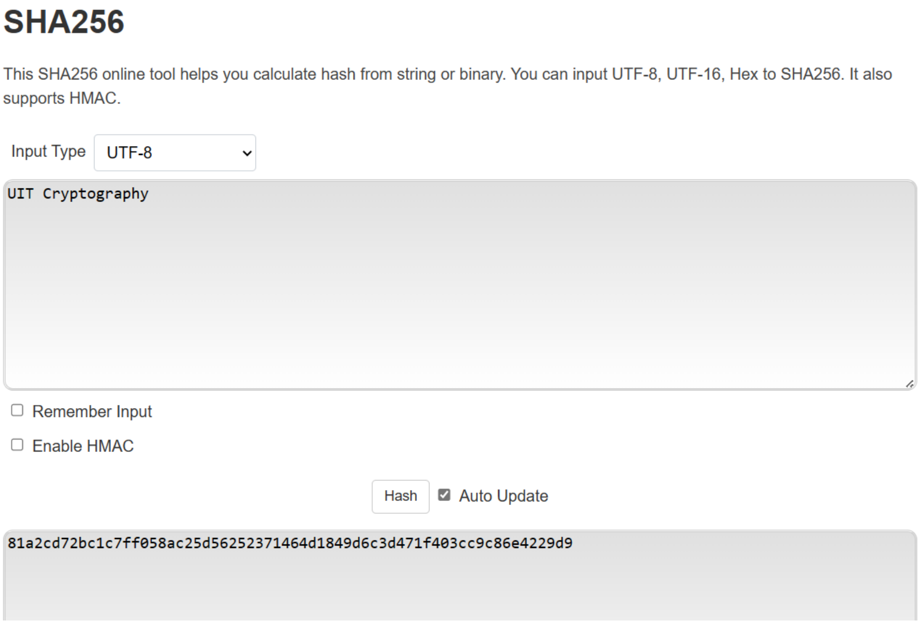
```
UIT Cryptography
```

□ Remember Input

□ Enable HMAC

☑ Auto Update

```
f73f57a04d0dce50b899cdd8252529c0327ef7de
```



SHA256

This SHA256 online tool helps you calculate hash from string or binary. You can input UTF-8, UTF-16, Hex to SHA256. It also supports HMAC.

Input Type UTF-8

```
UIT Cryptography
```

□ Remember Input

□ Enable HMAC

☑ Auto Update

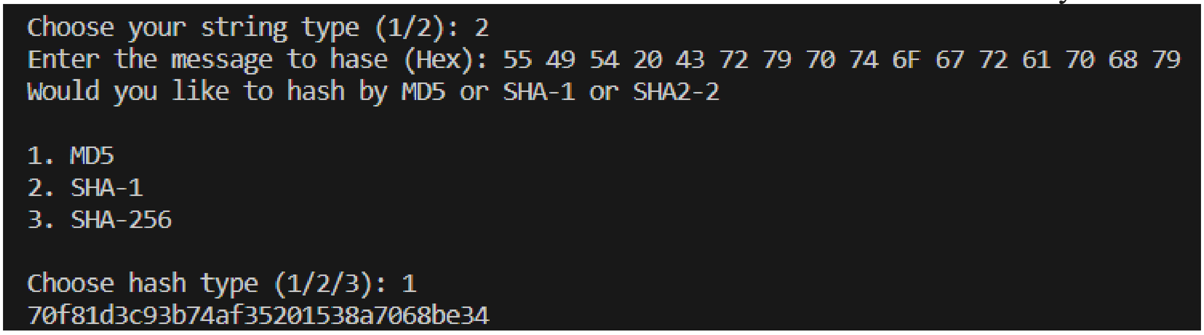
```
81a2cd72bc1c7ff058ac25d56252371464d1849d6c3d471f403cc9c86e4229d9
```

As we can see both result is identical in all 3 MD5, SHA-1 and SHA-256

Hex string

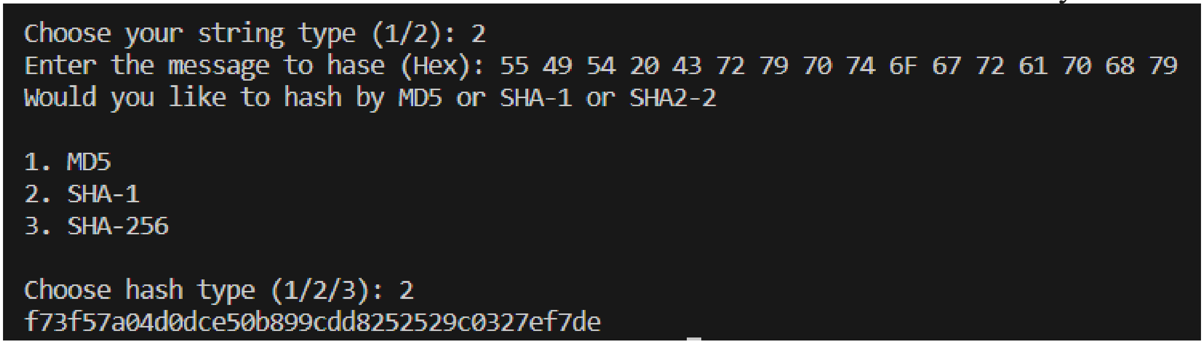
At first, let’s convert “UIT Cryptography” into hex format, we have the result “55 49 54 20 43 72 79 70 74 6F 67 72 61 70 68 79”

This is the hash value of “55 49 54 20 43 72 79 70 74 6F 67 72 61 70 68 79” by MD5



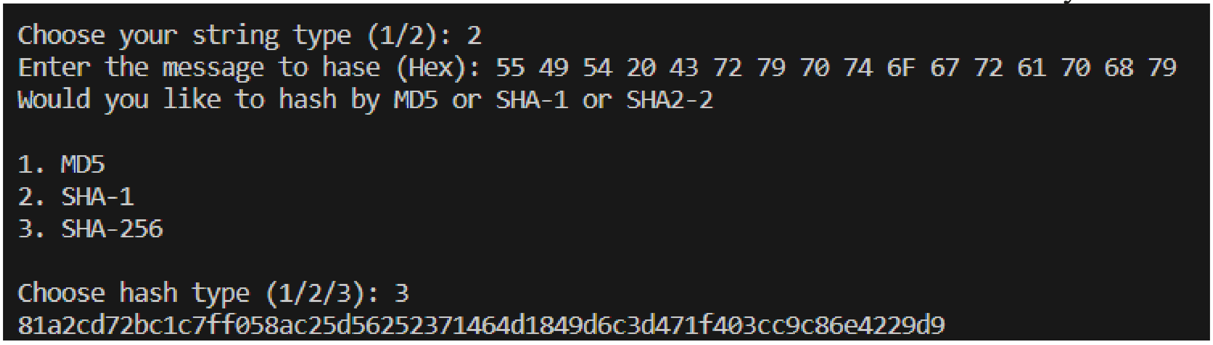
```
Choose your string type (1/2): 2
Enter the message to hase (Hex): 55 49 54 20 43 72 79 70 74 6F 67 72 61 70 68 79
Would you like to hash by MD5 or SHA-1 or SHA2-2

1. MD5
2. SHA-1
3. SHA-256

Choose hash type (1/2/3): 1
70f81d3c93b74af35201538a7068be34
```

This is the hash value of “55 49 54 20 43 72 79 70 74 6F 67 72 61 70 68 79” by SHA-1



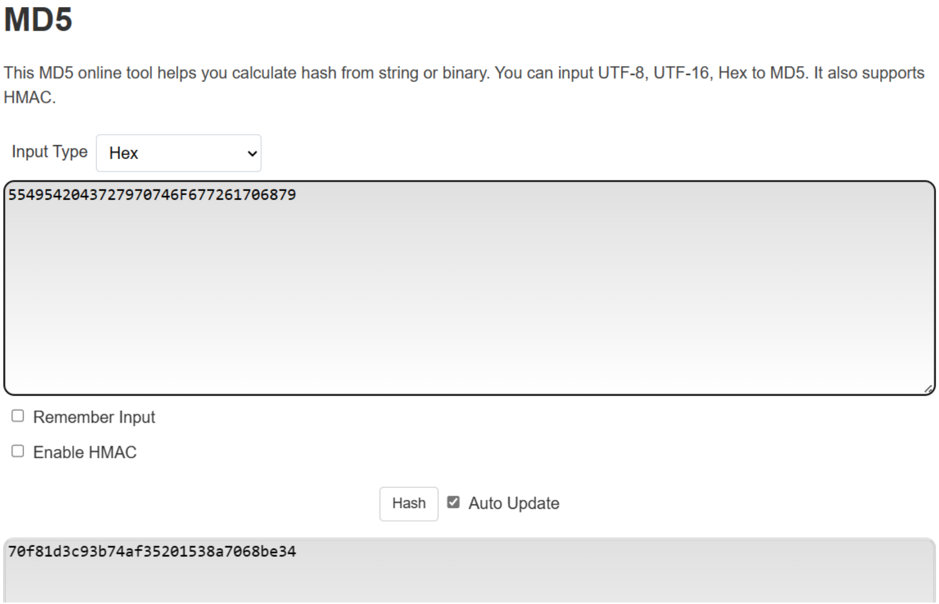
```
Choose your string type (1/2): 2
Enter the message to hase (Hex): 55 49 54 20 43 72 79 70 74 6F 67 72 61 70 68 79
Would you like to hash by MD5 or SHA-1 or SHA2-2

1. MD5
2. SHA-1
3. SHA-256

Choose hash type (1/2/3): 2
f73f57a04d0dce50b899cdd8252529c0327ef7de
```

This is the hash value of “55 49 54 20 43 72 79 70 74 6F 67 72 61 70 68 79” by SHA-256



```
Choose your string type (1/2): 2
Enter the message to hase (Hex): 55 49 54 20 43 72 79 70 74 6F 67 72 61 70 68 79
Would you like to hash by MD5 or SHA-1 or SHA2-2

1. MD5
2. SHA-1
3. SHA-256

Choose hash type (1/2/3): 3
81a2cd72bc1c7ff058ac25d56252371464d1849d6c3d471f403cc9c86e4229d9
```

We also use Online Tools (emn178.github.io) to compare the result with out application.



MD5

This MD5 online tool helps you calculate hash from string or binary. You can input UTF-8, UTF-16, Hex to MD5. It also supports HMAC.

Input Type: Hex

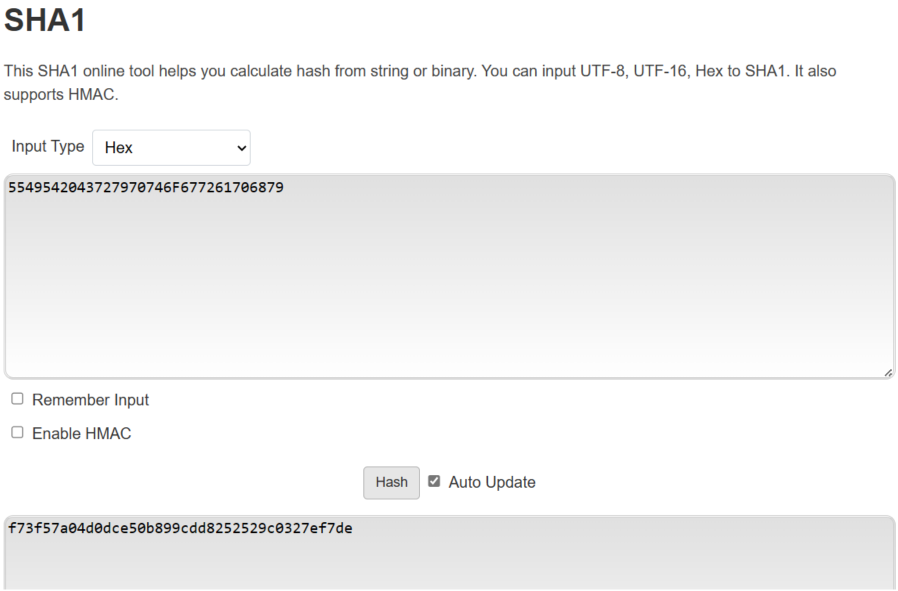
```
5549542043727970746F677261706879
```

☐ Remember Input

☐ Enable HMAC

☑ Auto Update

```
70f81d3c93b74af35201538a7068be34
```



SHA1

This SHA1 online tool helps you calculate hash from string or binary. You can input UTF-8, UTF-16, Hex to SHA1. It also supports HMAC.

Input Type: Hex

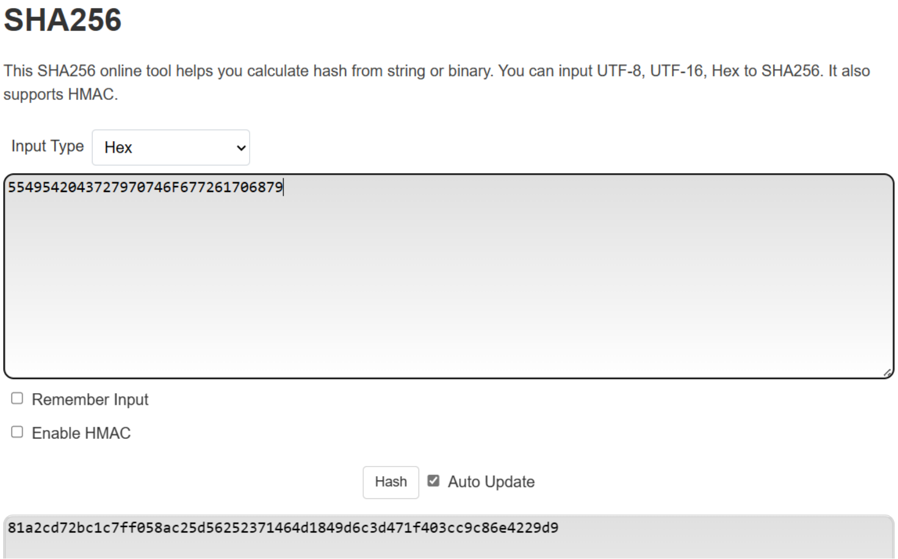
```
5549542043727970746F677261706879
```

☐ Remember Input

☐ Enable HMAC

☑ Auto Update

```
f73f57a04d0dce50b899cdd8252529c0327ef7de
```



SHA256

This SHA256 online tool helps you calculate hash from string or binary. You can input UTF-8, UTF-16, Hex to SHA256. It also supports HMAC.

Input Type: Hex

```
5549542043727970746F677261706879
```

☐ Remember Input

☐ Enable HMAC

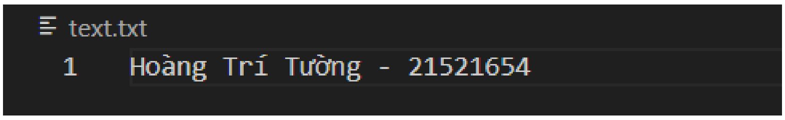
☑ Auto Update

```
81a2cd72bc1c7ff058ac25d56252371464d1849d6c3d471f403cc9c86e4229d9
```

The result of out application not only identical as the tool but also the same as the result of the input in text string “UIT Cryptography”

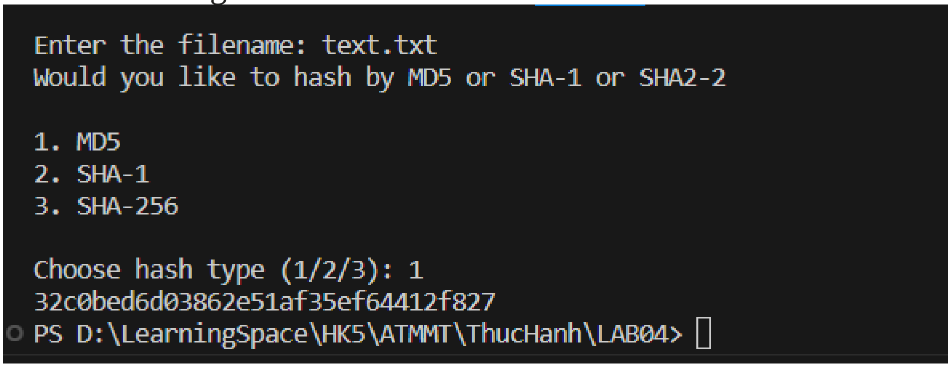
File

The Content of text file a.k.a text.txt in this scenario



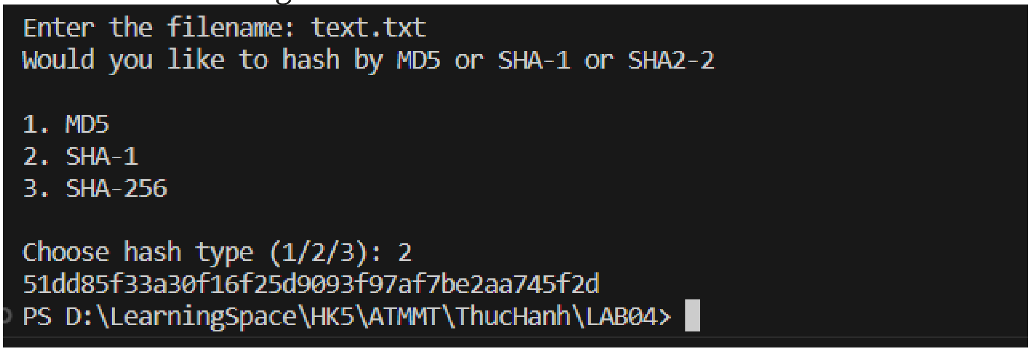
```
text.txt
1    Hoàng Trí Tường - 21521654
```

The MD5 hash value of original text file



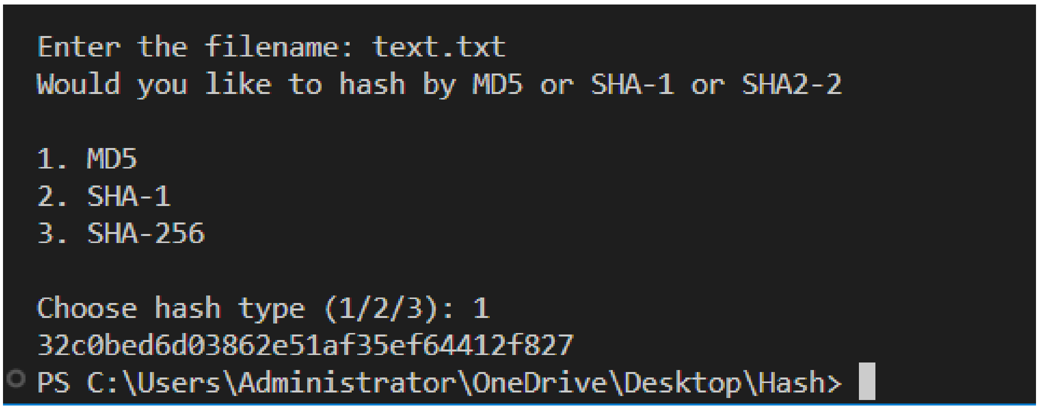
```
Enter the filename: text.txt
Would you like to hash by MD5 or SHA-1 or SHA2-2

1. MD5
2. SHA-1
3. SHA-256

Choose hash type (1/2/3): 1
32c0bed6d03862e51af35ef64412f827
PS D:\LearningSpace\HK5\ATMMT\ThucHanh\LAB04>
```

The SHA-1 hash value of original text file



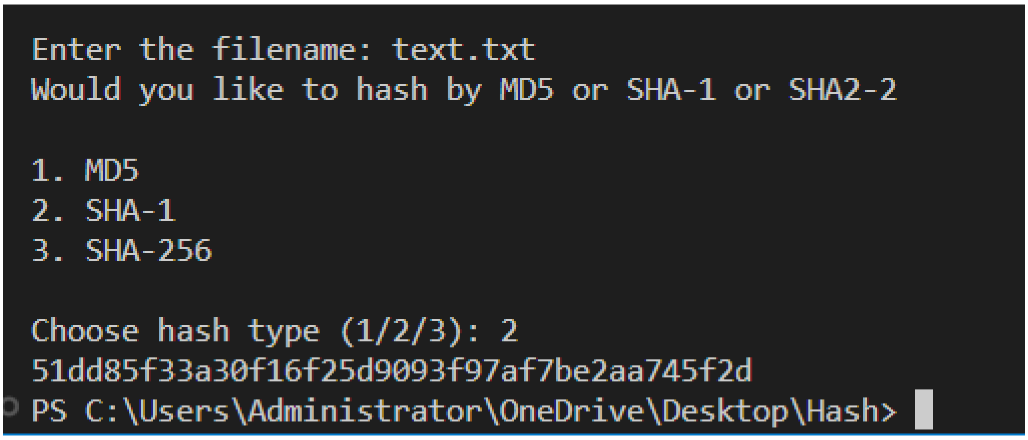
```
Enter the filename: text.txt
Would you like to hash by MD5 or SHA-1 or SHA2-2

1. MD5
2. SHA-1
3. SHA-256

Choose hash type (1/2/3): 2
51dd85f33a30f16f25d9093f97af7be2aa745f2d
PS D:\LearningSpace\HK5\ATMMT\ThucHanh\LAB04>
```

The MD5 hash value of downloaded text file



```
Enter the filename: text.txt
Would you like to hash by MD5 or SHA-1 or SHA2-2

1. MD5
2. SHA-1
3. SHA-256

Choose hash type (1/2/3): 1
32c0bed6d03862e51af35ef64412f827
PS C:\Users\Administrator\OneDrive\Desktop\Hash>
```

The SHA-1 hash value of downloaded text file

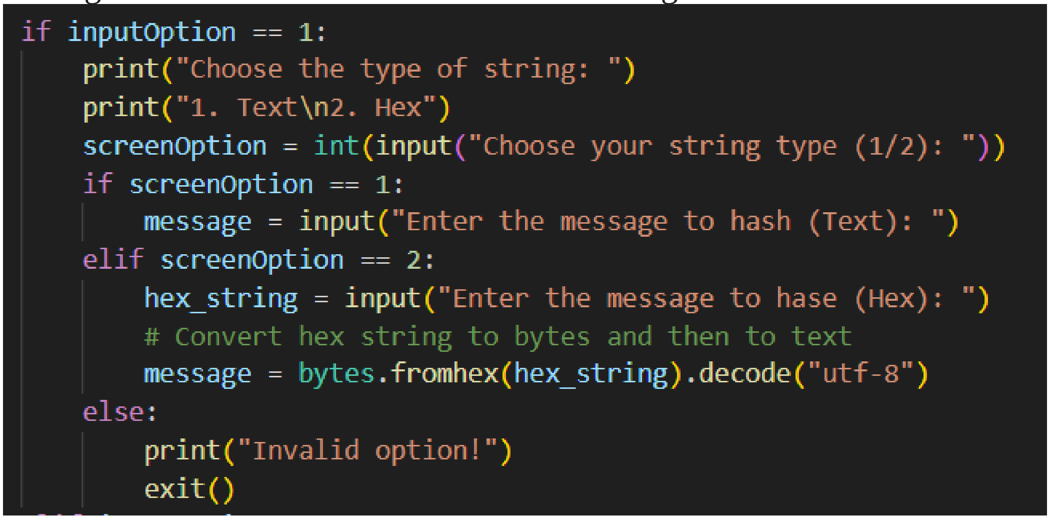


```
Enter the filename: text.txt
Would you like to hash by MD5 or SHA-1 or SHA2-2

1. MD5
2. SHA-1
3. SHA-256

Choose hash type (1/2/3): 2
51dd85f33a30f16f25d9093f97af7be2aa745f2d
PS C:\Users\Administrator\OneDrive\Desktop\Hash>
```

The hash values on original file and downloaded file are similar.

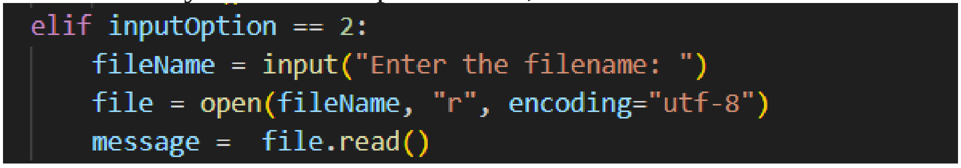
1.2 Code Explanation.

The code below is when you choose input from screen it is just take the input from screen and if the string is in hex format it will convert hex string to text.



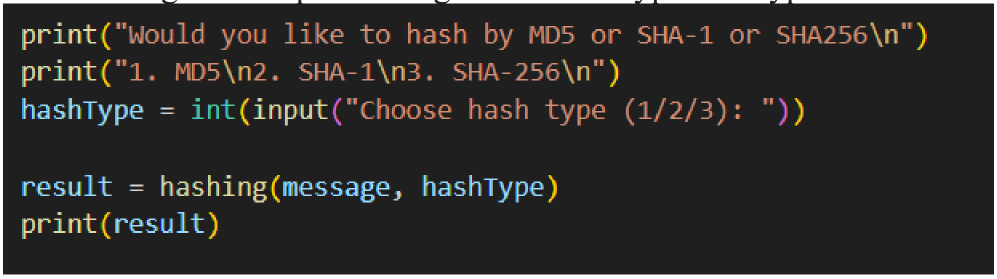
```
if inputOption == 1:
    print("Choose the type of string: ")
    print("1. Text\n2. Hex")
    screenOption = int(input("Choose your string type (1/2): "))
    if screenOption == 1:
        message = input("Enter the message to hash (Text): ")
    elif screenOption == 2:
        hex_string = input("Enter the message to hase (Hex): ")
        # Convert hex string to bytes and then to text
        message = bytes.fromhex(hex_string).decode("utf-8")
    else:
        print("Invalid option!")
        exit()
```

The code below is when you choose input the file, it takes the file and read the content.



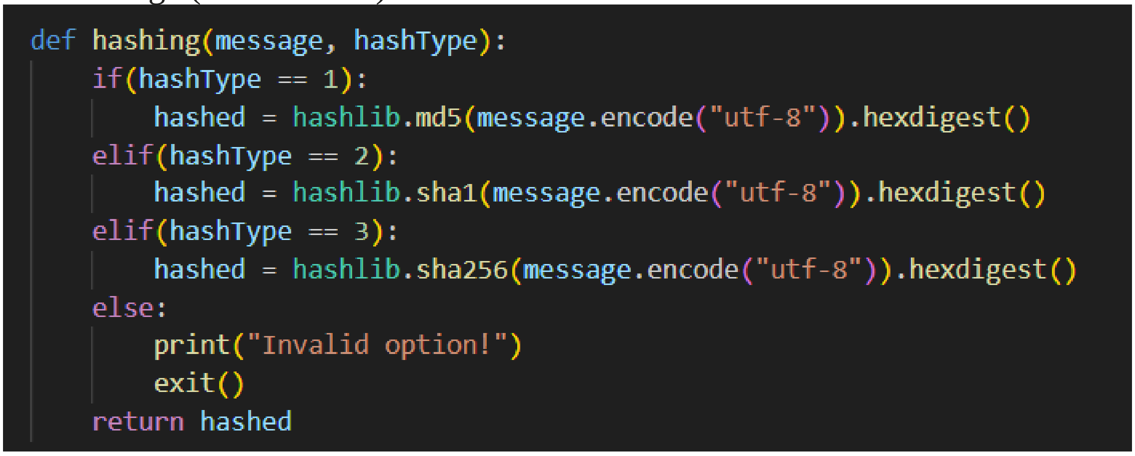
```
elif inputOption == 2:
    fileName = input("Enter the filename: ")
    file = open(fileName, "r", encoding="utf-8")
    message =  file.read()
```

The code asked you to choose the hash function the call the hashing function take the arguments are “message” the input message and “hashType” the type of hash function.



```
print("Would you like to hash by MD5 or SHA-1 or SHA256\n")
print("1. MD5\n2. SHA-1\n3. SHA-256\n")
hashType = int(input("Choose hash type (1/2/3): "))

result = hashing(message, hashType)
print(result)
```

Take the message and hashType then hashing the right type for the message then just return the hased message (encode utf-8).



```python
def hashing(message, hashType):
    if(hashType == 1):
        hashed = hashlib.md5(message.encode("utf-8")).hexdigest()
    elif(hashType == 2):
        hashed = hashlib.sha1(message.encode("utf-8")).hexdigest()
    elif(hashType == 3):
        hashed = hashlib.sha256(message.encode("utf-8")).hexdigest()
    else:
        print("Invalid option!")
        exit()
    return hashed
```

2. Hash properties: One-way vs Collision-free

Task 2.1

1. Two HEX messages

Message 1 and 2 are difference in 3 bytes (6 characters)

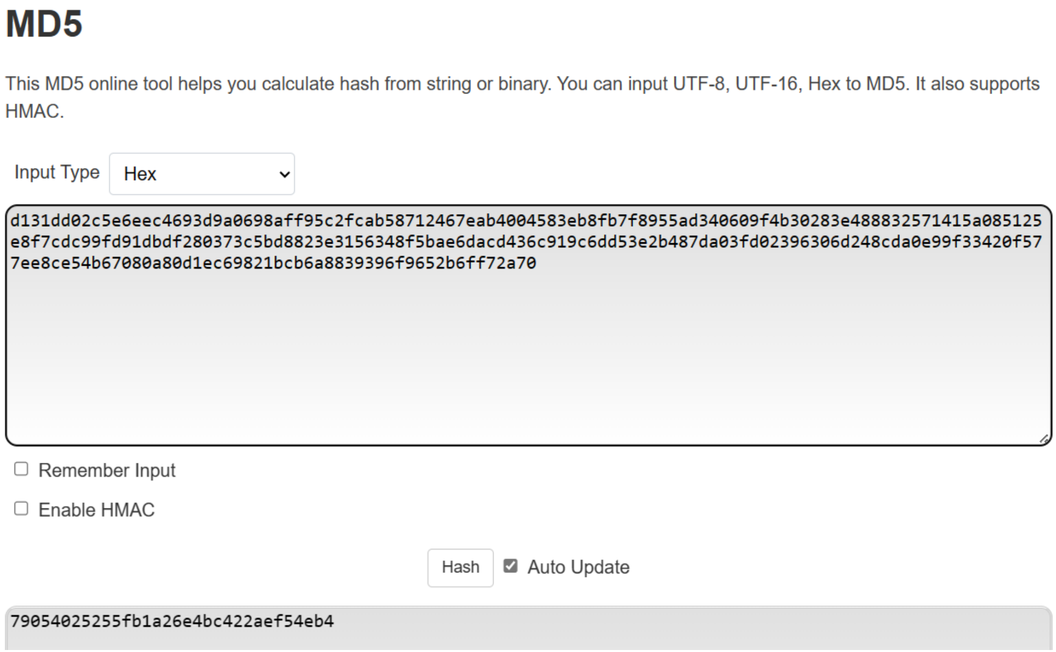
Message 1:

d131dd02c5e6eec4693d9a0698aff95c2fcab58712467eab4004583eb8fb7f8955ad340609f4b30283e488832571415a085125e8f7cdc99fd91dbdf280373c5bd8823e3156348f5bae6dacd436c919c6dd53e2b487da03fd02396306d248cda0e99f33420f577ee8ce54b67080a80d1ec69821bcb6a8839396f9652b6ff72a70

Message 2:

d131dd02c5e6eec4693d9a0698aff95c2fcab50712467eab4004583eb8fb7f8955ad340609f4b30283e4888325f1415a085125e8f7cdc99fd91dbd7280373c5bd8823e3156348f5bae6dacd436c919c6dd53e23487da03fd02396306d248cda0e99f33420f577ee8ce54b67080280d1ec69821bcb6a8839396f965ab6ff72a70

Message 1:



MD5

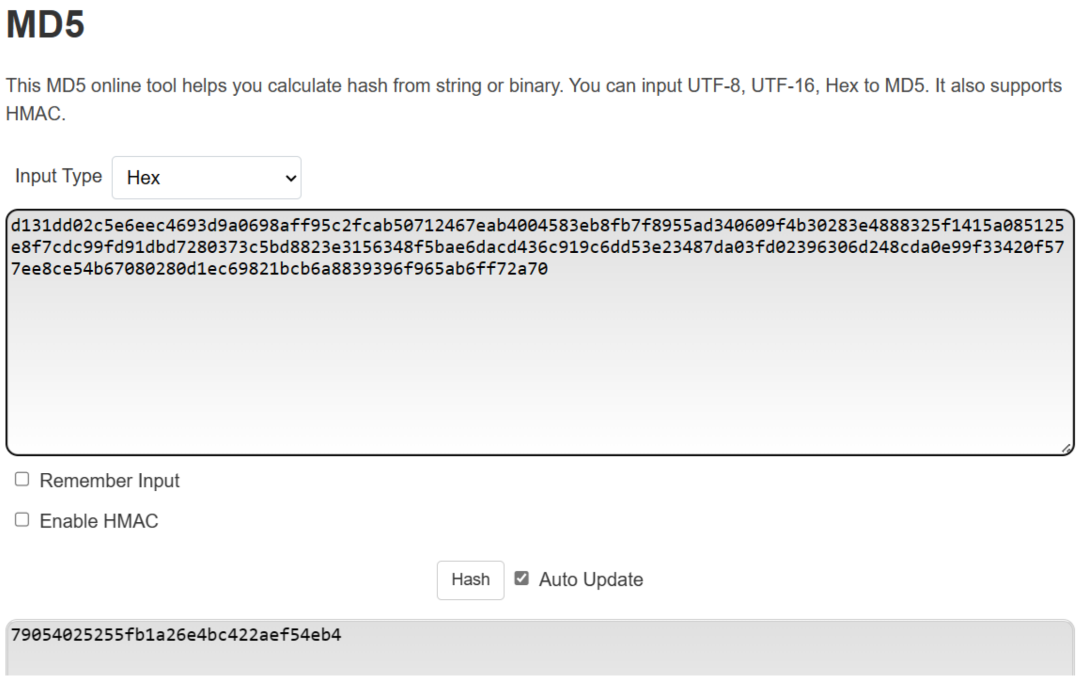
This MD5 online tool helps you calculate hash from string or binary. You can input UTF-8, UTF-16, Hex to MD5. It also supports HMAC.

Input Type: Hex

```
d131dd02c5e6eec4693d9a0698aff95c2fcab58712467eab4004583eb8fb7f8955ad340609f4b30283e488832571415a085125
e8f7cdc99fd91dbdf280373c5bd8823e3156348f5bae6dacd436c919c6dd53e2b487da03fd02396306d248cda0e99f33420f57
7ee8ce54b67080a80d1ec69821bcb6a8839396f9652b6ff72a70
```

```
79054025255fb1a26e4bc422aef54eb4
```

Message 2:



MD5

This MD5 online tool helps you calculate hash from string or binary. You can input UTF-8, UTF-16, Hex to MD5. It also supports HMAC.

Input Type: Hex

```
d131dd02c5e6eec4693d9a0698aff95c2fcab50712467eab4004583eb8fb7f8955ad340609f4b30283e4888325f1415a085125
e8f7cdc99fd91dbd7280373c5bd8823e3156348f5bae6dacd436c919c6dd53e23487da03fd02396306d248cda0e99f33420f57
7ee8ce54b67080280d1ec69821bcb6a8839396f965ab6ff72a70
```

□ Remember Input

□ Enable HMAC

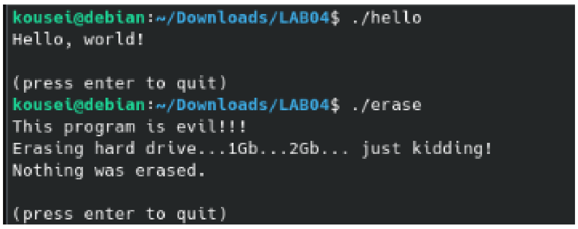
☑ Auto Update

```
79054025255fb1a26e4bc422aef54eb4
```

The MD5 hash values for both messages are the same even though the messages are different.

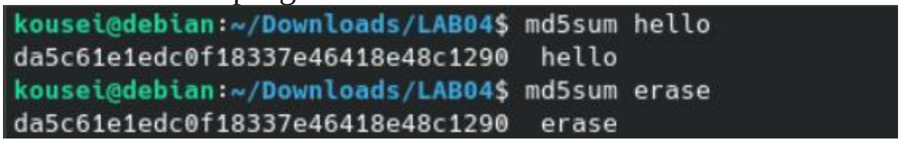
2. Two executable programs

The contents of two programs are different.



```
kousei@debian:~/Downloads/LAB04$ ./hello
Hello, world!

(press enter to quit)
kousei@debian:~/Downloads/LAB04$ ./erase
This program is evil!!!
Erasing hard drive...1Gb...2Gb... just kidding!
Nothing was erased.

(press enter to quit)
```

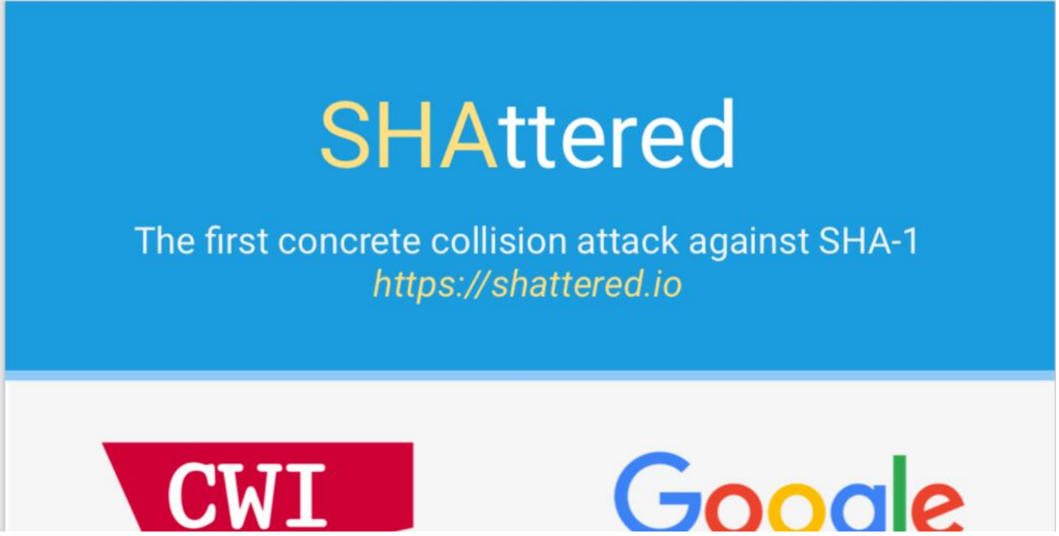
The MD5 hash values for two programs are similar.



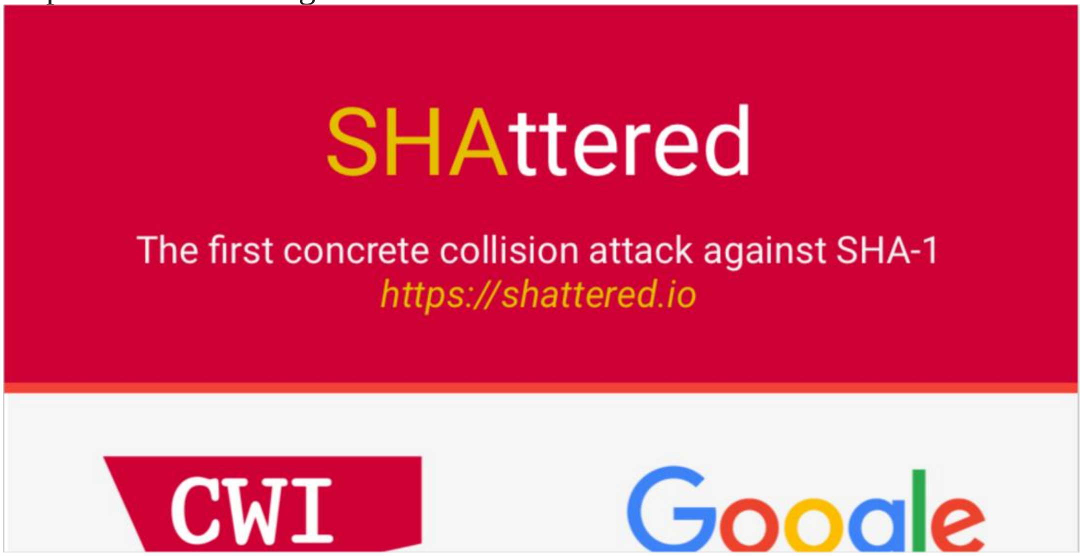
```
kousei@debian:~/Downloads/LAB04$ md5sum hello
da5c61e1edc0f18337e46418e48c1290  hello
kousei@debian:~/Downloads/LAB04$ md5sum erase
da5c61e1edc0f18337e46418e48c1290  erase
```

3. Two PDF files

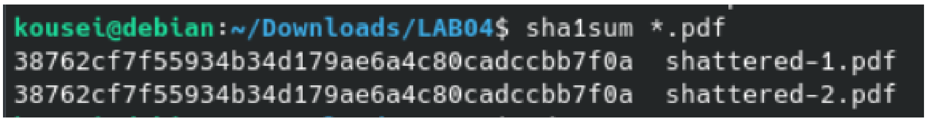
shatterd-1.pdf has a blue background.



shatterd-2.pdf has a red background



The SHA-1 hash values for both PDF files are identical



```
kousei@debian:~/Downloads/LAB04$ sha1sum *.pdf
38762cf7f55934b34d179ae6a4c80cadccbb7f0a  shattered-1.pdf
38762cf7f55934b34d179ae6a4c80cadccbb7f0a  shattered-2.pdf
```

Conclusion:

The fact that two different HEX messages, two different executable files, and two different PDF files have the same MD5 and SHA-1 hash values indicates that these hash functions are not collision resistant. This means that it is possible to find two different inputs that produce the same hash output. This is a serious security vulnerability, as it can be used to forge digital signatures or tamper with data without being detected. Steps should be taken to migrate to more secure hash functions and to avoid using MD5 or SHA-1 for applications that require strong collision resistance.

MD5 Collisions:

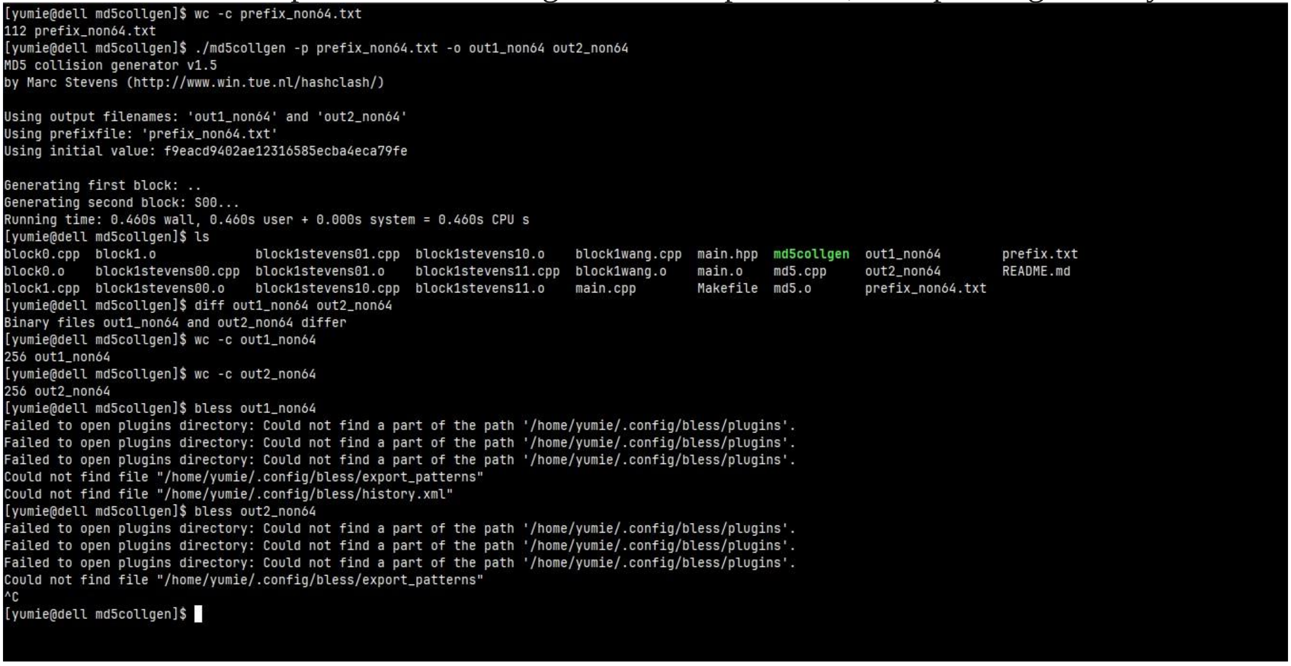
The reason why MD5 collisions are possible is because the algorithm is not computationally infeasible. This means that it is possible to find collisions by brute-force searching. The researchers who published the MD5 collision attack used a technique called differential cryptanalysis to find their collisions. This is a mathematical technique that can be used to find weaknesses in hash functions.

SHA-1 Collisions:

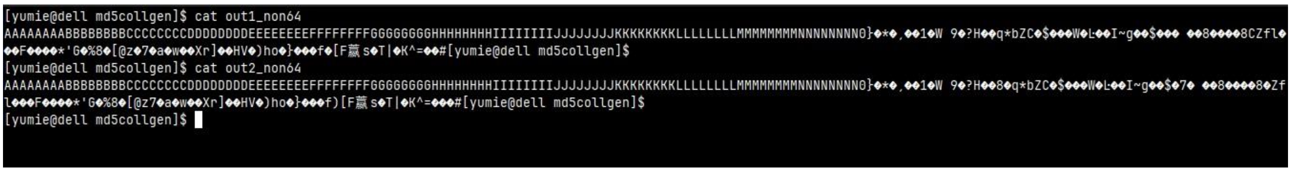
The reason why SHA-1 collisions are possible is similar to the reason why MD5 collisions are possible. The algorithm is not computationally infeasible, and it is possible to find collisions by brute-force searching. The researchers who published the SHA-1 collision attack used a technique called multi-target preimage attack to find their collisions. This is a more sophisticated attack than differential cryptanalysis, and it can be used to find collisions for hash functions that are more resistant to differential cryptanalysis.

Task 2.2

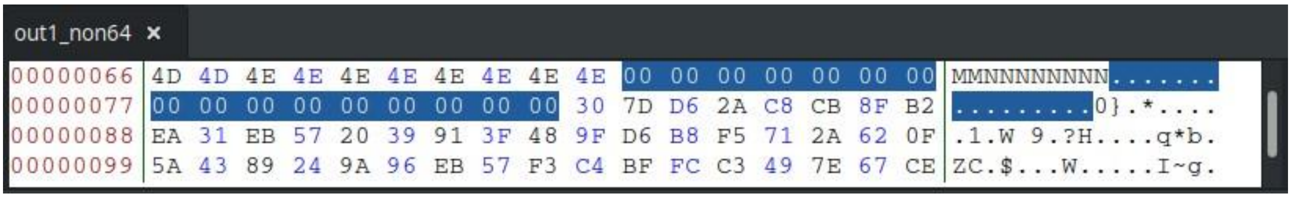
1. If the length of your prefix file is not multiple of 64, what is going to happen?
   Zeros will be padded so that length is a multiple of 64, here padding is 16 bytes



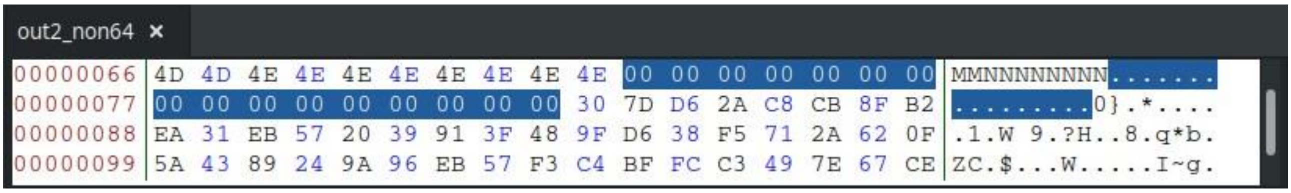
```
[yumie@dell md5collgen]$ wc -c prefix_non64.txt
112 prefix_non64.txt
[yumie@dell md5collgen]$ ./md5collgen -p prefix_non64.txt -o out1_non64 out2_non64
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'out1_non64' and 'out2_non64'
Using prefixfile: 'prefix_non64.txt'
Using initial value: f9eacd9402ae12316585ecba4eca79fe

Generating first block: ..
Generating second block: S00...
Running time: 0.460s wall, 0.460s user + 0.000s system = 0.460s CPU s
[yumie@dell md5collgen]$ ls
block0.cpp  block1.o             block1stevens01.cpp  block1stevens10.o    block1wang.cpp  main.hpp  md5collgen  out1_non64        prefix.txt
block0.o    block1stevens00.cpp  block1stevens01.o    block1stevens11.cpp  block1wang.o    main.o    md5.cpp     out2_non64        README.md
block1.cpp  block1stevens00.o    block1stevens10.cpp  block1stevens11.o    main.cpp        Makefile  md5.o       prefix_non64.txt
[yumie@dell md5collgen]$ diff out1_non64 out2_non64
Binary files out1_non64 and out2_non64 differ
[yumie@dell md5collgen]$ wc -c out1_non64
256 out1_non64
[yumie@dell md5collgen]$ wc -c out2_non64
256 out2_non64
[yumie@dell md5collgen]$ bless out1_non64
Failed to open plugins directory: Could not find a part of the path '/home/yumie/.config/bless/plugins'.
Failed to open plugins directory: Could not find a part of the path '/home/yumie/.config/bless/plugins'.
Failed to open plugins directory: Could not find a part of the path '/home/yumie/.config/bless/plugins'.
Could not find file "/home/yumie/.config/bless/export_patterns"
Could not find file "/home/yumie/.config/bless/history.xml"
[yumie@dell md5collgen]$ bless out2_non64
Failed to open plugins directory: Could not find a part of the path '/home/yumie/.config/bless/plugins'.
Failed to open plugins directory: Could not find a part of the path '/home/yumie/.config/bless/plugins'.
Failed to open plugins directory: Could not find a part of the path '/home/yumie/.config/bless/plugins'.
Could not find file "/home/yumie/.config/bless/export_patterns"
^C
[yumie@dell md5collgen]$ █
```



```
[yumie@dell md5collgen]$ cat out1_non64
AAAAAAAABBBBBBBBCCCCCCCCDDDDDDDDEEEEEEEEFFFFFFFFGGGGGGGGHHHHHHHHIIIIIIIIJJJJJJJJKKKKKKKKLLLLLLLLMMMMMMMMNNNNNNNN0}�*�,��1�W 9�?H��q*bZC�$���W�L��I~g��$��� ��8����8CZfl�
��F����*'G�%8�[@z�7�a�w��Xr]��HV�)ho�}���f�[F蠃s�T|�K^=��#[yumie@dell md5collgen]$
[yumie@dell md5collgen]$ cat out2_non64
AAAAAAAABBBBBBBBCCCCCCCCDDDDDDDDEEEEEEEEFFFFFFFFGGGGGGGGHHHHHHHHIIIIIIIIJJJJJJJJKKKKKKKKLLLLLLLLMMMMMMMMNNNNNNNN0}�*�,��1�W 9�?H��8�q*bZC�$���W�L��I~g��$�7� ��8����8�Zf
l���F����*'G�%8�[@z7�a�w��Xr]��HV�)ho�}���f)[F蠃s�T|�K^=���#[yumie@dell md5collgen]$
[yumie@dell md5collgen]$ █
```

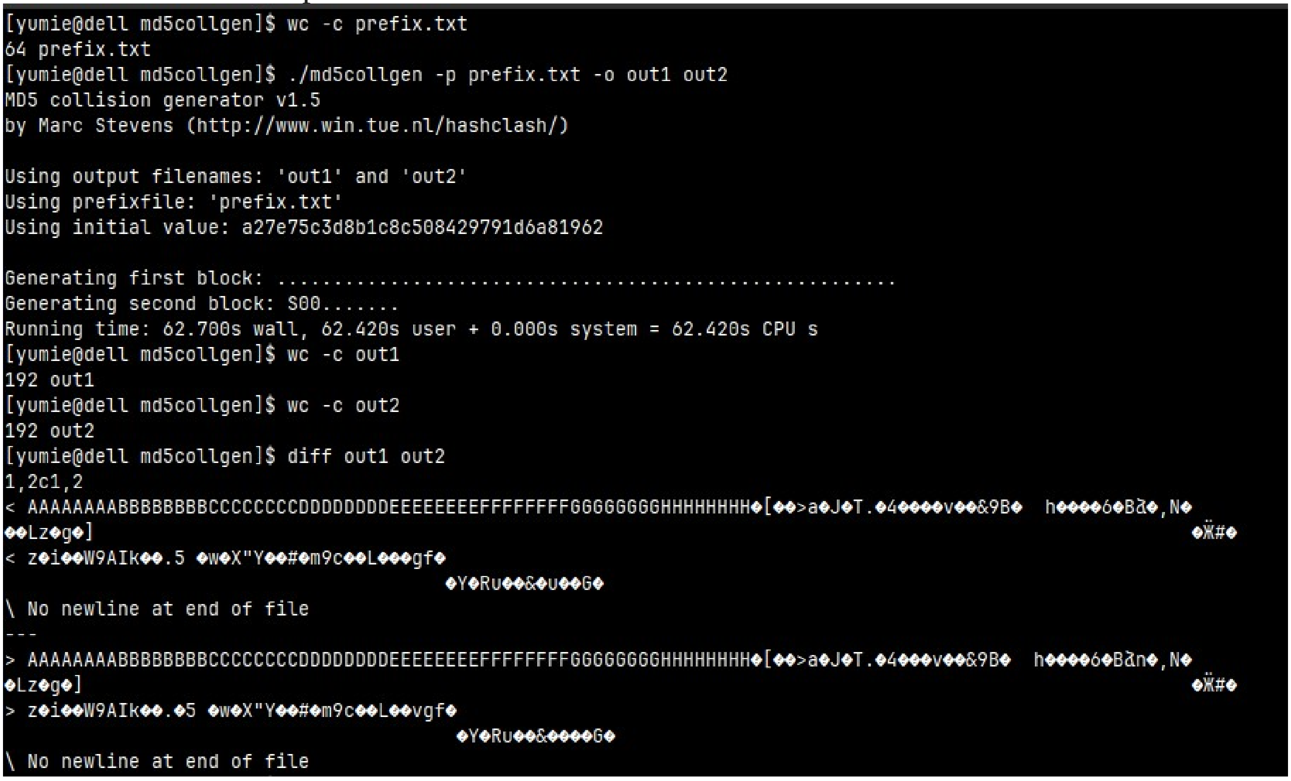


```
00000066 4D 4D 4E 4E 4E 4E 4E 4E 4E 4E 00 00 00 00 00 00 00 MMNNNNNNNN.......
00000077 00 00 00 00 00 00 00 00 00 30 7D D6 2A C8 CB 8F B2 .........0}.*....
00000088 EA 31 EB 57 20 39 91 3F 48 9F D6 B8 F5 71 2A 62 0F .1.W 9.?H....q*b.
00000099 5A 43 89 24 9A 96 EB 57 F3 C4 BF FC C3 49 7E 67 CE ZC.$...W.....I~g.
```

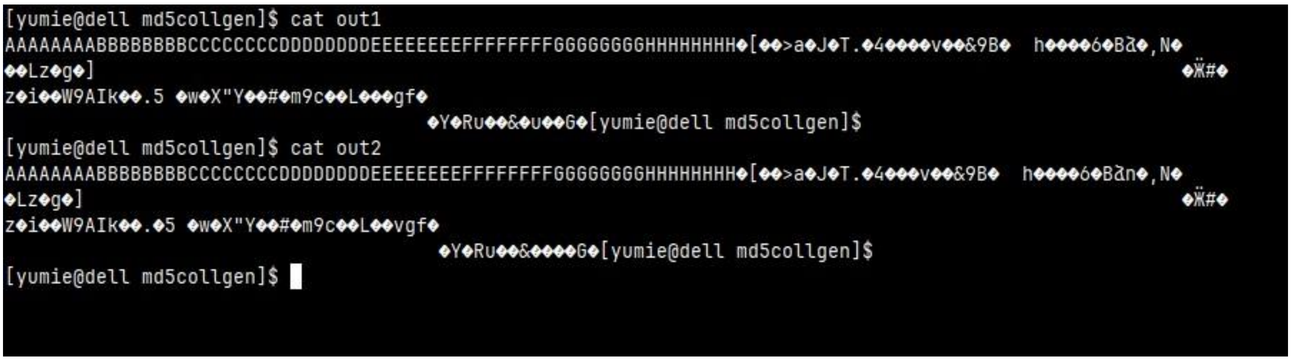


```
00000066 4D 4D 4E 4E 4E 4E 4E 4E 4E 4E 00 00 00 00 00 00 00 MMNNNNNNNN.......
00000077 00 00 00 00 00 00 00 00 00 30 7D D6 2A C8 CB 8F B2 .........0}.*....
00000088 EA 31 EB 57 20 39 91 3F 48 9F D6 38 F5 71 2A 62 0F .1.W 9.?H..8.q*b.
00000099 5A 43 89 24 9A 96 EB 57 F3 C4 BF FC C3 49 7E 67 CE ZC.$...W.....I~g.
```

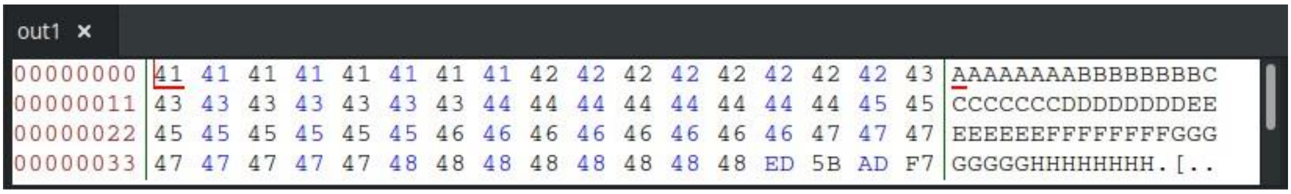
2. Create a prefix file with exactly 64 bytes, and run the collision tool again, and see what happens.
   None of zeros padded



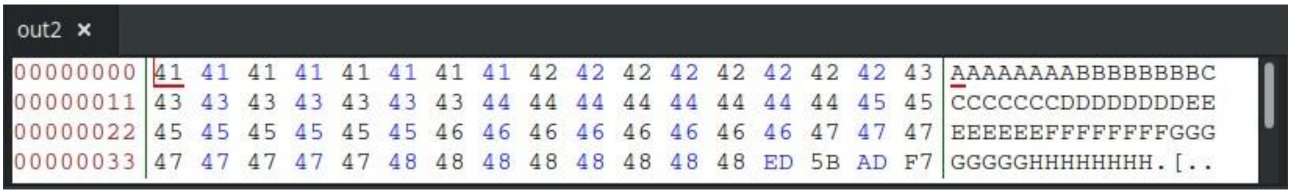
```
[yumie@dell md5collgen]$ wc -c prefix.txt
64 prefix.txt
[yumie@dell md5collgen]$ ./md5collgen -p prefix.txt -o out1 out2
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'out1' and 'out2'
Using prefixfile: 'prefix.txt'
Using initial value: a27e75c3d8b1c8c508429791d6a81962

Generating first block: ....................................................
Generating second block: S00.......
Running time: 62.700s wall, 62.420s user + 0.000s system = 62.420s CPU s
[yumie@dell md5collgen]$ wc -c out1
192 out1
[yumie@dell md5collgen]$ wc -c out2
192 out2
[yumie@dell md5collgen]$ diff out1 out2
1,2c1,2
< AAAAAAAABBBBBBBBCCCCCCCCDDDDDDDDEEEEEEEEFFFFFFFFGGGGGGGGHHHHHHHH�[��>a�J�T.�4����v��&9B�  h����6�Bd�,N�
��Lz�g�]                                                                                                  �Ӂ#�
< z�i��W9AIk��.5 �w�X"Y��#�m9c��L���gf�
                                     �Y�Ru��&�u��G�
\ No newline at end of file
---
> AAAAAAAABBBBBBBBCCCCCCCCDDDDDDDDEEEEEEEEFFFFFFFFGGGGGGGGHHHHHHHH�[��>a�J�T.�4���v��&9B�  h����6�Bdn�,N�
�Lz�g�]                                                                                                   �Ӂ#�
> z�i��W9AIk��.�5 �w�X"Y��#�m9c��L��vgf�
                                      �Y�Ru��&����G�
\ No newline at end of file
```



```
[yumie@dell md5collgen]$ cat out1
AAAAAAAABBBBBBBBCCCCCCCCDDDDDDDDEEEEEEEEFFFFFFFFGGGGGGGGHHHHHHHH�[��>a�J�T.�4����v��&9B�  h����6�Bd�,N�
��Lz�g�]                                                                                                 �Ӂ#�
z�i��W9AIk��.5 �w�X"Y��#�m9c��L���gf�
                                   �Y�Ru��&�u��G�[yumie@dell md5collgen]$
[yumie@dell md5collgen]$ cat out2
AAAAAAAABBBBBBBBCCCCCCCCDDDDDDDDEEEEEEEEFFFFFFFFGGGGGGGGHHHHHHHH�[��>a�J�T.�4���v��&9B�  h����6�Bdn�,N�
�Lz�g�]                                                                                                  �Ӂ#�
z�i��W9AIk��.�5 �w�X"Y��#�m9c��L��vgf�
                                    �Y�Ru��&����G�[yumie@dell md5collgen]$
[yumie@dell md5collgen]$ █
```



```
00000000 41 41 41 41 41 41 41 41 42 42 42 42 42 42 42 42 43 AAAAAAAABBBBBBBBC
00000011 43 43 43 43 43 43 43 44 44 44 44 44 44 44 44 45 45 CCCCCCCDDDDDDDDEE
00000022 45 45 45 45 45 45 46 46 46 46 46 46 46 46 47 47 47 EEEEEEFFFFFFFFGGG
00000033 47 47 47 47 47 48 48 48 48 48 48 48 48 ED 5B AD F7 GGGGGHHHHHHHH.[..
```



```
00000000 41 41 41 41 41 41 41 41 42 42 42 42 42 42 42 42 43 AAAAAAAABBBBBBBBC
00000011 43 43 43 43 43 43 43 44 44 44 44 44 44 44 44 45 45 CCCCCCCDDDDDDDDEE
00000022 45 45 45 45 45 45 46 46 46 46 46 46 46 46 47 47 47 EEEEEEFFFFFFFFGGG
00000033 47 47 47 47 47 48 48 48 48 48 48 48 48 ED 5B AD F7 GGGGGHHHHHHHH.[..
```

3. Can one make 2 different files get the same hash by appending stuff? Explain.
   File out1_non64 has length of 256 bytes, meanwhile file out1 has length of 192 bytes, so we can conclude that tool require additional 128 bytes to produce hash-collision

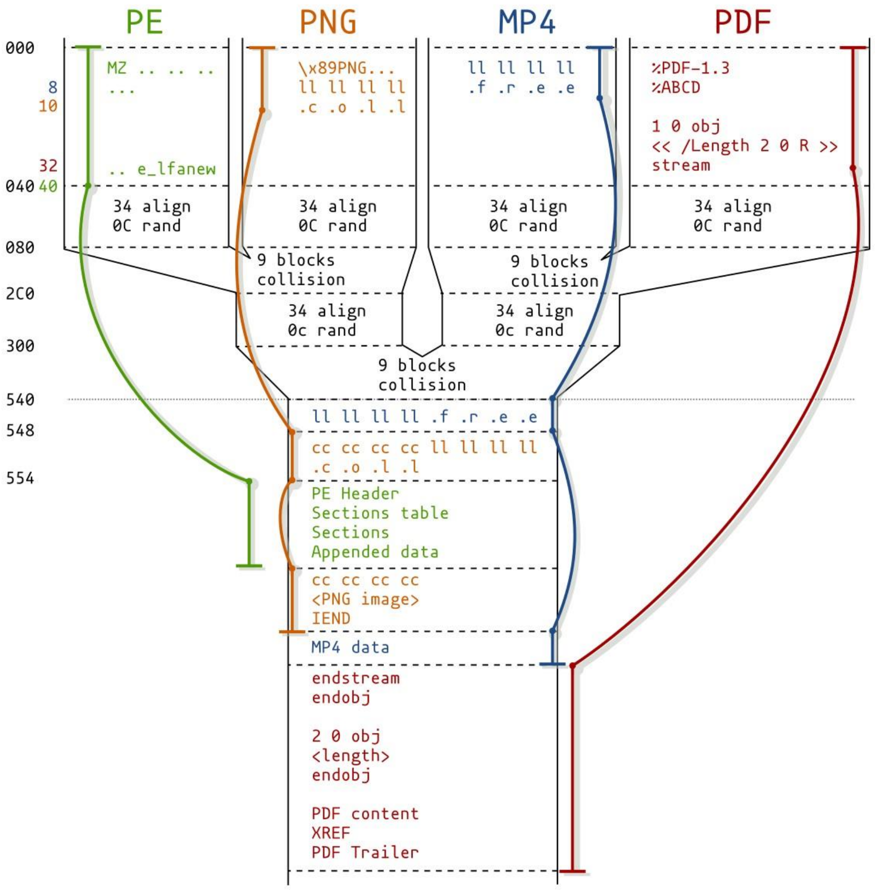
Task 2.3

It is possible to create two different files with arbitrary contents and the same hash. This is known as a hash collision. Hash collisions are a serious security vulnerability, as they can be used to forge digital signatures or tamper with data without being detected.

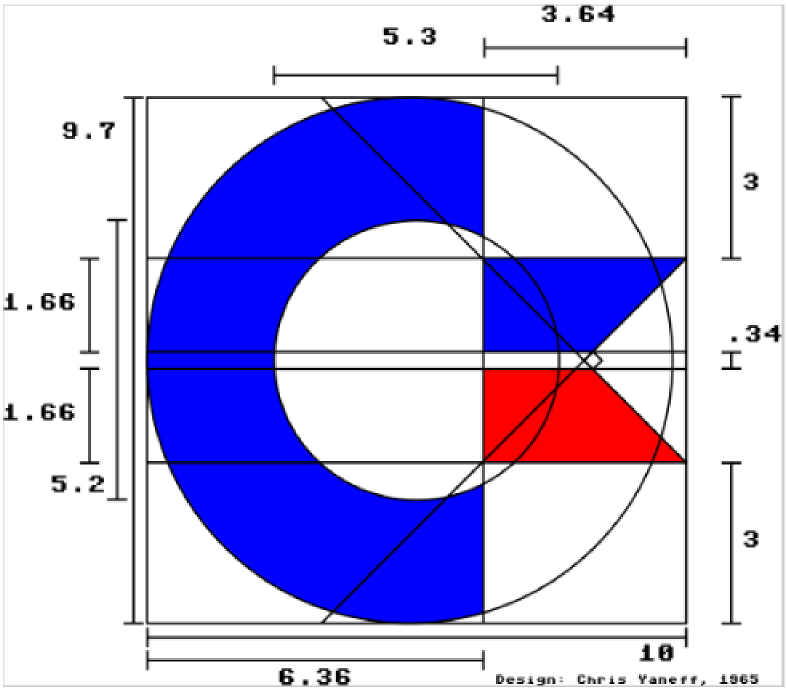
One example of how a hacker could abuse a hash collision is to create a malicious file that has the same hash value as a legitimate file. The hacker could then replace the legitimate file with the malicious file, and the system would not be able to detect the difference. This could allow the hacker to execute arbitrary code on the system.

Take pileup.png and pileup.pdf at Hash collisions and exploitations for example

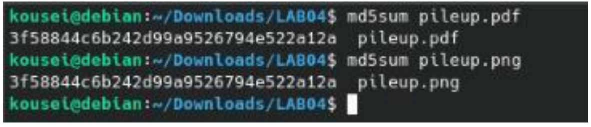
- pileup.png content



- pileup.pdf content.



The MD5 hash values of these two files are both: 3f58844c6b242d99a9526794e522a12a. This means that the two files have the same hash value, even though they have different contents and different formats,

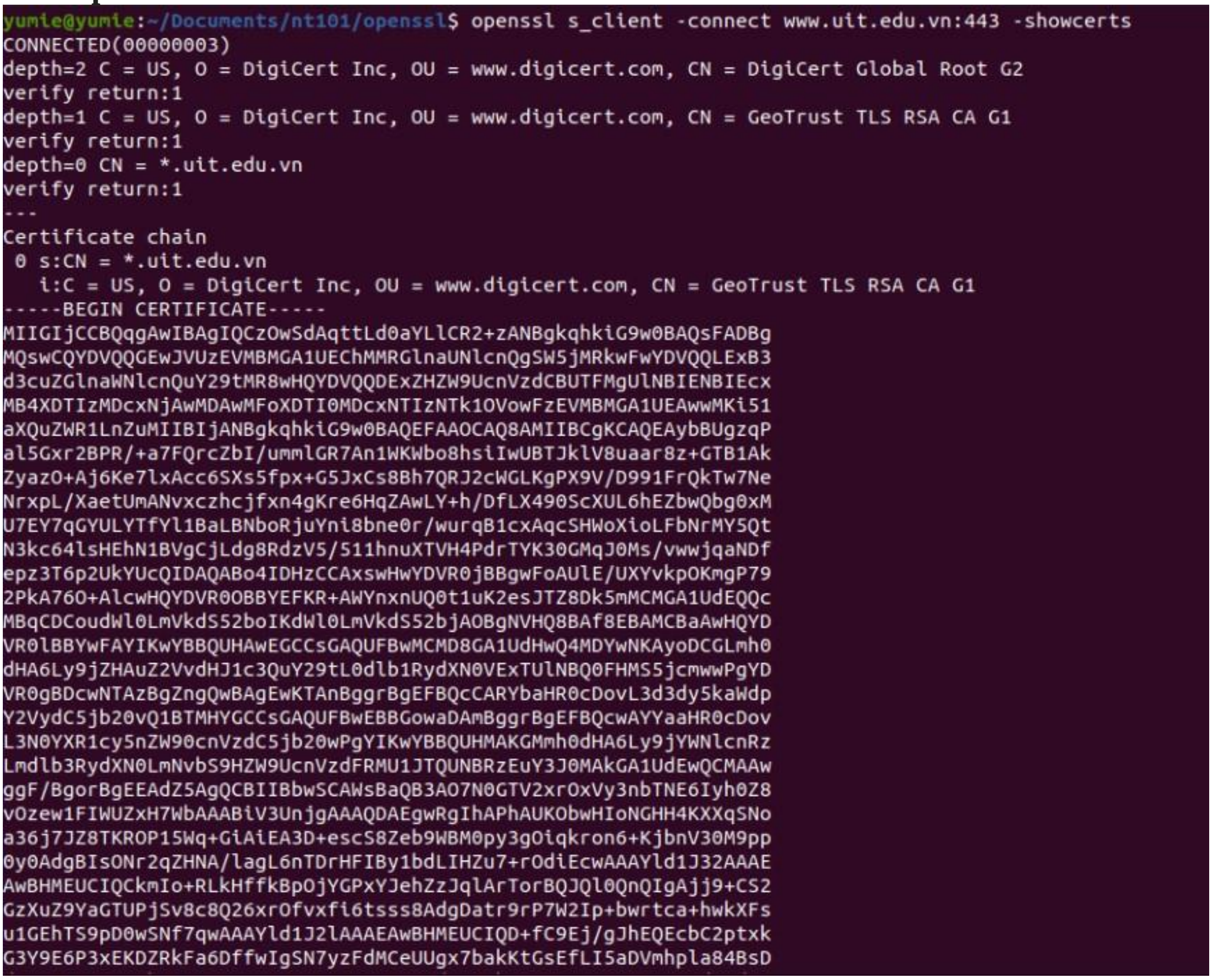


```
kousei@debian:~/Downloads/LAB04$ md5sum pileup.pdf
3f58844c6b242d99a9526794e522a12a  pileup.pdf
kousei@debian:~/Downloads/LAB04$ md5sum pileup.png
3f58844c6b242d99a9526794e522a12a  pileup.png
kousei@debian:~/Downloads/LAB04$ 
```

In 2012, a group of hackers used a hash collision to create a piece of malware called Flame. Flame was a sophisticated cyber espionage tool that was used to target organizations in the Middle East. The hackers used a hash collision to create a malicious file that had the same hash value as a legitimate file. This allowed them to install Flame on the target systems without being detected.

3. Manually Verifying an X.509 Certificate

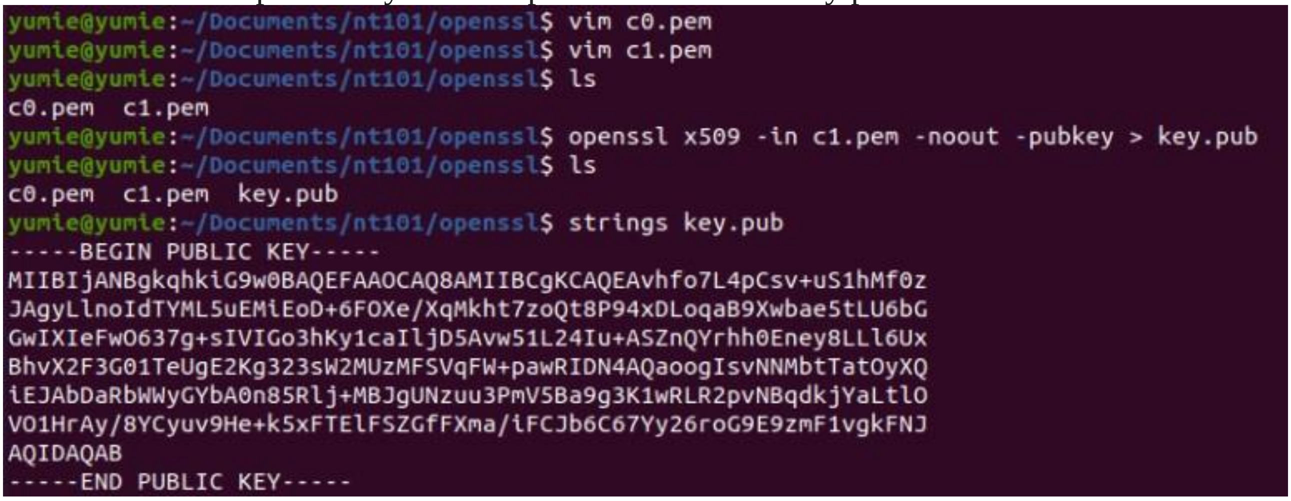
Step 1: Save the whole chain's certificates



```
yumie@yumie:~/Documents/nt101/openssl$ openssl s_client -connect www.uit.edu.vn:443 -showcerts
CONNECTED(00000003)
depth=2 C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert Global Root G2
verify return:1
depth=1 C = US, O = DigiCert Inc, OU = www.digicert.com, CN = GeoTrust TLS RSA CA G1
verify return:1
depth=0 CN = *.uit.edu.vn
verify return:1
---
Certificate chain
 0 s:CN = *.uit.edu.vn
   i:C = US, O = DigiCert Inc, OU = www.digicert.com, CN = GeoTrust TLS RSA CA G1
-----BEGIN CERTIFICATE-----
MIIGIjCCBQqgAwIBAgIQCzOwSdAqttLd0aYLlCR2+zANBgkqhkiG9w0BAQsFADBg
MQswCQYDVQQGEwJVUzEVMBMGA1UEChMMRGlnaUNlcnQgSW5jMRkwFwYDVQQLExB3
d3cuZGlnaWNlcnQuY29tMR8wHQYDVQQDExZHZW9UcnVzdCBUTFMgUlNBIENBIEcx
MB4XDTIzMDcxNjAwMDAwMFoXDTI0MDcxNTIzNTk1OVowFzEVMBMGA1UEAwwMKi51
aXQuZWR1LnZuMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAybBUgzqP
al5Gxr2BPR/+a7FQrcZbI/ummlGR7An1WKWbo8hsiIwUBTJklV8uaar8z+GTB1Ak
ZyazO+Aj6Ke7lxAcc6SXs5fpx+G5JxCs8Bh7QRJ2cWGLKgPX9V/D991FrQkTw7Ne
NrxpL/XaetUmANvxczhcjfxn4gKre6HqZAwLY+h/DfLX490ScXUL6hEZbwQbg0xM
U7EY7qGYULYTfYl1BaLBNboRjuYni8bne0r/wurqB1cxAqcSHWoXioLFbNrMY5Qt
N3kc64lsHEhN1BVgCjLdg8RdzV5/511hnuXTVH4PdrTYK30GMqJ0Ms/vwwjqaNDf
epz3T6p2UkYUcQIDAQABo4IDHzCCAxswHwYDVR0jBBgwFoAUlE/UXYvkpOKmgP79
2PkA76O+AlcwHQYDVR0OBBYEFKR+AWYnxnUQ0t1uK2esJTZ8Dk5mMCMGA1UdEQQc
MBqCDCoudWl0LmVkdS52boIKdWl0LmVkdS52bjAOBgNVHQ8BAf8EBAMCBaAwHQYD
VR0lBBYwFAYIKwYBBQUHAwEGCCsGAQUFBwMCMD8GA1UdHwQ4MDYwNKAyoDCGLmh0
dHA6Ly9jZHAuZ2VvdHJ1c3QuY29tL0dlb1RydXN0VExTUlNBQ0FHMS5jcmwwPgYD
VR0gBDcwNTAzBgZngQwBAgEwKTAnBggrBgEFBQcCARYbaHR0cDovL3d3dy5kaWdp
Y2VydC5jb20vQ1BTMHYGCCsGAQUFBwEBBGowaDAmBggrBgEFBQcwAYYaaHR0cDov
L3N0YXR1cy5nZW90cnVzdC5jb20wPgYIKwYBBQUHMAKGMmh0dHA6Ly9jYWNlcnRz
Lmdlb3RydXN0LmNvbS9HZW9UcnVzdFRMU1JTQUNBRzEuY3J0MAkGA1UdEwQCMAAw
ggF/BgorBgEEAdZ5AgQCBIIBbwSCAWsBaQB3AO7N0GTV2xrOxVy3nbTNE6Iyh0Z8
vOzew1FIWUZxH7WbAAABiV3UnjgAAAQDAEgwRgIhAPhAUKObwHIoNGHH4KXXqSNo
a36j7JZ8TKROP15Wq+GiAiEA3D+escS8Zeb9WBM0py3gOiqkron6+KjbnV30M9pp
0y0AdgBIsONr2qZHNA/lagL6nTDrHFIBy1bdLIHZu7+rOdiEcwAAAYld1J32AAAE
AwBHMEUCIQCkmIo+RLkHffkBpOjYGPxYJehZzJqlArTorBQJQl0QnQIgAjj9+CS2
GzXuZ9YaGTUPjSv8c8Q26xrOfvxfi6tsss8AdgDatr9rP7W2Ip+bwrtca+hwkXFs
u1GEhTS9pD0wSNf7qwAAAYld1J2lAAAEAwBHMEUCIQD+fC9Ej/gJhEQEcbC2ptxk
G3Y9E6P3xEKDZRkFa6DffwIgSN7yzFdMCeUUgx7bakKtGsEfLI5aDVmhpla84BsD
```

Step 2: Extract the public key from c1.pem

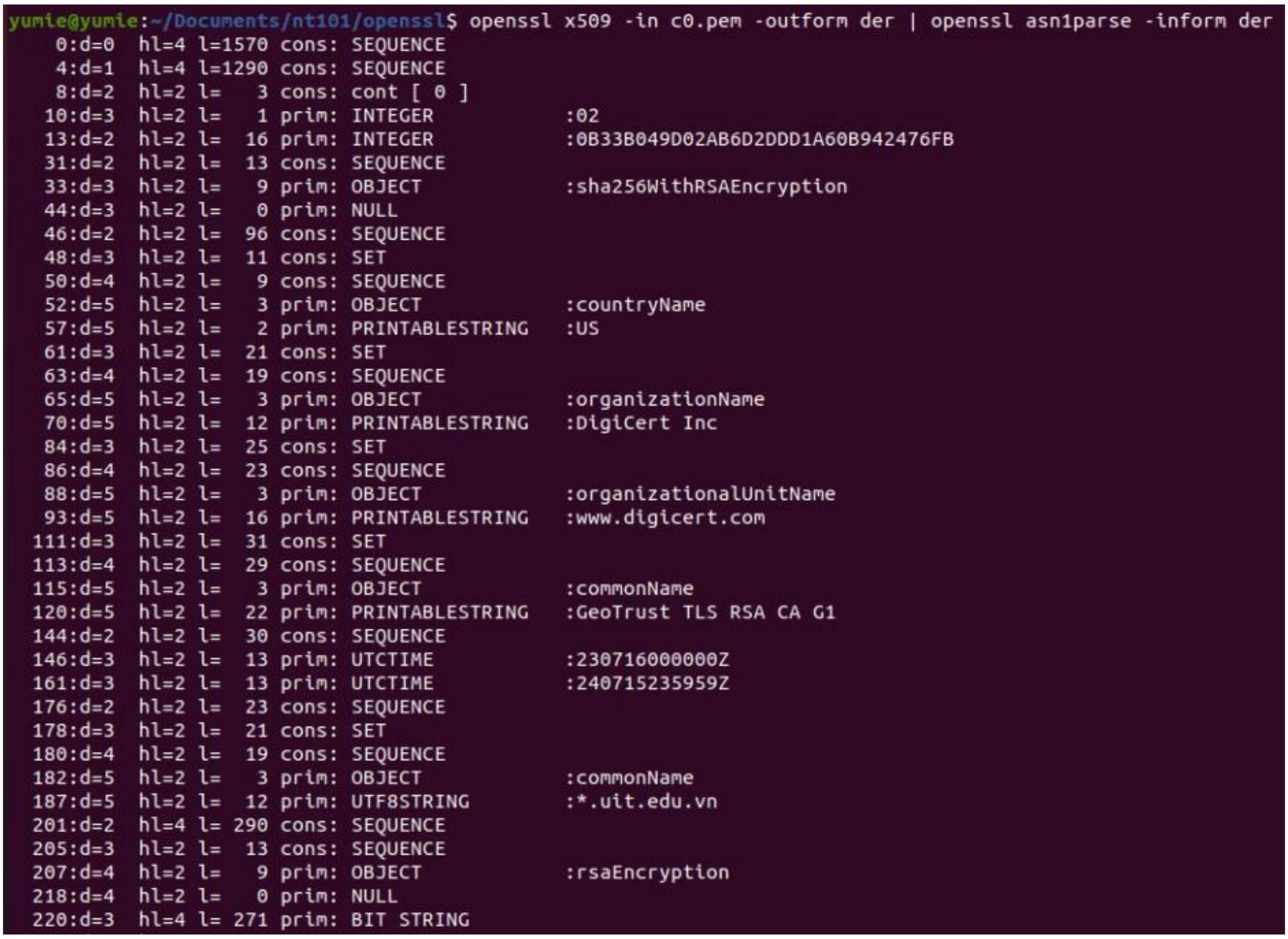
- The first certificate is saved to "c0.pem"
- The second certificate is saved to "c1.pem"
- Extract public key from c1.pem and saved to "key.pub"



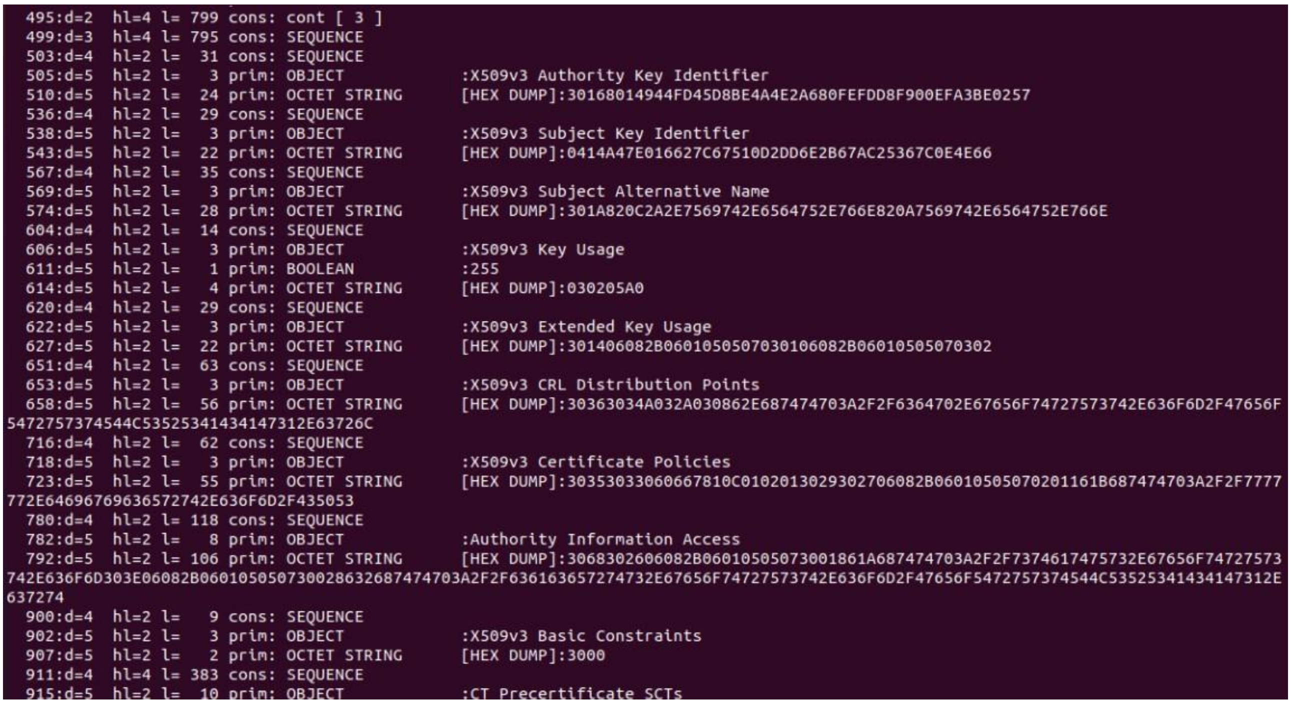
```
yumie@yumie:~/Documents/nt101/openssl$ vim c0.pem
yumie@yumie:~/Documents/nt101/openssl$ vim c1.pem
yumie@yumie:~/Documents/nt101/openssl$ ls
c0.pem  c1.pem
yumie@yumie:~/Documents/nt101/openssl$ openssl x509 -in c1.pem -noout -pubkey > key.pub
yumie@yumie:~/Documents/nt101/openssl$ ls
c0.pem  c1.pem  key.pub
yumie@yumie:~/Documents/nt101/openssl$ strings key.pub
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAvhfo7L4pCsv+uS1hMf0z
JAgyLlnoIdTYML5uEMiEoD+6FOXe/XqMkht7zoQt8P94xDLoqaB9Xwbae5tLU6bG
GwIXIeFwO637g+sIVIGo3hKy1caIljD5Avw51L24Iu+ASZnQYrhh0Eney8LLl6Ux
BhvX2F3G01TeUgE2Kg323sW2MUzMFSVqFW+pawRIDN4AQaoogIsvNNMbtTatOyXQ
iEJAbDaRbWWyGYbA0n85Rlj+MBJgUNzuu3PmV5Ba9g3K1wRLR2pvNBqdkjYaLtlO
VO1HrAy/8YCyuv9He+k5xFTElFSZGfFXma/iFCJb6C67Yy26roG9E9zmF1vgkFNJ
AQIDAQAB
-----END PUBLIC KEY-----
```

Step 3: Extract the signature from c0.pem

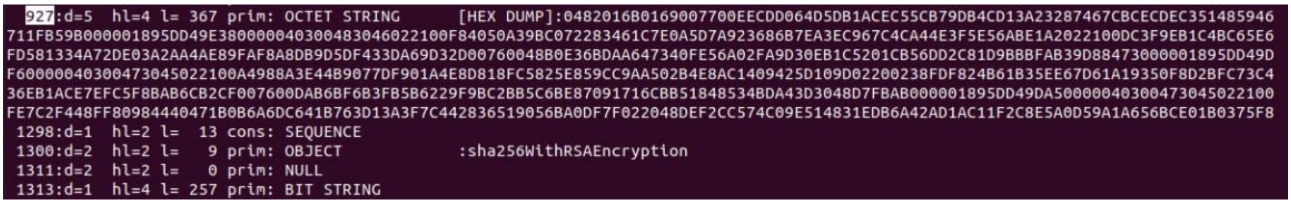
- Parse c0.pem with DER format by using ans.1 to find header length(hl) and content length(l) of tbsCertificate, signatureAlgorithm, signatureValue of the certificate



```
yumie@yumie:~/Documents/nt101/openssl$ openssl x509 -in c0.pem -outform der | openssl asn1parse -inform der
    0:d=0  hl=4 l=1570 cons: SEQUENCE
    4:d=1  hl=4 l=1290 cons: SEQUENCE
    8:d=2  hl=2 l=   3 cons: cont [ 0 ]
   10:d=3  hl=2 l=   1 prim: INTEGER           :02
   13:d=2  hl=2 l=  16 prim: INTEGER           :0B33B049D02AB6D2DDD1A60B942476FB
   31:d=2  hl=2 l=  13 cons: SEQUENCE
   33:d=3  hl=2 l=   9 prim: OBJECT            :sha256WithRSAEncryption
   44:d=3  hl=2 l=   0 prim: NULL
   46:d=2  hl=2 l=  96 cons: SEQUENCE
   48:d=3  hl=2 l=  11 cons: SET
   50:d=4  hl=2 l=   9 cons: SEQUENCE
   52:d=5  hl=2 l=   3 prim: OBJECT            :countryName
   57:d=5  hl=2 l=   2 prim: PRINTABLESTRING   :US
   61:d=3  hl=2 l=  21 cons: SET
   63:d=4  hl=2 l=  19 cons: SEQUENCE
   65:d=5  hl=2 l=   3 prim: OBJECT            :organizationName
   70:d=5  hl=2 l=  12 prim: PRINTABLESTRING   :DigiCert Inc
   84:d=3  hl=2 l=  25 cons: SET
   86:d=4  hl=2 l=  23 cons: SEQUENCE
   88:d=5  hl=2 l=   3 prim: OBJECT            :organizationalUnitName
   93:d=5  hl=2 l=  16 prim: PRINTABLESTRING   :www.digicert.com
  111:d=3  hl=2 l=  31 cons: SET
  113:d=4  hl=2 l=  29 cons: SEQUENCE
  115:d=5  hl=2 l=   3 prim: OBJECT            :commonName
  120:d=5  hl=2 l=  22 prim: PRINTABLESTRING   :GeoTrust TLS RSA CA G1
  144:d=2  hl=2 l=  30 cons: SEQUENCE
  146:d=3  hl=2 l=  13 prim: UTCTIME           :230716000000Z
  161:d=3  hl=2 l=  13 prim: UTCTIME           :240715235959Z
  176:d=2  hl=2 l=  23 cons: SEQUENCE
  178:d=3  hl=2 l=  21 cons: SET
  180:d=4  hl=2 l=  19 cons: SEQUENCE
  182:d=5  hl=2 l=   3 prim: OBJECT            :commonName
  187:d=5  hl=2 l=  12 prim: UTF8STRING        :*.uit.edu.vn
  201:d=2  hl=4 l= 290 cons: SEQUENCE
  205:d=3  hl=2 l=  13 cons: SEQUENCE
  207:d=4  hl=2 l=   9 prim: OBJECT            :rsaEncryption
  218:d=4  hl=2 l=   0 prim: NULL
  220:d=3  hl=4 l= 271 prim: BIT STRING
```

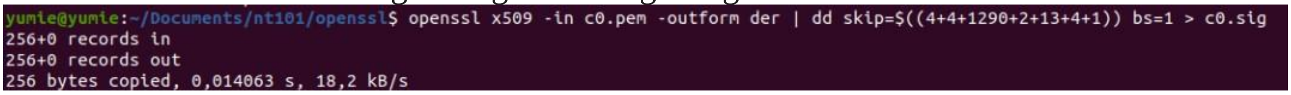


```
  495:d=2  hl=4 l= 799 cons: cont [ 3 ]
  499:d=3  hl=4 l= 795 cons: SEQUENCE
  503:d=4  hl=2 l=  31 cons: SEQUENCE
  505:d=5  hl=2 l=   3 prim: OBJECT            :X509v3 Authority Key Identifier
  510:d=5  hl=2 l=  24 prim: OCTET STRING      [HEX DUMP]:30168014944FD45D8BE4A4E2A680FEFDD8F900EFA3BE0257
  536:d=4  hl=2 l=  29 cons: SEQUENCE
  538:d=5  hl=2 l=   3 prim: OBJECT            :X509v3 Subject Key Identifier
  543:d=5  hl=2 l=  22 prim: OCTET STRING      [HEX DUMP]:0414A47E016627C67510D2DD6E2B67AC25367C0E4E66
  567:d=4  hl=2 l=  35 cons: SEQUENCE
  569:d=5  hl=2 l=   3 prim: OBJECT            :X509v3 Subject Alternative Name
  574:d=5  hl=2 l=  28 prim: OCTET STRING      [HEX DUMP]:301A820C2A2E7569742E6564752E766E820A7569742E6564752E766E
  604:d=4  hl=2 l=  14 cons: SEQUENCE
  606:d=5  hl=2 l=   3 prim: OBJECT            :X509v3 Key Usage
  611:d=5  hl=2 l=   1 prim: BOOLEAN           :255
  614:d=5  hl=2 l=   4 prim: OCTET STRING      [HEX DUMP]:030205A0
  620:d=4  hl=2 l=  29 cons: SEQUENCE
  622:d=5  hl=2 l=   3 prim: OBJECT            :X509v3 Extended Key Usage
  627:d=5  hl=2 l=  22 prim: OCTET STRING      [HEX DUMP]:301406082B0601050507030106082B06010505070302
  651:d=4  hl=2 l=  63 cons: SEQUENCE
  653:d=5  hl=2 l=   3 prim: OBJECT            :X509v3 CRL Distribution Points
  658:d=5  hl=2 l=  56 prim: OCTET STRING      [HEX DUMP]:30363034A032A030862E687474703A2F2F6364702E67656F74727573742E636F6D2F47656F
5472757374544C53525341434147312E63726C
  716:d=4  hl=2 l=  62 cons: SEQUENCE
  718:d=5  hl=2 l=   3 prim: OBJECT            :X509v3 Certificate Policies
  723:d=5  hl=2 l=  55 prim: OCTET STRING      [HEX DUMP]:30353033060667810C0102013029302706082B06010505070201161B687474703A2F2F7777
772E64696769636572742E636F6D2F435053
  780:d=4  hl=2 l= 118 cons: SEQUENCE
  782:d=5  hl=2 l=   8 prim: OBJECT            :Authority Information Access
  792:d=5  hl=2 l= 106 prim: OCTET STRING      [HEX DUMP]:3068302606082B06010505073001861A687474703A2F2F7374617475732E67656F74727573
742E636F6D303E06082B060105050730028632687474703A2F2F636163657274732E67656F74727573742E636F6D2F47656F5472757374544C53525341434147312E
637274
  900:d=4  hl=2 l=   9 cons: SEQUENCE
  902:d=5  hl=2 l=   3 prim: OBJECT            :X509v3 Basic Constraints
  907:d=5  hl=2 l=   2 prim: OCTET STRING      [HEX DUMP]:3000
  911:d=4  hl=4 l= 383 cons: SEQUENCE
  915:d=5  hl=2 l=  10 prim: OBJECT            :CT Precertificate SCTs
```



```
  927:d=5  hl=4 l= 367 prim: OCTET STRING      [HEX DUMP]:0482016B0169007700EECDD064D5DB1ACEC55CB79DB4CD13A23287467CBCECDEC351485946
711FB59B000001895DD49E380000040300483046022100F84050A39BC072283461C7E0A5D7A923686B7EA3EC967C4CA44E3F5E56ABE1A2022100DC3F9EB1C4BC65E6
FD581334A72DE03A2AA4AE89FAF8A8DB9D5DF433DA69D32D00760048B0E36BDAA647340FE56A02FA9D30EB1C5201CB56DD2C81D9BBBFAB39D88473000001895DD49D
F60000040300473045022100A4988A3E44B9077DF901A4E8D818FC5825E859CC9AA502B4E8AC1409425D109D02200238FDF824B61B35EE67D61A19350F8D2BFC73C4
36EB1ACE7EFC5F8BAB6CB2CF007600DAB6BF6B3FB5B6229F9BC2BB5C6BE87091716CBB51848534BDA43D3048D7FBAB000001895DD49DA50000040300473045022100
FE7C2F448FF80984440471B0B6A6DC641B763D13A3F7C442836519056BA0DF7F022048DEF2CC574C09E514831EDB6A42AD1AC11F2C8E5A0D59A1A656BCE01B0375F8
 1298:d=1  hl=2 l=  13 cons: SEQUENCE
 1300:d=2  hl=2 l=   9 prim: OBJECT            :sha256WithRSAEncryption
 1311:d=2  hl=2 l=   0 prim: NULL
 1313:d=1  hl=4 l= 257 prim: BIT STRING
```

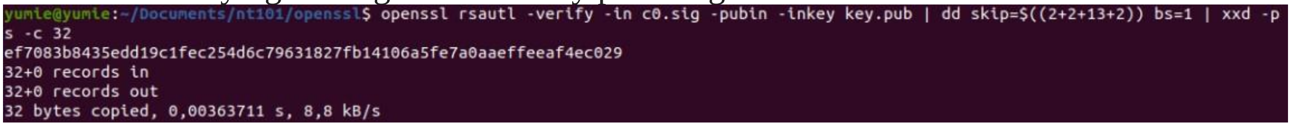
- Perform extracting the signature to get .sig file



```
yumie@yumie:~/Documents/nt101/openssl$ openssl x509 -in c0.pem -outform der | dd skip=$((4+4+1290+2+13+4+1)) bs=1 > c0.sig
256+0 records in
256+0 records out
256 bytes copied, 0,014063 s, 18,2 kB/s
```

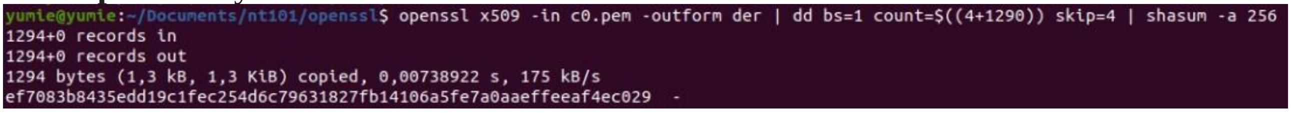
Step 4: Decrypt the signature

- Verifying the .sig file with "key.pub" to get the hash



```
yumie@yumie:~/Documents/nt101/openssl$ openssl rsautl -verify -in c0.sig -pubin -inkey key.pub | dd skip=$((2+2+13+2)) bs=1 | xxd -p
s -c 32
ef7083b8435edd19c1fec254d6c79631827fb14106a5fe7a0aaeffeeaf4ec029
32+0 records in
32+0 records out
32 bytes copied, 0,00363711 s, 8,8 kB/s
```

Step 5: Verify the hash



```
yumie@yumie:~/Documents/nt101/openssl$ openssl x509 -in c0.pem -outform der | dd bs=1 count=$((4+1290)) skip=4 | shasum -a 256
1294+0 records in
1294+0 records out
1294 bytes (1,3 kB, 1,3 KiB) copied, 0,00738922 s, 175 kB/s
ef7083b8435edd19c1fec254d6c79631827fb14106a5fe7a0aaeffeeaf4ec029  -
```