REPORT TASK 5, 6

Môn học: Mật mã học

Giảng viên: Nguyễn Ngọc Tự

Thái Vĩnh Đạt – 22520235 – ATTN2022

Thái Vĩnh Đạt - 22520235

1. Hardware resource

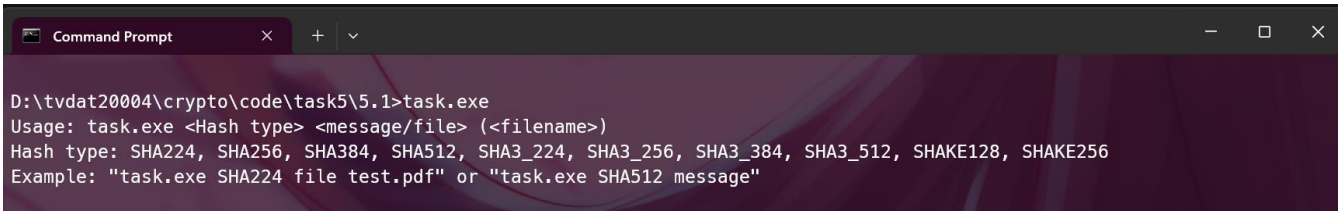
System Information

Current Date/Time:	Monday, December 18, 2023, 11:22:54 PM
Computer Name:	HNO3
Operating System:	Windows 11 Home Single Language 64-bit (10.0, Build 22631)
Language:	English (Regional Setting: English)
System Manufacturer:	LENOVO
System Model:	82L5
BIOS:	GSCN35WW
Processor:	AMD Ryzen 5 5600H with Radeon Graphics (12 CPUs), ~3.3GHz
Memory:	16384MB RAM
Page file:	20259MB used, 5160MB available
DirectX Version:	DirectX 12

2. Task 5: PKI and Hash Functions

a. 5.1 - Hash function

Đây là demo cách sử dụng của chương trình:

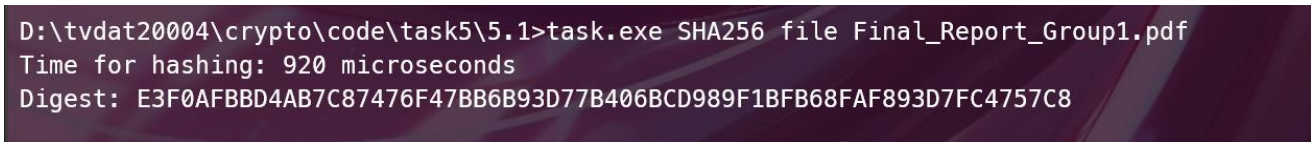


Chương trình được yêu cầu hash input (có thể nhập từ file hoặc bàn phím), hỗ trợ các thuật toán hash như SHA224, SHA256, SHA384, SHA512, SHA3_224, SHA3_256, SHA3_384, SHA3_512, SHAKE128, SHAKE256.

Cách hoạt động của chương trình:

- Hash message:

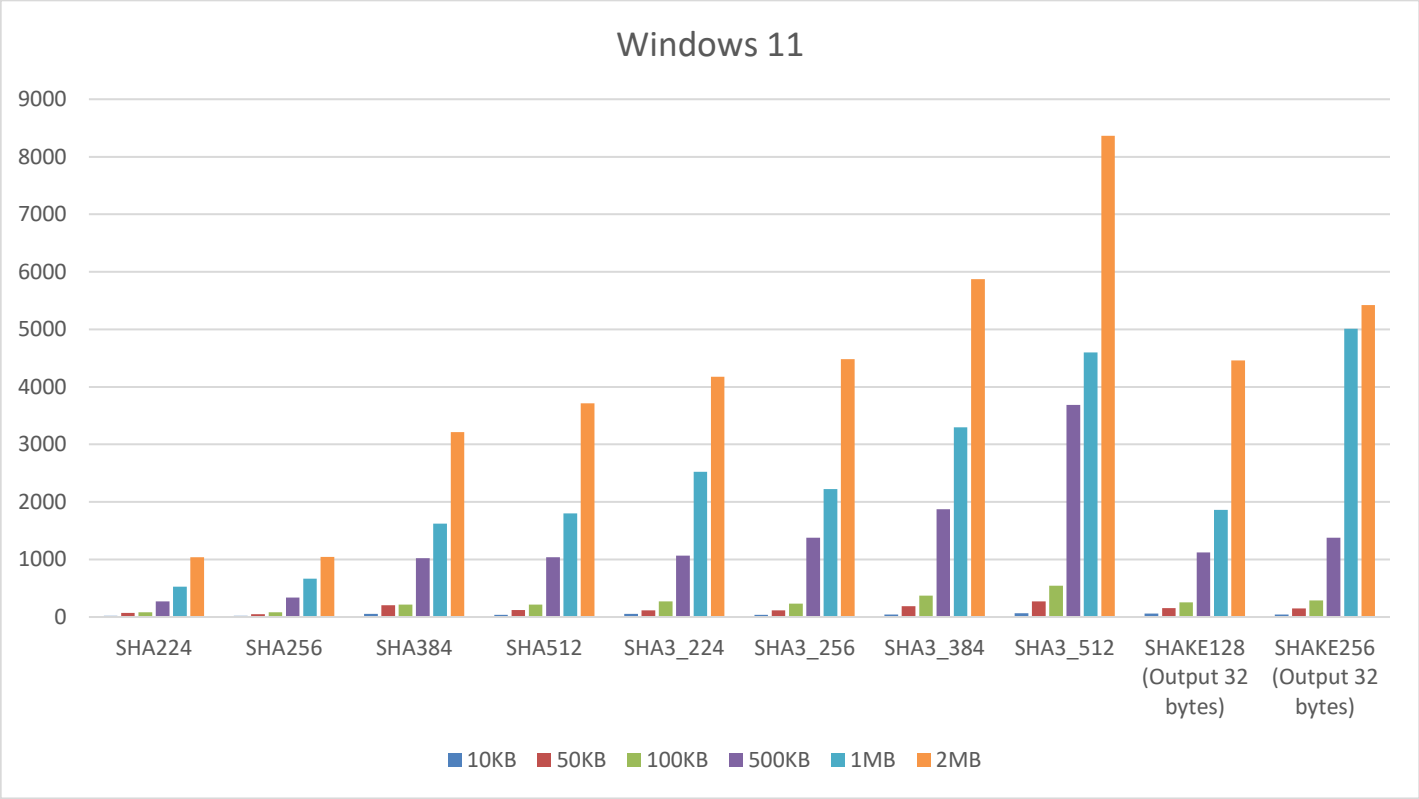
- Hash file:



Còn đây là số liệu thống kê về thời gian chạy các thuật toán hash trên các input có kích thước khác nhau, ở đây em do với 6 kích thước: 10KB, 50KB, 100KB, 500KB, 1MB, 2MB. Việc đo đạc được thực hiện trên cả Windows là Linux.

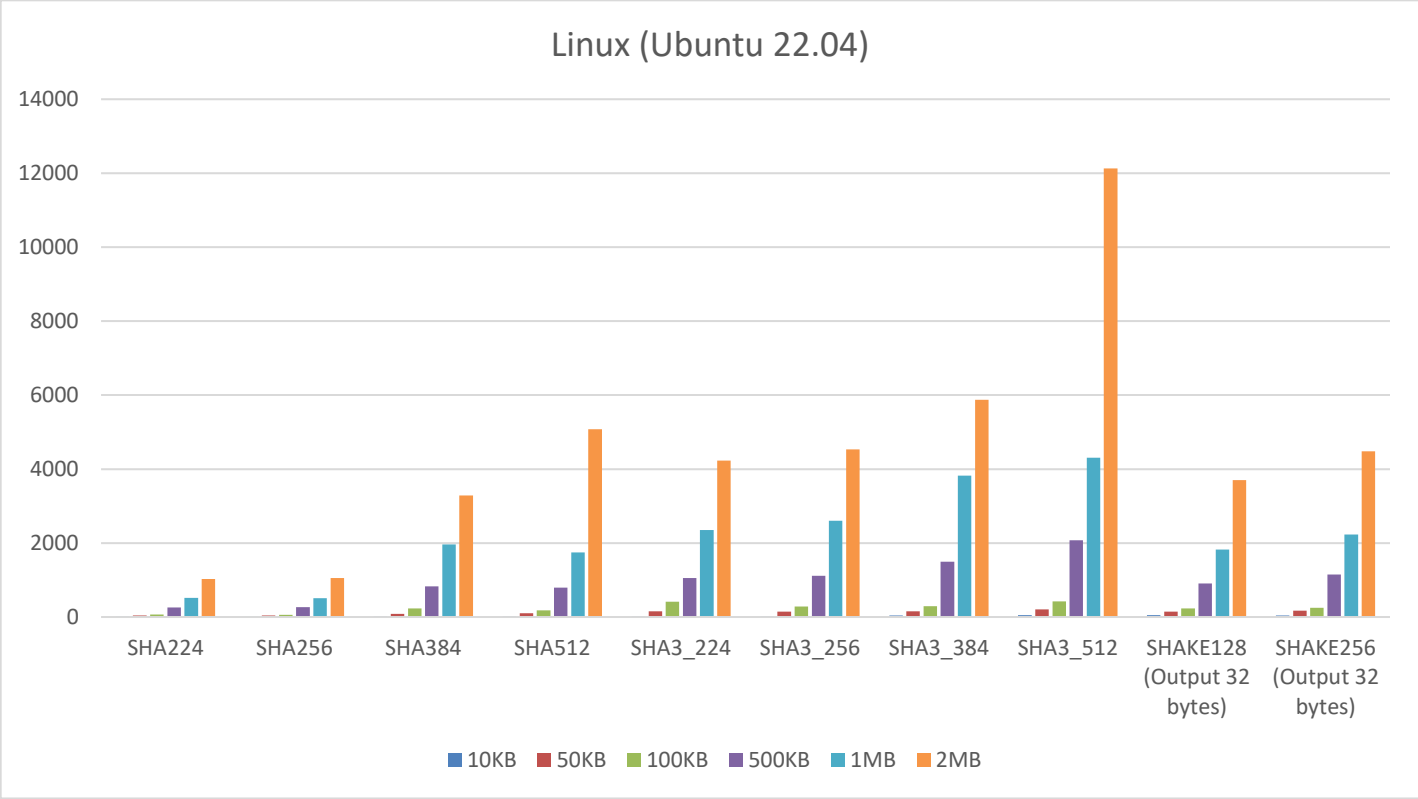
- Trên Windows 11: (đơn vị: microsecond (10^{-6} s))

Hash type	10KB	50KB	100KB	500KB	1MB	2MB
SHA224	22	73	85	272	529	1041
SHA256	22	49	80	336	665	1042
SHA384	54	202	218	1025	1624	3215
SHA512	38	119	218	1038	1800	3713
SHA3_224	52	114	271	1067	2525	4174
SHA3_256	38	114	234	1379	2226	4483
SHA3_384	45	190	369	1871	3300	5873
SHA3_512	68	272	542	3686	4599	8363
SHAKE128 (Output 32 bytes)	59	153	253	1123	1864	4462
SHAKE256 (Output 32 bytes)	46	149	286	1381	5009	5423



- Trên Linux (Ubuntu 22.04): đơn vị (microsecond = 10^{-6} s)

Hash type	10KB	50KB	100KB	500KB	1MB	2MB
SHA224	12	38	70	262	520	1026
SHA256	11	38	56	270	510	1058
SHA384	21	85	233	825	1966	3282
SHA512	21	99	176	793	1745	5076
SHA3_224	24	152	414	1051	2352	4230
SHA3_256	26	144	280	1118	2600	4534
SHA3_384	40	150	292	1492	3823	5876
SHA3_512	46	210	418	2079	4309	12125
SHAKE128 (Output 32 bytes)	46	149	229	907	1821	3698
SHAKE256 (Output 32 bytes)	39	171	250	1152	2232	4476



- **Nhận xét:**

- Trên cả 2 hệ điều hành, ta có thể thấy với input càng lớn, thời gian chạy càng tăng. Khi so sánh thời gian chạy giữa các thuật toán với nhau trên cùng một độ lớn input, ta thấy với input nhỏ thì chênh lệch thời gian không đáng kể, tuy nhiên khi input càng lớn, sự khác biệt thời gian cũng lớn dần. Các thuật toán hash càng phức tạp thì sẽ cho thời gian chạy càng lâu, tuy nhiên đánh đổi lại đó là độ bảo mật được tăng lên đáng kể. Do đó khi áp dụng thực tế, người dùng có thể linh động sử dụng các thuật toán khác nhau dựa vào nhu cầu.
- Nếu xét thời gian chạy của 2 hệ điều hành trên cùng 1 thuật với và cùng 1 độ dài input, ta thấy rằng đa phần Ubuntu nhanh hơn Windows, nhưng chênh lệch thời gian cũng không đáng kể.

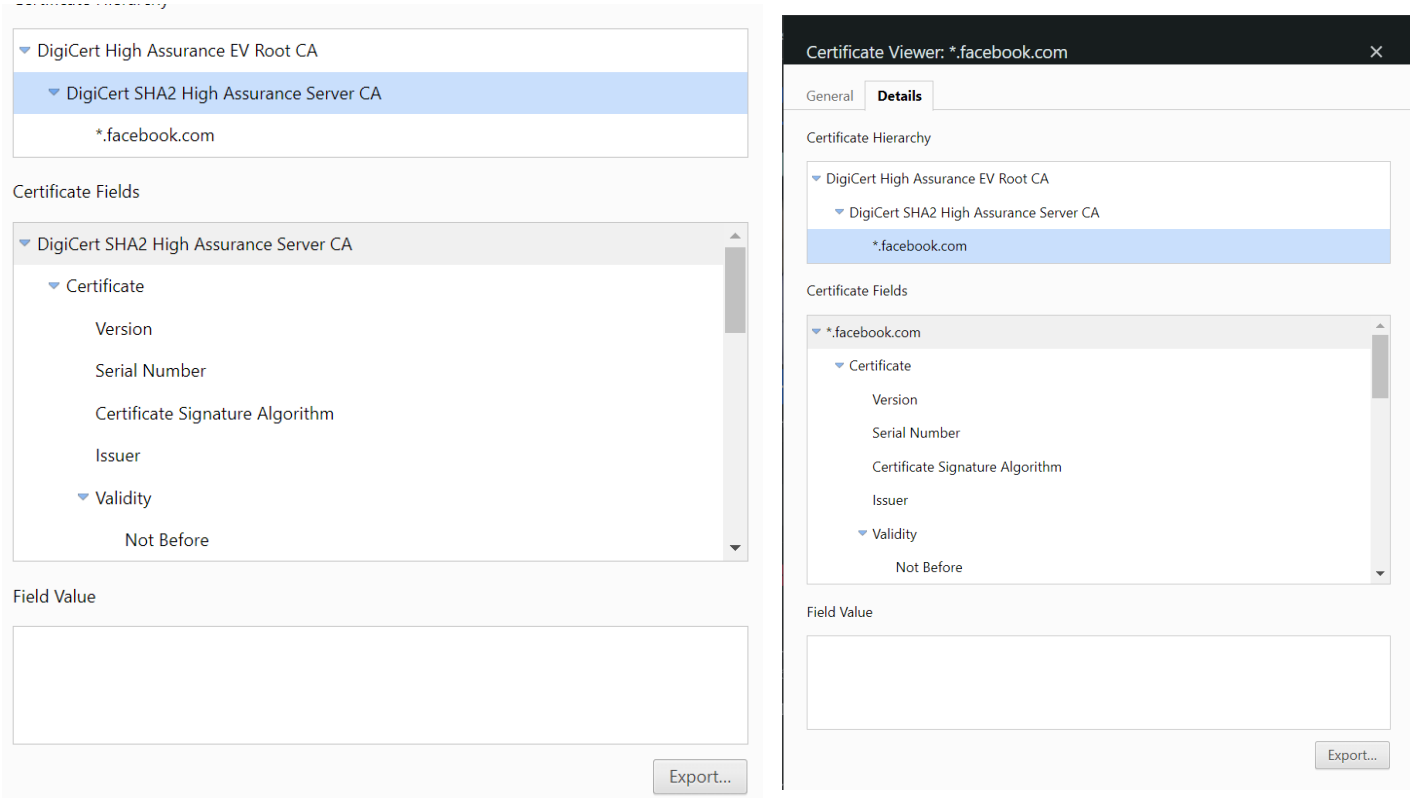
b. 5.2: PKI and digital certificate

Task này yêu cầu tạo một tool sử dụng thư viện Openssl hoặc CryptoPP nhằm 2 mục đích:

- Kiểm tra tính xác thực của X509 certificate.
- Phân tích tất cả các trường của X.509 certificate, bao gồm tên chủ đề, tên nhà phát hành, public key, signature, thuật toán signature và các tham số, mục đích của nó, khoảng thời gian hợp lệ, …

Sau đây là demo của chương trình:

- Lên facebook.com lấy certificate của facebook và DigiCert SHA2 High Assurance Server CA về.



- Sau đó chạy chương trình, nhập certificate của facebook vào mục "Enter certificate file name:", certificate của server vào "Enter intermediate certificate:".

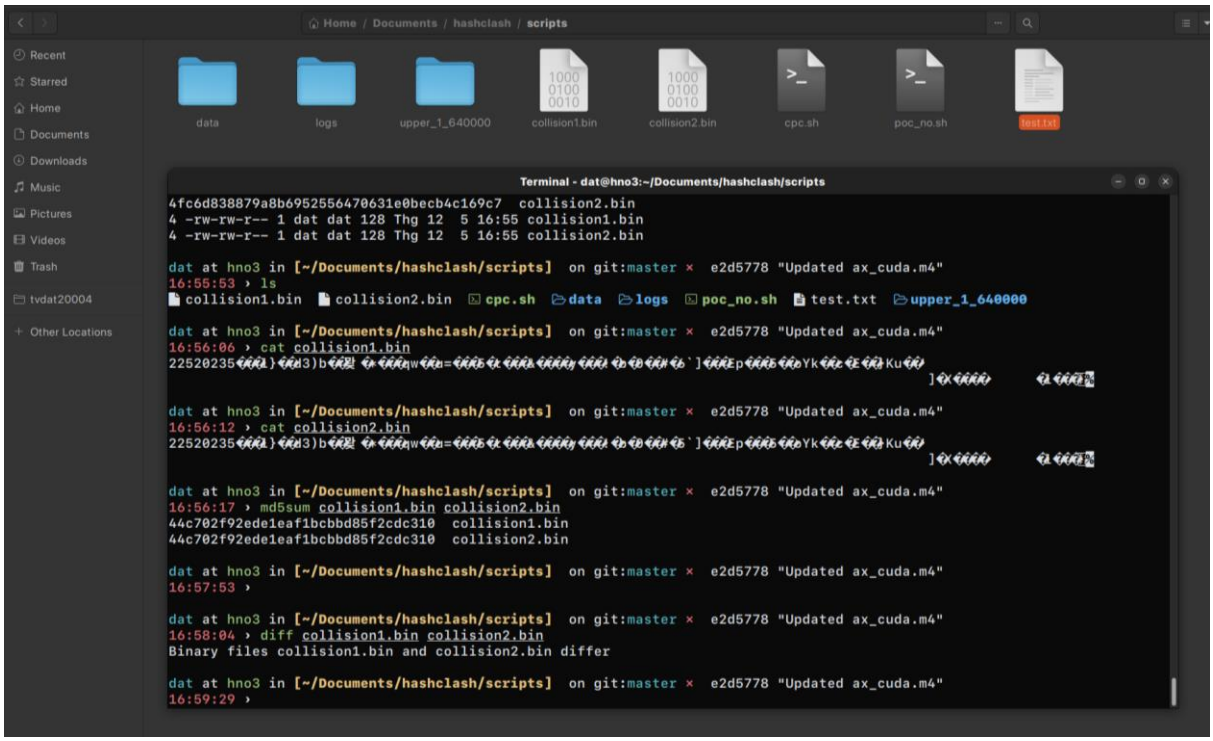
```
D:\tvdat20004\crypto\code\task5\5.2>task.exe
Enter certificate file name: fb.crt
Enter intermediate certificate: inter_fb.crt

Validate certificate sucessfully!!!
Subject: /C=US/ST=California/L=Menlo Park/O=Meta Platforms, Inc./CN=*.facebook.com
Issuer: /C=US/O=DigiCert Inc/OU=www.digicert.com/CN=DigiCert SHA2 High Assurance Server CA
Subject Public Key Info:
        Public Key Algorithm: id-ecPublicKey
        Public-Key: (256 bit)
        04:E8:85:9F:24:DD:DE:D1:66:C2:DB:CA:2E:FF:25:
        24:87:DA:F7:D8:5C:F9:50:13:43:57:42:8D:F8:60:
        59:DF:B9:70:B8:E2:58:13:DA:1E:E9:55:3D:4A:40:
        4C:BC:F6:83:8B:CF:6C:A0:E6:48:37:4A:41:82:82:
        81:C7:CC:1B:D7
        ASN1 OID: prime256v1
NIST CURVE: prime256v1
Signature:
A86065B9DDBB003E1150734F9F459AE3605F456FC7FE8630238E805EF7C7CE3F0D0E72\
E111BDA439932C5C19BFE53EF6520B135D2B87C973C3BEF594AA2931D5497AFD945F32\
9F800C29358D6797F7FA30F288E1AA61A5292B56D52AD8A8AC00FADBBF62DE9BA4126C\
4C98FD2534CC8D13A63F20850E16D5728C978A5F2430C1CF161D369F60071E8224B124\
74CB590FBE204762377877808D664B99380C305D978FB49C40D99655C5125141627B1C\
9074234620D0A2861FAD6C758060736FB3AE2B4884578FC31C7C5953D00737009B6D14\
7AF66693A0230A0ACA8831334D19C2A298E6049AA1180DAD715A8ADC4D3BE35B9321BF\
D739BEE82A7456CD9EDB73
Signature algorithm: sha256WithRSAEncryption
Validity:
        Not before: Sep 26 00:00:00 2023 GMT
        Not after: Dec 25 23:59:59 2023 GMT
Purpose: SSL clientSSL serverAny PurposeOCSP helper
```

- Kiểm tra bằng cách chạy lệnh của openssl:
  $ openssl x509 -in fb.crt -inform PEM -text -noout

3. Task 6: Collision and length extension attacks on Hash functions

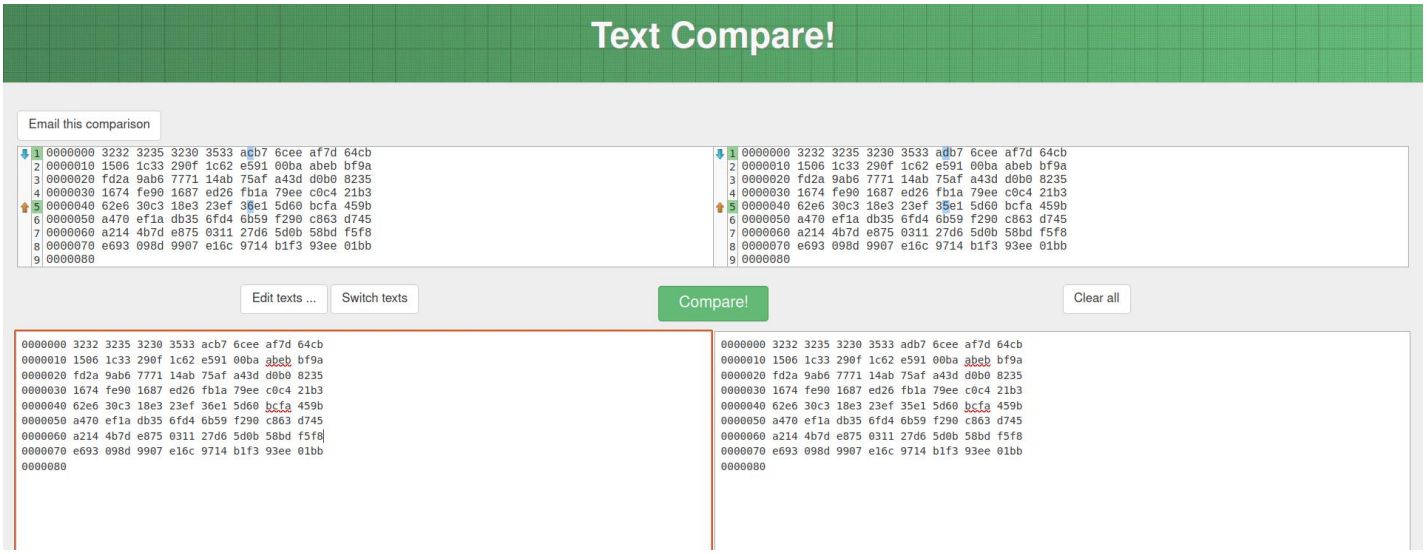
a. 6.1: MD5 collision attacks

- Đây là kết quả sau khi chạy tool hashclash để tạo 2 file có cùng giá trị hash MD5 và có cùng prefix cho trước:



Kiểm tra lại bằng nội dung 2 file bằng cách check hexdump:

```
dat at hno3 in [~/Documents/crypto/code_ubuntu/task6/6.1/1]
10:54:53 › hexdump collision1.bin
0000000 3232 3235 3230 3533 acb7 6cee af7d 64cb
0000010 1506 1c33 290f 1c62 e591 00ba abeb bf9a
0000020 fd2a 9ab6 7771 14ab 75af a43d d0b0 8235
0000030 1674 fe90 1687 ed26 fb1a 79ee c0c4 21b3
0000040 62e6 30c3 18e3 23ef 36e1 5d60 bcfa 459b
0000050 a470 ef1a db35 6fd4 6b59 f290 c863 d745
0000060 a214 4b7d e875 0311 27d6 5d0b 58bd f5f8
0000070 e693 098d 9907 e16c 9714 b1f3 93ee 01bb
0000080

dat at hno3 in [~/Documents/crypto/code_ubuntu/task6/6.1/1]
10:55:02 › hexdump collision2.bin
0000000 3232 3235 3230 3533 adb7 6cee af7d 64cb
0000010 1506 1c33 290f 1c62 e591 00ba abeb bf9a
0000020 fd2a 9ab6 7771 14ab 75af a43d d0b0 8235
0000030 1674 fe90 1687 ed26 fb1a 79ee c0c4 21b3
0000040 62e6 30c3 18e3 23ef 35e1 5d60 bcfa 459b
0000050 a470 ef1a db35 6fd4 6b59 f290 c863 d745
0000060 a214 4b7d e875 0311 27d6 5d0b 58bd f5f8
0000070 e693 098d 9907 e16c 9714 b1f3 93ee 01bb
0000080
```



- Đây là kết quả sau khi chạy tool hashclash để tạo ra file MD5 collision với 2 prefix là 2 file thực thi của 2 chương trình C++ khác nhau (thời gian chạy: 304 phút)

```
Found collision!
[*] Step 6 completed
[*] Number of backtracks until now: 2
[*] Collision generated: ../../2/test1.coll ../../2/test2.coll
e827e00c3111863043c36c39232b3828  ../../2/test1.coll
e827e00c3111863043c36c39232b3828  ../../2/test2.coll
[*] Process completed in 304 minutes (2 backtracks).

dat at hno3 in [~/Documents/crypto/code_ubuntu/task6/6.1/hashclash/scripts]
3:17:59 › cd ../../2

dat at hno3 in [~/Documents/crypto/code_ubuntu/task6/6.1/2]
10:18:53 › md5sum test1 test2
0e76bd512c049e93775683be8ecddb36  test1
8311b09476a50a40c5659e3ee1d9d895  test2

dat at hno3 in [~/Documents/crypto/code_ubuntu/task6/6.1/2]
10:19:04 › md5sum test1.coll test2.coll
e827e00c3111863043c36c39232b3828  test1.coll
e827e00c3111863043c36c39232b3828  test2.coll
```

b. 6.2: Length extension attacks on MAC in form: H(k||m), k is secret key.

- Sau đây là demo length extension attack trên các thuật toán hash sau khi chạy bằng tool hashpump:
  - Cách sử dụng:

```
dat at hno3 in [~/Documents/crypto/code_ubuntu/task6/6.2]
11:03:19 › hashpump -h
HashPump [-h help] [-t test] [-s signature] [-d data] [-a additional] [-k keylength]
     HashPump generates strings to exploit signatures vulnerable to the Hash Length Extension Attack.
     -h --help          Display this message.
     -t --test          Run tests to verify each algorithm is operating properly.
     -s --signature     The signature from known message.
     -d --data          The data from the known message.
     -a --additional    The information you would like to add to the known message.
     -k --keylength     The length in bytes of the key being used to sign the original message with.
     -u --unknown     number if leading hash bits unknown (EXPERIMENTAL HACK)
     -z --sig2     target signature (EXPERIMENTAL HACK)
     Version 1.0 with MD5, SHA1, SHA256 and SHA512 support.
     <Developed by bwall(@bwallHatesTwits)>
```

  - Attack SHA1

```
dat at hno3 in [~/Documents/crypto/code_ubuntu/task6/6.2]
11:02:31 › hashpump -s 1dad0d78421c0cb3b04fa241fbdd297ca07a5f2c --data thaivinhdat -a 22520235 -k 14
mask: ffffffff
predicted sig: 5a87d5a8124e5859bf4fd4a3e14951c6a2920f1b
thaivinhdat\x80\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\xc822520235
```

  - Attack SHA256

```
dat at hno3 in [~/Documents/crypto/code_ubuntu/task6/6.2]
11:04:30 › hashpump -s 263a8fa1e67a446b38d47efc3e766fe978058e8bcebce84ff89c075de4fdaf86 --data thaivinhdat -a 22520235 -
k 14
mask: ffffffff
predicted sig: 56567feea36c1587c1e24e559b8f958d61221b56317102fdc816ea895914c342
thaivinhdat\x80\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\xc822520235
```

  - Attack SHA512

```
dat at hno3 in [~/Documents/crypto/code_ubuntu/task6/6.2]
13:13:50 › hashpump -s 34e4b20041ad0f4ccfd2e7f5d5ab178b616db22785aeefeb5922512c75819b18591d1e395ccf25b3a1e6f4b3ed97de8e0
0a694afb470587c9ea39f7a5e7ca3ac --data thaivinhdat -a 22520235 -k 14
mask: ffffffff
predicted sig: 8577bce1c8e158ea404be30ff447c90b7b862bbea881eb71a587b19247876b112c18e333a684652944586f8795ef287fe4c076990
1e91f603cfefd7b27d755e3
thaivinhdat\x80\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\xc822520235
```

- **Bonus point:**

Sau đây là demo chương trình mà em tự xây dựng cũng như so sánh với kết quả khi chạy bằng hashpump: