REPORT CRYTOGRAPHY – TASK 4

Student: Huỳnh Trung Thuận

ID: 22521444

Lecturer: Nguyễn Ngọc Tự

---

1. Hardware resources.

a. Windows

System Information

Current Date/Time:	Saturday, June 15, 2024, 3:25:43 PM
Computer Name:	LAPTOP-B42TB1HN
Operating System:	Windows 11 Home Single Language 64-bit (10.0, Build 22631)
Language:	English (Regional Setting: English)
System Manufacturer:	ASUSTeK COMPUTER INC.
System Model:	Vivobook_ASUSLaptop X1403ZA_A1403ZA
BIOS:	X1403ZA.300
Processor:	12th Gen Intel(R) Core(TM) i5-12500H (16 CPUs), ~2.5GHz
Memory:	16384MB RAM
Page file:	17426MB used, 12686MB available
DirectX Version:	DirectX 12

b. Linux (ubuntu)

```
thuanht@HuynhTrungThuan-22521444:~$ lscpu
Architecture:            x86_64
  CPU op-mode(s):        32-bit, 64-bit
  Address sizes:         39 bits physical, 48 bits virtual
  Byte Order:            Little Endian
CPU(s):                  16
  On-line CPU(s) list:   0-15
Vendor ID:               GenuineIntel
  Model name:            12th Gen Intel(R) Core(TM) i5-12500H
    CPU family:          6
    Model:               154
    Thread(s) per core:  2
    Core(s) per socket:  12
    Socket(s):           1
    Stepping:            3
    CPU max MHz:         4500.0000
    CPU min MHz:         400.0000
    BogoMIPS:            6220.80
    Flags:               fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mc
                         a cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss
                         ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc art
                          arch_perfmon pebs bts rep_good nopl xtopology nonstop_
                         tsc cpuid aperfmperf tsc_known_freq pni pclmulqdq dtes6
                         4 monitor ds_cpl vmx smx est tm2 ssse3 sdbg fma cx16 xt
                         pr pdcm sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_
                         timer aes xsave avx f16c rdrand lahf_lm abm 3dnowprefet
                         ch cpuid_fault epb cat_l2 cdp_l2 ssbd ibrs ibpb stibp i
                         brs enhanced_tpr shadow flexpriority ept vpid ept_ad fs
                         gsbase tsc_adjust bmi1 avx2 smep bmi2 erms invpcid rdt_
                         a rdseed adx smap clflushopt clwb intel_pt sha_ni xsave
                         opt xsavec xgetbv1 xsaves split_lock_detect avx_vnni dt
                         herm ida arat pln pts hwp hwp_notify hwp_act_window hwp
                         _epp hwp_pkg_req hfi vnmi umip pku ospke waitpkg gfni v
                         aes vpclmulqdq rdpid movdiri movdir64b fsrm md_clear se
                         rialize arch_lbr ibt flush_l1d arch_capabilities
Virtualization features:
  Virtualization:        VT-x
Caches (sum of all):
  L1d:                   448 KiB (12 instances)
  L1i:                   640 KiB (12 instances)
  L2:                    9 MiB (6 instances)
  L3:                    18 MiB (1 instance)
NUMA:
  NUMA node(s):          1
  NUMA node0 CPU(s):     0-15
Vulnerabilities:
  Gather data sampling:  Not affected
```

2. Giới thiệu.

Báo cáo này trình bày quá trình thực hiện và kết quả kiểm thử hiệu suất của các hàm băm SHA-224, SHA-256, SHA-384, SHA-512, SHA3-224, SHA3-256, SHA3-384, SHA3-512, SHAKE128 và SHAKE256. Ngoài ra, báo cáo cũng khám phá sâu hơn vào lĩnh vực Hạ tầng Khóa công khai (PKI) và các phương pháp tấn công mật mã. Mục tiêu là so sánh thời gian băm của từng hàm khi thực thi trên hai hệ điều hành khác nhau, Windows và Linux, với mỗi tác vụ để đảm bảo tính chính xác và ổn định của kết quả. Thông qua việc sử dụng các công cụ mật mã như Crypto++ và OpenSSL, chúng tôi đã triển khai và đánh giá các hàm băm này với các kích thước đầu vào khác nhau.

Ngoài ra, báo cáo cũng xem xét các chiến lược tấn công mật mã như các cuộc tấn công như collision attack và Length extension attack, nhằm đánh giá sự an toàn và độ tin cậy của chúng trong các môi trường khác nhau. Kết quả thu được sẽ giúp đưa ra nhận định chi tiết về hiệu suất và khả năng ứng dụng của từng hàm băm và PKI trong thực tế.

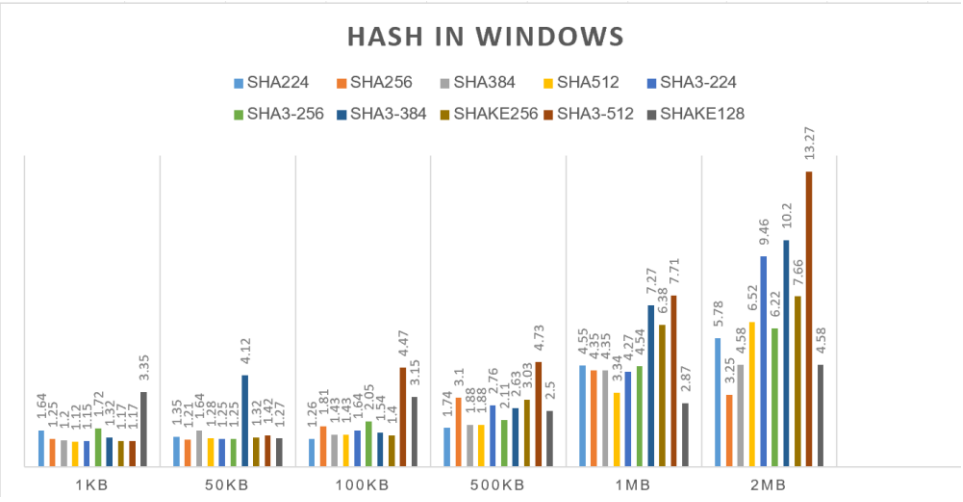
3. PKI and Hash Functions

3.1 Hash Functions

a. Thống kê thời gian.

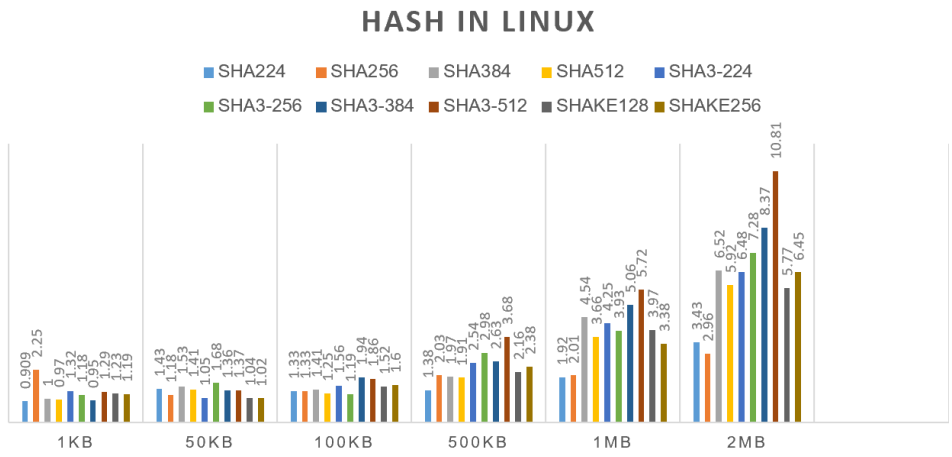
Windows - Hash: đơn vị (ms)

Runtime Hash in Windows (ms)						
Hash Type	**1Kb**	**50kb**	**100kb**	**500kb**	**1Mb**	**2Mb**
SHA224	1.64	1.35	1.26	1.74	4.55	5.78
SHA256	1.25	1.21	1.81	3.1	4.35	3.25
SHA384	1.2	1.64	1.43	1.88	4.35	4.58
SHA512	1.12	1.28	1.43	1.88	3.34	6.52
SHA3-224	1.15	1.25	1.64	2.76	4.27	9.46
SHA3-256	1.72	1.25	2.05	2.11	4.54	6.22
SHA3-384	1.32	4.12	1.54	2.63	7.27	10.2
SHA3-512	1.17	1.42	4.47	4.73	7.71	13.27
SHAKE128	3.35	1.27	3.15	2.5	2.87	4.58
SHAKE256	1.17	1.32	1.4	3.03	6.38	7.66



Linux – Hash: đơn vị (ms)

Runtime Hash in Linux (ms)						
Hash Type	**1Kb**	**50kb**	**100kb**	**500kb**	**1Mb**	**2Mb**
SHA224	0.909	1.43	1.33	1.38	1.92	3.43
SHA256	2.25	1.18	1.33	2.03	2.01	2.96
SHA384	1	1.53	1.41	1.97	4.54	6.52
SHA512	0.97	1.41	1.25	1.91	3.66	5.92
SHA3-224	1.32	1.05	1.56	2.54	4.25	6.48
SHA3-256	1.18	1.68	1.19	2.98	3.93	7.28
SHA3-384	0.95	1.36	1.94	2.63	5.06	8.37
SHA3-512	1.29	1.37	1.86	3.68	5.72	10.81
SHAKE128	1.23	1.04	1.52	2.16	3.97	5.77
SHAKE256	1.19	1.02	1.6	2.38	3.38	6.45



b. So sánh:

Hiệu suất trên Linux:

Hầu hết các thuật toán băm trên Linux có hiệu suất tốt hơn so với Windows, đặc biệt là với các kích thước tệp nhỏ như 1KB và 50KB.

SHA-2 (SHA224, SHA256, SHA384, SHA512) thường có thời gian băm nhanh hơn các biến thể SHA-3 và SHAKE trên Linux.

Hiệu suất trên Windows:

Với các kích thước tệp lớn hơn như 1MB và 2MB, sự khác biệt về hiệu suất giữa Linux và Windows trở nên rõ rệt hơn. Các thuật toán băm trên Windows có xu hướng mất nhiều thời gian hơn so với Linux.

SHA-3 và SHAKE trên Windows có xu hướng mất nhiều thời gian hơn so với các biến thể SHA-2.

c. Nhận Xét :

Trên cả hai hệ điều hành, khi kích thước đầu vào càng lớn, thời gian xử lý càng tăng. Đối với các đầu vào nhỏ, sự chênh lệch thời gian giữa các thuật toán băm là không đáng kể. Tuy nhiên, với các đầu vào lớn, sự khác biệt về thời gian xử lý trở nên rõ rệt hơn. Các thuật toán băm phức tạp hơn thường mất nhiều thời gian hơn, nhưng lại cung cấp mức độ bảo mật cao hơn. Vì vậy, trong thực tế, người dùng nên chọn thuật toán băm phù hợp với yêu cầu bảo mật và hiệu suất cụ thể của mình.

Khi so sánh thời gian xử lý trên cùng một thuật toán và cùng kích thước đầu vào giữa hai hệ điều hành, Ubuntu thường có thời gian xử lý nhanh hơn Windows, nhưng sự khác biệt này thường không quá lớn.

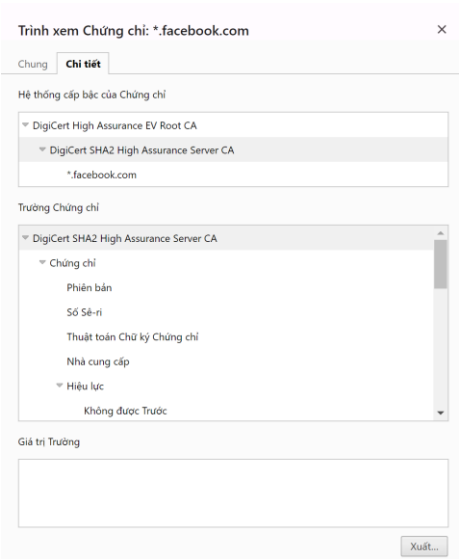
3.2 PKI and digital certificate

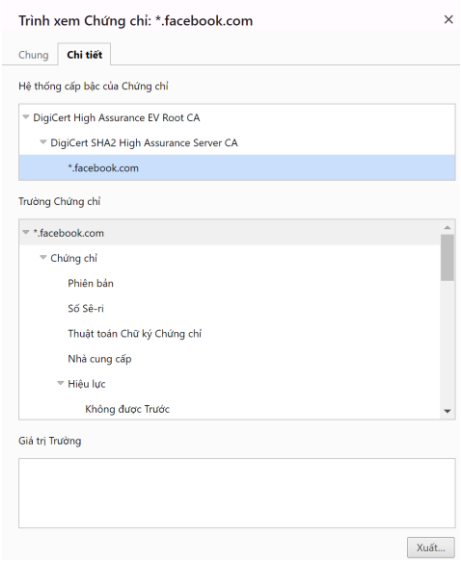
Task này yêu cầu bạn xây dựng một công cụ sử dụng thư viện OpenSSL hoặc CryptoPP để thực hiện hai nhiệm vụ chính:

- Xác minh tính hợp lệ của chứng chỉ X.509.
- Phân tích các trường thông tin trong chứng chỉ X.509, bao gồm tên chủ thể, tên nhà phát hành, khóa công khai, chữ ký, thuật toán chữ ký và các tham số, mục đích sử dụng, khoảng thời gian hợp lệ, v.v.

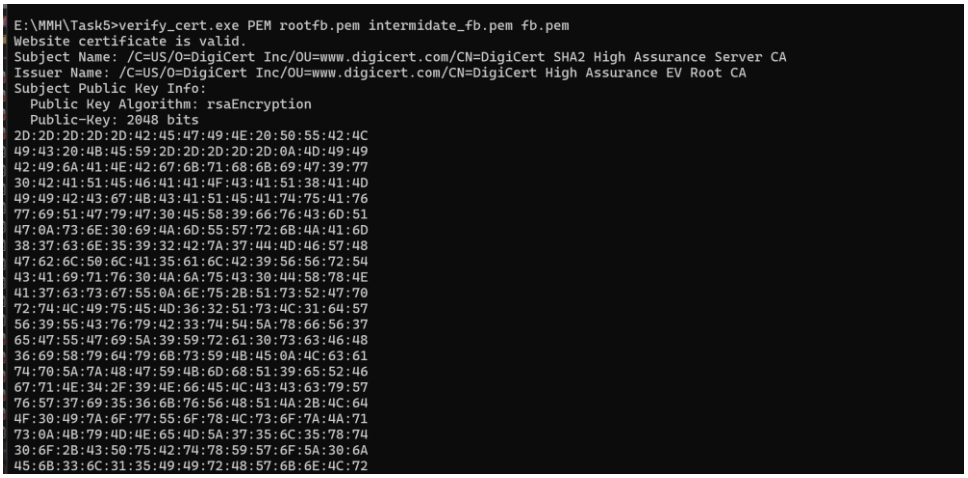
Dưới đây là một ví dụ minh họa về cách thức hoạt động của chương trình:

- Lấy chứng chỉ của trang facebook.com và chứng chỉ của DigiCert SHA2 High Assurance Server CA từ Facebook để kiểm tra và phân tích.



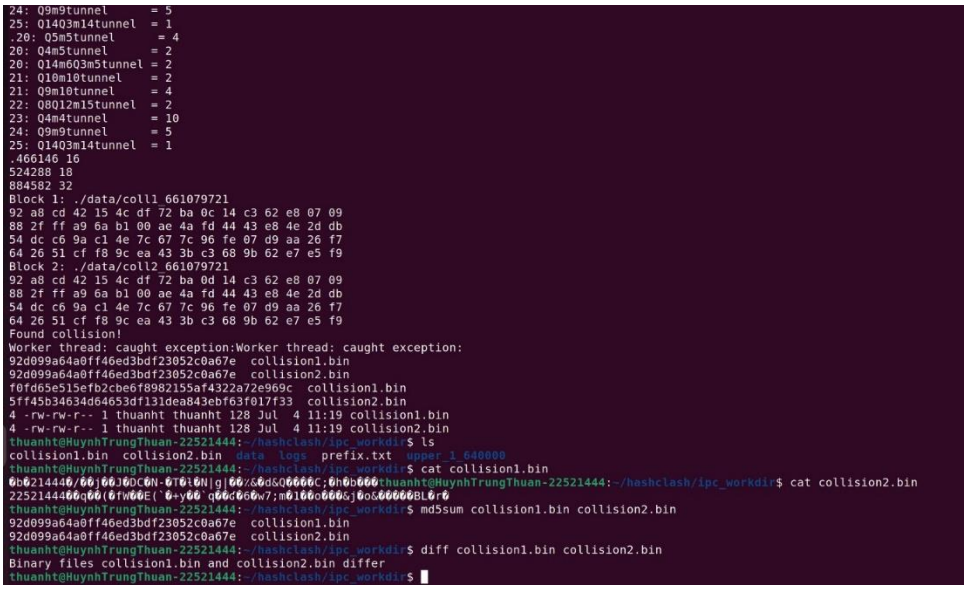


- Lấy Kiểm tra bằng cách nhập vào lần lượt là mode , RootCACert, Intermediate Cert, Website Cert



3.3 Collision and length extension attacks on Hash functions

- Chạy HashClash để tạo 2 file có cùng giá trị băm MD5 và có cùng prefix



- Kiểm tra 2 file bằng hexdump

```
thuanht@HuynhTrungThuan-22521444:~/hashclash/ipc_workdir$ hexdump collision1.bin
0000000 3232 3235 3431 3434 ecf7 b771 9cd8 28e0
0000010 a61d 5766 0591 03c4 4512 6028 2bd8 8179
0000020 60e9 cf71 cad8 8c9b 3604 77dd 3b37 986d
0000030 d731 6ff8 f81d a6fc 6a26 168a 266f b6b6
0000040 a892 42cd 4c15 72df 0dba c314 e862 0907
0000050 2f88 a9ff b16a ae00 fd4a 4344 4ee8 db2d
0000060 dc54 9ac6 4ec1 677c 967c 07fe aad9 f726
0000070 2664 cf51 9cf8 43ea c33b 9b68 e762 f9e5
0000080
thuanht@HuynhTrungThuan-22521444:~/hashclash/ipc_workdir$ hexdump collision2.bin
0000000 3232 3235 3431 3434 edf7 b771 9cd8 28e0
0000010 a61d 5766 0591 03c4 4512 6028 2bd8 8179
0000020 60e9 cf71 cad8 8c9b 3604 77dd 3b37 986d
0000030 d731 6ff8 f81d a6fc 6a26 168a 266f b6b6
0000040 a892 42cd 4c15 72df 0cba c314 e862 0907
0000050 2f88 a9ff b16a ae00 fd4a 4344 4ee8 db2d
0000060 dc54 9ac6 4ec1 677c 967c 07fe aad9 f726
0000070 2664 cf51 9cf8 43ea c33b 9b68 e762 f9e5
0000080
thuanht@HuynhTrungThuan-22521444:~/hashclash/ipc_workdir$
```

- 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: 415 phút)

```
thuanht@HuynhTrungThuan-22521444: ~/hashclash/cpc_workdir
[*] Time before backtrack: 2090 s
[*] Time before backtrack: 2080 s
16777216 127
16846284 128
Block 1: workdir6/coll1_3218359301
90 15 53 c8 37 c0 87 88 a2 d7 8d 4a 31 27 92 98
aa 75 78 b4 65 7d 94 b3 81 d4 db f4 2d 3a b5 af
9b be 1d 8c e2 e8 46 0c e7 a5 3c 84 ee 99 de c7
d6 e4 3f 14 72 01 52 98 09 90 e9 6d 1f 03 67 d9
Block 2: workdir6/coll2_3218359301
90 15 53 c8 37 c0 87 88 a2 d7 8d 4a 31 27 92 98
aa 75 78 b4 65 7d 94 b3 81 d4 db f4 2d 3a b5 af
9b be 1d 8c e2 e8 46 0c e7 a5 3c 84 de 99 de c7
d6 e4 3f 14 72 01 52 98 09 90 e9 6d 1f 03 67 d9
Found collision!
[*] Step 6 completed
[*] Number of backtracks until now: 2
[*] Collision generated: test1.coll test2.coll
bd0241b4c0c4ed73f53429c8505ece4f  test1.coll
bd0241b4c0c4ed73f53429c8505ece4f  test2.coll
[*] Process completed in 415 minutes (2 backtracks).
thuanht@HuynhTrungThuan-22521444:~/hashclash/cpc_workdir$ md5sum test1 test2
5cb31c6bff8705daae15f698986f3bf5  test1
5c942b52160b551b13b6b3f57400a643  test2
thuanht@HuynhTrungThuan-22521444:~/hashclash/cpc_workdir$ md5sum test1.coll tes
2.coll
bd0241b4c0c4ed73f53429c8505ece4f  test1.coll
bd0241b4c0c4ed73f53429c8505ece4f  test2.coll
thuanht@HuynhTrungThuan-22521444:~/hashclash/cpc_workdir$
```

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

- Hướng dẫn dùng tool hashpump

```
thuanht@HuynhTrungThuan-22521444:~/Bàn tài về/HashPump-partialhash-master$ ./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:

- Attach SHA256:

- Attack SHA512: