UNIVERSITY OF INFORMATION TECHNOLOGY

FACULTY OF COMPUTER NETWORK AND COMMUNICATION

Logo, company name

Description automatically generated

**REPORT**

Subject: Cryptography

Semester II (2021 – 2022)

**Report Lab 5 – 6**

Student: Võ Anh Kiệt

Student ID Number: 20520605

Class: NT219.M21.ANTN

University of Information Technology

Lecturer: Nguyễn Ngọc Tự

**Hồ Chí Minh City, May 2022**

Report Lab 5 – 6

**Student Information**

Full Name: Võ Anh Kiệt

Student ID Number: 20520605

Class: ANTN2020

**Device Information**

CPU: Intel core i5 – 8250U @ 1.60 GHz

Ram: 16 GB DDR3L

SSD M2 SATA: 500GB

HDD: 1000GB

Display chip name: UHD 620

**Lab 5: Hash function**

Testcase1: Võ Anh Kiệt - 20520605 - ANTN2020

TimeCounter: mili second

Window

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Hash function | 1KB | 100KB | 1MB | 10MB | 100MB | 1GB |
| SHA224 | 0.0011 | 0.0011 | 0.0010 | 0.0016 | 0.0009 | 0.0017 |
| SHA256 | 0.0010 | 0.0009 | 0.0009 | 0.0009 | 0.0014 | 0.0012 |
| SHA384 | 0.0013 | 0.0010 | 0.0016 | 0.0016 | 0.0013 | 0.0016 |
| SHA512 | 0.0021 | 0.0014 | 0.0016 | 0.0012 | 0.0014 | 0.0014 |
| SHA3-224 | 0.0020 | 0.0023 | 0.0023 | 0.0014 | 0.0015 | 0.0023 |
| SHA3-256 | 0.0021 | 0.0020 | 0.0019 | 0.0015 | 0.0015 | 0.0023 |
| SHA3-384 | 0.0015 | 0.0019 | 0.0016 | 0.0022 | 0.0015 | 0.0026 |
| SHA3-512 | 0.0016 | 0.0026 | 0.0016 | 0.0022 | 0.0014 | 0.0014 |
| SHAKE128  (Digest size: 64 bytes) | 0.0024 | 0.0016 | 0.0015 | 0.0020 | 0.0014 | 0.0015 |
| SHAKE256  (Digest size: 64 bytes) | 0.0016 | 0.0019 | 0.0015 | 0.0015 | 0.0024 | 0.0014 |

Linux

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Hash function | 1KB | 100KB | 1MB | 10MB | 100MB | 1GB |
| SHA224 | 0.0011023 | 0.0011043 | 0.0011304 | 0.0010723 | 0.0011466 | 0.0011235 |
| SHA256 | 0.0010611 | 0.0011198 | 0.0011735 | 0.0010754 | 0.0011526 | 0.0011305 |
| SHA384 | 0.0011874 | 0.0012717 | 0.0012427 | 0.0012436 | 0.0013000 | 0.0012898 |
| SHA512 | 0.0012427 | 0.0012651 | 0.0012348 | 0.0012517 | 0.0012151 | 0.0012587 |
| SHA3-224 | 0.0014128 | 0.0013878 | 0.0014530 | 0.0014461 | 0.0013777 | 0.0013560 |
| SHA3-256 | 0.0014235 | 0.0013346 | 0.0013714 | 0.0013320 | 0.0013525 | 0.0014016 |
| SHA3-384 | 0.0015383 | 0.0013972 | 0.0013342 | 0.0013543 | 0.0013609 | 0.0013521 |
| SHA3-512 | 0.0018721 | 0.0013674 | 0.0013612 | 0.0013270 | 0.0013808 | 0.0013590 |
| SHAKE128  (Digest size: 64 bytes) | 0.0014478 | 0.0013584 | 0.0013890 | 0.0013660 | 0.0013508 | 0.0013897 |
| SHAKE256  (Digest size: 64 bytes) | 0.0013604 | 0.0013412 | 0.0013553 | 0.0014071 | 0.0014218 | 0.0014589 |

**PKI and digital certificate**

Create cert

Text

Description automatically generated

Check cert

Text

Description automatically generated

**Lab 6.1: MD5 collision attacks**

Task 1: Two collision messages have the same prefix string (Generate 2 different files with the same MD5 hash)

Prefix.txt:

Text

Description automatically generated

Solution: Use md5\_fastcoll

Step 1: Run md5\_fastcoll to create 2 different files with the same prefix and the same MD5 hash digest.

Text

Description automatically generated

Step 2: Compare the two out files.



Step 3: Use md5sum and compare hash.



That is collision!

Full step:

Text

Description automatically generated

Task 2: Understanding MD5's property

Solution 1: use md5collgen

Text

Description automatically generated

Solution 2: use hashclash (md5\_fastcoll)

Text

Description automatically generated

Task 3: Two different C++ programs but have the same MD5 (Generating two executable files with the same MD5 hash)

Solution: md5\_fastcoll

Step 1: implement the program

A screenshot of a computer

Description automatically generated with medium confidence

Step 2: Build the \*.o file with the gcc



Step 3: Check the \*.o file with bless and spot the location of a continuous block of A's, the byte offset is 1040 (4160). Slipt the \*.o into 3 parts: prefix, suffix, and the var. The var is required that MD5(prefix || variant1 || suffix) = MD5(prefix || variant2 || suffix).

The prefix is chosen to be a multiple of 64. The offset is 4224.



The suffix is kept with 10FF because of ending byte offset as 10FF in files



Step 4: Using the md5\_fastcoll to get the same hash transfer to file1 and file2

Text

Description automatically generated

Step 5: Create 2 binaries transfer to code1 and code2



Step 6: Check the binary

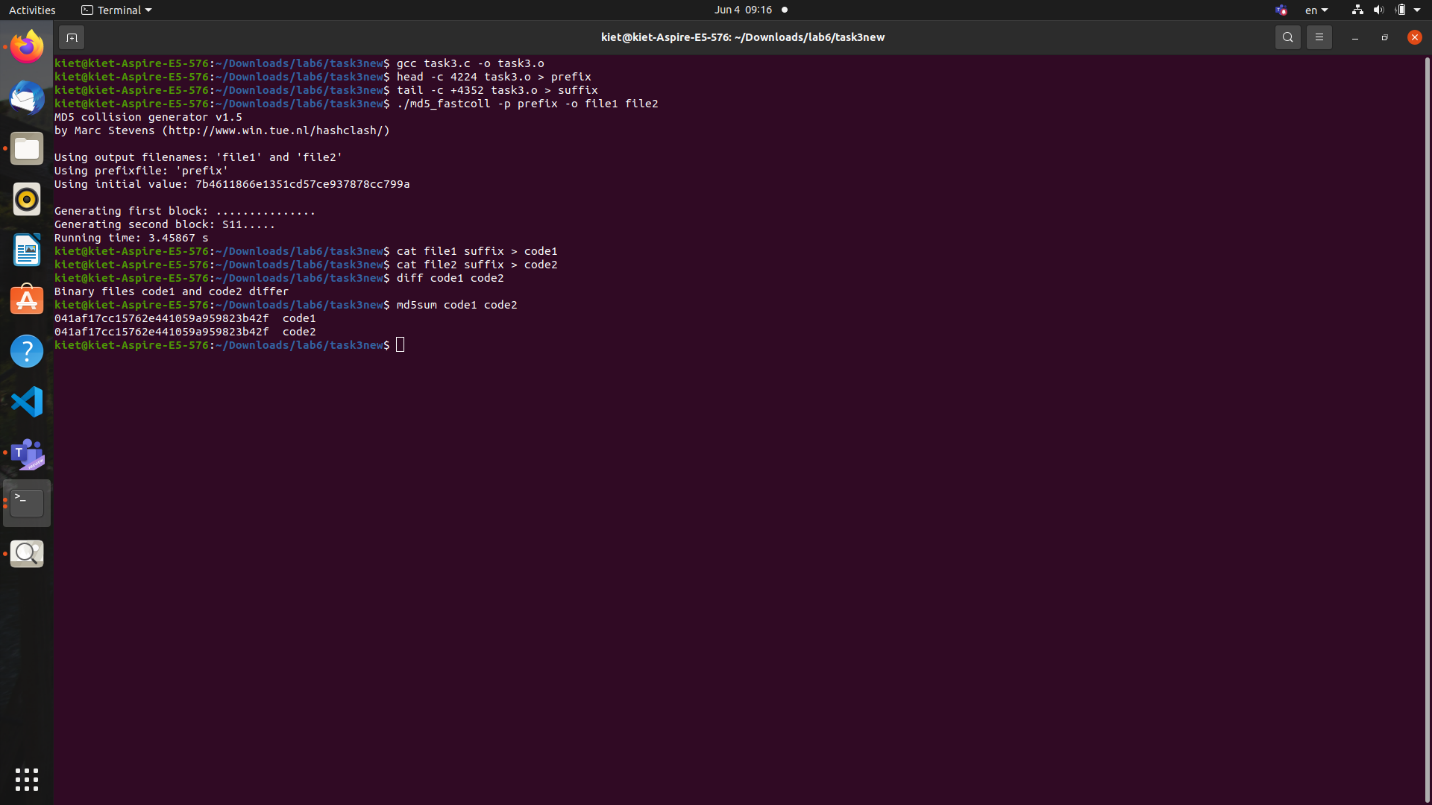


Step 7: Use md5sum and compare hash



That is collision!

Full step:



**Lab 6.2: Length extension attacks on MAC in form**

Text

Description automatically generated with medium confidence