Phòng thí nghiệm An toàn Thông tin InsecLab
Trường Đại học Công nghệ Thông tin (UIT)

BÁO CÁO THỰC HÀNH

Môn học: Nhập môn mạng

Tên chủ đề: Lab 4

GVHD: Tô Trọng Nghĩa

Nhóm: Mệt mỏi

1. THÔNG TIN CHUNG:

(Liệt kê tất cả các thành viên trong nhóm)
Lớp: ATTT2021

STT	Họ và tên	MSSV	Email
1	Nguyễn Thị Minh Châu	21520645	21520645@gm.uit.edu.vn
2	Lưu Thị Huỳnh Như	21521242	21521242@gm.uit.edu.vn

2. NỘI DUNG THỰC HIỆN:[1]

STT	Nội dung	Tình trạng	Trang
1	Yêu cầu 1	100%	2 – 4
2	Yêu cầu 2	100%	4 – 5
3	Yêu cầu 3	100%	5 – 9
4	Yêu cầu 4	100%	9 – 12
5	Yêu cầu 5	100%	12 – 17
6	Yêu cầu 6	100%	17 – 19
Điểm tự đánh giá			**10/10**

Phần bên dưới của báo cáo này là tài liệu báo cáo chi tiết của nhóm thực hiện.

[1] Ghi nội dung công việc, các kịch bản trong bài Thực hành

BÁO CÁO CHI TIẾT

Task 1: Generating message digests (hash values) and HMAC

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Security.Cryptography;
using HashLib;
using System.IO;
using System.Text.RegularExpressions;

namespace HashCalculate
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        string md5_output;
        string sha1_output;
        string sha256_output;
        private static object encoding;

        private void btnCalculate_Click(object sender, EventArgs e)
        {
            string input = tbData.Text;

            switch (comboBoxType.Text.ToString())
            {
                case "Text":
                    md5_output = TextMd5Hash(input);
                    sha1_output = TextSHA1Hash(input);
                    sha256_output = TextSha256Hash(input);
                    break;
                case "Hex":
                    md5_output = HexMd5Hash(input);
                    sha1_output = HexSHA1Hash(input);
                    sha256_output = HexSha256Hash(input);
                    break;
                case "File":

                    break;
            }
            tbMD5.Text = md5_output;
            tbSHA1.Text = sha1_output;
            tbSHA3.Text = sha256_output;
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            comboBoxType.SelectedIndex = 0;
        }
```

```
public static string TextMd5Hash(string message)
{
    using (MD5 md5 = MD5.Create())
    {
        byte[] input = Encoding.UTF8.GetBytes(message);
        byte[] hash = md5.ComputeHash(input);

        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < hash.Length; i++)
        {
            sb.Append(hash[i].ToString("X2")); // print in hexadecimal format
        }
        return sb.ToString().ToLower();
    }
}

public static string HextoString(string message)
{
    int numberChars = message.Length;
    byte[] bytes = new byte[numberChars / 2];
    for (int i = 0; i < numberChars; i += 2)
    {
        bytes[i / 2] = Convert.ToByte(message.Substring(i, 2), 16);
    }
    return Encoding.UTF8.GetString(bytes);
}
public static string HexMd5Hash(string message)
{
    return TextMd5Hash(HextoString(message));
}

static string TextSHA1Hash(string message)
{
    using (SHA1Managed sha1 = new SHA1Managed())
    {
        var hash = sha1.ComputeHash(Encoding.UTF8.GetBytes(message));
        var sb = new StringBuilder(hash.Length * 2);

        foreach (byte b in hash)
        {
            sb.Append(b.ToString("X2"));
        }

        return sb.ToString().ToLower();
    }
}

public static string HexSHA1Hash(string message)
{
    return TextSHA1Hash(HextoString(message));
}

static string TextSha256Hash(string message)
{
    var hashAlgorithm = new Org.BouncyCastle.Crypto.Digests.Sha3Digest(256);
    byte[] input = Encoding.ASCII.GetBytes(message);

    hashAlgorithm.BlockUpdate(input, 0, input.Length);

    byte[] result = new byte[32]; // 256 / 8 = 32
    hashAlgorithm.DoFinal(result, 0);
```

```
            string hashString = BitConverter.ToString(result);
            hashString = hashString.Replace("-", "").ToLowerInvariant();

            return hashString;
        }

        static string HexSha256Hash(string message)
        {
            return TextSha256Hash(HextoString(message));
        }

        private void btnOpenfile_Click(object sender, EventArgs e)
        {
            OpenFileDialog ofd = new OpenFileDialog();
            ofd.Filter = "|*.txt";
            if (ofd.ShowDialog() == DialogResult.OK)
            {
                tbData.Text = ofd.FileName;
                StreamReader srd = new StreamReader(ofd.FileName);
                String data = srd.ReadToEnd();
                srd.Close();

                md5_output = TextMd5Hash(data);
                sha1_output = TextSHA1Hash(data);
                sha256_output = TextSha256Hash(data);
            }
        }


    }
}
```

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

So sánh 2 message: có 6 bytes khác nhau

d1 31 dd 02 c5 e6 ee c4 69 3d	d1 31 dd 02 c5 e6 ee c4 69 3d
9a 06 98 af f9 5c 2f ca b5 87	9a 06 98 af f9 5c 2f ca b5 07
12 46 7e ab 40 04 58 3e b8 fb	12 46 7e ab 40 04 58 3e b8 fb
7f 89 55 ad 34 06 09 f4 b3 02	7f 89 55 ad 34 06 09 f4 b3 02
83 e4 88 83 25 71 41 5a 08 51	83 e4 88 83 25 f1 41 5a 08 51
25 e8 f7 cd c9 9f d9 1d bd f2	25 e8 f7 cd c9 9f d9 1d bd 72
80 37 3c 5b d8 82 3e 31 56 34	80 37 3c 5b d8 82 3e 31 56 34
8f 5b ae 6d ac d4 36 c9 19 c6	8f 5b ae 6d ac d4 36 c9 19 c6
dd 53 e2 b4 87 da 03 fd 02 39	dd 53 e2 34 87 da 03 fd 02 39
63 06 d2 48 cd a0 e9 9f 33 42	63 06 d2 48 cd a0 e9 9f 33 42
0f 57 7e e8 ce 54 b6 70 80 a8	0f 57 7e e8 ce 54 b6 70 80 28
0d 1e c6 98 21 bc b6 a8 83 93	0d 1e c6 98 21 bc b6 a8 83 93

96 f9 65 2b 6f f7 2a 70	96 f9 65 ab 6f f7 2a 70

```
┌──(kali㉿kali)-[~]
└─$ openssl dgst -md5 file1 file2
MD5(file1)= 79054025255fb1a26e4bc422aef54eb4
MD5(file2)= 79054025255fb1a26e4bc422aef54eb4
```

MD5(file1)= 79054025255fb1a26e4bc422aef54eb4

MD5(file2)= 79054025255fb1a26e4bc422aef54eb4

- Nhận xét: dù Hex của các message khác nhau nhưng lại có cùng MD5.

```
┌──(kali㉿kali)-[~]
└─$ sha1sum shattered-1.pdf
38762cf7f55934b34d179ae6a4c80cadccbb7f0a  shattered-1.pdf

┌──(kali㉿kali)-[~]
└─$ sha1sum shattered-2.pdf
38762cf7f55934b34d179ae6a4c80cadccbb7f0a  shattered-2.pdf
```

SHA(shattered-1.pdf) = 38762cf7f55934b34d179ae6a4c80cadccbb7f0a

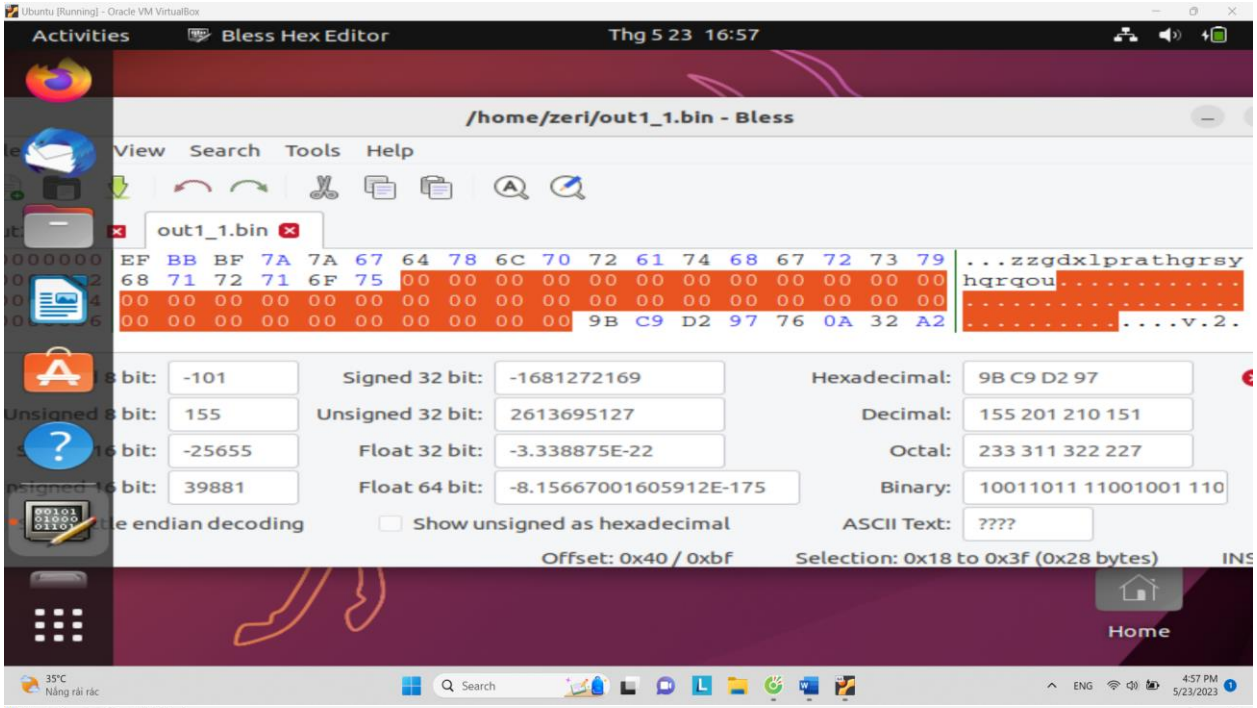
SHA(shattered-1.pdf) = 38762cf7f55934b34d179ae6a4c80cadccbb7f0a

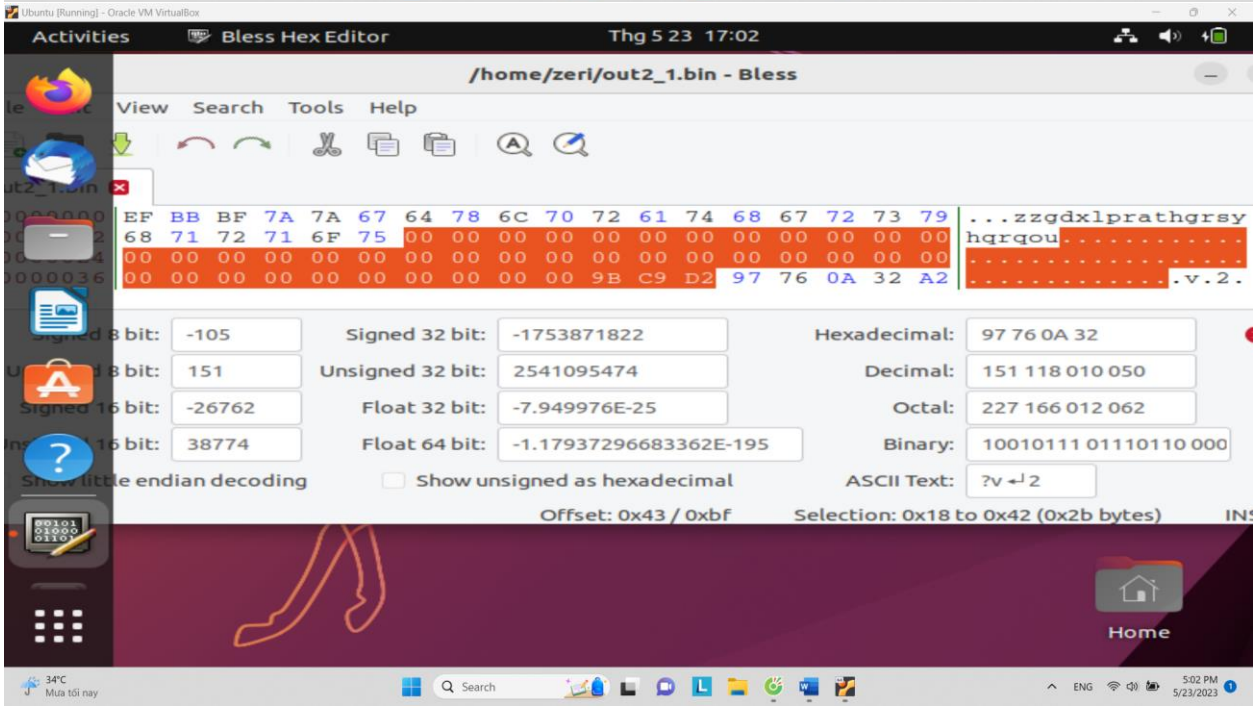
Collison xảy ra khi 2 đầu vào khác nhau lại cho ra cùng một kết quả băm. Vì hàm băm ánh xạ bất kỳ đầu vào nào (bất kể độ dài) cho ra thành một mã có độ dài cố định, nên với một tập đầu vào vô hạn, các hàm băm cuối cùng sẽ cho ra một kết quả lặp lại.

Task 3. Generating Two Different Files with the Same MD5 Hash

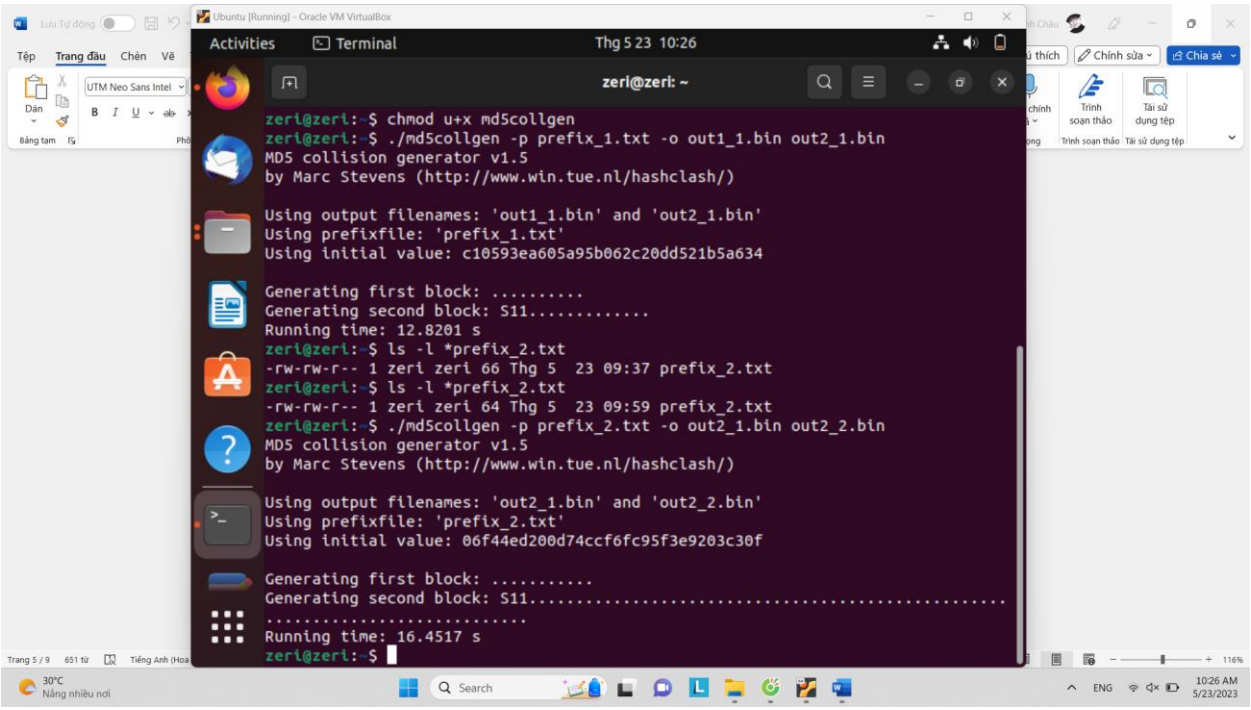
1. Phần padding thêm vào sẽ là những kí tự '0'.
   Khởi tạo file prefix_1 24 bytes, sau đó dùng mdcollgen sinh ra 2 file out1_1bin và out2_1.bin, thực hiện đọc 2 file này bằng bless.

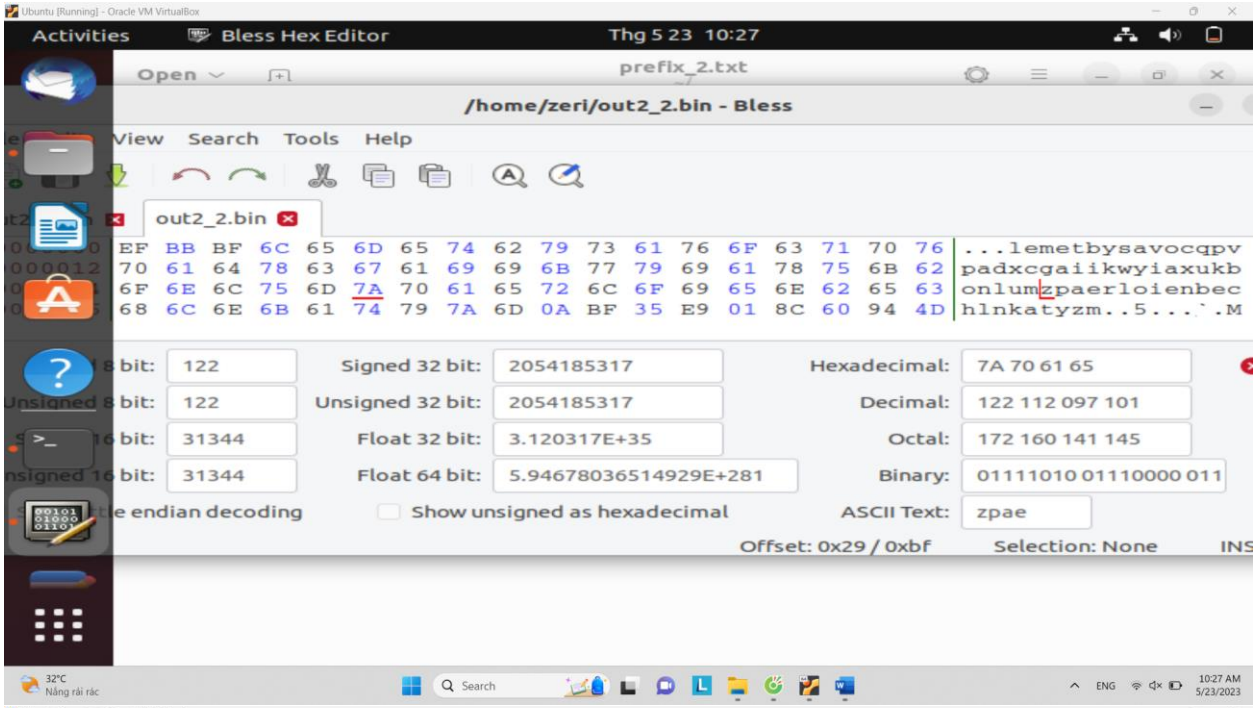
```
zeri@zeri: ~
zeri@zeri:~$ ./md5collgen -p prefix_1.txt  -o out1_1.bin out2_1.bin
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'out1_1.bin' and 'out2_1.bin'
Using prefixfile: 'prefix_1.txt'
Using initial value: c10593ea605a95b062c20dd521b5a634

Generating first block: .
Generating second block: S10.............
Running time: 2.13662 s
zeri@zeri:~$
```

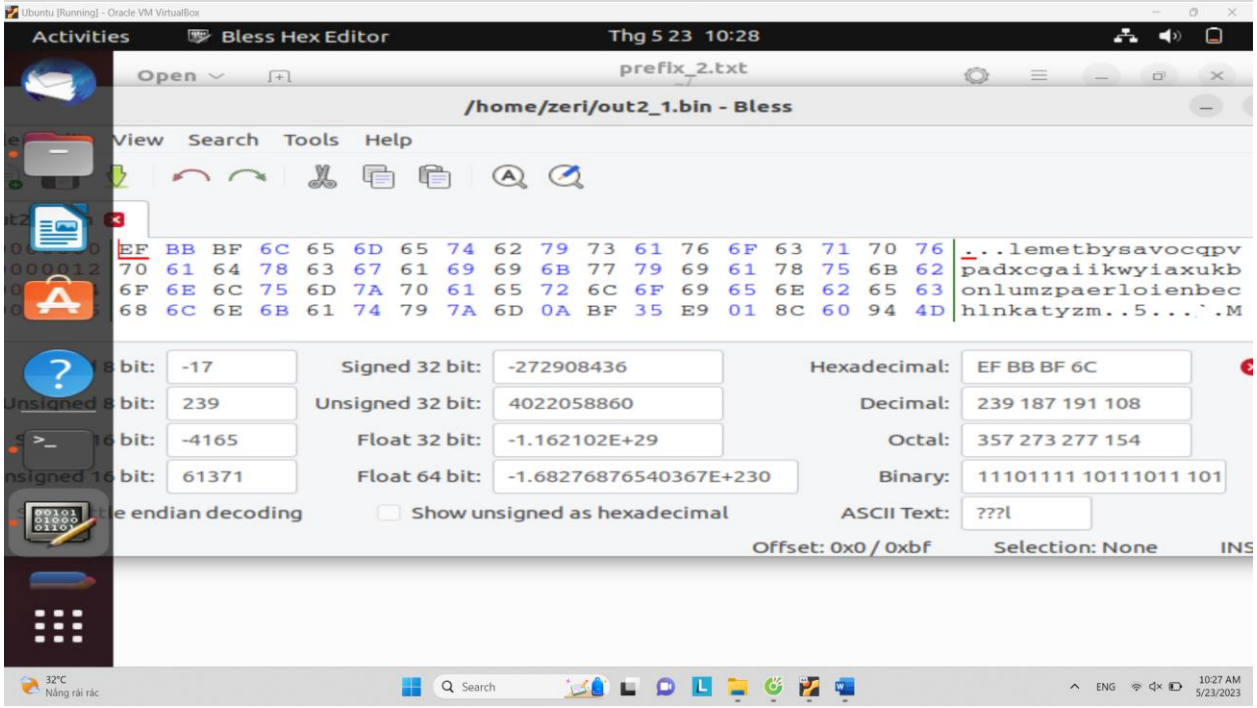




2. Phần padding thêm vào là những kí tự

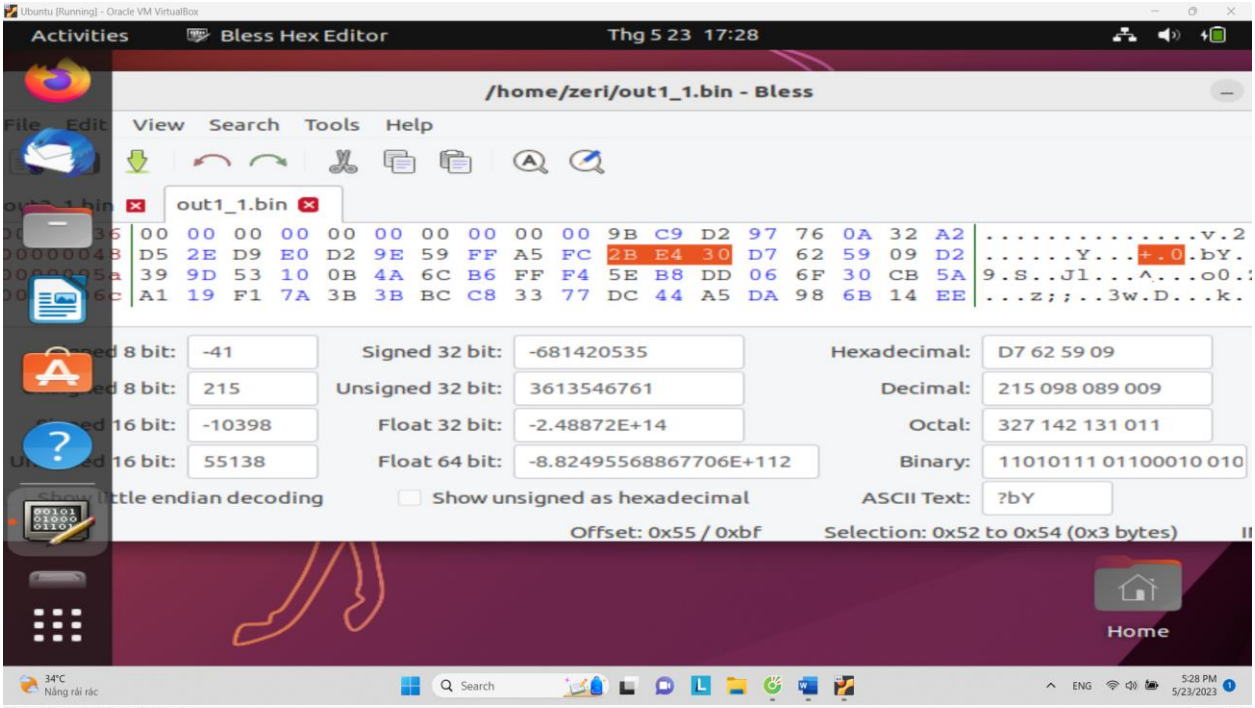

Ubuntu [Running] - Oracle VM VirtualBox
Activities
Terminal
Thg 5 23 10:26
zeri@zeri: ~
zeri@zeri:~$ chmod u+x md5collgen
zeri@zeri:~$ ./md5collgen -p prefix_1.txt -o out1_1.bin out2_1.bin
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)
Using output filenames: 'out1_1.bin' and 'out2_1.bin'
Using prefixfile: 'prefix_1.txt'
Using initial value: c10593ea605a95b062c20dd521b5a634
Generating first block: .....
Generating second block: S11.....
Running time: 12.8201 s
zeri@zeri:~$ ls -l *prefix_2.txt
-rw-rw-r-- 1 zeri zeri 66 Thg 5 23 09:37 prefix_2.txt
zeri@zeri:~$ ls -l *prefix_2.txt
-rw-rw-r-- 1 zeri zeri 64 Thg 5 23 09:59 prefix_2.txt
zeri@zeri:~$ ./md5collgen -p prefix_2.txt -o out2_1.bin out2_2.bin
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)
Using output filenames: 'out2_1.bin' and 'out2_2.bin'
Using prefixfile: 'prefix_2.txt'
Using initial value: 06f44ed200d74ccf6fc95f3e9203c30f
Generating first block: .....
Generating second block: S11.....
Running time: 16.4517 s
zeri@zeri:~$

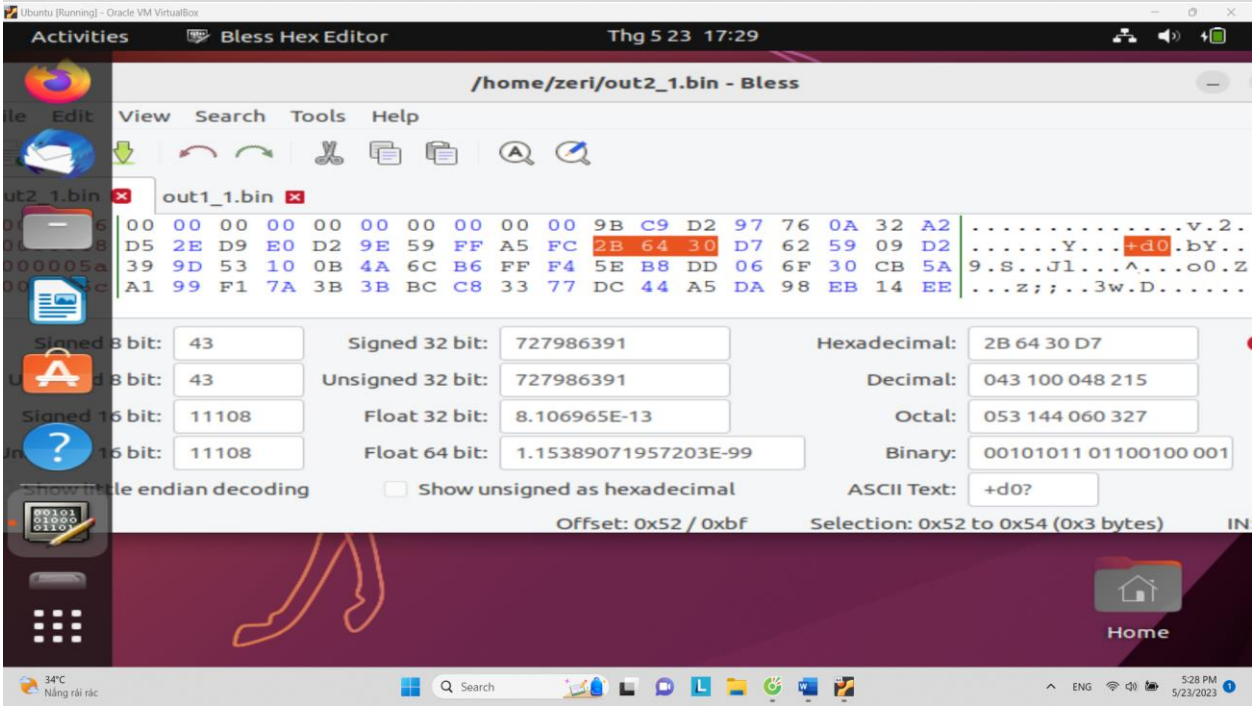




3. Không phải tất cả các byte trong 2 file output sinh ra từ md5collgen hoàn toàn giống nhau (khác ở một số byte). VD ở 2 output1_1 và output2_1 sinh ra từ prefix2 khác nhau ở bye dưới

4.





Task 4. Generating Two Executable Files with the same MD5 Hash

Tạo một file thực thi cho chương trình:

```
┌──(kali㉿kali)-[~]
└─$ gcc lab4.c -o lab
```

Byte bắt đầu chứa mảng là byte thứ 12352

```
00003040  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
00003050  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
00003060  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
00003070  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
00003080  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
00003090  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
000030A0  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
000030B0  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
000030C0  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
000030D0  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
000030E0  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
000030F0  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
00003100  41 41 41 41 41 41 41 41 47 43 43 3A 20 28 44 65  AAAAAAAAGCC: (De
00003110  62 69 61 6E 20 31 31 2E 33 2E 30 2D 35 29 20 31  bian 11.3.0-5) 1
00003120  31 2E 33 2E 30 00 00 00 00 00 00 00 00 00 00 00  1.3.0...........
00003130  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00003140  01 00 00 00 04 00 F1 FF 00 00 00 00 00 00 00 00  ................
00003150  00 00 00 00 00 00 00 00 09 00 00 00 01 00 04 00  ................
00003160  7C 03 00 00 00 00 00 00 20 00 00 00 00 00 00 00  |....... .......
00003170  13 00 00 00 04 00 F1 FF 00 00 00 00 00 00 00 00  ................
00003180  00 00 00 00 00 00 00 00 1E 00 00 00 02 00 0F 00  ................

Offset: 0x3040
```

Nhưng vì prefix cần phải có một phần của mảng và phải chia hết cho 64 nên ta lấy 12416 byte prefix

```
┌──(kali㉿kali)-[~]
└─$ head -c 12416 lab > prefix
```

Tạo 2 file output với prefix vừa tạo.

```
┌──(kali㉿kali)-[~]
└─$ ./md5collgen -p prefix -o out1.bin out2.bin
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'out1.bin' and 'out2.bin'
Using prefixfile: 'prefix'
Using initial value: 8c275ecc21f3274344bdc39c3d8bc37e

Generating first block: .......
Generating second block: S00...........................
Running time: 11.5724 s
```

Lấy từ byte 12544 đến cuối để làm suffix, lấy 128 byte cuối của out1.bin vào file p và 128 byte cuối của out2.bin vào file q

```
┌──(kali㉿kali)-[~]
└─$ tail -c +12544 lab > suffix

┌──(kali㉿kali)-[~]
└─$ tail -c 128 out1.bin > p

┌──(kali㉿kali)-[~]
└─$ tail -c 128 out2.bin > q
```

Nối file lại với nhau:

```
┌──(kali㉿kali)-[~]
└─$ cat prefix p suffix > kq1

┌──(kali㉿kali)-[~]
└─$ cat prefix q suffix > kq2
```

Kết quả md5:

```
┌──(kali㉿kali)-[~]
└─$ md5sum kq1
9437f1f0dbfe5f3d959080d0baf5e87b  kq1

┌──(kali㉿kali)-[~]
└─$ md5sum kq2
9437f1f0dbfe5f3d959080d0baf5e87b  kq2
```

Kết quả chạy chương trình:

```
┌──(kali㉿kali)-[~]
└─$ chmod +x kq1

┌──(kali㉿kali)-[~]
└─$ ./kq1
414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414
1414141414141414141414141414141414141c53aa67d66c2b59adbdffe39737cb61d652893c7c085514d47a678add1
af2f2c6819f6e4d2ca177f6654b28a95565d959842bb5773d4f78ca1c67d5bfd2d7010ef1e7aa5ae1fdb69e435493
bef9e6784e5f6ba6019c5bbf5c0511979b239715fe2202382457175ccf51aa67e4a735d5ff6f49c1d92d0966cbc6d
e21e4141414141414141

┌──(kali㉿kali)-[~]
└─$ chmod +x kq2

┌──(kali㉿kali)-[~]
└─$ ./kq2
414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414
1414141414141414141414141414141414141c53aa67d66c2b59adbdffe39737cb61d65289347c085514d47a678add1
af2f2c6819f6e4d2ca177f6654b28a15575d959842bb5773d4f78ca1467d5bfd2d7010ef1e7aa5ae1fdb69e435493
bef9e678465f6ba6019c5bbf5c0511979b239715fe2202382457175ccf512a67e4a735d5ff6f49c1d92d096ecbc6d
e21e4141414141414141
```

Kết quả so sánh:

```
┌──(kali㉿kali)-[~]
└─$ ./kq1 > kq1.txt

┌──(kali㉿kali)-[~]
└─$ ./kq2 > kq2.txt

┌──(kali㉿kali)-[~]
└─$ diff kq1.txt kq2.txt
1c1
< 4141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141c53aa67d66c2b59adbdffe39737cb61d652893c7c085514d47a678add1af2f2c6819f6e4d2ca177f6654b28a95565d959842bb5773d4f78ca1c67d5bfd2d7010ef1e7aa5ae1fdb69e435493bef9e6784e5f6ba6019c5bbf5c0511979b239715fe2202382457175ccf51aa67e4a735d5ff6f49c1d92d0966cbc6de21e4141414141414141
---
> 4141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141414141c53aa67d66c2b59adbdffe39737cb61d65289347c085514d47a678add1af2f2c6819f6e4d2ca177f6654b28a15575d959842bb5773d4f78ca1467d5bfd2d7010ef1e7aa5ae1fdb69e435493bef9e678465f6ba6019c5bbf5c0511979b239715fe2202382457175ccf512a67e4a735d5ff6f49c1d92d096ecbc6de21e4141414141414141
```

Task 5. Manually Verifying an X.509 Certificate

Các bước theo hướng dẫn như sau:

```
zeri@zeri:~$ openssl s_client -connect www.seedsecuritylabs.org:443 -showcerts
CONNECTED(00000003)
depth=2 C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert Global Root CA
verify return:1
depth=1 C = US, O = DigiCert Inc, CN = DigiCert TLS RSA SHA256 2020 CA1
verify return:1
depth=0 C = US, ST = California, L = San Francisco, O = "GitHub, Inc.", CN = *.github.io
verify return:1
---
Certificate chain
 0 s:C = US, ST = California, L = San Francisco, O = "GitHub, Inc.", CN = *.github.io
   i:C = US, O = DigiCert Inc, CN = DigiCert TLS RSA SHA256 2020 CA1
   a:PKEY: rsaEncryption, 2048 (bit); sigalg: RSA-SHA256
   v:NotBefore: Feb 21 00:00:00 2023 GMT; NotAfter: Mar 20 23:59:59 2024 GMT
-----BEGIN CERTIFICATE-----
MIIHEjCCBfqgAwIBAgIQBE1y13zdpwLdWmfyoju92TANBgkqhkiG9w0BAQsFADBP
MQswCQYDVQQGEwJVUzEVMBMGA1UEChMMRGlnaUNlcnQgSW5jMSkwJwYDVQQDEyBE
aWdpQ2VydCBUTFMgUlNBIFNIQTI1NiAyMDIwIENBMTAeFw0yMzAyMjEwMDAwMDBa
Fw0yNDAzMjAyMzU5NTlaMGcxCzAJBgNVBAYTAlVTMRMwEQYDVQQIEwpDYWxpZm9y
bmlhMRYwFAYDVQQHEw1TYW4gRnJhbmNpc2NvMRUwEwYDVQQKEwxHaXRIdWIsIElu
Yy4xFDASBgNVBAMMCyouZ2l0aHViLmlvMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8A
MIIBCgKCAQEAuLBgDhov8bGGS2TsEZ+meb7oh/GIxbRJmxC7yq/qr75UDHhDf8p7
TkVbCyQp8bsj/Bmkx2xwSXZT0wkjZbJIe7Ycqgca4nka+Xpe5xb4pkrVOaPiDfdX
7+34CHZbUtqL0OYebi/5D5lLalLKNOGkySAz05foenfFAxAmQYJhR6KvxFY/dqI4
y7JwrnJ6Q8F+J6Ne1uP256UwcL0qlid6e/tA0ld3ryMSJ0I6xgtqjL26Le4/nxXu
YlekppVQr0OwrHa44Q7Z/1bsdFCGtR+WLNGVBeW3BWeTTp7yWjgfp49DWt48V9pI
elDGiDgVyJcsLOz4OQk2vRmNA1ZBZgck4wIDAQABo4ID0DCCA8wwHwYDVR0jBBgw
FoAUt2ui6qiqhIx56rTaD5iyxZV2ufQwHQYDVR0OBBYEFI0CHHVazcamQXhpKMP3
qqeYO9W7MHsGA1UdEQR0MHKCCyouZ2l0aHViLmlvgglnaXRodWIuaW+CDCouZ2l0
aHViLmNvbYIKZ2l0aHViLmNvbYIOd3d3LmdpdGh1Yi5jb22CFyouZ2l0aHVidXNl
cmNvbnRlbnQuY29tghVnaXRodWJ1c2VyY29udGVudC5jb20wDgYDVR0PAQH/BAQD
AgWgMB0GA1UdJQQWMBQGCCsGAQUFBwMBBggrBgEFBQcDAjCBjwYDVR0fBIGHMIGE
MECgPqA8hjpodHRwOi8vY3JsMy5kaWdpY2VydC5jb20vRGlnaUNlcnRUTFNSU0FT
SEEyNTYyMDIwQ0ExLTQuY3JsMECgPqA8hjpodHRwOi8vY3JsNC5kaWdpY2VydC5j
b20vRGlnaUNlcnRUTFNSU0FTSEEyNTYyMDIwQ0ExLTQuY3JsMD4GA1UdIAQ3MDUw
MwYGZ4EMAQICMCkwJwYIKwYBBQUHAgEWG2h0dHA6Ly93d3cuZGlnaWNlcnQuY29t
L0NQUzB/BggrBgEFBQcBAQRzMHEwJAYIKwYBBQUHMAGGGGh0dHA6Ly9vY3NwLmRp
Z2ljZXJ0LmNvbTBJBggrBgEFBQcwAoY9aHR0cDovL2NhY2VydHMuZGlnaWNlcnQu
Y29tL0RpZ2lDZXJ0VExTUlNBU0hBMjU2MjAyMENBMS0xLmNydDAJBgNVHRMEAjAA
MIIBfgYKKwYBBAHWeQIEAgSCAW4EggFqAWgAdwB2/4g/Crb7lVHCYcz1h7o0tKTN
uyncaEIKn+ZnTFo6dAAAAYZ0gHV7AAAEAwBIMEYCIQCqfmfSO8MxeeVZ/fJzqqBB
p+VqeRDUOUBVGyTTOn43ewIhAJT0S27mmGUlpqNiDADP+Jo8C6kYHF+7U6TY74bH
XHAaAHYAc9meiRtMlnigIH1Hneayxhz QUV5xGSqMa4AQesF3crUAAAGGdIB1agAA
BAMARzBFAiEAguB+XQVANBj2MPcJzbz+LBPrkDDOEO3op52jdHUSW3ICIF0fnYdW
qvdtmgQNSns13pAppdQWp4/f/jerNYskI7krAHUASLDja9qmRzQP5WoC+p0w6xxS
ActW3SyB2bu/qznYhHMAAAGGdIB1SgAABAMARjBEAiAT/wA2qGGHSKZqBAm84z6q
E+dGPQZ1aCMY52pFSfcw8QIgP/SciuZG02X2mBO/miDT2hCp4y5d2sc7FE5PThyC
pbMwDQYJKoZIhvcNAQELBQADggEBADekGxEin/yfyWcHj6qGE5/gCB1uDI1l+wN5
UMZ2ujCQoKQceRMHuVoYjZdMBXGK0CIXxhmiIosD9iyEcWxV3+KZQ2Xl17e3N0zG
yOXx2Kd7B13ruBxQpKOO8Ez4uGpyWb5DDoretV6Pnj9aQ2SCzODedvS+phIKBmi7
d+FM70tNZ6/2csdrG5xIU6d/7XYYXPD2xkwkU1dX4UKmPa7h9ZPyavopcgE+twbx
LxoOkcXsNb/12jOV3iQSDfXDI41AgtFc694KCOjlg+UKizpemE53T5/cq37OqChP
qnlPyb6PYIhua/kgbH84ltba1xEDQ9i4UYfOMiJNZEzEdSfQ498=
```

```
zeri@zeri:~$ openssl x509 -in c1.pem -noout -modulus
Modulus=C14BB3654770BCDD4F58DBEC9CEDC366E51F311354AD4A66461F2C0AEC6407E52EDCDCB90A20EDDFE3C4D09E9AA97
A1D8288E51156DB1E9F58C251E72C340D2ED292E156CBF1795FB3BB87CA25037B9A52416610604F571349F0E8376783DFE7D3
4B674C2251A6DF0E9910ED57517426E27DC7CA622E131B7F238825536FC13458008B84FFF8BEA75849227B96ADA2889B15BCA
07CDFE951A8D5B0ED37E236B4824B62B5499AECC767D6E33EF5E3D6125E44F1BF71427D58840380B18101FAF9CA32BBB48E27
8727C52B74D4A8D697DEC364F9CACE53A256BC78178E490329AEFB494FA415B9CEF25C19576D6B79A72BA2272013B5D03D40D
321300793EA99F5
zeri@zeri:~$ openssl x509 -in c1.pem -text
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            06:d8:d9:04:d5:58:43:46:f6:8a:2f:a7:54:22:7e:c4
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert Global Root CA
        Validity
            Not Before: Apr 14 00:00:00 2021 GMT
            Not After : Apr 13 23:59:59 2031 GMT
        Subject: C = US, O = DigiCert Inc, CN = DigiCert TLS RSA SHA256 2020 CA1
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:c1:4b:b3:65:47:70:bc:dd:4f:58:db:ec:9c:ed:
                    c3:66:e5:1f:31:13:54:ad:4a:66:46:1f:2c:0a:ec:
                    64:07:e5:2e:dc:dc:b9:0a:20:ed:df:e3:c4:d0:9e:
                    9a:a9:7a:1d:82:88:e5:11:56:db:1e:9f:58:c2:51:
                    e7:2c:34:0d:2e:d2:92:e1:56:cb:f1:79:5f:b3:bb:
                    87:ca:25:03:7b:9a:52:41:66:10:60:4f:57:13:49:
                    f0:e8:37:67:83:df:e7:d3:4b:67:4c:22:51:a6:df:
                    0e:99:10:ed:57:51:74:26:e2:7d:c7:ca:62:2e:13:
                    1b:7f:23:88:25:53:6f:c1:34:58:00:8b:84:ff:f8:
                    be:a7:58:49:22:7b:96:ad:a2:88:9b:15:bc:a0:7c:
                    df:e9:51:a8:d5:b0:ed:37:e2:36:b4:82:4b:62:b5:
                    49:9a:ec:c7:67:d6:e3:3e:f5:e3:d6:12:5e:44:f1:
                    bf:71:42:7d:58:84:03:80:b1:81:01:fa:f9:ca:32:
                    bb:b4:8e:27:87:27:c5:2b:74:d4:a8:d6:97:de:c3:
                    64:f9:ca:ce:53:a2:56:bc:78:17:8e:49:03:29:ae:
                    fb:49:4f:a4:15:b9:ce:f2:5c:19:57:6d:6b:79:a7:
                    2b:a2:27:20:13:b5:d0:3d:40:d3:21:30:07:93:ea:
                    99:f5
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints: critical
```

```
                Policy: 2.23.140.1.2.3
    Signature Algorithm: sha256WithRSAEncryption
    Signature Value:
        80:32:ce:5e:0b:dd:6e:5a:0d:0a:af:e1:d6:84:cb:c0:8e:fa:
        85:70:ed:da:5d:b3:0c:f7:2b:75:40:fe:85:0a:fa:f3:31:78:
        b7:70:4b:1a:89:58:ba:80:bd:f3:6b:1d:e9:7e:cf:0b:ba:58:
        9c:59:d4:90:d3:fd:6c:fd:d0:98:6d:b7:71:82:5b:cf:6d:0b:
        5a:09:d0:7b:de:c4:43:d8:2a:a4:de:9e:41:26:5f:bb:8f:99:
        cb:dd:ae:e1:a8:6f:9f:87:fe:74:b7:1f:1b:20:ab:b1:4f:c6:
        f5:67:5d:5d:9b:3c:e9:ff:69:f7:61:6c:d6:d9:f3:fd:36:c6:
        ab:03:88:76:d2:4b:2e:75:86:e3:fc:d8:55:7d:26:c2:11:77:
        df:3e:02:b6:7c:f3:ab:7b:7a:86:36:6f:b8:f7:d8:93:71:cf:
        86:df:73:30:fa:7b:ab:ed:2a:59:c8:42:84:3b:11:17:1a:52:
        f3:c9:0e:14:7d:a2:5b:72:67:ba:71:ed:57:47:66:c5:b8:02:
        4a:65:34:5e:8b:d0:2a:3c:20:9c:51:99:4c:e7:52:9e:f7:6b:
        11:2b:0d:92:7e:1d:e8:8a:eb:36:16:43:87:ea:2a:63:bf:75:
        3f:eb:de:c4:03:bb:0a:3c:f7:30:ef:eb:af:4c:fc:8b:36:10:
        73:3e:f3:a4
```

```
zeri@zeri:~$ openssl x509 -in c0.pem -text
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            04:4d:72:d7:7c:dd:a7:02:dd:5a:67:f2:a2:3b:bd:d9
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = US, O = DigiCert Inc, CN = DigiCert TLS RSA SHA256 2020 CA1
        Validity
            Not Before: Feb 21 00:00:00 2023 GMT
            Not After : Mar 20 23:59:59 2024 GMT
        Subject: C = US, ST = California, L = San Francisco, O = "GitHub, Inc.", CN = *.github.io
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:b8:b0:60:0e:1a:2f:f1:b1:86:4b:64:ec:11:9f:
                    a6:79:be:e8:87:f1:88:c5:b4:49:9b:10:bb:ca:af:
                    ea:af:be:54:0c:78:43:7f:ca:7b:4e:45:5b:0b:24:
                    29:f1:bb:23:fc:19:a4:c7:6c:70:49:76:53:d3:09:
                    23:65:b2:48:7b:b6:1c:aa:07:1a:e2:79:1a:f9:7a:
                    5e:e7:16:f8:a6:4a:d5:39:a3:e2:0d:f7:57:ef:ed:
                    f8:08:76:5b:52:da:8b:d0:e6:1e:6e:2f:f9:0f:99:
                    4b:6a:52:ca:34:e1:a4:c9:20:33:d3:97:e8:7a:77:
                    c5:03:10:26:41:82:61:47:a2:af:c4:56:3f:76:a2:
                    38:cb:b2:70:ae:72:7a:43:c1:7e:27:a3:5e:d6:e3:
                    f6:e7:a5:30:70:bd:2a:96:27:7a:7b:fb:40:d2:57:
                    77:af:23:12:27:42:3a:c6:0b:6a:8c:bd:ba:2d:ee:
                    3f:9f:15:ee:62:57:a4:a6:95:50:af:43:b0:ac:76:
                    b8:e1:0e:d9:ff:56:ec:74:50:86:b5:1f:96:2c:d1:
                    95:05:e5:b7:05:67:93:4e:9e:f2:5a:38:1f:a7:8f:
                    43:5a:de:3c:57:da:48:7a:50:c6:88:38:15:c8:97:
                    2c:2c:ec:f8:39:09:36:bd:19:8d:03:56:41:66:07:
                    24:e3
                Exponent: 65537 (0x10001)
```

```
                4f:4E:1C:82:A5:B3
    Signature Algorithm: sha256WithRSAEncryption
    Signature Value:
        37:a4:1b:11:22:9f:fc:9f:c9:67:07:8f:aa:86:13:9f:e0:08:
        1d:6e:0c:8d:65:fb:03:79:50:c6:76:ba:30:90:a0:a4:1c:79:
        13:07:b9:5a:18:8d:97:4c:05:71:8a:d0:22:17:c6:19:a2:22:
        8b:03:f6:2c:84:71:6c:55:df:e2:99:43:65:e5:d7:b7:b7:37:
        4c:c6:c8:e5:f1:d8:a7:7b:07:5d:eb:b8:1c:50:a4:a3:8e:f0:
        4c:f8:b8:6a:72:59:be:43:0e:8a:de:b5:5e:8f:9e:3f:5a:43:
        64:82:cc:e0:de:76:f4:be:a6:12:0a:06:68:bb:77:e1:4c:ef:
        4b:4d:67:af:f6:72:c7:6b:1b:9c:48:53:a7:7f:ed:76:18:5c:
        f0:f6:c6:4c:24:53:57:57:e1:42:a6:3d:ae:e1:f5:93:f2:6a:
        fa:29:72:01:3e:b7:06:f1:2f:1a:0e:91:c5:ec:35:bf:f5:da:
        33:95:de:24:12:0d:f5:c3:23:8d:40:82:d1:5c:eb:de:0a:08:
        e8:e5:83:e5:0a:8b:3a:5e:98:4e:77:4f:9f:dc:ab:7e:ce:a8:
        28:4f:aa:79:4f:c9:be:8f:60:88:6e:6b:f9:20:6c:7f:38:96:
        d6:da:d7:11:03:43:d8:b8:51:87:ce:32:22:4d:64:4c:c4:75:
        27:d0:e3:df
    -----BEGIN CERTIFICATE-----
```

```
zeri@zeri:~$ cat signature | tr -d '[:space:]:'
37a41b11229ffc9fc967078faa86139fe0081d6e0c8d65fb037950c676ba3090a0a41c791307b95a188d974c05718ad02217c
619a2228b03f62c84716c55dfe2994365e5d7b7b7374cc6c8e5f1d8a77b075debb81c50a4a38ef04cf8b86a7259be430e8ade
b55e8f9e3f5a436482cce0de76f4bea6120a0668bb77e14cef4b4d67aff672c76b1b9c4853a77fed76185cf0f6c64c2453575
7e142a63daee1f593f26afa2972013eb706f12f1a0e91c5ec35bff5da3395de24120df5c3238d4082d15cebde0a08e8e583e5
0a8b3a5e984e774f9fdcab7ecea8284faa794fc9be8f60886e6bf9206c7f3896d6dad7110343d8b85187ce32224d644cc4752
7d0e3dfzeri@zeri:~$
```

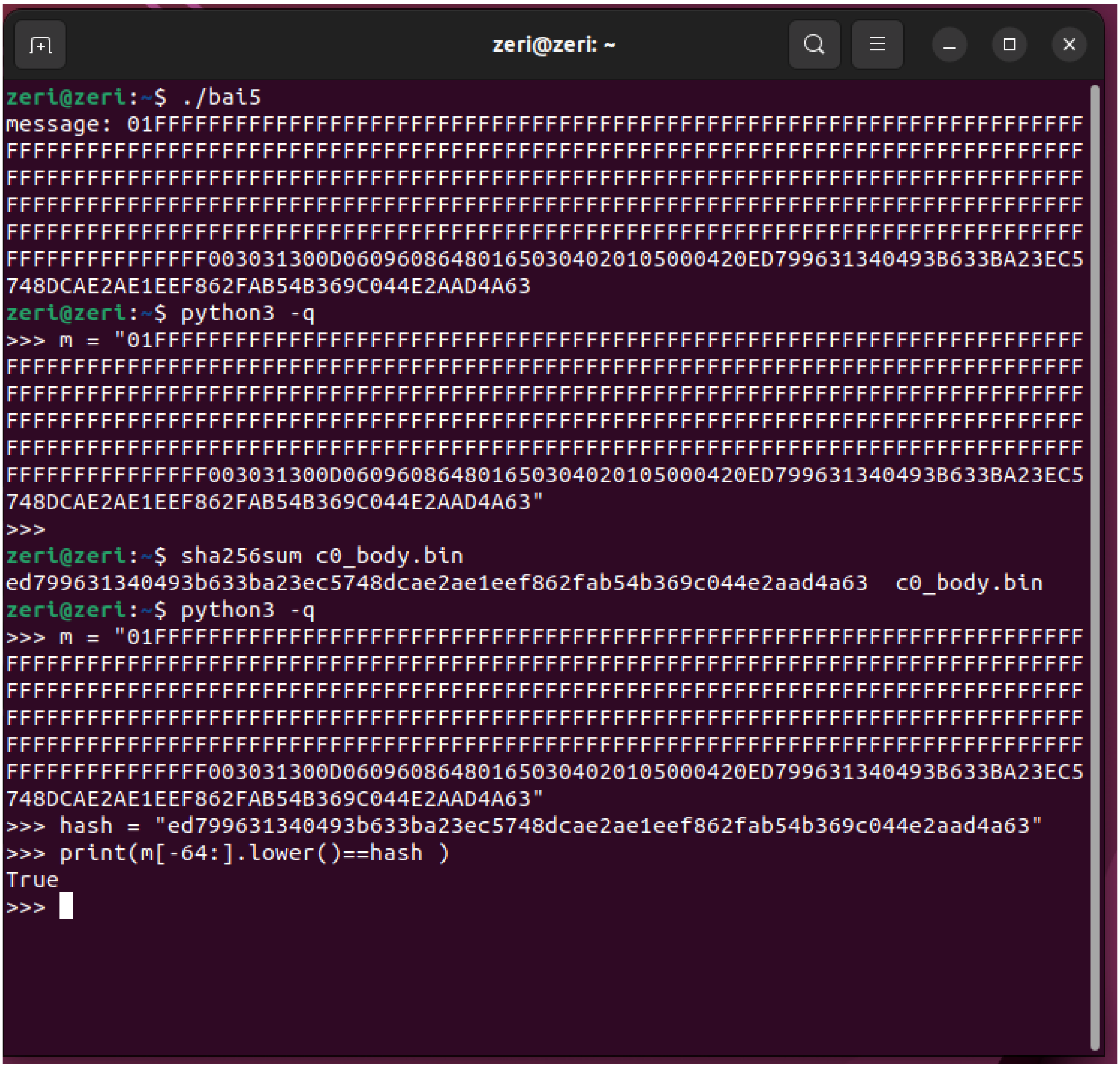
```
zeri@zeri:~$ openssl asn1parse -i -in c0.pem -strparse 4 -out c0_body.bin -noout
zeri@zeri:~$ sha256sum c0_body.bin
ed799631340493b633ba23ec5748dcae2ae1eef862fab54b369c044e2aad4a63  c0_body.bin
zeri@zeri:~$
```

Viết chương trình đơn giản để tính m:

bai5.c

```c
#include <stdio.h>
#include <openssl/bn.h>
#define NBITS 128
void printBN(char *msg, BIGNUM *a)
{ /* Use BN_bn2hex(a) for hex string
        * Use BN_bn2dec(a) for decimal string */
    char *number_str = BN_bn2hex(a);
    printf("%s %s\n", msg, number_str);
    OPENSSL_free(number_str);
}
int main()
{
    BN_CTX *ctx = BN_CTX_new();

    BIGNUM *n = BN_new();
    BIGNUM *e = BN_new();
    BIGNUM *sig = BN_new();
    BIGNUM *m = BN_new();

    BN_hex2bn(&n, "c14bb3654770bcdd4f58dbec9cedc366e51f311354ad4a66461f2c0aec6407e52edcdcb90a20eddfe3c4d09e9aa97a1d
    BN_hex2bn(&e, "010001");
    BN_hex2bn(&sig, "37a41b11229ffc9fc967078faa86139fe0081d6e0c8d65fb037950c676ba3090a0a41c791307b95a188d974c05718ad0

    BN_mod_exp(m, sig, e, n, ctx);

    printBN("message:", m);

    return 0;
}
```

Ta thu được kết quả true như bên dưới:



```
zeri@zeri:~$ ./bai5
message: 01FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFF003031300D060960864801650304020105000420ED799631340493B633BA23EC5
748DCAE2AE1EEF862FAB54B369C044E2AAD4A63
zeri@zeri:~$ python3 -q
>>> m = "01FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFF003031300D060960864801650304020105000420ED799631340493B633BA23EC5
748DCAE2AE1EEF862FAB54B369C044E2AAD4A63"
>>> 
zeri@zeri:~$ sha256sum c0_body.bin
ed799631340493b633ba23ec5748dcae2ae1eef862fab54b369c044e2aad4a63  c0_body.bin
zeri@zeri:~$ python3 -q
>>> m = "01FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFF003031300D060960864801650304020105000420ED799631340493B633BA23EC5
748DCAE2AE1EEF862FAB54B369C044E2AAD4A63"
>>> hash = "ed799631340493b633ba23ec5748dcae2ae1eef862fab54b369c044e2aad4a63"
>>> print(m[-64:].lower()==hash )
True
>>> 
```

Task 6. Programming application for X509 Certificate verification

```
//  Defines the entry point for the console application
/*ECC parameters p,a,b, P (or G), n, h where p=h.n*/

/* Source, Sink */
#include "include/cryptopp/filters.h"
#include "include/cryptopp/x509cert.h"
#include <ctime>
#include <iostream>
#include <string>
using namespace std;

/* Randomly generator*/
#include "include/cryptopp/osrng.h"
```

```
using CryptoPP::AutoSeededRandomPool;
#include "include/cryptopp/files.h"
/* Integer arithmatics*/
#include "include/cryptopp/cryptlib.h"
#include "include/cryptopp/secblock.h"
#include "include/cryptopp/rsa.h"
#include "include/cryptopp/sha.h"
#include "include/cryptopp/hex.h"
#include "include/cryptopp/pem.h"
/* standard curves*/

#include <fstream>
namespace ASN1 = CryptoPP::ASN1;
using CryptoPP::OID;
using namespace CryptoPP;


int main(int argc, char* argv[])
{
    ifstream c1;
    c1.open("c0.txt");
    string cert1, line;
    while (getline(c1, line))
    {
        cert1 += line+"\r\n";
    }
    c1.close();

    ifstream c2;
    c2.open("c1.txt");
    string cert0;
    while (getline(c2, line))
    {
        cert0 += line + "\r\n";
    }
    c2.close();
    StringSource ss1(cert1 , true);
    X509Certificate serverCert;
    PEM_Load(ss1, serverCert);

    StringSource ss2(cert0, true);
    X509Certificate caCert;
    PEM_Load(ss2, caCert);
    const SecByteBlock& signature = serverCert.GetCertificateSignature();
    const SecByteBlock& toBeSigned = serverCert.GetToBeSigned();
    const X509PublicKey& publicKey = caCert.GetSubjectPublicKey();

    RSASS<PKCS1v15, SHA256>::Verifier verifier(publicKey);
```

```
    bool result = verifier.VerifyMessage(toBeSigned, toBeSigned.size(), signature,
signature.size());

    if (result)
        std::cout << "Verified certificate" << std::endl;
    else
        std::cout << "Failed to verify certificate" << std::endl;

    cout<<"c0: \n"<<serverCert.GetSubjectIdentities();
    cout<<"\nc1: \n"<<caCert.GetSubjectIdentities();

    return 0;
}
```

Thử chạy chương trình ta được kết quả như sau: