4. Hàm băm và chứng chỉ số

A. NHIỆM VỤ PHÒNG THÍ NGHIỆM

1. Tạo bản tóm tắt tin nhắn (giá trị băm) và HMAC

Nhiệm vụ của bạn là viết một ứng dụng (bằng C/C++/C#) để tính toán các giá trị băm (ít nhất ba loại khác nhau: MD5, SHA-1/SHA-2, SHA-3). Đối với đầu vào, có thể là:

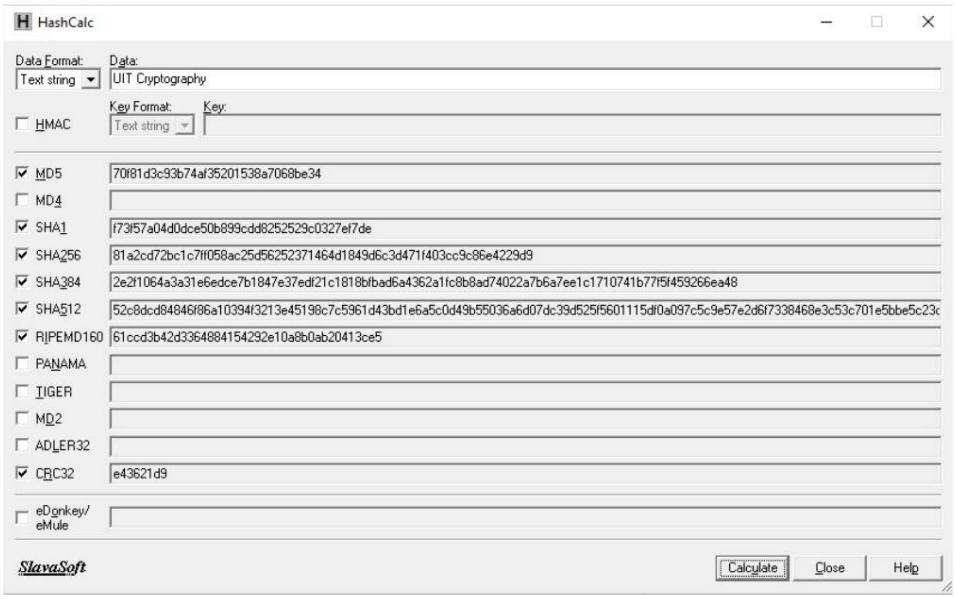
- § Chuỗi văn bản
- § Chuỗi lục giác
- § Tệp (hỗ trợ cả tệp văn bản và tệp nhị phân)

Bạn có thể sử dụng bất kỳ thư viện băm nào cho ngôn ngữ lập trình bạn chọn. Sau đó, hãy kiểm tra ứng dụng của bạn bằng bài tập sau:

1. Tạo giá trị băm của tin nhắn tùy ý có chứa ID sinh viên của bạn. Sau đó so sánh kết quả với các công cụ khác để xác minh.
2. Tạo ba tệp có kích thước lên đến 10 KB, 10 MB và 10 GB. Tạo băm giá trị của các tập tin này.

Mẹo: Bạn có thể tham khảo ứng dụng tương tự như HashCalc

(https://www.slavasoft.com/hashcalc/)



Hình 1: Ứng dụng HashCalc trên hệ điều hành Windows

2. Thuộc tính băm: Một chiều so với Không va chạm

Hiện nay, người ta đã biết rõ rằng hàm băm mật mã MD5 và SHA-1 đã bị phá vỡ rõ ràng (về mặt tính chất chống va chạm). Chúng ta sẽ tìm hiểu về va chạm MD5 và SHA-1 trong nhiệm vụ này bằng cách thực hiện các bài tập sau:

1. Xem xét hai thông điệp HEX như sau:

   Tin nhắn 1

   d131dd02c5e6eec4693d9a0698aff95c2fcab58712467eab4004583eb8fb7f89
   55ad340609f4b30283e488832571415a085125e8f7cdc99fd91dbdf280373c5b
   d8823e3156348f5bae6dacd436c919c6dd53e2b487da03fd02396306d248cda0
   e99f33420f577ee8ce54b67080a80d1ec69821bcb6a8839396f9652b6ff72a70

   Tin nhắn 2

   d131dd02c5e6eec4693d9a0698aff95c2fcab50712467eab4004583eb8fb7f89
   55ad340609f4b30283e4888325f1415a085125e8f7cdc99fd91dbd7280373c5b
   d8823e3156348f5bae6dacd436c919c6dd53e23487da03fd02396306d248cda0
   e99f33420f577ee8ce54b67080280d1ec69821bcb6a8839396f965ab6ff72a70

   Có bao nhiêu bit/byte khác nhau giữa hai thông điệp? Hãy tạo các giá trị băm MD5 cho mỗi thông điệp. Vui lòng quan sát xem MD5 này có giống nhau hay không và mô tả các quan sát của bạn trong báo cáo phòng thí nghiệm.

2. Tải xuống hai tệp PDF: broken-1 và broken-2.pdf:

   - broken-1.pdf: https://shattered.io/static/shattered-1.pdf
   - broken-2.pdf: https://shattered.io/static/shattered-2.pdf

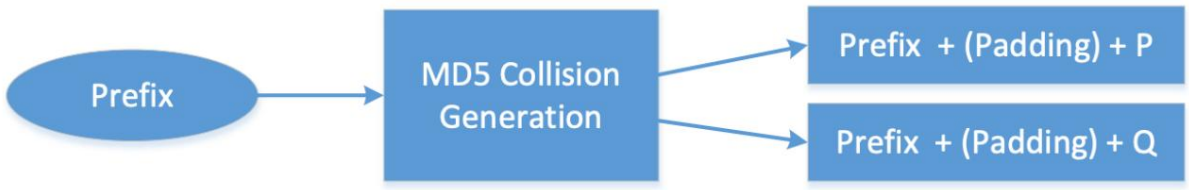
Mở các tệp này để kiểm tra sự khác biệt. Sau đó tạo hàm băm SHA-1 cho chúng và quan sát kết quả.

3. Rút ra kết luận dựa trên quan sát của bạn. Bạn có thể giải thích lý do tồn tại va chạm trong MD5 và SHA-1 không?

3. Tạo hai tệp khác nhau với cùng một mã băm MD5

Trong nhiệm vụ này, chúng ta sẽ tạo hai tệp khác nhau có cùng giá trị băm MD5. Phần đầu của hai tệp này cần phải giống nhau, tức là chúng chia sẻ cùng một tiền tố. Chúng ta có thể thực hiện điều này bằng chương trình md5collgen , cho phép chúng ta cung cấp tệp tiền tố với bất kỳ nội dung tùy ý nào. Cách thức hoạt động của chương trình được minh họa trong Hình 2. Lệnh sau tạo hai tệp đầu ra, out1.bin và out2.bin, cho một tiền tố nhất định là prefix.txt:

```
$ md5collgen -p prefix.txt -o out1.bin out2.bin
```



Hình 2: Tạo va chạm MD5 từ tiền tố

Chúng ta có thể kiểm tra xem các tập tin đầu ra có khác biệt hay không bằng cách sử dụng lệnh diff . Chúng ta cũng có thể sử dụng lệnh md5sum để kiểm tra mã băm MD5 của mỗi tệp đầu ra. Xem lệnh sau.

```
$ diff ra1.bin ra2.bin
$ md5sum ra1.bin
$ md5sum out2.bin
```

Vì out1.bin và out2.bin là nhị phân, chúng ta không thể xem chúng bằng chương trình xem văn bản, chẳng hạn như cat hoặc more; chúng ta cần sử dụng trình soạn thảo nhị phân để xem (và chỉnh sửa) chúng. Bạn có thể sử dụng trình soạn thảo hex có tên là bless.

Hãy thực hành và trả lời các câu hỏi dưới đây:

64 bytes

1. Nếu độ dài tệp tiền tố của bạn không phải là bội số của 64 thì điều gì sẽ xảy ra?
2. Tạo một tệp tiền tố có đúng 64 byte, chạy lại công cụ va chạm và xem điều gì xảy ra.
3. Dữ liệu (128 byte) do md5collgen tạo ra có hoàn toàn khác nhau đối với hai tệp đầu ra không? Vui lòng xác định tất cả các byte khác nhau.
4. (Nhiệm vụ nâng cao) Viết lại chương trình md5collgen bằng ngôn ngữ lập trình của riêng bạn ngôn ngữ.

4. Tạo hai tệp thực thi có cùng mã băm MD5

Trong nhiệm vụ này, bạn được cung cấp chương trình C sau. Nhiệm vụ của bạn là tạo ra hai phiên bản khác nhau của chương trình này, sao cho nội dung của mảng xyz của chúng khác nhau, nhưng giá trị băm của các tệp thực thi thì giống nhau:

```
#include <stdio.h>

ký tự không dấu xyz[200] = {
   /* Nội dung thực tế của mảng này tùy thuộc vào bạn */
};

int chính()
{
   số nguyên i;
   đối với (i=0; i<200; i++){
      printf("%x", xyz[i]);
   }
   inf("\n");
}
```

Bạn có thể chọn làm việc ở cấp độ mã nguồn, tức là tạo ra hai phiên bản của chương trình C ở trên, sao cho sau khi biên dịch, các tệp thực thi tương ứng của chúng có cùng giá trị băm MD5. Tuy nhiên, có thể dễ dàng hơn khi làm việc trực tiếp ở cấp độ nhị phân. Bạn có thể đặt một số giá trị tùy ý vào mảng xyz và biên dịch mã ở trên thành nhị phân. Sau đó, bạn có thể sử dụng công cụ biên tập hex để sửa đổi nội dung của mảng xyz trực tiếp trong tệp nhị phân.

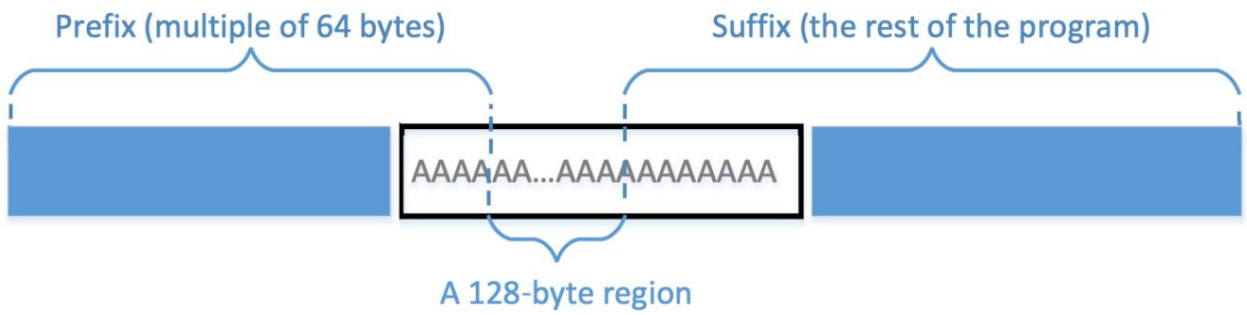
Việc tìm ra vị trí lưu trữ nội dung của mảng trong tệp nhị phân không phải là điều dễ dàng. Tuy nhiên, nếu chúng ta điền một số giá trị cố định vào mảng, chúng ta có thể dễ dàng tìm thấy chúng trong nhị phân. Ví dụ, đoạn mã sau điền mảng bằng 0x41, đây là giá trị ASCII của chữ cái A. Sẽ không khó để định vị 200 A trong nhị phân.

```
ký tự không dấu xyz[200] = {
   0x41, 0x41, 0x41,
   0x41, 0x41, 0x41,
    ... (bỏ qua)...
   0x41, 0x41, 0x41,

}
```

Hướng dẫn: Từ bên trong mảng, chúng ta có thể tìm thấy hai vị trí, từ đó chúng ta có thể chia tệp thực thi thành ba phần: tiền tố, vùng 128 byte và hậu tố.

Độ dài của tiền tố cần phải là bội số của 64 byte. Xem Hình 3 để biết minh họa về cách chia tệp.



Hình 3: Chia tệp thực thi thành ba phần

Chúng ta có thể chạy md5collgen trên tiền tố để tạo ra hai đầu ra có cùng giá trị băm MD5. Hãy sử dụng P và Q để biểu diễn phần thứ hai (mỗi phần có 128 byte) của các đầu ra này (tức là phần sau tiền tố). Do đó, chúng ta có như sau:

```
MD5 (tiền tố || P) = MD5 (tiền tố || Q)
```

Dựa trên đặc tính của MD5, chúng ta biết rằng nếu chúng ta thêm cùng một hậu tố vào hai đầu ra trên, dữ liệu kết quả cũng sẽ có cùng một giá trị băm. Về cơ bản, điều sau đây đúng với bất kỳ hậu tố nào:

```
MD5 (tiền tố || P || hậu tố) = MD5 (tiền tố || Q || hậu tố)
```

Do đó, chúng ta chỉ cần sử dụng P và Q để thay thế 128 byte của mảng (giữa hai điểm phân chia) và chúng ta sẽ có thể tạo ra hai chương trình nhị phân có cùng giá trị băm. Kết quả của chúng khác nhau, vì mỗi chương trình in ra các mảng riêng của chúng, có nội dung khác nhau.

Công cụ: Bạn có thể sử dụng bless để xem tệp thực thi nhị phân và tìm vị trí cho mảng. Để chia tệp nhị phân, có một số công cụ mà chúng ta có thể sử dụng để chia tệp từ một vị trí cụ thể. Các lệnh head và tail là những công cụ hữu ích như vậy.
Bạn có thể xem hướng dẫn sử dụng của họ để biết cách sử dụng chúng. Chúng tôi đưa ra ba ví dụ sau:

```
$ head -c 3200 a.out > tiền tố
$ tail -c 100 a.out > hậu tố
$ tail -c +3300 a.out > hậu tố
```

Lệnh đầu tiên ở trên lưu 3200 byte đầu tiên của a.out vào prefix. Lệnh thứ hai lưu 100 byte cuối cùng của a.out vào suffix. Lệnh thứ ba lưu dữ liệu từ byte thứ 3300 vào cuối tệp a.out vào suffix. Với hai lệnh này, chúng ta có thể chia tệp nhị phân thành các phần từ bất kỳ vị trí nào. Nếu chúng ta cần dán một số phần lại với nhau, chúng ta có thể sử dụng lệnh cat.

Nếu bạn sử dụng bless để sao chép và dán một khối dữ liệu từ tệp nhị phân này sang tệp khác, mục menu “Chỉnh sửa à Chọn phạm vi” khá tiện dụng, vì bạn có thể chọn một khối dữ liệu bằng cách sử dụng điểm bắt đầu và phạm vi, thay vì phải đếm thủ công số byte được chọn.

5. Xác minh thủ công chứng chỉ X.509

Trong nhiệm vụ này, chúng ta sẽ xác minh thủ công chứng chỉ X.509. X.509 chứa dữ liệu về khóa công khai và chữ ký của bên phát hành trên dữ liệu. Chúng ta sẽ tải xuống chứng chỉ X.509 thực từ máy chủ web và lấy khóa công khai của bên phát hành, sau đó sử dụng khóa công khai này để xác minh chữ ký trên chứng chỉ.

Bước 1: Tải xuống chứng chỉ từ máy chủ web thực

Chúng tôi sử dụng máy chủ www.example.org trong tài liệu này. Học viên nên chọn một máy chủ web khác có chứng chỉ khác với máy chủ được sử dụng trong tài liệu này.

tài liệu (cần lưu ý rằng www.example.com có thể chia sẻ cùng một

chứng chỉ với www.example.org). Chúng ta có thể tải xuống chứng chỉ bằng trình duyệt hoặc sử dụng lệnh sau:

```
$ openssl s_client -connect www.example.org:443 -showcerts

Chuỗi chứng chỉ
 0 giây:/C=Mỹ/ST=California/L=Los
Angeles/O=Internet\xC2\xA0Corporation\xC2\xA0for\xC2\xA0Assigned\xC2\xA0Names
\xC2\xA0và\xC2\xA0Số/CN=www.example.org
   i:/C=US/O=DigiCert Inc/CN=DigiCert TLS RSA SHA256 2020 CA1
-----BẮT ĐẦU CHỨNG NHẬN-----
MIIHRzCCBi+gAwIBAgIQD6pjEJMHvD1BSJJkDM1NmjANBgkqhkiG9w0BAQsFADBP
MQswCQYDVQQGEwJVUzEVMBMGA1UEChMMRGlnaUNlcnQgSW5jMSkwJwYDVQQDEyBE
.....
0XELBnGQ666tr7pfx9trHniitNEGI6dj87VD+laMUBd7HBtOEGsiDoRSlA==
-----KẾT THÚC CHỨNG NHẬN-----
 1 s:/C=US/O=DigiCert Inc/CN=DigiCert TLS RSA SHA256 2020 CA1
   i:/C=US/O=DigiCert Inc/OU=www.digicert.com/CN=DigiCert Global Root CA
-----BẮT ĐẦU CHỨNG NHẬN-----
MIIEvjCCA6agAwIBAgIQBtjZBNVYQ0b2ii+nVCJ+xDANBgkqhkiG9w0BAQsFADBh
MQswCQYDVQQGEwJVUzEVMBMGA1UEChMMRGlnaUNlcnQgSW5jMRkwFwYDVQQLExB3
.....
A7sKPPcw7+uvTPyLNhBzPvOk
-----KẾT THÚC CHỨNG NHẬN--
```

Kết quả của lệnh chứa hai chứng chỉ. Trường chủ đề (mục bắt đầu bằng s:) của chứng chỉ là www.example.org, tức là đây là chứng chỉ của www.example.org. Trường phát hành (mục bắt đầu bằng i:) cung cấp thông tin của đơn vị phát hành. Trường chủ đề của chứng chỉ thứ hai giống với trường phát hành của chứng chỉ thứ nhất. Về cơ bản, chứng chỉ thứ hai thuộc về một CA trung gian. Trong nhiệm vụ này, chúng ta sẽ sử dụng chứng chỉ của CA để xác minh chứng chỉ máy chủ.

Sao chép và dán từng chứng chỉ (văn bản giữa dòng chứa "Begin CERTIFICATE" và dòng chứa "END CERTIFICATE", bao gồm cả hai dòng này) vào một tệp. Chúng ta hãy gọi chứng chỉ đầu tiên là c0.pem và chứng chỉ thứ hai là c1.pem.

Bước 2: Trích xuất khóa công khai (e, n) từ chứng chỉ của bên phát hành

Openssl cung cấp các lệnh để trích xuất một số thuộc tính nhất định từ chứng chỉ x509.
Chúng ta có thể trích xuất giá trị của n bằng cách sử dụng -modulus. Không có lệnh cụ thể nào để trích xuất e, nhưng chúng ta có thể in ra tất cả các trường và có thể dễ dàng tìm ra giá trị của e.

```
Đối với môđun (n):
$ openssl x509 -in c1.pem -noout -modulus

In ra tất cả các trường, tìm số mũ (e):
$ openssl x509 -trong c1.pem -văn bản
```

Bước 3: Trích xuất chữ ký từ chứng chỉ của máy chủ

Không có lệnh openssl cụ thể nào để trích xuất trường chữ ký. Tuy nhiên, chúng ta có thể in ra tất cả các trường rồi sao chép và dán khối chữ ký vào một tệp (lưu ý: nếu thuật toán chữ ký được sử dụng trong chứng chỉ không dựa trên RSA, bạn có thể tìm một chứng chỉ khác)

```
$ openssl x509 -trong 1.pem -văn bản
...
Thuật toán chữ ký: sha256WithRSAEncryption
          aa:9f:be:5d:91:1b:ad:e4:4e:4e:cc:8f:07:64:44:35:b4:ad:
          3b:13:3f:c1:29:d8:b4:ab:f3:42:51:49:46:3b:d6:cf:1e:41:
          ......
          0 giờ:84:52:94
```

Chúng ta cần xóa khoảng trắng và dấu hai chấm khỏi dữ liệu để có thể có chuỗi hex mà chúng ta có thể đưa vào chương trình. Các lệnh sau có thể đạt được mục tiêu này.
Lệnh tr là một công cụ tiện ích Linux cho các hoạt động chuỗi. Trong trường hợp này, tùy chọn -d được sử dụng để xóa ":" và "khoảng trắng" khỏi dữ liệu.

```
$ chữ ký mèo | tr -d '[:space:]:'

84a89a11a7d8bd0b267e52247bb2559dea30895108876fa9ed10ea5b3e0bc7
......
5c045564ce9db365fdf68f5e99392115e271aa6a8882
```

Bước 4: Trích xuất phần thân chứng chỉ của máy chủ

Cơ quan cấp chứng chỉ (CA) tạo chữ ký cho chứng chỉ máy chủ bằng cách đầu tiên tính toán hàm băm của chứng chỉ, sau đó ký hàm băm. Để xác minh chữ ký, chúng ta cũng cần tạo hàm băm từ chứng chỉ. Vì hàm băm được tạo trước khi chữ ký được tính toán, chúng ta cần loại trừ khối chữ ký của chứng chỉ khi tính toán hàm băm. Việc tìm ra phần nào của chứng chỉ được sử dụng để tạo hàm băm khá khó khăn nếu không hiểu rõ về định dạng của chứng chỉ.

Chứng chỉ X.509 được mã hóa bằng chuẩn ASN.1 (Abstract Syntax Notation.One), vì vậy nếu chúng ta có thể phân tích cú pháp cấu trúc ASN.1, chúng ta có thể dễ dàng trích xuất bất kỳ trường nào từ chứng chỉ. Openssl có lệnh gọi là asn1parse dùng để trích xuất dữ liệu từ dữ liệu định dạng ASN.1 và có thể phân tích chứng chỉ X.509 của chúng ta.

```
$ openssl asn1parse -i -in c0.pem
    0:d=0  hl=4 l=1522 cons: SEQUENCE
    4:d=1  hl=4 l=1242 cons:  SEQUENCE          ❶
    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
   :0E64C5FBC236ADE14B172AEB41C78CB0
   ... ...
 1236:d=4  hl=2 l=  12 cons:     SEQUENCE
 1238:d=5  hl=2 l=   3 prim:      OBJECT            :X509v3 Basic Constraints
 1243:d=5  hl=2 l=   1 prim:      BOOLEAN           :255
 1246:d=5  hl=2 l=   2 prim:      OCTET STRING      [HEX DUMP]:3000
 1250:d=1  hl=2 l=  13 cons:  SEQUENCE          ❷
 1252:d=2  hl=2 l=   9 prim:   OBJECT            :sha256WithRSAEncryption
 1263:d=2  hl=2 l=   0 prim:   NULL
 1265:d=1  hl=4 l= 257 prim:  BIT STRING
```

Trường bắt đầu từ là phần thân của chứng chỉ được sử dụng để tạo băm; trường bắt đầu từ là khối chữ ký. Các offset của chúng là các số ở đầu các dòng. Trong trường hợp của chúng ta, phần thân chứng chỉ là từ offset 4 đến 1249, trong khi khối chữ ký là từ 1250 đến cuối tệp. Đối với chứng chỉ X.509, offset bắt đầu luôn giống nhau (tức là 4), nhưng phần cuối phụ thuộc vào độ dài nội dung của chứng chỉ. Chúng ta có thể sử dụng tùy chọn -strparse để lấy trường từ offset 4, điều này sẽ cung cấp cho chúng ta phần thân của chứng chỉ, không bao gồm khối chữ ký.

```
$ openssl asn1parse -i -in c0.pem -strparse 4 -out c0_body.bin -noout
```

Khi đã có được phần thân của chứng chỉ, chúng ta có thể tính toán hàm băm của nó bằng lệnh sau:

```
$ sha256sum c0_body.bin
```

Bước 5: Xác minh chữ ký

Bây giờ chúng ta có tất cả thông tin, bao gồm khóa công khai của CA, chữ ký của CA và nội dung chứng chỉ của máy chủ. Chúng ta có thể chạy chương trình của riêng mình để xác minh xem chữ ký có hợp lệ hay không. Openssl cung cấp lệnh để xác minh chứng chỉ cho chúng ta, nhưng sinh viên phải sử dụng chương trình của riêng mình để thực hiện.

6. Lập trình ứng dụng xác minh chứng chỉ X509

> Viết một ứng dụng (sử dụng C++ với thư viện cryptopp) để lấy tất cả
> thông tin và xác minh chứng chỉ x509.
>
> Đầu vào: Đường dẫn đến tất cả các tệp chứng chỉ trong chuỗi từ máy tính của bạn.
> Đầu ra: Mỗi chứng chỉ trong chuỗi hiển thị thông tin (ví dụ: phiên bản, danh tính, ngày hiệu lực, v.v.) và kết quả đã xác minh.

B. YÊU CẦU

Bạn được yêu cầu hoàn thành tất cả các nhiệm vụ trong phần B (Nhiệm vụ trong phòng thí nghiệm). Nhiệm vụ nâng cao là tùy chọn và bạn có thể nhận được điểm thưởng khi hoàn thành các nhiệm vụ đó. Chúng tôi muốn bạn làm việc theo nhóm gồm 2 thành viên để đạt hiệu quả cao nhất.

Bài nộp của bạn phải đáp ứng các yêu cầu sau:

- § Bạn cần nộp báo cáo thực hành chi tiết ở định dạng .docx (Word Document) , sử dụng mẫu báo cáo được cung cấp trên trang web Khóa học UIT.
- § Báo cáo bằng tiếng Việt hoặc tiếng Anh đều được chấp nhận. Tùy bạn. Không được phép sử dụng nhiều hơn một ngôn ngữ trong báo cáo (trừ từ khóa không thể dịch được).
- § Khi nói đến các nhiệm vụ lập trình (yêu cầu bạn viết một ứng dụng hoặc tập lệnh), vui lòng đính kèm tất cả mã nguồn và tệp thực thi (nếu có) trong bài nộp của bạn. Vui lòng cũng liệt kê các đoạn mã quan trọng theo sau là giải thích và ảnh chụp màn hình khi chạy ứng dụng của bạn. Chỉ đính kèm mã mà không có bất kỳ giải thích nào sẽ không nhận được điểm.
- § Nộp bài làm mà bạn tự hào – đừng cẩu thả và lười biếng!

  Bài nộp của bạn phải là của riêng bạn. Bạn được tự do thảo luận với các bạn cùng lớp khác để tìm ra giải pháp. Tuy nhiên, việc sao chép báo cáo là bị cấm, ngay cả khi chỉ là một phần báo cáo của bạn. Cả báo cáo của chủ sở hữu và người sao chép đều sẽ bị từ chối. Vui lòng nhớ trích dẫn bất kỳ nguồn tài liệu nào (trang web, sách,.) có ảnh hưởng đến giải pháp của bạn.

> Lưu ý: Gộp báo cáo phòng thí nghiệm và tất cả các tệp liên quan vào một tệp ZIP duy nhất (.zip), đặt tên như sau:
>
> StudentID1_StudentID2_ReportLabX.zip

C. TÀI LIỆU THAM KHẢO

[1]MD5CollisionAttackLab, WenliangDu (Đại học Syracuse), SEEDCryptographyLabs https://seedsecuritylabs.org/Labs_20.04/Crypto/Crypto_MD5_Collision/

[2]RSAEncryptionandSignature, WenliangDu (Đại học Syracuse), SEEDCryptographyLabs: https://seedsecuritylabs.org/Labs_20.04/Crypto/Crypto_RSA/

[3]X509Chứng chỉ
https://www.cryptopp.com/wiki/X509Certificate

Lưu ý: Không chia sẻ bất kỳ tài liệu nào (slide, bài đọc, bài tập, phòng thí nghiệm, v.v.) ra khỏi lớp học mà không có sự cho phép của tôi!