BÁO CÁO THỰC HÀNH

**Bài thực hành số 2: Classical cryptography**

**Môn học:** An toàn mạng máy tính

**Lớp:** NT101.N12.MMCL.1

**THÀNH VIÊN THỰC HIỆN (Nhóm TH1.02):**

|  |  |  |
| --- | --- | --- |
| **STT** | **Họ và tên** | **MSSV** |
| 1 | Hoàng Văn Anh Đức | 20520890 |
| 2 | Nguyễn Mạnh Cường | 20520421 |
| 3 | Nguyễn Trần Đức Anh | 20520392 |
| 4 | Lê Quang Minh | 20520245 |

|  |
| --- |
| **Điểm tự đánh giá** |
| **10** |

**ĐÁNH GIÁ KHÁC:**

|  |  |
| --- | --- |
| Tổng thời gian thực hiện |  |
| Phân chia công việc |  |
| Ý kiến *(nếu có)*  + Khó khăn  + Đề xuất, kiến nghị |  |

MỤC LỤC

[**A.** BÁO CÁO CHI TIẾT 2](#_Toc115332408)

[1. Nội dung 1 2](#_Toc115332409)

[a. Nội dung a 2](#_Toc115332410)

[b. Hé lô 2](#_Toc115332411)

[c. Nội dung b 2](#_Toc115332412)

[2. Nội dung 2 2](#_Toc115332413)

[a. Nội dung a 2](#_Toc115332414)

[**B.** TÀI LIỆU THAM KHẢO 2](#_Toc115332415)

# BÁO CÁO CHI TIẾT

## Nội dung 1 : Lý thuyết số (number theory)

### Nội dung : *Generate random large numbers*

***2 bytes***

Text

Description automatically generated

Hàm **ReturnPrimeNumber\_2bytes()** sẽ tạo số ngẫu nhiên trongkhoảng từ 2->99 và bỏ số đó vào hàm kiểm tra số nguyên tố.

***8 bytes***

Với việc tạo ra 1 số nguyên tố ngẫu nhiên thì cũng tương tự như 2 bytes

***32 bytes***

Tại vì 32 bytes là quá lớn so với kiểu **long long int** (giá trị lớn nhất của kiểu **int**) ,ta sẽ không thể làm đc theo cách trên .

Text

Description automatically generated

Hàm này sẽ tạo ra 1 mảng 32 số nguyên , mỗi số sẽ được tạo ra ngẫu nhiên trong khoảng từ 0->9 . Ngoại trừ số đầu tiên là : 1->9

* Ta sẽ tạo được 1 số nguyên 32 bytes

### Nội dung : Xác định ước chung lớn nhất (GCD) của 2 lớn tùy ý

Text

Description automatically generated

ll là : long long int

A screenshot of a computer

Description automatically generated with medium confidence

// Ở đây 'a' là số nguyên và 'b' là chuỗi.

// Ý tưởng là tạo ra số thứ hai nhỏ hơn hoặc bằng số đầu tiên bằng cách tính toán mod của nó với số nguyên đầu tiên .Bằng cách này thì b sẽ <= a . Từ đó ta có thể dễ dàng tính đc gcd của a và b

Text

Description automatically generated

// Hàm này trả về GCD của 'a' và 'b'

// trong đó b có thể rất lớn và được biểu diễn dưới dạng một mảng hoặc chuỗi ký tự

A screenshot of a computer

Description automatically generated with medium confidenceA screen shot of a computer

Description automatically generated with low confidence

### Nội dung : Tính a^b mod p ( b > 80 )

Tính (a ^ b) % p

*//a, b > 0, p phải là số nguyên tố*

Text

Description automatically generated

Đặt x = a % m.

Ta có a^b ≡ x^b (mod m)

Bước này giảm a từ 10^18 xuống x còn cỡ ~10^9

Định lý Fermat nhỏ cho biết: a^p ≡ a (mod p) với a bất kì và p là số nguyên tố, hay nói cách khác a^(p-1) ≡ 1 (mod p),

hay a^( q(p-1) )≡ 1^q ≡ 1 (mod p),

hay a^( q(p-1) + r ) = a^(q(p-1)).a^r ≡ 1 \* a^r (mod p) = ar (mod p).

Đặt r = b % (m - 1). Ta có x^b ≡ x^r (mod m) với m là số nguyên tố.

Bước này giảm b từ 10^18 xuống r còn cỡ ~10^9

Để tính ab lẹ thì ta xài cách tính a^(b/2 )\* a^(b/2), ở đây có mod m vẫn xài được, vì ab mod m = a mod m \* b mod m.

Để tránh đệ quy thì ta phân tích b thành tổng của các số mũ 2

. Ví dụ b = 12345 thì b = 1 + 8 + 16 + 32 + 4096 + 8192, vậy a^(12345) = a^(1 + 8 + 16 + 32 + 4096 + 8192) = a^1.a^8.a^16.a^32.a^4096.a^8192.

Để phân tích b ra thành tổng của các số mũ 2 thì ta chuyển b về nhị phân, từ đó ta biết được ở vị trí i nào (từ phải qua trái, bắt đầu từ 0) có số 1 thì tổng b có 1 số hạng 2^i.

Ví dụ ở đây b = (12345)10 = (11000000111001)2 có số 1 ở vị trí thứ 0, 3, 4, 5, 12, 13 từ phải qua thì tổng b = 2^0 + 2^3 + 2^4 + 2^5 + 2^12 + 2^13 = 1 + 8 + 16 + 32 + 4096 + 8192.

### \*Chú thích trong ảnh ….

### Nội dung : Primary number

### Kiểm tra số nguyên bất kì (>2^89) là số nguyên tố hay không

Text

Description automatically generated

Text

Description automatically generated

* ***Dựa vào lý thuyết xây dựng thuật toán để kiểm tra số nguyên tố***

### Nội dung : *Generate random large prime numbers.*

Text

Description automatically generated

Dùng vòng lặp random ra 1 số lớn rồi kiểm tra 10 lần bằng miller test .

### Nội dung :

+Đánh giá ứng dụng : (Core intel i2 8th gen)

1. Generate a random large number : 0.002s
2. Determine the Greatest Common Divisor (GCD) of 2 arbitrary large numbers : 0.2721s
3. Compute the modular exponentiation ax mod p. Your program should be able to compute in case of “large” exponents (x > 80), for example, 794 mod19 : 0.00003s
4. Check if an arbitrary integer larger than 289 is prime or not : 0.008697s
5. Generate random large prime numbers : 0.001379s

## Nội dung : RSA Application

### Viết một ứng dụng mật mã để mô phỏng hoạt động của Mật mã RSA đơn giản. Các ngôn ngữ lập trình được phép là C, C ++ và C #. Ứng dụng của bạn cần chỉ ra cách đạt được mục tiêu của các yêu cầu sau một cách chi tiết:

* ***Tạo một cặp khóa (PU, PR) từ các đầu vào “hợp lệ” đã cho (p, q, e) hoặc tạo ngẫu nhiên.***

Text

Description automatically generated

Chương trình cho người dùng lựa chọn

**Trường hợp chọn 1** : Thì chương trình sẽ giải mã và mã hóa bằng các input mà người dùng nhập vào

Text

Description automatically generated

Tại lựa chọn này chúng ta sẽ cho người dùng nhập vào q,e,p và chương trình sẽ tự động tính các biến cần dùng để **encrypt** và **decrypt** **( phi n , n , d )**

Text

Description automatically generated

Ở đây chúng ta sẽ kiểm tra e nhập vào có đúng dạng để chương trình có thể hoạt động được **gcd (phi(n), e) = 1 and 1 < e < phi(n)**

* Xuất ra các biến được tính toán để chương trình có thể dùng

Text

Description automatically generated

Tiếp theo người dùng sẽ nhập **message** để mã hóa và giải mã : Có 2 lựa chọn là **message** dạng **string** và **dạng số**

***Ở dạng số*** thì **Ciphertxt** sẽ được tính bằng công thức **( m^e ) mod n**

Và **Plaintext sẽ = ( Ciphertxt ^ d ) mod n**

**n ,e** sẽ là **publickey và d** sẽ là **private key**

Text

Description automatically generated

Text

Description automatically generated

* ***Sử dụng các khóa đã tạo để mã hóa / giải mã tin nhắn. Tin nhắn có thể là số hoặc chuỗi.***

Khi người dùng chọn message nhập vào dạng string thì :

Text

Description automatically generated

Chúng ta sẽ tạo ra 2 mảng kí tự có số lượng kí tự = với độ dài của string mà người dùng nhập vào

**cipher** để lưu chuỗi msg sau khi được mã hóa

**plain** để lưu chuỗi msg sau khi được giải mã

**Cùng với công thức cipher =** (m^e) mod n

**Plain =** (c^d) mod n

Khác nhau ở chỗ khi ta xử lý 1 string thì ta phải đổi chúng dưới dạng decimal và xử lí từng kí tự

Sau khi mã hóa thì từng kí tự đc mã được lưu vào **cipher**

Giải mã thì từng kí tự đc mã được lưu vào **plain**

Text

Description automatically generated

Việc còn lại chỉ là xuất ra màn hình

Text

Description automatically generated

**Đối với lựa chọn người dùng không cần nhập vào từ bàn phím thì ta sẽ có hàm để tạo ra các biến cần thiết để chương trình có thể mã hóa nội dung mà người dùng nhập vào dưới 2 dạng ( số và string )**

Text

Description automatically generated

Hàm này sẽ tạo ra q,e,p bằng cách random các số từ 1 -> 99 và kiểm tra nó có phải là số nguyên tố không

Text

Description automatically generated

Hàm kiểm tra số nguyên tố sau khi số ngẫu nhiên dược tạo ra

* P,q,e sẽ được tạo ra là 1 số nguyên tố
* Và sau khi có đủ dữ kiện để mã hóa và giải mã thì các bước làm hoàn toàn giống với các bước mà người dùng nhập vào

Text

Description automatically generated

Text

Description automatically generated

* Text

  Description automatically generated

## Nội dung : Ứng dụng của mật mã đối xứng

### Trong nhiệm vụ này, đề xuất giải pháp và phát triển ứng dụng cho một trong những các chủ đề bên dưới. Không bị giới hạn trong việc sử dụng các ngôn ngữ lập trình, thư viện, hoặc các khuôn khổ.

### Trò chuyện an toàn trong môi trường mạng không an toàn:

### Tính năng: Thiết lập hệ thống trò chuyện để người dùng có thể gửi hoặc nhận tin nhắn an toàn với nhau. Ứng dụng sẽ hoạt động như một mô hình Máy khách-Máy chủ.

### Bạn phải đảm bảo rằng khi người dùng trò chuyện, không ai khác có thể nhận được tin nhắn của họ ở dạng bản rõ.

### Hệ thống của bạn phải hỗ trợ nhiều người dùng trò chuyện đồng thời. Các tính năng

### Nâng cao: Các tính năng nâng cao: Tích hợp tính năng End-to-End đơn giản tính năng mã hóa để tăng cường quyền riêng tư. Chỉ có hai thiết bị đầu cuối có thể giải mã thông điệp.

* ***Bước 1:*** Thiết kế giao diện của **Client** và **Server**
* Graphical user interface, application, Word

  Description automatically generatedGraphical user interface, application, Word

  Description automatically generated
* ***Bước 2:*** Tạo **Client**

+ Giao diện cần có:

* 1 listView để hiện tin nhắn của **Server** và các **client** khác
* 2 Label có tên “Your name” và “Message”
* 2 textbox ứng với 2 label : “txbName” để xác định tên của người gửi , “txbMessage” để nhập nội dung cần gửi
* 1 button có tên “btnSend” để nhấn gửi đến Server và tất cả các client
* Thư viện cần dùng:

+ using System.Net.Sockets;

+ using System.Threading.Tasks;

+ using System.Runtime.Serialization.Formatters.Binary;

+ using System.IO;

Text

Description automatically generated

* Khi chạy chương trình của **Client** thì **Client** sẽ cố gắng kết nối với **Server** :

Text

Description automatically generated

* Tạo 1 **Socket client** có ip : 127.0.0.1 và port :8080 để kết nối tới server
* Kết nối bằng **client.Connect()**
* Trong hàm **Connect()** còn có cả **Thread(Receive)** để tạo luồng lắng nghe khi vừa kết nối tới **Server .**

Text

Description automatically generated

Hàm **Connect()**

* Hàm **Receive()** để nhận các thông tin từ **Server** hoặc các **Client** khác, khi các **Server** hoặc **Client** khác gửi tin thì sẽ có hàm **Serialize()** để phân mảnh dữ liệu từ 1 mảng byte sang 1 object và hàm **Deserialize()** sẽ gom mảnh lại thành 1 chuỗi byte trước khi **Client** nhận được

Text

Description automatically generated

Hàm **Receive()**

* Phân mảnh 1 object thành 1 chuỗi byte để có thể gửi cho **Server** và các **Client** khác

A screenshot of a computer

Description automatically generated with medium confidence

* Hàm **Deserialize()** sẽ thực hiện gom mảnh các object đã đc phân mảnh lại thành 1 chuỗi byte để **Server** và các **Client** khác có thể nhận được

Text

Description automatically generated

* Hàm **AddMessage()** để gửi thông tin từ các Server hoặc client khác lên trên **lsvMessage**

Text

Description automatically generated

* Ta có sự kiện **btnSend()** để gửi dữ liệu đến các **Client** và **Server** khác

Text

Description automatically generated

* Hàm **Send()** sẽ kiểm tra xem “**txbMessage”** , nếu không rỗng thì khi nhấp btnSend thì dữ liệu sẽ được gửi đến các **Client** và **Server** khác

Text

Description automatically generated

***Bước 3***:Tạo **Server**

* Khi chạy **Server** thì **Server** cũng cố gắng kết nối tới các **Client**

Text

Description automatically generated

* Ở phần **Server** chúng ta phải tạo 1 List các **Client :**Sau khi connect đến **Client** thì sẽ sẽ bắt đầu lắng nghe tất cả các **Client .** Nếu “lắng nghe” thành công thì sẽ thêm Client vào **ClientList .**Trong khi đó thì ta cũng sẽ tạo luồng lắng nghe xem các **Client** có gửi gì đến không .Ta dùng Try catch để khi kết nối đến **client** mà có 1 **client** disconnect thì **server** sẽ chạy vòng lặp while liên tục để chương trình ko bị crash

Text

Description automatically generated

Text

Description automatically generated

* Ta dùng try catch để Remove các thành client phát sinh lỗi khi **Server** đang lắng nghe thông tin từ các **Client** .Đầu tiên ta khai báo 1 **Socket** để nhận dữ liệu => Khởi tạo mảng byte để **Socket** đó nhận dữ liệu => Chuyển dữ liệu đó sang dạng string
* Ta dùng Forech để gửi **Message** nhận được này đến các **Client** và hiển thị message nhận được lên trên **lsvMessage**
* Hàm **Serialize()** và **Deserialize()** của **Server** tương tự như của **Client**

Text

Description automatically generated

* Sự kiện **btnSend** để gửi thông tin cho các **Client** trong **ClientList .** Duyệt từng phần tử trong **ClientList** và gửi thông tin nhập vào trong **“txbMessage”** của **Server** và gửi nó đến các **Client**

Text

Description automatically generated

* Trong hàm **Send()** của Server thì tham số truyền vào sẽ là **Socket** tên **Client** .Kiểm tra nếu Socket đó khác null và **txbMessage** khác rỗng thì phân mảnh và gửi đến các **Client**

Text

Description automatically generated

**+Phần mã hóa**

Graphical user interface, application

Description automatically generated

**Nút ENCRYPT**

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generatedText

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

**NÚT\_DECRYPT**Text

Description automatically generatedText

Description automatically generatedText

Description automatically generated

Text

Description automatically generated

***DEMO:***

* Khi chạy **Server** thì **Server** sẽ lắng nghe xem có **Client** nào kết nối tới nó không .Khi ta chạy 2 **client** thì bên **Server** có thông báo kết nối thành công

Graphical user interface, application

Description automatically generated

* Khi **Client** đầu tiên gửi tin nhắn thì cả **Server** và các **client** khác cũng nhận được thông qua việc **Client** này gửi đến **Server** và **Server** sẽ gửi tới các **Client** khác (**Client** còn lại).Khi gửi thì message sẽ được phân mảnh và tới đích sẽ được gom mảnh lại .

Graphical user interface, application

Description automatically generated

* Khi Client còn lại gửi thì cả Server và Client đầu tiên cũng nhận đc

Graphical user interface, application

Description automatically generated

Khi **Server** gửi thì các **Client** cũng sẽ nhận được tin nhắn

Graphical user interface, application

Description automatically generated

+Khi mã hóa :

Mã hóa trước khi gửi

Graphical user interface

Description automatically generated

Người nhận sẽ giải mã bằng khóa đã trao đổi từ trước

A picture containing table

Description automatically generated