**VIETNAM GENERAL CONFEDERATION OF LABOR**

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**

Logo

Description automatically generated

**FINAL REPORT**

**DISCRETE STRUCTURE**

*Instructor:* MAI DUY TÂN

*Student:* NGUYỄN HOÀNG ANH TÚ

*Student ID:*521H0177

*Class:*21H50203

**HO CHI MINH CITY, 2022**

**VIETNAM GENERAL CONFEDERATION OF LABOR**

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**

Logo

Description automatically generated

**FINAL REPORT**

**DISCRETE STRUCTURE**

*Instructor:* MAI DUY TÂN

*Student:* NGUYỄN HOÀNG ANH TÚ

*Student ID:*521H0177

*Class:*21H50203

**HO CHI MINH CITY, 2022**

# Acknowledgement

First of all, I would like to express my sincere and deep gratitude to the subject lecturer - Mr. Mai Duy Tan for enthusiastically guiding the important, valuable lessons and experiences during the past practice period and for this essay. Because there are many limitations in knowledge as well as not having equipped myself with much experience to do the essay. I personally hope to receive your comments, suggestions and criticisms to improve these reports.

Finally, I would like to wish you good health and happiness.

**TABLE OF CONTENTS**

[Acknowledgement 1](#_Toc132318794)

[Part 1 : Finding an inverse modulo n 3](#_Toc132318795)

[Introduction about the basic and extended Euclidean algorithm : 3](#_Toc132318796)

[Introduction about inverse and inverse modulo : 3](#_Toc132318797)

[a) Review about an inverse and its examples ? 3](#_Toc132318798)

[b) What is an inverse modulo ? 4](#_Toc132318799)

[Method finding an inverse modulo using the extended Euclidean algorithm : 4](#_Toc132318800)

[Python implementation : 5](#_Toc132318801)

[Part 2 : RSA cryptosystem 6](#_Toc132318802)

[Prime Number Generation : 6](#_Toc132318803)

[Modular Arithmetic : 6](#_Toc132318804)

[Extended Euclidean Algorithm : 7](#_Toc132318805)

[Prime Factorization : 7](#_Toc132318806)

[Encryption and decryption: 7](#_Toc132318807)

[RSA algorithm step by step procedure: 7](#_Toc132318808)

[Implement a Python program to encrypt and decrypt a message : 8](#_Toc132318809)

[The efficiency and security of RSA cryptosystem : 10](#_Toc132318810)

[The considered security threats and limitations of the RSA cryptosystem : 10](#_Toc132318811)

[Enhance the security of the RSA cryptosystem implementation : 10](#_Toc132318812)

[SELF-EVALUATION FORM 11](#_Toc132318813)

# Part 1 : Finding an inverse modulo n

## Introduction about the basic and extended Euclidean algorithm :

Basic Euclidean algorithm is an effective method that allows the user to quickly get the GCD which is the greatest common divisor of two positive integers provided. It can be observed that a simple approach to find GCD is to factorize both numbers and multiply common prime factors together.

For example:

84 = 2 x 2 x 3 x 7

56 = 2 x 2 x 2 x 7

GCD = 2 x 2 x 7 = 28

Likewise, the other algorithm called the extended Euclidean algorithm not only has the same function but also could finding the integer coefficients that express the GCD as a linear combination of the two integers x and y such that: ax + by = gcd(a, b) .

For instance:

Give a = 84, b = 56

Then we have the output after using the extended algolrithm :

gcd = 28, x = 1, y = -1

which can be expressed as : 84(1) + 56(-1) = 28

## Introduction about inverse and inverse modulo :

### **a) Review about an inverse and its examples ?**

Recall that a number multiplied by its inverse equals one.

By this, it has some following statements :

* All real numbers other than number zero have their inverse.
* The inverse of a number A is 1/A since A x 1/A = 1

Example :

The inverse of 3 is 1/3 since 3 x 1/3 = 1

The inverse of 2/3 is 3/2 since 2/3 x 3/2 = 1

### **b) What is an** **in****verse modulo ?**

To begin, there is no division operation in modular arithmetic, however, it has modular inverses. For more information, A^ -1 is the inverse modulo of A (mod C) and the statement (A \* A^ -1) ≡ 1 (mod C) is equivelently to (A \* A^-1) mod C = 1.

This means that if the gcd(A, C) is not equal to 1, A does not have its inverse modulo . In addition, only numbers that share no prime factors with C have a modular inverse (mod C).

To illustrate, let take C = 15 as an example, we know that 15 is the product of 3 and 5, which are prime factors. Like the above definition, numbers that share no prime factors with 15 are those not divisible by 3 or 5 .

## Method finding an inverse modulo using the extended Euclidean algorithm :

Step 1: Use the Euclidean algorithm to find gcd(a, b).

Step 2: Write down the equation ax + by = gcd(a, b) and solve for x and y.

Step 3: If gcd(a, b) = 1, then x is the inverse of a modulo b, if x is negative, add b to it until you get a positive value.

Let choose number 2 to check if it has a inverse modulo (mod 15) or not and then caculate the inverse modulo if it exists.

Firstly, we need to calculate its gcd with 15 using the extended Euclidean algorithm :

gcd(2,15) = gcd (15,2)

15 = 7 x 2 + 1

2 = 2 x 1 + 0

=> gcd (15,2) = 1, so that 2 has a inverse modulo ( mod 15 )

Then, the following step is solving for x and y in the equation 2x + 15y = 1 :

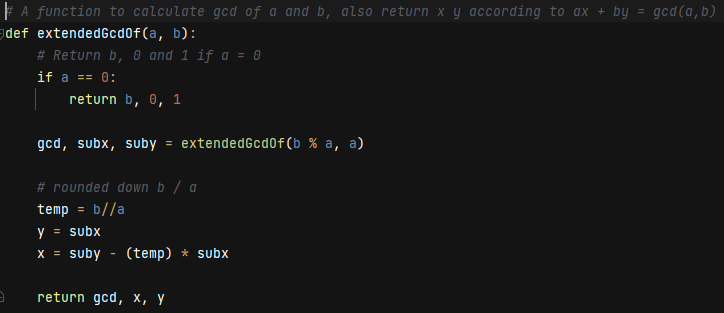
1 = 2 x (- 7) + 15

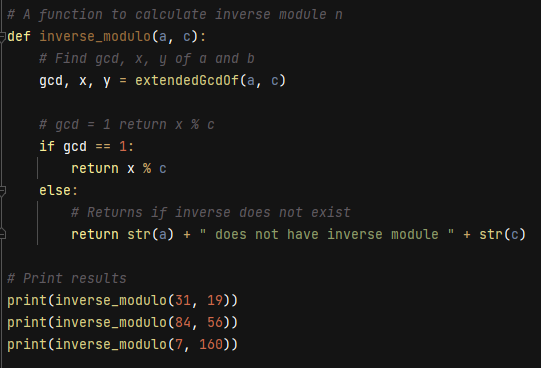
So x = -7 , add 15 to get new x as a positive integer

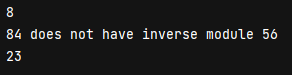
x = -7 + 15 = 8

Therefore, the inverse of 2 modulo 15 is 8 : 2 x 8 ≡ 16 ≡ 1 (mod 15)

## Python implementation :



  
Result :



# Part 2 : RSA cryptosystem

RSA cryptosystem is a widely used public-key encryption algorithm that was invented by Ron Rivest, Adi Shamir and Leonard Adleman in 1977.

In addition, public-key encryption algorithm, which is also called the asymmetric algorithm, are those algorithms in which sender and receiver use different keys for encryption and decryption. A pair of keys is given to each sender, consisting of a public key utilized for encrypting data and a private key utilized for decrypting it.

The two keys are linked, but it is impossible to derive the private key from the public key. While the public key is accessible to anyone, the private key is confidential and known only to its owner. It means that everybody can send a message to the user using user's public key. But however only the owner can decrypt the message using their private key.

For more information, this crytosystem is based on the mathematical concepts of prime number generation, modular arithmetic, extended Euclidean algorithm, and prime factorization.

## Prime Number Generation :

There is a truth that cryptosystem is computationally hard to factorize a large composite number. Therefore, the intitial step in the RSA cryptosystem is to generate two large prime numbers, p and q. These primes must be kept secret, and only their product, n = p \* q, is made public.

Example : Let p = 31 and q = 19, the product n = p \* q= 589 is made public.

## Modular Arithmetic :

This cryptosystem uses modular arithmetic to encrypt and decrypt messages. In modular arithmetic, we perform arithmetic operations (addition, subtraction, multiplication, and division) on remainders obtained after dividing by a fixed positive integer called the modulus.

Example : 5 mod 3 = 2, 31 mod 19 = 12

## Extended Euclidean Algorithm :

The extended Euclidean algorithm is used to find the modular multiplicative inverse of a number modulo n. The modular multiplicative inverse of a number a modulo n is another number x such that a \* x mod n = 1.

For instance, the modular multiplicative inverse of 5 modulo 7 is 7 because 5 \* 7 mod 7 = 1

## Prime Factorization :

The security strength of RSA cryptosystem is related to the difficulty of the factoring a large composite number into its prime factors. In the RSA encryption system, the private key consists of two integers, d and n, where d is the modular multiplicative inverse of e modulo (p - 1) \* (q - 1), and n is the product of two prime numbers, p and q.

## Encryption and decryption:

To encrypt a message m using the public key (e, n) into ciphertext, the system compute c = m^e mod n. While to decrypt a ciphertext c using the private key (d, n) into message, it compute m = c^d mod n.

## RSA algorithm step by step procedure:

The RSA algorithm follows a set of steps to create public and private keys.

First, two prime numbers p and q are selected and multiplied to obtain the modulus, n.

Then, a number e is chosen such that it is less than n and relatively prime to

( p – 1 ) \* ( q – 1 ).

The public key is < e, n >, and plaintext messages are encrypted using this key with the formula c = me mod n. For larger messages, they are broken down into smaller messages and encrypted separately.

To find the private key, a formula is used to calculate d such that de mod φ (n) = 1. The private key is < d, n >, and ciphertext messages are decrypted using this key with the formula m = cd mod n.

Example :

Let choose p = 17 and q = 11 for simple instance.

n = p \* q = 17 \* 11 = 187

φ (n) = ( p – 1 ) \* ( q – 1 )

φ (n) = ( 17 – 1 ) \* ( 11 – 1 )

φ (n) = 160

Choose e that 1 < e < φ (n) and it should be relatively prime φ (n). 7 and 160 are relatively prime since they have no common factors other than 1. So let e = 7.

Therefore the public key is < e, n > = (7, 187)

Imagine there is a plaintext message “2” that sender A need to send it to receiver B

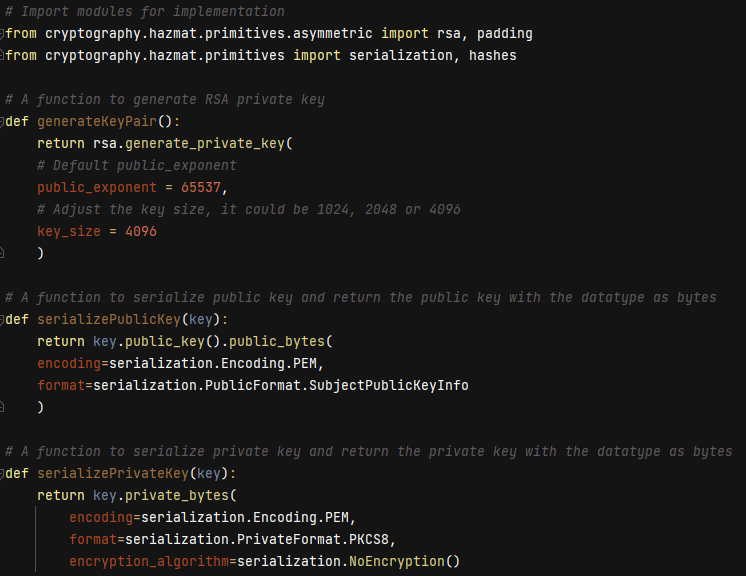
Encrypt the plaintext message using public key : c = me mod n = 27 mod 187 = 128

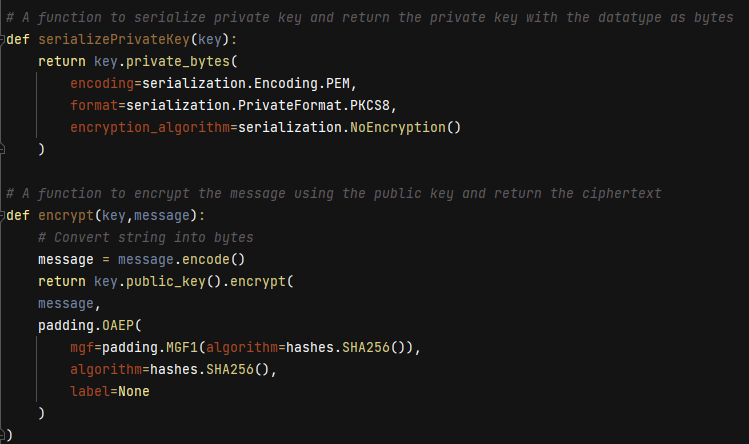
de mod φ (n) = e.d mod φ (n) = 1 = 7d mod 160 = 1, which gives d = 23

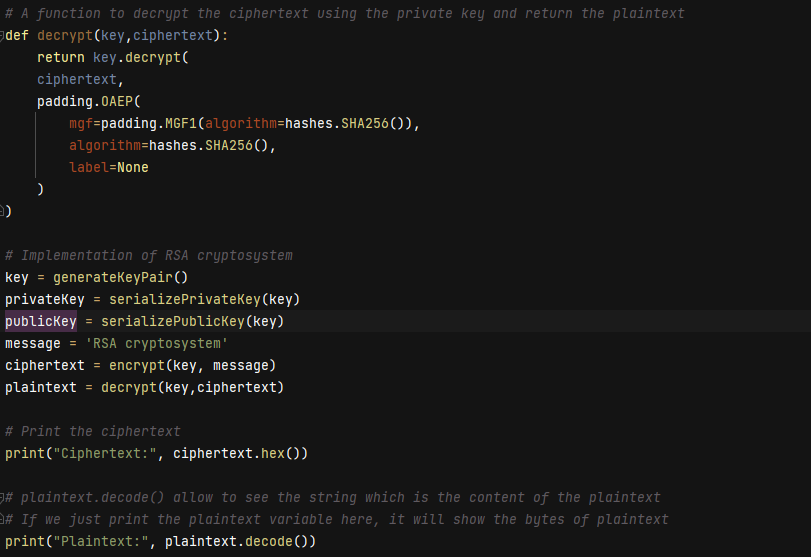
Therefore the private key is < d, n > = (23, 77)

Decrypt the ciphertext message c using private key : m = cd mod n = 12823 mod 187 = 2

## Implement a Python program to encrypt and decrypt a message :







**Result:**

Original message: Dung Chi Luong

Ciphertext: 

Plaintext: Dung Chi Luong

## The efficiency and security of RSA cryptosystem :

The time needed to complete encryption and decryption operations is used to measure the effectiveness of the RSA cryptosystem. A greater key length can improve security but could also cause the system to run more slowly. Therefore, by choosing an acceptable key length, we need to strike a balance between security and effectiveness.

The security of the RSA cryptosystem is determined by the difficulty of factoring large composite numbers, which forms the foundation of RSA encryption. If the prime factors of the public key are found, the system becomes vulnerable to attacks. Therefore, it's necessary to use large prime numbers to ensure the system's security.

## The considered security threats and limitations of the RSA cryptosystem :

- Key length : the security of RSA depends on the key length used. If the key length is too short, the system can be easily compromised.

- Key Exchange: RSA requires secure key exchange to establish a secure communication channel. If an attacker intercepts the keys during transmission, they can easily decrypt the messages.

- Vulnerability to attacks: RSA is vulnerable to a range of attacks, including chosen-plaintext attacks, chosen-ciphertext attacks, and side-channel attacks. These attacks can be used to reveal the private key and decrypt the encrypted data.

## Enhance the security of the RSA cryptosystem implementation :

To improve the security of the RSA cryptosystem implementation, the following recommendations should be considered :

- Expand size of the key : the key size used for encryption and decryption should be long enough to resist attacks. Larger key sizes could be necessary for more sensitive applications; a minimum key size of 2048 bits is advised.

- Key management: access to private keys should be strictly controlled and handled securely.

- Side-channel attack mitigation: timing or power analysis attacks are two potential examples of side-channel attacks that countermeasures should be put in place to protect against these threats.. This can be achieved by implementing the hardware or software-based countermeasures like blinding or masking.

# SELF-EVALUATION FORM

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Criteria** | **Scale** | **1** | **2** | **3** | **Self-evalutaion** | **Reason** |
|  | **Score /10** | **0 score** | **1/2 score** | **Full score** |  |  |
| Part 1 | 4 | Do nothing or wrongly. | Correct calculation but wrong result or conclusion. | Correct calculation, detailed explanation. | 3 |  |
| Part 2 | 4 | Do nothing or wrongly. | Correct calculation but wrong result or conclusion. | Correct calculation, detailed explanation. | 4 |  |
| **Total** | 8 | Result | | | 7 |  |