**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**KHOA CÔNG NGHỆ THÔNG TIN 1**


# BÀI TẬP THỰC HÀNH MÔN:
# AN TOÀN VÀ BẢO MẬT HỆ THỐNG THÔNG TIN

## Đề tài: Tìm hiểu thuật toán Rabin và ElGamal

**Giảng viên hướng dẫn** : **QUẢN TRỌNG THẾ**
**Nhóm lớp** : **INT1303 - 20242 - 09**

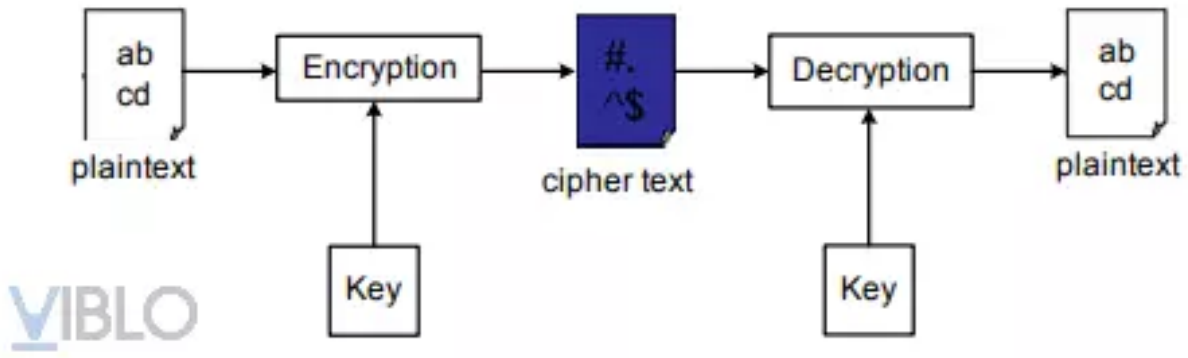
**Sinh viên thực hiện** : **Bùi Ngọc Hiếu**
**Mã sinh viên** : **B22DCCN300**

*Hà Nội – 2025*

# MỤC LỤC

## I. Giới thiệu về hệ mật mã

- Ta đã biết các gói tin thường được truyền trong mạng qua các kênh, trong đó việc đảm bảo an toàn trên các kênh là rất khó. Do đó để thực hiện truyền tin an toàn, người ta thường mã hóa thông tin theo các quy tắc nhất định gọi là hệ thống mật mã.
- Một hệ thống mật mã được định nghĩa là một bộ **(P, C, K. E, D)** với 5 thành phần chính:
    - **P (Plaintext): Bản rõ:** Dạng ban đầu của thông báo.
    - **C (Ciphertext): Bản mã:** Dạng mã của bản rõ ban đầu.
    - **K (Key): Khóa:** thông tin tham số dùng để mã hóa.
    - **E (Encryption):** Quá trình biến đổi thông tin từ dạng bản rõ sang bản mã bằng khóa hoặc không cần khóa.
    - **D (Decryption):** Quá trình ngược lại biến đổi thông tin từ dạng bản mã sang bản rõ.




- Quá trình mã hóa được tiến hành bằng cách áp dụng hàm toán học E lên bản rõ P, trở thành thông tin đã được mã hóa C. Quá trình giải mã là việc áp dụng hàm D lên thông tin C để thông tin được giải mã về dạng P.
- **Thám mã (cryptanalysis):** là việc nghiên cứu cách phá các hệ bảo mật nhằm phục hồi bản rõ ban đầu từ bản mã, nghiên cứu các nguyên lí và phương pháp giải mã mà không biết khóa. Có 3 phương pháp tấn công cơ bản của thám mã:
    - Tìm khóa vét cạn.
    - Phân tích thống kê.
    - Phân tích toán.
- Có nhiều cách phân loại thuật toán mã hóa nhưng cách phân loại theo khóa được sử dụng phổ biến hơn bao gồm 2 loại thuật toán chính:
    - **Mã hóa khóa đối xứng:** còn có một số tên gọi khác như Secret Key Cryptography (hay Private Key Cryptography), sử dụng cùng một khóa cho cả hai quá trình mã hóa và giải mã. Một số thuật toán nổi tiếng như DES (Data Encryption Standard), Triple DES (3DES), RC4,...
    - **Mã hóa bất đối xứng:** Hay còn được gọi với một cái tên khác là mã hóa khóa công khai (Public Key Cryptography), nó được thiết kế sao cho

khóa sử dụng trong quá trình mã hóa khác biệt với khóa được sử dụng trong quá trình giải mã.

- Khóa dùng trong quá trình giải mã không thể được tính toán hay suy luận từ khóa dùng để mã hóa và ngược lại, tức là hai khóa này có quan hệ với nhau về mặt toán học nhưng không thể suy diễn được ra nhau.
- Khóa dùng để mã hóa còn được gọi là public key (khóa công khai) vì khóa dùng cho việc mã hóa được công khai cho tất cả mọi người. Một người hòan toàn xa lạ có thể dùng khóa này để mã hóa dữ liệu nhưng chỉ duy nhất người mà có khóa giải mã tương ứng mới có thể đọc được dữ liệu.

## III. Thuật toán Rabin

- Do ông Michael O.Rabin đề xuất năm 1978.
- Rabin là một **hệ mã hóa bất đối xứng**.
- Rabin có độ an toàn dựa trên việc **không thể tính căn bậc hai modulo n** nếu không biết phân tích n=p*q. Tuy nhiên người gửi cần thêm thông tin để biết nghiệm đúng.

### 1. Mã hóa

- Ban đầu chọn hai số nguyên tố lớn p và q sao cho p ≡ q ≡ 3 mod 4 (p và q đồng dư: p và q đều là số nguyên tố lớn chia 4 dư 3). Khi đó thực hiện tính n = p * q;
  - **Khóa công khai: n**
  - **Khóa bí mật: p, q**
- Với bản rõ m. tính $c = m^2 \ mod \ n$ → c là bản mã để gửi đi.

### 2. Giải mã

- Để giải mã, ta cần dùng thuật toán Euclide mở rộng tìm 2 số nguyên a và b thỏa mãn ap + bq = 1.
  - Tính $r = c^{(p+1)/4} \ mod \ p$
  - Tính $s = c^{(q+1)/4} \ mod \ q$
  - Tính $x = (aps + bqr) \ mod \ n$
  - Tính $y = (aps - bqr) \ mod \ n$
  - Có 4 nghiệm: x, -x mod n, y, -y mod n

→ Với 4 nghiệm tương ứng với 4 giá trị m, chỉ 1 trong số đó là bản rõ. → Cần thêm thông tin xác thực để nhân biết nghiệm đúng. Giải mã phức tạp hơn RSA vì cần xử lý nhiều nghiệm.

**Ví dụ:** với p = 331, q = 311, m = 633. Giả sử 6 bit cuối cùng của thông điệp ban đầu cần phải được lặp lại trước khi mã hóa.

**Bài làm**

**Mã hóa:**

- Thông điệp $633_{(10)} = 1001111001_{(2)}$. Lặp lại 6 bit cuối cùng nhận được $m = 1001111001111001_{(2)} = 40569_{(10)}$.
- **Khóa công khai** của a là n = p * q = 102941.
- **Khóa bí mật** là (331,311).
- Tính $c = m^2 \, mod \, n$ = 23053.

**Giải mã:**

- Dùng thuật toán Euclide mở rộng tìm 2 số nguyên a và b thỏa mãn: ap + bq = 1. Tìm được a = 140, b = -149.
  - Tính $r = c^{(p+1)/4} \, mod \, p = 144$
  - Tính $s = c^{(q+1)/4} \, mod \, q$ = 139
  - Tính $x = (aps + bqr) \, mod \, n = -25674$
  - Tính $y = (aps - bqr) \, mod \, n = 40569$

  → Có 4 nghiệm x, -x mod n, y, -y mod n tương ứng với m1, m2, m3, m4.
  - m1 = $25674_{(10)} = 0110010001001010_{(2)}$
  - m2 = $77267_{(10)} = 0010110111010011_{(2)}$
  - m3 = $40569_{(10)} = 1001111001111001_{(2)}$
  - m4 = $62372_{(10)} = 1111001110100100_{(2)}$
- Ta thấy chỉ có m3 dư thừa dữ liệu yêu cầu → bỏ 6 bit lặp cuối cùng và phục hồi bản rõ ban đầu là m = $633_{(10)} = 1001111001_{(2)}$.

## II. Thuật toán ElGamal

- Do ông Teher ElGamal người Ai Cập đề xuất năm 1984.
- ElGamal là một **hệ mã hóa bất đối xứng**.
- ElGamal dựa trên bài toán logarithm. Tính an toàn của nó phụ thuộc vào **độ phức tạp của bài toán logarithm**.

### 1. Mã hóa

- Ban đầu người ta sẽ chọn một số nguyên tố lớn p và hai số nguyên tố nhỏ hơn là alpha và a (khóa bí mật của người nhận). Khi đó, khóa công khai sẽ là:

  $beta = alpha^a \, mod \, p$
- Để mã hóa một thông điệp M thành bản mã C người gửi chọn một số ngẫu nhiên k nhỏ hơn p và tính cặp bản mã:

$$C_1 = alpha^k \bmod p$$

$$C_2 = (M * beta^k) \bmod p$$

- Sau đó gửi bản mã $C = (C_1, C_2)$ và k bị hủy đi.

**2. Giải mã**

- Để giải mã thông điệp M đầu tiên ta dùng khóa bí mật a và tính theo công thức:

$$M = (C_2 * (C_1^a)^{-1}) \bmod p$$

Với $((C_1^a)^{-1}) \bmod p = (C_1^{p-1-a}) \bmod p$

→ Kết luận: Xây dựng được hệ mã Elgamal bộ khóa K = (p, alpha, a, beta) với:

- **Khóa công khai:** $K_U = (alpha, beta, p)$
- **Khóa bí mật:** $K_R = (a, p)$

**Ví dụ**: Cho hệ ElGamal có p=2579 , alpha = 2, a=765. Chọn k ngẫu nhiên là 853. Bản rõ M=1299. Tìm khóa của hệ mã trên?

**Bài làm**

**Mã hóa**:

- Trước hết ta tính: $Beta = alpha^a \bmod p = 2^{765} \bmod 2579 = 949$.
- Để mã hóa thông điệp M =1299 ta tính theo k=853:

- $C_1 = alpha^k \bmod p = 2^{853} \bmod 2579 = 435$
- $C_2 = (M * beta^k) \bmod p = (1299 * 949^{853}) \bmod 2579 = 2396$

→ Vậy bản mã được gửi đi sẽ là C = (435, 2396).

**Giải mã:**

- Với khóa bí mật a = 765:

$$(C_1^a)^{-1} \bmod p = (C_1^{p-1-a}) \bmod p = (435^{2579-1-765}) \bmod 2579$$

$$= (435^{1813}) \bmod 2579 = 1980$$

$$M = (C_2 * (C_1^a)^{-1}) \bmod p = (2396 * 1980) \bmod 2579 = 1299$$

→ Kết luận: Xây dựng được hệ mã ElGamal bộ khóa: K = (p, alpha, a, beta) = (2579, 2, 765, 949) với:

- Thành phần **khóa công khai**: $K_U = (alpha, beta, p) = (2, 949, 2579)$
- Thành phần **khóa bí mật**: $K_R = (a, p) = (765, 2579)$
- Mã hóa M = 1299 với $C(C1, C2) = (435, 2396)$

## 3. Thám mã ElGamal

- Trong quá trình mã hóa ElGamal, ta có các thành phần alpha, beta, a, p với a nằm trong thành phần khóa bí mật. Ban đầu với các phép toán modulo ta nhanh chóng tính được các thành phần của khóa công khai và khóa bí mật. Nhưng khi kẻ tấn công có khóa công khai và cả bản mã $C = (C_1, C_2)$, việc tìm ra bản rõ M mà không có khóa bí mật hay tìm ra khóa bí mật (tìm ra a) là rất khó với p đủ lớn. $beta = alpha^a \bmod p \rightarrow a = log_{alpha}(beta) \bmod p$.
- Qua đó, để thám mã thuật toán ElGamal, ta có giải bài toán logarit rời rạc với hai thuật toán: **Thuật toán Shank** và **thuật toán Pohlig-Hellman**. Mục tiêu: Tìm x sao cho: $alpha^x = beta \bmod p$.

### 3.1. Thuật toán Shank

- **Ý tưởng:** dùng bộ nhớ dung lượng lớn để giảm thời gian thực hiện từ *O(p)* xuống $O(sqrt(p))$ bằng cách chia nhỏ không gian tìm kiếm.
  - Đặt $N = \lceil\sqrt{p-1}\rceil$
  - Tạo bảng, tính và lưu các giá trị:

    $(j, alpha^j \bmod p)$ $với$ $j = 0 \rightarrow N$

  - Tính $alpha^{-N} \bmod p$ dùng nghịch đảo modulo.
  - Lặp qua i: 0 → N
    - Tính $gamma = beta * (alpha^{-N})^i \bmod p$
    - Nếu *gamma* trùng với một giá trị trong bảng tại j, ta có $x = iN + j$
  - Khi đó tìm được x thỏa mãn $alpha^x = beta \bmod p$

### 3.2. Thuật toán Pohlig-Hellman

- **Ý tưởng:** Phân tích bài toán thành các logarit nhỏ hơn với p-1 ra nhiều ước nhỏ, rồi hợp các kết quả.
  - Giả sử $p - 1 = q_1^{e_1} q_2^{e_2} \dots q_k^{e_k}$
  - Với mỗi thừa số $q_i^{e_i}$ :
    - Tính $x_i = log_{alpha}(beta) \bmod q_i^{e_i}$
    - Dùng kỹ thuật đặc biệt để giải logarit modulo $q_i^{e_i}$
  - Áp dụng CRT để tìm $x \bmod (p-1)$

## IV. Thực hiện mô phỏng tạo - kiểm tra chữ ký số sử dụng ElGamal và Rabin.

- Chuẩn bị 2 máy ảo Ubuntu LTS 20.04 với tên gọi Ubuntu Linux 1 (Recipient - nhận file thông điệp và public key) và Ubuntu Linux 2 (Sender - viết code tạo khóa, ký thông điệp bằng python).

### 1. Thực hiện các cấu hình cần thiết

- Thực hiện cài đặt OpenSSH và OpenSSL trên cả hai máy:

```
sudo apt update && sudo apt install openssh-server openssl -y
```

- Thực hiện lấy địa chỉ IP máy Recipient (192.168.30.132) và kết nối ssh sender.
  - Máy recipient:

```
b22dccn300@ubuntu:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP gro
up default qlen 1000
    link/ether 00:0c:29:b0:a5:18 brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet 192.168.30.132/24 brd 192.168.30.255 scope global dynamic noprefixroute
 ens33
       valid_lft 1763sec preferred_lft 1763sec
    inet6 fe80::e3a5:b1ab:b9b6:e10d/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOW
N group default
    link/ether 56:8e:5a:27:48:a9 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
```

  - Máy sender: `ssh b22dccn300@192.168.30.132`

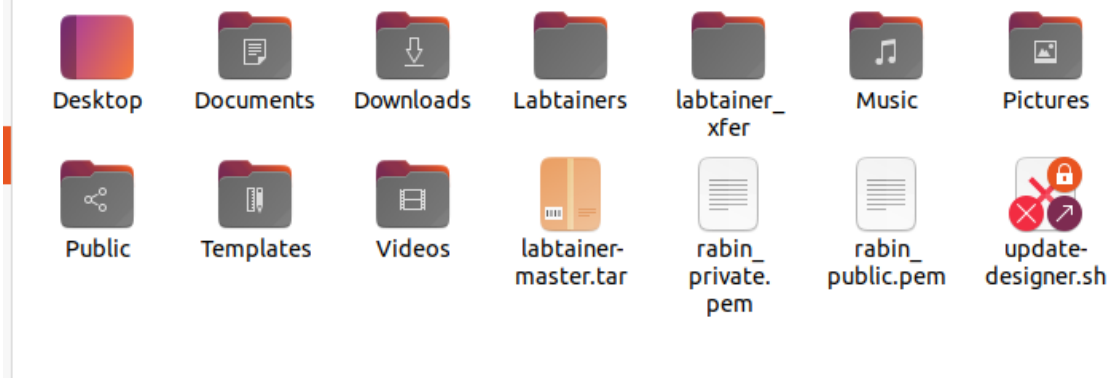
### 2. Thuật toán Rabin

- Tạo cặp khóa Rabin (private key và public key) trên **Sender**:

```
b22dccn300@ubuntu:~/Desktop$ openssl genpkey -algorithm RSA -pkeyopt
 rsa_keygen_bits:2048 -out rabin_private.pem
```

```
b22dccn300@ubuntu:~/Desktop$ openssl rsa -pubout -in rabin_private.p
em -out rabin_public.pem
```

- Khi đó xuất hiện hai file khóa rabin_private.pem và rabin_public trên trang home:



- Thực hiện trên máy **Sender** gửi file sang máy Recipient với lệnh:

```
b22dccn300@ubuntu:~$ scp rabin_public.pem b22dccn300@192.168.30.132:/home/b22dcc
n300/
b22dccn300@192.168.30.132's password:
rabin_public.pem                              100%  451   254.5KB/s   00:00
```

- Kiểm tra trên máy **Recipient** xem đã nhận được file chưa:

```
b22dccn300@ubuntu:~$ ls ~/rabin_public.pem
/home/b22dccn300/rabin_public.pem
```

- Trên máy **Sender**:
  - Tạo file thông điệp:

```
b22dccn300@ubuntu:~/Desktop$ echo "Hello, this is a
private message from Sender" >message.txt
```

  - Ký số bằng private key Rabin:

```
b22dccn300@ubuntu:~/Desktop$ openssl dgst -sha256 -sign
rabin_private.pem -out message_rabin.sig message.txt
```

  - Thực hiện gửi thông điệp và chữ ký cho Recipient:

```
b22dccn300@ubuntu:~/Desktop$ scp message.txt message_rab
in.sig b22dccn300@192.168.30.132:/home/b22dccn300/
```

- Trên máy **Recipient**:
- Kiểm tra chữ ký Rabin, nếu kết quả trả về Verified OK → Thông điệp toàn vẹn và đúng nguồn gốc.

```
b22dccn300@ubuntu:~$ openssl dgst -sha256 -verify rabin_public.pem -signature me
ssage_rabin.sig message.txt
Verified OK
```

### 3. Thuật toán ElGamal

- OpenSSL không hỗ trợ trực tiếp ElGamal, do đó ta cần thực hiện mô phỏng bằng cách sử dụng tham số DH.
- Thực hiện tạo cặp khóa ElGamal trên máy **Sender**:
  - Tạo tham số DH: tương đương tạo p và alpha

```
b22dccn300@ubuntu:~$ openssl dsaparam -out dsaparam.pem 2048
```

  - Tạo private key: tương đương tạo a

```
openssl gendsa -out private_key.pem dsaparam.pem
```

  - Trích xuất public key: $beta = alpha^a \ mod \ p$

```
b22dccn300@ubuntu:~$ openssl dsa -in private_key.pem -pu
bout -out public_key.pem
read DSA key
writing DSA key
```

- **Sender** thực hiện gửi public key cho Recipient qua SSH:

```
b22dccn300@ubuntu:~$ scp public_key.pem b22dccn300@192.168.30.132:/home/b22dccn3
00/
b22dccn300@192.168.30.132's password:
public_key.pem                                100% 1194   654.6KB/s   00:00
```

- Trên máy **Sender**:
    - Tạo thông điệp:

```
b22dccn300@ubuntu:~$ echo "Thong diep quan trong can duo
c bao mat" >message.txt
```

    - Tạo chữ ký số:

```
b22dccn300@ubuntu:~$ openssl dgst -sha256 -sign private_
key.pem -out signature.bin message.txt
```

    - Gửi thông điệp và chữ ký cho Recipient:

```
b22dccn300@ubuntu:~$ scp message.txt signature.bin b22dc
cn300@192.168.30.132:/home/b22dccn300
```

- Trên máy **Recipient**: Kiểm tra chữ ký ElGamal, nếu kết quả trả về Verified OK → Thông điệp toàn vẹn và đúng nguồn gốc.

```
b22dccn300@ubuntu:~$ openssl dgst -sha256 -verify public_key.pem -signature sign
ature.bin message.txt
Verified OK
```