Bài Tiểu Luận: Mã Hóa ElGamal và Ứng Dụng Trong Bảo Mật Thông Tin

[I. Cơ sở lý thuyết: 3](#_Toc163430422)

[1. Mã Hóa ElGamal: 3](#_Toc163430423)

[2. Ứng Dụng Trong Bảo Mật Thông Tin: 4](#_Toc163430424)

[3. Ưu và Nhược Điểm: 4](#_Toc163430425)

[II. Thực Nghiệm Bằng Python. 5](#_Toc163430426)

[III. Kết Luận: 9](#_Toc163430427)

# I. Cơ sở lý thuyết:

Trong thế giới kỹ thuật số hiện đại, việc bảo vệ thông tin cá nhân và dữ liệu quan trọng là một vấn đề cực kỳ quan trọng. Mã hóa ElGamal, một thuật toán mã hóa khóa công khai, đóng vai trò quan trọng trong việc đảm bảo tính bảo mật của dữ liệu trong quá trình truyền tải trên mạng internet và các hệ thống thông tin khác. Trong bài tiểu luận này, chúng ta sẽ tìm hiểu về cách hoạt động của mã hóa ElGamal, ứng dụng của nó trong bảo mật thông tin và một số ưu và nhược điểm của thuật toán này.

## 1. Mã Hóa ElGamal:

Mã hóa ElGamal là một thuật toán mã hóa khóa công khai, được phát triển bởi Taher Elgamal vào những năm 1980. Nó dựa trên tính chất khó của việc giải phương trình logarit ở trên một trường số nguyên tố. Thuật toán này sử dụng một cặp khóa: khóa công khai (public key) và khóa riêng (private key), giúp cho việc mã hóa và giải mã thông tin trở nên an toàn và bảo mật hơn.

* Khóa Công Khai (Public Key): Là một cặp giá trị gồm một số nguyên p và một số nguyên g (gọi là gốc nguyên thủy modulo p).
* Khóa Riêng (Private Key): Là một số nguyên x, được chọn một cách ngẫu nhiên từ một khoảng giá trị nhất định.

*Quá trình mã hóa thông tin bằng ElGamal bao gồm các bước sau:*

1. Chọn một số nguyên k ngẫu nhiên từ khoảng [1, p - 2].
2. Tính toán điểm mã hóa c1 = g^k mod p.
3. Tính toán điểm s = y^k mod p, trong đó y là khóa công khai.
4. Tính toán điểm mã hóa c2 = (plaintext \* s) mod p, với plaintext là giá trị ASCII của ký tự cần mã hóa.
5. Kết quả của quá trình mã hóa là cặp (c1, c2), được gửi đi.

*Quá trình giải mã thông tin sử dụng ElGamal là ngược lại:*

1. Sử dụng khóa riêng x để tính toán s = c1^x mod p.
2. Tính toán nghịch đảo của s theo modulo p.
3. Tính toán giá trị ASCII của ký tự đã mã hóa: plaintext = (c2 \* nghịch đảo của s) mod p.

**VÍ DỤ**

Giả sử Alice muốn gửi một tin nhắn cho Bob mà chỉ có Bob mới có thể đọc được. Alice và Bob đã thống nhất sử dụng mã hóa ElGamal để bảo vệ tính bảo mật của thông tin. Dưới đây là quá trình mã hóa thông tin của Alice:

Bước 1: Tạo Khóa Công Khai và Khóa Riêng:

* Alice chọn một số nguyên tố lớn p và một số nguyên g, là gốc nguyên thủy modulo p.
* Alice chọn p = 23 và g = 5.
* Alice chọn một số nguyên x ngẫu nhiên từ khoảng [1, p - 1] để tạo khóa riêng của mình.
* Alice chọn x = 6.
* Alice tính toán khóa công khai y = g^x mod p.
* Alice tính y = 5^6 mod 23 = 8.

Bước 2: Mã Hóa Tin Nhắn:

* Alice chọn một số ngẫu nhiên k từ khoảng [1, p - 2].
* Alice chọn k = 3.
* Alice chọn một thông điệp cần gửi và chuyển đổi mỗi ký tự của nó thành giá trị ASCII.
* Alice muốn gửi thông điệp "HELLO".
* ASCII của "H" là 72, "E" là 69, "L" là 76, và "O" là 79.
* Đối với mỗi giá trị ASCII, Alice thực hiện các phép tính sau:
* Đầu tiên, Alice tính c1 = g^k mod p.
* Với ASCII của "H" (72), c1 = 5^3 mod 23 = 10.
* Sau đó, Alice tính s = y^k mod p.
* s = 8^3 mod 23 = 17.

Cuối cùng

* Alice tính c2 = (giá trị ASCII \* s) mod p.
* Với ASCII của "H" (72), c2 = (72 \* 17) mod 23 = 16.
* Alice gửi cặp giá trị (c1, c2) cho Bob.
* Đối với "H", Alice gửi (10, 16) cho Bob.
* Alice lặp lại quá trình này cho mỗi ký tự trong tin nhắn của mình.
* Sau khi hoàn thành quá trình mã hóa, Alice gửi các cặp giá trị (c1, c2) cho Bob qua kênh bảo mật.

## 2. Ứng Dụng Trong Bảo Mật Thông Tin:

Mã hóa ElGamal có nhiều ứng dụng trong bảo mật thông tin, bao gồm:

* Bảo mật giao tiếp: ElGamal được sử dụng để bảo vệ tính bí mật của thông tin trong giao tiếp điện tử, đặc biệt là trong việc truyền tải thông tin mật qua internet.
* Chữ ký số: Thuật toán này cũng được sử dụng để tạo chữ ký số, giúp xác nhận tính toàn vẹn và nguồn gốc của tài liệu điện tử.
* Mã hóa tập tin và email: ElGamal được sử dụng để mã hóa dữ liệu trong các tệp tin và email, giúp bảo vệ thông tin cá nhân của người dùng.
* Ứng dụng trong blockchain: ElGamal cũng được sử dụng trong việc tạo và quản lý các cặp khóa cho ví điện tử trong blockchain.

## 3. Ưu và Nhược Điểm:

**Ưu điểm:**

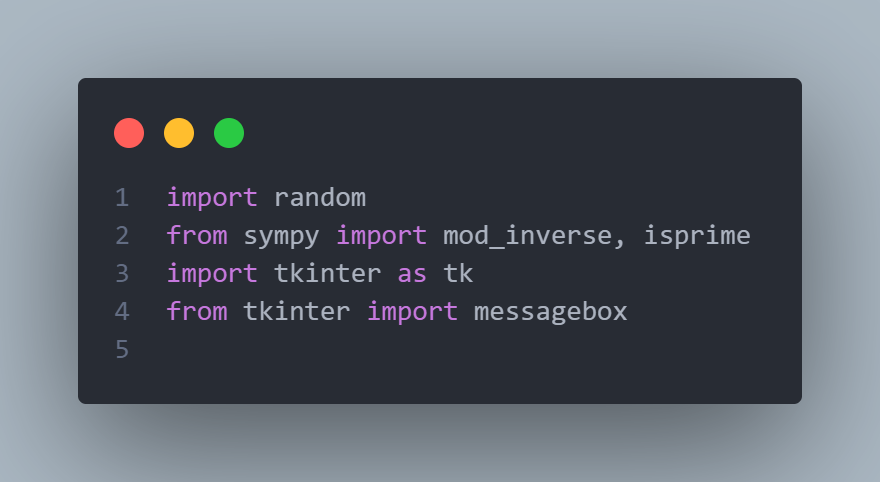
* Bảo mật cao: ElGamal dựa trên tính chất khó của việc giải phương trình logarit ở trên một trường số nguyên tố, làm cho việc tấn công trở nên khó khăn.
* Mã hóa khóa công khai: Không cần truyền tải khóa riêng giữa các bên, giúp giảm thiểu rủi ro bảo mật.

**Nhược điểm:**

* Tính toán phức tạp: Các phép tính toán trong quá trình mã hóa và giải mã có thể tốn nhiều thời gian và tài nguyên máy tính.
* Khối lượng dữ liệu lớn: ElGamal không hiệu quả với việc mã hóa các khối lớn của dữ liệu do tốn kém về tài nguyên tính toán.

# II. Thực Nghiệm Bằng Python.

**Import Thư Viện.**



random để tạo số ngẫu nhiên.

sympy để tính modular inverse và kiểm tra số nguyên tố.

tkinter để tạo giao diện người dùng.

**Hàm Tạo Khóa.**

def tao\_khoa(p, g, x):

    """Tạo khóa công cộng và khóa riêng."""

    y = pow(g, x, p)  *# Khóa công cộng*

    return (y, x)

Hàm này tạo khóa công cộng và khóa riêng dựa trên số nguyên tố p, gốc nguyên thủy g, và khóa riêng x.

**Hàm Tìm Gốc Nguyên Thủy.**

def tim\_goc\_nguyen\_thuy(p):

    """Tìm gốc nguyên thủy theo modulo p."""

    if p == 2:

        return 1

    phi = p - 1

    for candidate in range(2, p):

        if pow(candidate, phi, p) == 1:

            return candidate

    return None

Hàm ““tim\_goc\_nguyen\_thuy”“ được viết để tìm gốc nguyên thủy modulo ““p”“, trong đó ““p”“ là một số nguyên tố. Dưới đây là một phân tích chi tiết về cách hoạt động của hàm:

1. Kiểm tra nếu ““p”“ là 2: Nếu ““p”“ bằng 2, gốc nguyên thủy là 1, vì trong trường hợp này, 1 là phần tử duy nhất của modulo 2 và là gốc nguyên thủy.

2. Tính toán giá trị ““phi”“: ““phi”“ được tính bằng ““p - 1”“. Đây là một bước quan trọng trong thuật toán vì các phép tính sau đó sẽ dựa trên giá trị ““phi”“.

3. Kiểm tra từng số trong khoảng từ 2 đến p - 1: Vòng lặp kiểm tra từng số từ 2 đến ““p - 1”“ để xem nó có phải là gốc nguyên thủy hay không.

4. Sử dụng phép tính lũy thừa modulo để kiểm tra: Với mỗi số ““candidate”“, hàm sử dụng phép tính lũy thừa modulo ““candidate^phi % p”“ và kiểm tra xem kết quả có bằng 1 hay không. Nếu kết quả bằng 1, điều này có nghĩa là ““candidate”“ là một gốc nguyên thủy modulo ““p”“, và hàm sẽ trả về giá trị của ““candidate”“.

5. Trả về None nếu không tìm thấy gốc nguyên thủy: Nếu không có số nào từ 2 đến ““p - 1”“ là gốc nguyên thủy, hàm sẽ trả về ““None”“, đồng nghĩa với việc không tìm thấy gốc nguyên thủy cho ““p”“.

Hàm này có thể sử dụng trong các thuật toán mã hóa như ElGamal để tìm gốc nguyên thủy cần thiết để thực hiện các phép tính mã hóa và giải mã.

Đoạn mã trên là một ví dụ về cách thực hiện mã hóa và giải mã bằng thuật toán ElGamal trong Python. Dưới đây là phân tích chi tiết của từng hàm:

**1. Hàm "ma\_hoa\_ky\_tu":**

def ma\_hoa\_ky\_tu(ky\_tu, p, g, y):

    """Mã hóa một ký tự sử dụng ElGamal."""

    chu\_cai = ord(ky\_tu)  *# Giá trị ASCII của ký tự*

    k = random.randint(1, p - 2)  *# Tạo một khóa ngẫu nhiên*

    c1 = pow(g, k, p)

    s = pow(y, k, p)

    c2 = (chu\_cai \* s) % p

    return (c1, c2)

- Tham số:

- "ky\_tu": Ký tự cần mã hóa.

- "p": Số nguyên tố modulo.

- "g": Gốc nguyên thủy modulo p.

- "y": Khóa công cộng.

- Cách thực hiện:

- Chuyển đổi ký tự thành giá trị ASCII.

- Tạo một số ngẫu nhiên "k" trong khoảng từ 1 đến "p - 2".

- Tính "c1 = g^k mod p" và "s = y^k mod p".

- Tính "c2 = (giá trị ASCII của ký tự \* s) mod p".

- Trả về cặp giá trị "(c1, c2)".

**2. Hàm "giai\_ma\_ky\_tu":**

def giai\_ma\_ky\_tu(ky\_tu\_ma\_hoa, p, x):

    """Giải mã một ký tự đã được mã hóa bằng ElGamal."""

    c1, c2 = ky\_tu\_ma\_hoa

    s = pow(c1, x, p)

    nghich\_dao\_s = mod\_inverse(s, p)

    if nghich\_dao\_s is None:

        raise ValueError(f"Nghịch đảo của {s} (mod {p}) không tồn tại")

    chu\_cai = (c2 \* nghich\_dao\_s) % p

    return chr(chu\_cai)

- Mục đích: Giải mã một ký tự đã được mã hóa bằng ElGamal.

- Tham số:

- "ky\_tu\_ma\_hoa": Cặp giá trị "(c1, c2)" đã mã hóa.

- "p": Số nguyên tố modulo.

- "x": Khóa riêng.

- Cách thực hiện:

- Tính "s = c1^x mod p".

- Tính nghịch đảo của "s" modulo "p".

- Kiểm tra xem nghịch đảo có tồn tại hay không.

- Tính "chu\_cai = (c2 \* nghich\_dao\_s) mod p".

- Chuyển đổi giá trị ASCII thành ký tự và trả về kết quả.

**3. Hàm "giai\_ma\_van\_ban":**

def giai\_ma\_van\_ban(van\_ban\_ma\_hoa, p, x):

    """Giải mã một chuỗi đã được mã hóa bằng ElGamal."""

    van\_ban\_giai\_ma = ""

    for ky\_tu\_ma\_hoa in van\_ban\_ma\_hoa:

        ky\_tu = giai\_ma\_ky\_tu(ky\_tu\_ma\_hoa, p, x)

        van\_ban\_giai\_ma += ky\_tu

    return van\_ban\_giai\_ma

- Mục đích: Giải mã một chuỗi đã được mã hóa bằng ElGamal.

- Tham số:

- "van\_ban\_ma\_hoa": Chuỗi các ký tự đã mã hóa.

- "p": Số nguyên tố modulo.

- "x": Khóa riêng.

- Cách thực hiện:

- Duyệt qua từng ký tự trong chuỗi đã mã hóa.

- Sử dụng hàm "giai\_ma\_ky\_tu" để giải mã từng ký tự và thêm vào chuỗi kết quả.

- Trả về chuỗi kết quả sau khi giải mã.

**4. Hàm "tao\_so\_nguyen\_to":**

- Mục đích: Tạo một số nguyên tố ngẫu nhiên.

- Cách thực hiện:

- Tạo một số nguyên tố ngẫu nhiên trong khoảng từ 1000 đến 10000.

- Lặp lại quá trình cho đến khi có một số nguyên tố được tạo ra.

- Trả về số nguyên tố được tạo.

def tao\_so\_nguyen\_to():

    """Tạo một số nguyên tố ngẫu nhiên."""

    p = random.randint(1000, 10000)

    while not isprime(p):

        p = random.randint(1000, 10000)

    return p

**5. Hàm "ma\_hoa\_giai\_ma":**

def ma\_hoa\_giai\_ma():

    """Mã hóa hoặc giải mã văn bản đầu vào dựa trên các tham số do người dùng cung cấp."""

    try:

        p = int(p\_nhap.get())

        g = int(g\_nhap.get())

        van\_ban\_goc = van\_ban\_nhap.get()

        x = random.randint(2, p - 2)

        y, \_ = tao\_khoa(p, g, x)

        van\_ban\_ma\_hoa = [ma\_hoa\_ky\_tu(ky\_tu, p, g, y) for ky\_tu in van\_ban\_goc]

        van\_ban\_giai\_ma = giai\_ma\_van\_ban(van\_ban\_ma\_hoa, p, x)

        ket\_qua.config(text=f"Mã hóa: {van\_ban\_ma\_hoa}\nGiải mã: '{van\_ban\_giai\_ma}'")

        messagebox.showinfo("Thành công", "Mã hóa và giải mã hoàn tất thành công!")

    except Exception as e:

        messagebox.showerror("Lỗi", str(e))

- Mục đích: Mã hóa hoặc giải mã một văn bản dựa trên tham số người dùng cung cấp.

- Cách thực hiện:

- Lấy các thông số "p", "g" và văn bản cần xử lý từ các ô nhập trên giao diện.

- Tạo một số ngẫu nhiên "x" trong khoảng từ 2 đến "p - 2".

- Tính khóa công cộng "y" từ "p", "g" và "x".

- Mã hóa từng ký tự trong văn bản sử dụng "ma\_hoa\_ky\_tu" và thu thập các kết quả mã hóa.

- Giải mã các ký tự đã mã hóa sử dụng "giai\_ma\_van\_ban" và hiển thị kết quả trên giao diện.

# III. Kết Luận:

Mã hóa ElGamal là một thuật toán mã hóa khóa công khai mạnh mẽ, có nhiều ứng dụng trong bảo mật thông tin. Mặc dù có những hạn chế về tính toán và hiệu suất, nhưng ElGamal vẫn là một phương tiện hiệu quả để đảm bảo tính bảo mật và quyền riêng tư trong truyền tải và lưu trữ dữ liệu kỹ thuật số.