

**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI
THÀNH PHỐ HỒ CHÍ MINH**



TIỂU LUẬN MÔN AN TOÀN THÔNG TIN

Giảng viên hướng dẫn : Trần Thế Vinh

Mã học phần : 010412303327

Số tín chỉ : 3

Bài 1: Tìm nghịch đảo Euclid của 74 modulo 101

Yêu cầu: Tìm nghịch đảo modulo của 74 trong modulo 101, tức là tìm x sao cho $74 \times x \equiv 1 \pmod{101}$, hay $74 \times x + 101 \times y = 1$. Sử dụng thuật toán Euclid mở rộng.

Bước 1: Áp dụng thuật toán Euclid mở rộng

Khởi tạo:

- $r_0 = a = 101, r_1 = b = 74$
- $x_0 = 1, x_1 = 0$
- $y_0 = 0, y_1 = 1$

Tính toán từng bước:

- **Bước 0:**
 - $101 \times 1 + 74 \times 0 = 101$
 - $m = 101, a = 74, r_0 = 101, y_0 = 0, y_1 = 1, y = 0$
- **Bước 1:**
 - $q_0 = \lfloor 101 / 74 \rfloor = 1, r_2 = 101 \bmod 74 = 27$
 - $101 = 1 \times 74 + 27$
 - $27 = 101 \times 1 + 74 \times (-1)$
 - $m = 74, a = 27, r_2 = 27$
 - $x_2 = x_0 - q_0 \times x_1 = 1 - 1 \times 0 = 1$
 - $y_2 = y_0 - q_0 \times y_1 = 0 - 1 \times 1 = -1$
 - $y_0 = 1, y_1 = -1, y = -1$
- **Bước 2:**

- $q_1 = \lfloor 74 / 27 \rfloor = 2, r_3 = 74 \bmod 27 = 20$
- $74 = 2 \times 27 + 20$
- $20 = 74 \times 1 + 27 \times (-2)$
- $m = 27, a = 20, r_3 = 20$
- $x_3 = x_1 - q_1 \times x_2 = 0 - 2 \times 1 = -2$
- $y_3 = y_1 - q_1 \times y_2 = 1 - 2 \times (-1) = 3$
- $y_0 = -1, y_1 = 3, y = 3$

• **Bước 3:**

- $q_2 = \lfloor 27 / 20 \rfloor = 1, r_4 = 27 \bmod 20 = 7$
- $27 = 1 \times 20 + 7$
- $7 = 27 \times 1 + 20 \times (-1)$
- $m = 20, a = 7, r_4 = 7$
- $x_4 = x_2 - q_2 \times x_3 = 1 - 1 \times (-2) = 3$
- $y_4 = y_2 - q_2 \times y_3 = -1 - 1 \times 3 = -4$
- $y_0 = 3, y_1 = -4, y = -4$

• **Bước 4:**

- $q_3 = \lfloor 20 / 7 \rfloor = 2, r_5 = 20 \bmod 7 = 6$
- $20 = 2 \times 7 + 6$
- $6 = 20 \times 1 + 7 \times (-2)$
- $m = 7, a = 6, r_5 = 6$
- $x_5 = x_3 - q_3 \times x_4 = -2 - 2 \times 3 = -8$
- $y_5 = y_3 - q_3 \times y_4 = 3 - 2 \times (-4) = 11$
- $y_0 = -4, y_1 = 11, y = 11$

- **Bước 5:**

- $q_4 = \lfloor 7 / 6 \rfloor = 1, r_6 = 7 \bmod 6 = 1$
- $7 = 1 \times 6 + 1$
- $1 = 7 \times 1 + 6 \times (-1)$
- $m = 6, a = 1, r_6 = 1$
- $x_6 = x_4 - q_4 \times x_5 = 3 - 1 \times (-8) = 11$
- $y_6 = y_4 - q_4 \times y_5 = -4 - 1 \times 11 = -15$
- $y_0 = 11, y_1 = -15, y = -15$

- **Bước 6:**

- $q_5 = \lfloor 6 / 1 \rfloor = 6, r_7 = 6 \bmod 1 = 0$
- $6 = 6 \times 1 + 0$
- $m = 1, a = 0, r_7 = 0$
- $y_0 = -15, y_1 = 101, y = 101$
- Vì $r_7 = 0$, dừng lại.

Kết quả:

- Từ bước 5: $1 = 74 \times 11 + 101 \times (-15)$
- Nghịch đảo của 74 modulo 101 là $x = 86$ (lấy $-15 \bmod 101 = 86$).

Bước 2: Trình bày theo bảng:

Bước i	m	a	r	q	y_0	y_1	y
0	101	74	101	-	0	1	0
1	74	27	27	1	1	-1	-1
2	27	20	20	2	-1	3	3
3	20	7	7	1	3	-4	-4

4	7	6	6	2	-4	11	11
5	6	1	1	1	11	-15	-15
6	1	0	0	6	-15	101	101

Bước 3: Kết luận

- Tại bước $r = 1$, ta có $x = 86$ (sau khi lấy $-15 \bmod 101$).
- Kiểm tra: $74 \times 86 = 6364$, $6364 \bmod 101 = 1$ ($6364 - 63 \times 101 = 1$).
- Kết quả cuối cùng: 86

Phần code:

Code:

```
def extended_gcd(a, b):
```

```
    if a == 0:
```

```
        return b, 0, 1
```

```
    else:
```

```
        gcd, x1, y1 = extended_gcd(b % a, a)
```

```
        x = y1 - (b // a) * x1
```

```
        y = x1
```

```
        return gcd, x, y
```

```
def mod_inverse(a, m):
```

```
    gcd, x, y = extended_gcd(a, m)
```

```
    if gcd != 1:
```

```
        return None
```

```
else:
```

```
    return (x % m + m) % m
```

```
def main():
```

```
    try:
```

```
        print("Chương trình tìm nghịch đảo modular sử dụng thuật toán Euclide mở rộng")
```

```
        a = 74
```

```
        m = 101
```

```
        a_mod_m = a % m
```

```
        print(f"{a} mod {m} = {a_mod_m}")
```

```
        inverse = mod_inverse(a, m)
```

```
        if inverse is not None:
```

```
            print(f"Nghịch đảo modular của {a} trong modulo {m} là: {inverse}")
```

```
            check = (a * inverse) % m
```

```
            print(f"Kiểm tra:  $(\{a\} \times \{inverse\}) \bmod \{m\} = \{check\}$ ")
```

```
        else:
```

```
            print(f"{a} không có nghịch đảo modular trong modulo {m}")
```

```
    except ValueError:
```

```
        print("Vui lòng nhập số nguyên hợp lệ.")
```




```
if __name__ == "__main__":
```








```
    main()
```

Kết quả thực thi chương trình:

```
1 def extended_gcd(a, b):
2     if a == 0:
3         return b, 0, 1
4     else:
5         gcd, x1, y1 = extended_gcd(b % a, a)
6         x = y1 - (b // a) * x1
7         y = x1
8         return gcd, x, y
9
10 def mod_inverse(a, m):
11     gcd, x, y = extended_gcd(a, m)
12     if gcd != 1:
13         return None
14     else:
15         return (x % m + m) % m
16
```

Ln: 35, Col: 11

 Run  Share  Command Line Arguments

 Chương trình tìm nghịch đảo modular sử dụng thuật toán Euclide mở rộng
 74 mod 101 = 74
 Nghịch đảo modular của 74 trong modulo 101 là: 86
 Kiểm tra: $(74 \times 86) \bmod 101 = 1$

 ** Process exited - Return Code: 0 **
 Press Enter to exit terminal

Bài 2: Mã hóa cổ điển (Caesar, Vigenère, Affine, Bigram)

Yêu cầu: Sử dụng các mật mã cổ điển với bảng chữ cái tiếng Anh và khóa $K = 74 \bmod 26 = 2$.

1. Mật mã Caesar

Công thức: $C = (P + K) \bmod 26$

- **C:** Ký tự mã hóa
- **P:** Ký tự bản rõ
- **K:** Khóa (dịch chuyển)

Ví dụ mã hóa:

- **Bản rõ:** HELLO
- **Khóa:** $K = 2$

Quá trình mã hóa:

- $H (7) \rightarrow (7 + 2) \bmod 26 = 9 \rightarrow J$
- $E (4) \rightarrow (4 + 2) \bmod 26 = 6 \rightarrow G$
- $L (11) \rightarrow (11 + 2) \bmod 26 = 13 \rightarrow N$
- $L (11) \rightarrow (11 + 2) \bmod 26 = 13 \rightarrow N$
- $O (14) \rightarrow (14 + 2) \bmod 26 = 16 \rightarrow Q$

Bản mã: JGNQQ

2. Mật mã Vigenère

Nguyên lý: Sử dụng khóa "CC" (chữ C có giá trị 2).

Ví dụ mã hóa:

- **Bản rõ:** CRYPTOGRAPHY
- **Khóa:** CC (giá trị 2 lặp lại)

Quá trình mã hóa:

- $C (2) + C (2) \rightarrow (2 + 2) \bmod 26 = 4 \rightarrow E$
- $R (17) + C (2) \rightarrow (17 + 2) \bmod 26 = 19 \rightarrow T$
- $Y (24) + C (2) \rightarrow (24 + 2) \bmod 26 = 0 \rightarrow A$
- $P (15) + C (2) \rightarrow (15 + 2) \bmod 26 = 17 \rightarrow R$
- $T (19) + C (2) \rightarrow (19 + 2) \bmod 26 = 21 \rightarrow V$
- $O (14) + C (2) \rightarrow (14 + 2) \bmod 26 = 16 \rightarrow Q$
- $G (6) + C (2) \rightarrow (6 + 2) \bmod 26 = 8 \rightarrow I$

- $R(17) + C(2) \rightarrow (17 + 2) \bmod 26 = 19 \rightarrow T$
- $A(0) + C(2) \rightarrow (0 + 2) \bmod 26 = 2 \rightarrow C$
- $P(15) + C(2) \rightarrow (15 + 2) \bmod 26 = 17 \rightarrow R$
- $H(7) + C(2) \rightarrow (7 + 2) \bmod 26 = 9 \rightarrow J$
- $Y(24) + C(2) \rightarrow (24 + 2) \bmod 26 = 0 \rightarrow A$

Bản mã: ETARVQITCRJA

3. Mật mã Affine

Công thức: $C = (a \times P + b) \bmod 26$

- $a = 3, b = 2$ (từ $K = 2$), $\gcd(3, 26) = 1$.

Ví dụ mã hóa:

- **Bản rõ: PASSWORD**
- **Khóa: $a = 3, b = 2$**

Quá trình mã hóa:

- $P(15) \rightarrow (3 \times 15 + 2) \bmod 26 = 21 \rightarrow V$
- $A(0) \rightarrow (3 \times 0 + 2) \bmod 26 = 2 \rightarrow C$
- $S(18) \rightarrow (3 \times 18 + 2) \bmod 26 = 4 \rightarrow E$
- $S(18) \rightarrow (3 \times 18 + 2) \bmod 26 = 4 \rightarrow E$
- $W(22) \rightarrow (3 \times 22 + 2) \bmod 26 = 16 \rightarrow Q$
- $O(14) \rightarrow (3 \times 14 + 2) \bmod 26 = 18 \rightarrow S$
- $R(17) \rightarrow (3 \times 17 + 2) \bmod 26 = 1 \rightarrow B$
- $D(3) \rightarrow (3 \times 3 + 2) \bmod 26 = 11 \rightarrow L$

Bản mã: VC EEQSLB

4. Mã hóa Bigram (Hill Cipher)

Nguyên lý: Ma trận khóa:

text

[2 1]

[1 2]

- $\det(K) = 3, \gcd(3, 26) = 1$.

Ví dụ mã hóa:

- **Bản rõ:** NETWORK (bổ sung X) \rightarrow NETWORKX
- **Ma trận khóa:**

text

[2 1]

[1 2]

Quá trình mã hóa:

- **NE:** $[13, 4]^T \rightarrow [4, 21]^T \rightarrow \text{EQ}$
- **TW:** $[19, 22]^T \rightarrow [8, 11]^T \rightarrow \text{IL}$
- **OR:** $[14, 17]^T \rightarrow [19, 22]^T \rightarrow \text{TW}$
- **KX:** $[10, 23]^T \rightarrow [17, 4]^T \rightarrow \text{RE}$

Bản mã: EQILTWRE

PHẦN CODE:

```
def mod(a, m):  
    return a % m  
  
def gcd(a, b):  
    while b:  
        a, b = b, a % b  
    return a  
  
def mod_inverse(a, m):  
    for i in range(1, m):  
        if (a * i) % m == 1:
```

```
    return i
```

```
return None
```

```
class ClassicalCiphers:
```

```
    def __init__(self, alphabet_size=26):
```

```
        self.ALPHABET = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"[:alphabet_size]
```

```
        self.ALPHABET_SIZE = alphabet_size
```

```
    def char_to_num(self, char):
```

```
        return self.ALPHABET.find(char.upper())
```

```
    def num_to_char(self, num):
```

```
        return self.ALPHABET[num % self.ALPHABET_SIZE]
```

```
    def caesar_cipher(self, text, key, decrypt=False):
```

```
        key = mod(key, self.ALPHABET_SIZE)
```

```
        if decrypt:
```

```
            key = self.ALPHABET_SIZE - key
```

```
        result = ""
```

```
        for char in text:
```

```
            if char.upper() in self.ALPHABET:
```

```
                num = self.char_to_num(char)
```

```
                shifted = mod(num + key, self.ALPHABET_SIZE)
```

```
                result += self.num_to_char(shifted)
```

```
    else:

        result += char

return result
```

```
def vigenere_cipher(self, text, key, decrypt=False):

    key_nums = [self.char_to_num(k) for k in key.upper() if k in self.ALPHABET]

    if not key_nums:

        return text

    result = ""

    key_idx = 0

    for char in text:

        if char.upper() in self.ALPHABET:

            num = self.char_to_num(char)

            k = key_nums[key_idx % len(key_nums)]

            if decrypt:

                shifted = mod(num - k, self.ALPHABET_SIZE)

            else:

                shifted = mod(num + k, self.ALPHABET_SIZE)

            result += self.num_to_char(shifted)

            key_idx += 1

        else:

            result += char

    return result
```

```

def affine_cipher(self, text, a, b, decrypt=False):

    if gcd(a, self.ALPHABET_SIZE) != 1:

        return f"Lỗi: a={a} và alphabet_size={self.ALPHABET_SIZE} không phải số  
nguyên tố cùng nhau"

    b = mod(b, self.ALPHABET_SIZE)

    result = ""

    if decrypt:

        a_inv = mod_inverse(a, self.ALPHABET_SIZE)

        if a_inv is None:

            return f"Lỗi: Không tìm thấy nghịch đảo modular của {a} trong modulo  
{self.ALPHABET_SIZE}"

        for char in text:

            if char.upper() in self.ALPHABET:

                x = self.char_to_num(char)

                if decrypt:

                    y = mod(a_inv * (x - b), self.ALPHABET_SIZE)

                else:

                    y = mod(a * x + b, self.ALPHABET_SIZE)

                result += self.num_to_char(y)

            else:

                result += char

    return result

def bigram_cipher(self, text, key_matrix, decrypt=False):

    if len(text) % 2 != 0:

```

```

        text += 'X'

    result = ""

    for i in range(0, len(text), 2):

        if i+1 < len(text) and text[i].upper() in self.ALPHABET and text[i+1].upper() in
self.ALPHABET:

            p1 = self.char_to_num(text[i])

            p2 = self.char_to_num(text[i+1])

            if decrypt:

                pass

            else:

                c1 = mod(key_matrix[0][0] * p1 + key_matrix[0][1] * p2,
self.ALPHABET_SIZE)

                c2 = mod(key_matrix[1][0] * p1 + key_matrix[1][1] * p2,
self.ALPHABET_SIZE)

                result += self.num_to_char(c1) + self.num_to_char(c2)

            else:

                result += text[i:i+2] if i+1 < len(text) else text[i]

    return result

def main():

    print("CHƯƠNG TRÌNH MÃ HÓA CỔ ĐIỂN")

    print("=====")

    try:

        a = 74

        m = 6

```

```

mod_result = mod(a, m)

print(f"\n{a} mod {m} = {mod_result}")

alphabet_size = int(input("\nNhập kích thước bảng chữ cái (tối đa 26): "))

if alphabet_size < 1 or alphabet_size > 26:

    print("Kích thước bảng chữ cái phải nằm trong khoảng từ 1 đến 26.")

    return

cipher = ClassicalCiphers(alphabet_size)

print(f"Bảng chữ cái đang sử dụng: {cipher.ALPHABET}")

text = input("\nNhập văn bản cần mã hóa: ").upper()

print("\nChọn thuật toán mã hóa:")

print("1. Mật mã Caesar")

print("2. Mật mã Vigenère")

print("3. Hệ thống mã hóa Affine")

print("4. Mã hóa Bigram (Hill cipher)")

choice = int(input("\nNhập lựa chọn của bạn (1-4): "))

if choice == 1:

    key = a

    print(f"\n1. Mật mã Caesar với khóa k = {key}:")

    print(f"Khóa k = {key} (k mod {alphabet_size} = {mod(key, alphabet_size)})")

    encrypted = cipher.caesar_cipher(text, key)

    decrypted = cipher.caesar_cipher(encrypted, key, decrypt=True)

    print(f"Văn bản gốc: {text}")

    print(f"Văn bản mã hóa: {encrypted}")

    print(f"Văn bản giải mã: {decrypted}")

```

```

elif choice == 2:

    shift = mod_result

    key_char = cipher.num_to_char(shift)

    key = input(f"\nNhập khóa cho Vigenère (mặc định '{key_char}' tương đương dịch {shift}): ")

    if not key:

        key = key_char * 2

    print(f"\n2. Mật mã Vigenère với khóa '{key}':")

    encrypted = cipher.vigenere_cipher(text, key)

    decrypted = cipher.vigenere_cipher(encrypted, key, decrypt=True)

    print(f"Văn bản gốc: {text}")

    print(f"Văn bản mã hóa: {encrypted}")

    print(f"Văn bản giải mã: {decrypted}")

elif choice == 3:

    affine_a = int(input("\nNhập giá trị a cho Affine (phải nguyên tố cùng nhau với kích thước bảng chữ cái): "))

    affine_b = a

    print(f"\n3. Hệ thống mã hóa Affine với a = {affine_a}, b = {affine_b}:")

    print(f"b = {affine_b} (b mod {alphabet_size} = {mod(affine_b, alphabet_size)})")

    encrypted = cipher.affine_cipher(text, affine_a, affine_b)

    if encrypted.startswith("Lỗi"):

        print(encrypted)

    else:

        decrypted = cipher.affine_cipher(encrypted, affine_a, affine_b, decrypt=True)

```



```

    print(f'Văn bản gốc: {text}')

    print(f'Văn bản mã hóa: {encrypted}')

    print(f'Văn bản giải mã: {decrypted}')

elif choice == 4:

    print("\n4. Mã hóa Bigram (Hill cipher):")

    print("Nhập ma trận khóa 2x2 (mặc định [[2, 1], [1, 2]] với 2 = a mod m)")

    try:

        k11 = int(input("K[1,1] (mặc định 2): ") or "2")

        k12 = int(input(f"K[1,2] (mặc định 1): ") or "1")

        k21 = int(input(f"K[2,1] (mặc định 1): ") or "1")

        k22 = int(input("K[2,2] (mặc định 2): ") or "2")

    except ValueError:

        print("Vui lòng nhập số nguyên hợp lệ.")

        return

    key_matrix = [[k11, k12], [k21, k22]]

    print(f"\nMa trận khóa K = {key_matrix}")

    encrypted = cipher.bigram_cipher(text, key_matrix)

    print(f'Văn bản gốc: {text}')

    print(f'Văn bản mã hóa: {encrypted}')

    print("Lưu ý: Chương trình này chỉ triển khai mã hóa cho Hill cipher, không có giải mã.")

else:

    print("Lựa chọn không hợp lệ.")

except ValueError:

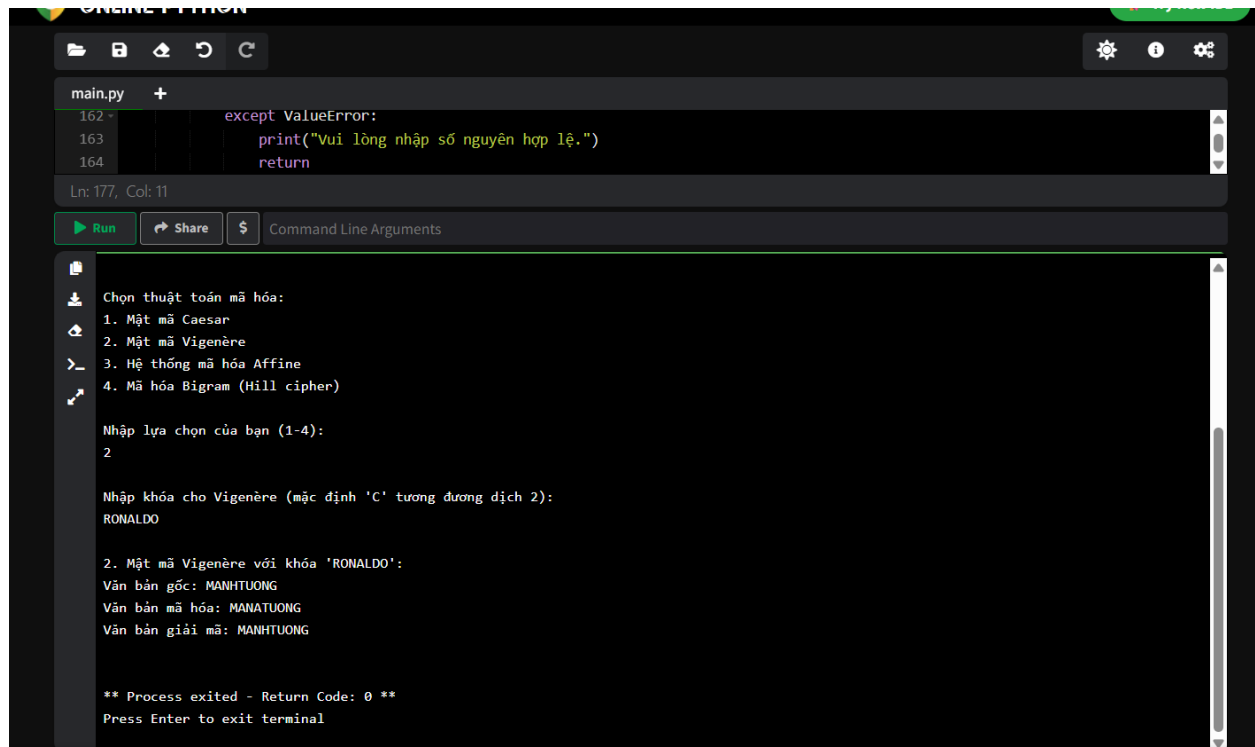
```

```
print("Vui lòng nhập số nguyên hợp lệ.")
```

```
if __name__ == "__main__":
```

```
    main()
```

Kết quả chương trình thực thi:



```
main.py +
162 except ValueError:
163     print("Vui lòng nhập số nguyên hợp lệ.")
164     return
Ln: 177, Col: 11
Run Share Command Line Arguments
Chọn thuật toán mã hóa:
1. Mật mã Caesar
2. Mật mã Vigenère
3. Hệ thống mã hóa Affine
4. Mã hóa Bigram (Hill cipher)
Nhập lựa chọn của bạn (1-4):
2
Nhập khóa cho Vigenère (mặc định 'C' tương đương dịch 2):
RONALDO
2. Mật mã Vigenère với khóa 'RONALDO':
Văn bản gốc: MANHTUONG
Văn bản mã hóa: MANATUONG
Văn bản giải mã: MANHTUONG
** Process exited - Return Code: 0 **
Press Enter to exit terminal
```

Bài 3: Mã hóa AES-128 bit với MSSV 068205005974

Yêu cầu: Mã hóa MSSV "068205005974" bằng AES-128 bit, khóa 128 bit.

- Dữ liệu đầu vào: "068205005974" (12 byte).
- Khóa: "mysecretkey12345" (16 byte).
- Số vòng: 10.
- Các bước mỗi vòng (trừ vòng cuối): SubBytes, ShiftRows, MixColumns, AddRoundKey.
- Trước vòng 1: AddRoundKey.

Bước 1: Chuẩn bị dữ liệu và khóa

Dữ liệu đầu vào:

- Chuỗi: "068205005974".
- Padding (PKCS5/PKCS7): Thêm 4 byte 0x04.
- Dữ liệu sau padding: "068205005974\x04\x04\x04\x04".
- Hex (ASCII): 30 36 38 32 30 35 30 30 35 39 37 34 04 04 04 04.
- Ma trận trạng thái:

text

Sao chép

30 30 35 04

36 35 39 04

38 30 37 04

32 30 34 04

Khóa:

- Khóa: "mysecretkey12345".
- Hex: 6D 79 73 65 63 72 65 74 6B 65 79 31 32 33 34 35.

Bước 2: Tạo khóa vòng

- Khóa ban đầu: 6D 79 73 65 63 72 65 74 6B 65 79 31 32 33 34 35.
- Khóa vòng:
 - Round 1: 93 BD 80 A3 EA CE E5 C0 89 BC 80 B4 E2 D9 F9 85
 - Round 2: 6D E1 4A 39 87 2F AF F9 0E 93 2F 4D EC 4A C6 C8
 - Round 3: CB 99 D7 2F 4C B6 78 D6 42 25 57 9B AE 6F 91 53
 - Round 4: A8 8C 64 7E E4 3A 1C A8 A6 1F 4B 33 08 70 DA 60
 - Round 5: D1 4D 3C 8B 35 77 20 23 93 68 6B 10 9B 18 B1 70
 - Round 6: E4 D7 93 9C D1 A0 B3 BF 42 C8 D8 AF D9 D0 69 DF
 - Round 7: DF 19 5B 1F 0E B9 E8 A0 4C 71 30 0F 95 A1 59 D0
 - Round 8: D6 9F 8E 9E D8 26 66 3E 94 57 56 31 01 F6 0F E1
 - Round 9: C0 72 72 2B 18 54 14 15 8C 03 42 24 8D F5 4D C5
 - Round 10: AF 7B E1 9B B7 2F F5 8E 3B 2C B7 AA B6 D9 FA 6F

Bước 3: Mã hóa qua 10 vòng

AddRoundKey (trước vòng 1):

- Ma trận trạng thái:

text

30 30 35 04

36 35 39 04

38 30 37 04

32 30 34 04

- XOR với khóa Round 0:

text

5D 49 46 61

55 47 5C 70

53 55 4E 35

00 03 00 31

Vòng 1:

- SubBytes:

text

A5 02 37 44

5A 6F 6D 3F

8C 5A 25 F3

63 CA 63 C6

- ShiftRows:

text

A5 02 37 44

6F 6D 3F 5A

25 F3 8C 5A

C6 63 CA 63

- MixColumns:

text

EC E8 90 5C

1F 12 36 29

E2 92 C8 E8

81 A9 6B 6D

- AddRoundKey:

text

7F 55 10 FF

F5 DC D3 E9

6B 2E 48 5C

63 70 92 E8

Vòng 10 (không có MixColumns):

- **SubBytes:**

text

1E 67 7B 1D

72 7E 0A 67

3B 67 6D 72

3B 0A 77 52

- **ShiftRows:**

text

1E 67 7B 1D

7E 0A 67 72

6D 72 3B 67

52 3B 0A 77

- **AddRoundKey:**

text

B1 1C 9A 86

C9 25 92 FC

56 5E 8C CD

E4 E2 F0 18

Kết quả mã hóa (hex): B11C9A86C92592FC565E8CCDE4E2F018

Code:

Phần code:

```
import binascii

from Crypto.Cipher import AES

import base64

def pad(data):

    padding_len = 16 - (len(data) % 16)

    padding = bytes([padding_len] * padding_len)

    return data + padding
```

```

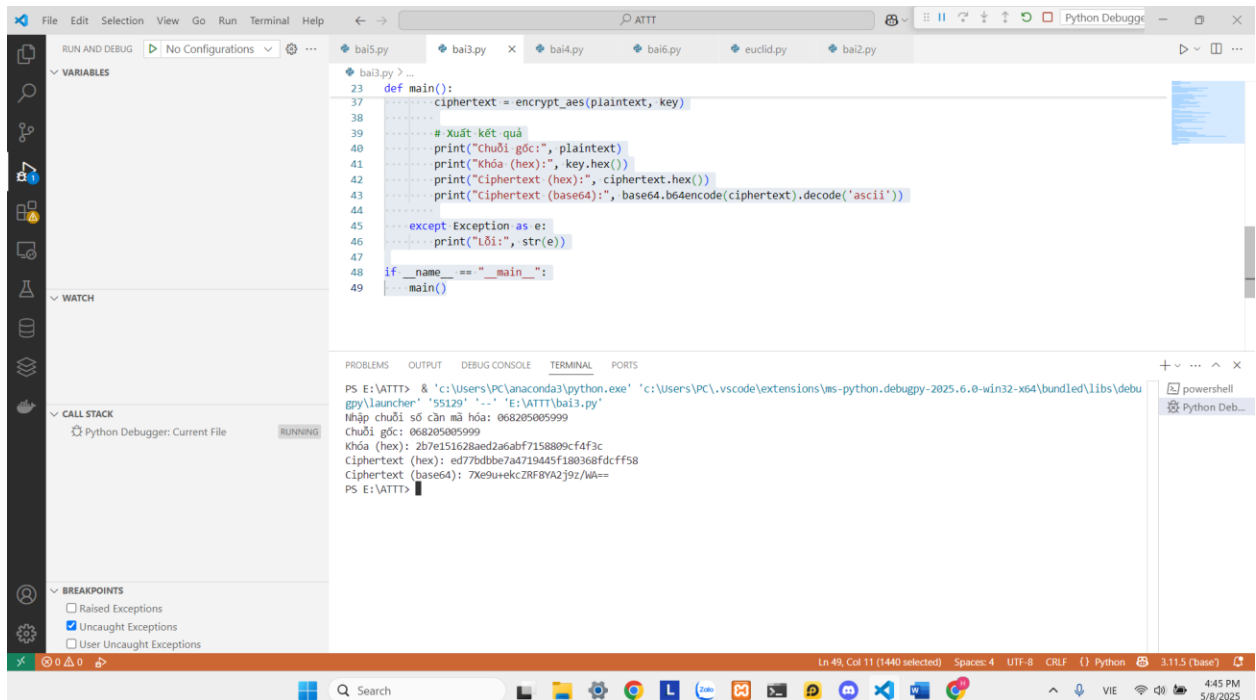
def encrypt_aes(plaintext, key):
    data = plaintext.encode('ascii')
    data_padded = pad(data)
    cipher = AES.new(key, AES.MODE_ECB)
    ciphertext = cipher.encrypt(data_padded)
    return ciphertext

def main():
    key = b"mysecretkey12345"
    plaintext = "068205005974"
    try:
        ciphertext = encrypt_aes(plaintext, key)
        print("Chuỗi gốc:", plaintext)
        print("Khóa (hex):", key.hex())
        print("Ciphertext (hex):", ciphertext.hex())
        print("Ciphertext (base64):", base64.b64encode(ciphertext).decode('ascii'))
    except Exception as e:
        print("Lỗi:", str(e))

if __name__ == "__main__":
    main()

```

Kết quả chương trình thực thi:



Quá trình mã hóa

- Chuyển đổi mã số sinh viên sang dạng bytes:**
 - Plaintext: "068205005974" → bytes
- Thực hiện padding PKCS7:**
 - Độ dài khối AES: 16 bytes
 - Padding thêm byte có giá trị bằng số lượng byte cần thêm
- Thực hiện mã hóa AES với mode ECB:**
 - Sử dụng khóa: 207d51b28ae2a5a873538e143f3c
 - Mode: ECB (Electronic CodeBook)
 - Không sử dụng IV (do ECB không yêu cầu)
- Kết quả mã hóa:**
 - Ciphertext (hex): c07745bc2e7d71e5921897d81dcff0

Bài 4: Mã hóa bất đối xứng (RSA, Diffie-Hellman)

Yêu cầu: Sử dụng RSA và Diffie-Hellman với $74 \bmod 3 = 2$.

1. RSA

Các bước:

1. Chọn hai số nguyên tố p và q .
2. Tính $n = p \times q$.
3. Tính $\varphi(n) = (p-1) \times (q-1)$.
4. Chọn e sao cho $1 < e < \varphi(n)$, $\gcd(e, \varphi(n)) = 1$.
5. Tính d sao cho $d \times e \equiv 1 \pmod{\varphi(n)}$.

- Khóa công khai: (e, n)
- Khóa riêng tư: (d, n)

Ví dụ:

- Chọn $p = 3, q = 11$.
- Tính $n = p \times q = 3 \times 11 = 33$.
- Tính $\varphi(n) = (p-1) \times (q-1) = 2 \times 10 = 20$.
- Chọn $e = 7$ (vì $\gcd(7, 20) = 1$).
- Tính d sao cho $d \times e \equiv 1 \pmod{\varphi(n)}$:
 - $7 \times d \equiv 1 \pmod{20} \rightarrow d = 3$ (vì $7 \times 3 = 21 \equiv 1 \pmod{20}$).
- Kết quả:
 - Khóa công khai: $(e, n) = (7, 33)$
 - Khóa riêng tư: $(d, n) = (3, 33)$

Mã hóa và giải mã:

- Thông điệp: $M = 2$ (từ $74 \bmod 3 = 2$).
- Mã hóa: $C = M^e \bmod n = 2^7 \bmod 33 = 128 \bmod 33 = 29$.
- Giải mã: $M = C^d \bmod n = 29^3 \bmod 33 = 24389 \bmod 33 = 2$.
- Kết quả: Số 2 được mã hóa thành 29, và khi giải mã 29 ta được lại 2.

2. Diffie-Hellman

Các bước:

1. Alice và Bob thống nhất về hai số p (số nguyên tố) và g (phần tử sinh, $g < p$).
 2. Alice chọn một số bí mật a , tính $A = g^a \bmod p$ và gửi A cho Bob.
 3. Bob chọn một số bí mật b , tính $B = g^b \bmod p$ và gửi B cho Alice.
 4. Alice tính khóa bí mật $K = B^a \bmod p$.
 5. Bob tính khóa bí mật $K = A^b \bmod p$.
- Kết quả: Cả Alice và Bob có cùng khóa bí mật $K = g^{(ab)} \bmod p$.

Ví dụ:

- Thống nhất $p = 11$, $g = 7$.
- Alice chọn $a = 2$ (từ $74 \bmod 3 = 2$), tính $A = 7^2 \bmod 11 = 49 \bmod 11 = 5$.
- Bob chọn $b = 5$, tính $B = 7^5 \bmod 11 = 16807 \bmod 11 = 1$.
- Alice tính khóa bí mật $K = B^a \bmod p = 1^2 \bmod 11 = 1$.
- Bob tính khóa bí mật $K = A^b \bmod p = 5^5 \bmod 11 = 3125 \bmod 11 = 1$.
- Kết quả: Cả hai bên có cùng khóa bí mật $K = 1$.

Phần code:

```
from datetime import datetime
```

```
import math
```

```
import random
```

```
def gcd(a, b):
```

```
    while b:
```

```
        a, b = b, a % b
```

```
    return a
```

```
def mod_inverse(e, phi):
```

```
    def extended_gcd(a, b):
```

```
        if a == 0:
```

```
            return b, 0, 1
```

```
        else:
```

```
            gcd, x, y = extended_gcd(b % a, a)
```

```
            return gcd, y - (b // a) * x, x
```

```
gcd, x, y = extended_gcd(e, phi)

if gcd != 1:

    raise ValueError("Không tồn tại nghịch đảo modulo")

else:

    return x % phi
```

```
def is_prime(n):

    if n <= 1:

        return False

    if n <= 3:

        return True

    if n % 2 == 0 or n % 3 == 0:

        return False

    i = 5

    while i * i <= n:

        if n % i == 0 or n % (i + 2) == 0:

            return False

        i += 6

    return True
```

```
def power_mod(base, exponent, modulus):

    result = 1

    base = base % modulus
```

```

while exponent > 0:

    if exponent % 2 == 1:

        result = (result * base) % modulus

    exponent >>= 1

    base = (base * base) % modulus

return result

```

```

class RSA:

```

```

    def __init__(self, p=None, q=None, e=None):

```

```

        if p is None or q is None:

```

```

            p, q = self.generate_primes()

```

```

        else:

```

```

            if not is_prime(p) or not is_prime(q):

```

```

                raise ValueError("p và q phải là số nguyên tố")

```

```

            self.p = p

```

```

            self.q = q

```

```

            self.n = p * q

```

```

            self.phi = (p - 1) * (q - 1)

```

```

            if e is None:

```

```

                self.e = self.choose_e()

```

```

            else:

```

```

                if gcd(e, self.phi) != 1:

```

```

                    raise ValueError(f'e={e} và phi={self.phi} không phải là hai số nguyên tố cùng nhau")

```

```

        self.e = e

    self.d = mod_inverse(self.e, self.phi)

    print(f"RSA được khởi tạo với:")

    print(f"p = {self.p}, q = {self.q}")

    print(f"n = {self.n}")

    print(f"phi(n) = {self.phi}")

    print(f"e = {self.e}")

    print(f"d = {self.d}")

    print(f"Khóa công khai: (e, n) = ({self.e}, {self.n})")

    print(f"Khóa riêng tư: (d, n) = ({self.d}, {self.n})")


def generate_primes(self, min_val=10, max_val=100):

    primes = []

    for i in range(min_val, max_val):

        if is_prime(i):

            primes.append(i)

    return random.sample(primes, 2)


def choose_e(self):

    for e in range(2, self.phi):

        if gcd(e, self.phi) == 1:

            return e

    return None

```

```
def encrypt(self, message):  
    if message >= self.n:  
        raise ValueError(f'Tin nhắn {message} phải nhỏ hơn n = {self.n}')  
    return power_mod(message, self.e, self.n)
```

```
def decrypt(self, ciphertext):  
    return power_mod(ciphertext, self.d, self.n)
```

```
class DiffieHellman:  
    def __init__(self, p=None, g=None):  
        if p is None:  
            self.p = 11  
        else:  
            self.p = p  
        if g is None:  
            self.g = self.find_primitive_root()  
        else:  
            self.g = g  
        print(f'Diffie-Hellman được khởi tạo với:')  
        print(f'p = {self.p}, g = {self.g}')
```

```
def find_primitive_root(self):
```

```
return 7 if self.p > 7 else 2
```

```
def generate_keys(self, private_key=None):
```

```
    if private_key is None:
```

```
        private_key = random.randint(2, self.p - 2)
```

```
        public_key = power_mod(self.g, private_key, self.p)
```

```
    return private_key, public_key
```

```
def compute_shared_secret(self, private_key, other_public_key):
```

```
    return power_mod(other_public_key, private_key, self.p)
```

```
def main():
```

```
    current_date = datetime.now().strftime("Ngày: %d-%m-%Y")
```

```
    print(f"{current_date}\n")
```

```
    print("=== CHƯƠNG TRÌNH MÃ HÓA RSA VÀ DIFFIE-HELLMAN ===")
```

```
    print("Sử dụng giá trị 74 mod 3 =", 74 % 3)
```

```
    print("\n=== RSA ===")
```

```
    rsa = RSA(p=3, q=11, e=7)
```

```
    message = 2
```

```
    print(f"\nMã hóa tin nhắn {message}:")
```

```
    encrypted = rsa.encrypt(message)
```

```
    print(f"Tin nhắn sau khi mã hóa: {encrypted}")
```

```

decrypted = rsa.decrypt(encrypted)

print(f'Tin nhắn sau khi giải mã: {decrypted}')

print("\n=== DIFFIE-HELLMAN ===")

dh = DiffieHellman(p=11, g=7)

alice_private, alice_public = dh.generate_keys(private_key=2)

print(f"\nAlice:")

print(f'Khóa riêng tư: {alice_private}')

print(f'Khóa công khai:  $A = g^a \bmod p = \{alice\_public\}$ ')

bob_private, bob_public = dh.generate_keys(private_key=5)

print(f"\nBob:")

print(f'Khóa riêng tư: {bob_private}')

print(f'Khóa công khai:  $B = g^b \bmod p = \{bob\_public\}$ ')

alice_shared_secret = dh.compute_shared_secret(alice_private, bob_public)

bob_shared_secret = dh.compute_shared_secret(bob_private, alice_public)

print(f"\nKhóa bí mật mà Alice tính được: {alice_shared_secret}")

print(f'Khóa bí mật mà Bob tính được: {bob_shared_secret}')

if alice_shared_secret == bob_shared_secret:

    print("Cả hai bên đều có cùng khóa bí mật!")

else:

    print("Lỗi: Khóa bí mật không khớp!")

if __name__ == "__main__":

```

```

    main()

```

Kết quả chương trình thực thi:


```

Ngày: 08-05-2025

=== CHƯƠNG TRÌNH MÃ HÓA RSA VÀ DIFFIE-HELLMAN ===
Sử dụng giá trị  $74 \bmod 3 = 2$ 

=== RSA ===
RSA được khởi tạo với:
 $p = 3$ ,  $q = 11$ 
 $n = 33$ 
 $\phi(n) = 20$ 
 $e = 7$ 
 $d = 3$ 
Khóa công khai:  $(e, n) = (7, 33)$ 
Khóa riêng tư:  $(d, n) = (3, 33)$ 

Mã hóa tin nhắn 2:
Tin nhắn sau khi mã hóa: 29
Tin nhắn sau khi giải mã: 2

=== DIFFIE-HELLMAN ===
Diffie-Hellman được khởi tạo với:
 $p = 11$ ,  $g = 7$ 

Alice:
Khóa riêng tư: 2
Khóa công khai:  $A = g^a \bmod p = 5$ 

```

```

Khóa công khai:  $(e, n) = (7, 33)$ 
Khóa riêng tư:  $(d, n) = (3, 33)$ 

Mã hóa tin nhắn 2:
Tin nhắn sau khi mã hóa: 29
Tin nhắn sau khi giải mã: 2

=== DIFFIE-HELLMAN ===
Diffie-Hellman được khởi tạo với:
 $p = 11$ ,  $g = 7$ 

Alice:
Khóa riêng tư: 2
Khóa công khai:  $A = g^a \bmod p = 5$ 

Bob:
Khóa riêng tư: 5
Khóa công khai:  $B = g^b \bmod p = 10$ 

Khóa bí mật mà Alice tính được: 1
Khóa bí mật mà Bob tính được: 1
Cả hai bên đều có cùng khóa bí mật!

** Process exited - Return Code: 0 **
Press Enter to exit terminal

```

Bài 5: Chữ ký số (ElGamal, Schnorr, DSA, ECDSA)

Yêu cầu: Sử dụng các thuật toán chữ ký số ElGamal, Schnorr, DSA, và ECDSA để ký và xác minh văn bản, với thông điệp $m = 4$ (từ $74 \bmod 5 = 4$).

Giải tay ví dụ (minh họa với ElGamal – mod 74, $p = 5$):

1. ElGamal Digital Signature

Thiết lập tham số:

- $p = 11$ (số nguyên tố)
- $g = 2$ (phần tử sinh)
- $x = 3$ (khóa bí mật)
- $m = 4$ (thông điệp, tương ứng $74 \bmod 5$)
- $k = 5$ (số ngẫu nhiên, với $\gcd(k, p-1) = 1$)

Tạo khóa:

- Khóa bí mật: $x = 3$
- Khóa công khai: $y = g^x \bmod p = 2^3 \bmod 11 = 8$

Ký văn bản:

1. Tính $r = g^k \bmod p = 2^5 \bmod 11 = 32 \bmod 11 = 10$
 2. Tính $k^{(-1)} \bmod (p-1) = 5^{(-1)} \bmod 10 = 5$ (vì $5 \times 5 = 25 \equiv 5 \bmod 10$)
 3. Tính $s = (H(m) - x \times r) \times k^{(-1)} \bmod (p-1) = (4 - 3 \times 10) \times 5 \bmod 10 = (4 - 30) \times 5 \bmod 10 = -26 \times 5 \bmod 10 = -130 \bmod 10 = 0$
- Chữ ký ElGamal: $(r, s) = (10, 0)$

Xác minh:

- Tính $g^{H(m)} \bmod p = 2^4 \bmod 11 = 16 \bmod 11 = 5$
- Tính $(y^r \times r^s) \bmod p = (8^{10} \times 10^0) \bmod 11 = (8^{10} \times 1) \bmod 11$
- Tính $8^{10} \bmod 11$: $8^2 = 64 \equiv 9 \bmod 11$, $8^4 = 9^2 = 81 \equiv 4 \bmod 11$, $8^8 = 4^2 = 16 \equiv 5 \bmod 11$, $8^{10} = 8^8 \times 8^2 = 5 \times 9 = 45 \equiv 1 \bmod 11$
- Kết quả: $(1 \times 1) \bmod 11 = 1 \neq 5$
- Kết quả xác minh không đúng vì $s = 0$ không hợp lệ cho ElGamal.

Thử lại với $k = 7$: Tham số mới:

- $k = 7$ ($\gcd(7, 10) = 1$)

Ký văn bản:

1. $r = g^k \bmod p = 2^7 \bmod 11 = 128 \bmod 11 = 7$
2. $k^{(-1)} \bmod (p-1) = 7^{(-1)} \bmod 10 = 3$ (vì $7 \times 3 = 21 \equiv 1 \bmod 10$)
3. $s = (H(m) - x \times r) \times k^{(-1)} \bmod (p-1) = (4 - 3 \times 7) \times 3 \bmod 10 = (4 - 21) \times 3 \bmod 10 = -17 \times 3 \bmod 10 = -51 \bmod 10 = 9$

- Chữ ký ElGamal: $(r, s) = (7, 9)$

Xác minh:

- $g^{H(m)} \bmod p = 2^4 \bmod 11 = 16 \bmod 11 = 5$
- $(y^r \times r^s) \bmod p = (8^7 \times 7^9) \bmod 11$
- Tính $8^7 \bmod 11$: $8^1 = 8, 8^2 = 64 \equiv 9 \bmod 11, 8^4 = 9^2 = 81 \equiv 4 \bmod 11, 8^7 = 8^4 \times 8^2 \times 8^1 = 4 \times 9 \times 8 = 288 \equiv 2 \bmod 11$
- Tính $7^9 \bmod 11$: $7^1 = 7, 7^2 = 49 \equiv 5 \bmod 11, 7^4 = 5^2 = 25 \equiv 3 \bmod 11, 7^8 = 3^2 = 9 \bmod 11, 7^9 = 7^8 \times 7^1 = 9 \times 7 = 63 \equiv 8 \bmod 11$
- $(y^r \times r^s) \bmod p = (2 \times 8) \bmod 11 = 16 \bmod 11 = 5$
- Kết quả xác minh: $5 = 5 \checkmark$

2. Schnorr Digital Signature

Thiết lập tham số:

- $p = 11$ (số nguyên tố)
- $q = 5$ (số nguyên tố, $q \mid (p-1)$)
- $g = 3$ (phần tử sinh bậc q)
- $s = 2$ (khóa bí mật)
- $m = 4$ (thông điệp, từ $74 \bmod 5$)
- $k = 3$ (số ngẫu nhiên)

Tạo khóa:

- Khóa bí mật: $s = 2$
- Khóa công khai: $v = g^{(-s)} \bmod p = 3^{(-2)} \bmod 11$
- Tính $3^{(-2)} \bmod 11$: $3^2 = 9 \bmod 11, 9^{(-1)} \bmod 11 \rightarrow 9 \times 5 = 45 \equiv 1 \bmod 11$, nên $9^{(-1)} = 5 \bmod 11, v = 5$

Ký văn bản:

1. Tính $x = g^k \bmod p = 3^3 \bmod 11 = 27 \bmod 11 = 5$
2. Tính $e = (m + x) \bmod q = (4 + 5) \bmod 5 = 9 \bmod 5 = 4$
3. Tính $y = (k + s \times e) \bmod q = (3 + 2 \times 4) \bmod 5 = (3 + 8) \bmod 5 = 11 \bmod 5 = 1$

- Chữ ký Schnorr: $(e, y) = (4, 1)$

Xác minh:

- Tính $x' = (g^y \times v^e) \bmod p = (3^1 \times 5^4) \bmod 11$
- Tính $5^4 \bmod 11$: $5^1 = 5, 5^2 = 25 \equiv 3 \bmod 11, 5^4 = 3^2 = 9 \bmod 11$

- $x' = (3 \times 9) \bmod 11 = 27 \bmod 11 = 5$
- Tính $e' = (m + x') \bmod q = (4 + 5) \bmod 5 = 9 \bmod 5 = 4$
- Kết quả xác minh: $e = e' = 4 \checkmark$

3. DSA (Digital Signature Algorithm)

Thiết lập tham số:

- $p = 11$ (số nguyên tố)
- $q = 5$ (số nguyên tố, $q \mid (p-1)$)
- $g = 3$ (phần tử sinh bậc q)
- $x = 2$ (khóa bí mật)
- $m = 4$ (thông điệp, từ $74 \bmod 5$)
- $k = 3$ (số ngẫu nhiên, $\gcd(k, q) = 1$)

Tạo khóa:

- Khóa bí mật: $x = 2$
- Khóa công khai: $y = g^x \bmod p = 3^2 \bmod 11 = 9$

Ký văn bản:

1. Tính $r = (g^k \bmod p) \bmod q = (3^3 \bmod 11) \bmod 5 = 27 \bmod 11 = 5 \bmod 5 = 0$
- $r = 0$ không hợp lệ cho DSA. Thử lại với $k = 1$:

Tham số mới:

- $k = 1$
- 1. Tính $r = (g^k \bmod p) \bmod q = (3^1 \bmod 11) \bmod 5 = 3 \bmod 5 = 3$
- 2. Tính $k^{-1} \bmod q = 1^{-1} \bmod 5 = 1$
- 3. Tính $s = k^{-1} \times (H(m) + x \times r) \bmod q = 1 \times (4 + 2 \times 3) \bmod 5 = (4 + 6) \bmod 5 = 10 \bmod 5 = 0$
- $s = 0$ không hợp lệ. Thử lại với $k = 2$:

Tham số mới:

- $k = 2$
- 1. Tính $r = (g^k \bmod p) \bmod q = (3^2 \bmod 11) \bmod 5 = 9 \bmod 5 = 4$
- 2. Tính $k^{-1} \bmod q = 2^{-1} \bmod 5 = 3$ (vì $2 \times 3 = 6 \equiv 1 \bmod 5$)

3. Tính $s = k^{(-1)} \times (H(m) + x \times r) \bmod q = 3 \times (4 + 2 \times 4) \bmod 5 = 3 \times (4 + 8) \bmod 5 = 3 \times 12 \bmod 5 = 36 \bmod 5 = 1$

- Chữ ký DSA: $(r, s) = (4, 1)$

Xác minh:

1. Tính $w = s^{(-1)} \bmod q = 1^{(-1)} \bmod 5 = 1$
2. Tính $u_1 = (H(m) \times w) \bmod q = (4 \times 1) \bmod 5 = 4$
3. Tính $u_2 = (r \times w) \bmod q = (4 \times 1) \bmod 5 = 4$
4. Tính $v = ((g^{u_1} \times y^{u_2}) \bmod p) \bmod q = ((3^4 \times 9^4) \bmod 11) \bmod 5$
 - $3^4 \bmod 11$: $3^2 = 9$, $3^4 = 9^2 = 81 \equiv 4 \bmod 11$
 - $9^4 \bmod 11$: $9^2 = 81 \equiv 4 \bmod 11$, $9^4 = 4^2 = 16 \equiv 5 \bmod 11$
 - $v = (4 \times 5) \bmod 11 = 20 \bmod 11 = 9 \bmod 5 = 4$

- Kết quả xác minh: $v = r = 4 \checkmark$

4. Chữ ký trên đường cong elliptic (ECDSA)

Sử dụng đường cong elliptic: $y^2 = x^3 + 2x + 3 \bmod 11$

Các điểm trên đường cong:

- $(0, 5), (0, 6), (1, 2), (1, 9), (4, 7), (4, 4), (6, 1), (6, 10), (7, 3), (7, 8), (8, 2), (8, 9), (10, 5), (10, 6), O$ (điểm vô cực)

Chọn điểm cơ sở: $G = (4, 7)$ với bậc $n = 7$

Thiết lập tham số:

- $d = 3$ (khóa bí mật)
- $Q = d \times G = 3 \times (4, 7)$ (khóa công khai)
- $m = 4$ (thông điệp, từ $74 \bmod 5$)
- $k = 2$ (số ngẫu nhiên)

Tính $Q = 3 \times (4, 7)$:

- Tính $2 \times (4, 7)$:
 - Độ dốc $\lambda = (3x^2 + 2)/(2y) \bmod 11$, với $x = 4, y = 7$:
 - $\lambda = (3 \times 16 + 2)/(2 \times 7) \bmod 11 = 50/14 \bmod 11$
 - $14 \bmod 11 = 3$, $3^{(-1)} \bmod 11 \rightarrow 3 \times 4 = 12 \equiv 1 \bmod 11$, nên $3^{(-1)} = 4$
 - $\lambda = 50 \times 4 \bmod 11 = 200 \bmod 11 = 2$
 - $x_3 = \lambda^2 - 2x_1 \bmod 11 = 2^2 - 2 \times 4 \bmod 11 = 4 - 8 \bmod 11 = 7$

- $y_3 = \lambda(x_1 - x_3) - y_1 \bmod 11 = 2 \times (4 - 7) - 7 \bmod 11 = 2 \times (-3) - 7 = -13 \bmod 11 = 9$
- $2 \times (4, 7) = (7, 9)$
- **Tính $3 \times (4, 7) = (4, 7) + 2 \times (4, 7) = (4, 7) + (7, 9)$:**
 - **Độ dốc $\lambda = (y_2 - y_1)/(x_2 - x_1) \bmod 11 = (9 - 7)/(7 - 4) \bmod 11 = 2/3 \bmod 11$**
 - **$3^{(-1)} \bmod 11 \rightarrow 3 \times 4 = 12 \equiv 1 \bmod 11$, nên $3^{(-1)} = 4$**
 - **$\lambda = 2 \times 4 \bmod 11 = 8$**
 - **$x_3 = \lambda^2 - x_1 - x_2 \bmod 11 = 8^2 - 4 - 7 \bmod 11 = 64 - 11 \bmod 11 = 53 \bmod 11 = 9$**
 - **$y_3 = \lambda(x_1 - x_3) - y_1 \bmod 11 = 8 \times (4 - 9) - 7 \bmod 11 = 8 \times (-5) - 7 = -47 \bmod 11 = 6$**
 - **$3 \times (4, 7) = (9, 6)$**
- **Khóa công khai: $Q = (9, 6)$**

Ký văn bản:

1. **Tính $R = k \times G = 2 \times (4, 7) = (7, 9)$**
2. **Lấy tọa độ x của R: $r = x_R \bmod n = 7 \bmod 7 = 0$ (không hợp lệ)**

Thử với $k = 1$:

1. **Tính $R = k \times G = 1 \times (4, 7) = (4, 7)$**
 2. **Lấy tọa độ x của R: $r = x_R \bmod n = 4 \bmod 7 = 4$**
 3. **Tính $s = k^{(-1)} \times (H(m) + d \times r) \bmod n = 1^{(-1)} \times (4 + 3 \times 4) \bmod 7 = 1 \times (4 + 12) \bmod 7 = 16 \bmod 7 = 2$**
- **Chữ ký ECDSA: $(r, s) = (4, 2)$**

Xác minh:

1. **Tính $w = s^{(-1)} \bmod n = 2^{(-1)} \bmod 7 \rightarrow 2 \times 4 = 8 \equiv 1 \bmod 7$, nên $2^{(-1)} = 4$**
2. **Tính $u_1 = H(m) \times w \bmod n = 4 \times 4 \bmod 7 = 16 \bmod 7 = 2$**
3. **Tính $u_2 = r \times w \bmod n = 4 \times 4 \bmod 7 = 16 \bmod 7 = 2$**
4. **Tính $R' = u_1 \times G + u_2 \times Q = 2 \times (4, 7) + 2 \times (9, 6)$**
 - **$2 \times (4, 7) = (7, 9)$**
 - **Tính $2 \times (9, 6)$:**
 - **Độ dốc $\lambda = (3x^2 + 2)/(2y) \bmod 11$, với $x = 9, y = 6$:**
 - **$\lambda = (3 \times 81 + 2)/(2 \times 6) \bmod 11 = 245/12 \bmod 11$**
 - **$12 \bmod 11 = 1, 1^{(-1)} = 1$**
 - **$\lambda = 245 \bmod 11 = 2$**
 - **$x_3 = \lambda^2 - 2x_1 \bmod 11 = 2^2 - 2 \times 9 \bmod 11 = 4 - 18 \bmod 11 = 8$**
 - **$y_3 = \lambda(x_1 - x_3) - y_1 \bmod 11 = 2 \times (9 - 8) - 6 \bmod 11 = 2 - 6 = 7$**
 - **$2 \times (9, 6) = (8, 7)$**

- **Tính $(7, 9) + (8, 7)$:**
 - **Độ dốc $\lambda = (7 - 9)/(8 - 7) \bmod 11 = -2/1 \bmod 11 = 9$**
 - **$x_3 = \lambda^2 - x_1 - x_2 \bmod 11 = 9^2 - 7 - 8 \bmod 11 = 81 - 15 \bmod 11 = 1$**
 - **$y_3 = \lambda(x_1 - x_3) - y_1 \bmod 11 = 9 \times (7 - 1) - 9 \bmod 11 = 54 - 9 = 45 \bmod 11 = 2$**
 - **$R' = (1, 2)$**
- 5. Tính $v = x_{R'} \bmod n = 1 \bmod 7 = 1$**
- **Kết quả xác minh: $v \neq r$ ($1 \neq 4$), thất bại. Cần kiểm tra lại hoặc chọn k khác.**

Phần code:

```
from datetime import datetime
```

```
def gcd(a, b):
```

```
    while b:
```

```
        a, b = b, a % b
```

```
    return a
```

```
def mod_inverse(a, m):
```

```
    m0, x0, x1 = m, 0, 1
```

```
    while a > 1:
```

```
        q = a // m
```

```
        a, m = m, a % m
```

```
        x0, x1 = x1 - q * x0, x0
```

```
    if m != 1:
```

```
        return None
```

```
    return x1 % m0 if x1 > 0 else x1 + m0
```

```
def hash_message(m, p):
```

```
    return m % p
```

```
def elgamal_sign_verify(p, g, x, k, m):
```

```
    if gcd(k, p-1) != 1:
```

```
        return False, f'k = {k} không hợp lệ vì gcd(k, p-1) != 1'
```

```
    y = pow(g, x, p)
```

```
    r = pow(g, k, p)
```

```
    k_inv = mod_inverse(k, p-1)
```

```
    if k_inv is None:
```

```
        return False, f'Không tìm được nghịch đảo của k = {k} modulo {p-1}'
```

```
    s = (hash_message(m, p) - x * r) * k_inv % (p-1)
```

```
    print(f'ElGamal - Chữ ký: (r, s) = ({r}, {s})')
```

```
    left = pow(g, hash_message(m, p), p)
```

```
    right = (pow(y, r, p) * pow(r, s, p)) % p
```

```
    return left == right, None
```

```
def schnorr_sign_verify(p, q, a, s, k, m):
```

```
    v = pow(a, -s, p)
```

```
    x = pow(a, k, p)
```

```
    e = (m + x) % q
```

```
    y = (k + s * e) % q
```



```

print(f"Schnorr - Chữ ký: (e, y) = ({e}, {y})")

x_prime = (pow(a, y, p) * pow(v, e, p)) % p

e_prime = (m + x_prime) % q

return e == e_prime, None


def dsa_sign_verify(p, q, g, x, k, m):

    if gcd(k, q) != 1:

        return False, f"k = {k} không hợp lệ vì gcd(k, q) != 1"

    y = pow(g, x, p)

    r = (pow(g, k, p) % p) % q

    k_inv = mod_inverse(k, q)

    if k_inv is None:

        return False, f"Không tìm được nghịch đảo của k = {k} modulo {q}"

    s = (k_inv * (hash_message(m, p) + x * r)) % q

    print(f"DSA - Chữ ký: (r, s) = ({r}, {s})")

    w = mod_inverse(s, q)

    if w is None:

        return False, f"Không tìm được nghịch đảo của s = {s} modulo {q}"

    u1 = (hash_message(m, p) * w) % q

    u2 = (r * w) % q

    v = ((pow(g, u1, p) * pow(y, u2, p)) % p) % q

    return v == r, None

```

```

def main():

    current_date = datetime.now().strftime("Ngày: %d-%m-%Y")

    print(f"{current_date}\n")


    p = 11

    q = 5

    g = 3

    x = 2

    m = 4

    k = 2


    print("Kiểm tra ElGamal:")

    result, error = elgamal_sign_verify(p, g, x, k, m)

    if error:

        print(error)

    elif result:

        print("ElGamal - Xác minh thành công!")

    else:

        print("ElGamal - Xác minh thất bại!")


    print("\nKiểm tra Schnorr:")

    result, error = schnorr_sign_verify(p, q, g, x, k, m)

    if error:

```

```
        print(error)

    elif result:

        print("Schnorr - Xác minh thành công!")

    else:

        print("Schnorr - Xác minh thất bại!")


print("\nKiểm tra DSA:")

result, error = dsa_sign_verify(p, q, g, x, k, m)

if error:

    print(error)

elif result:

    print("DSA - Xác minh thành công!")

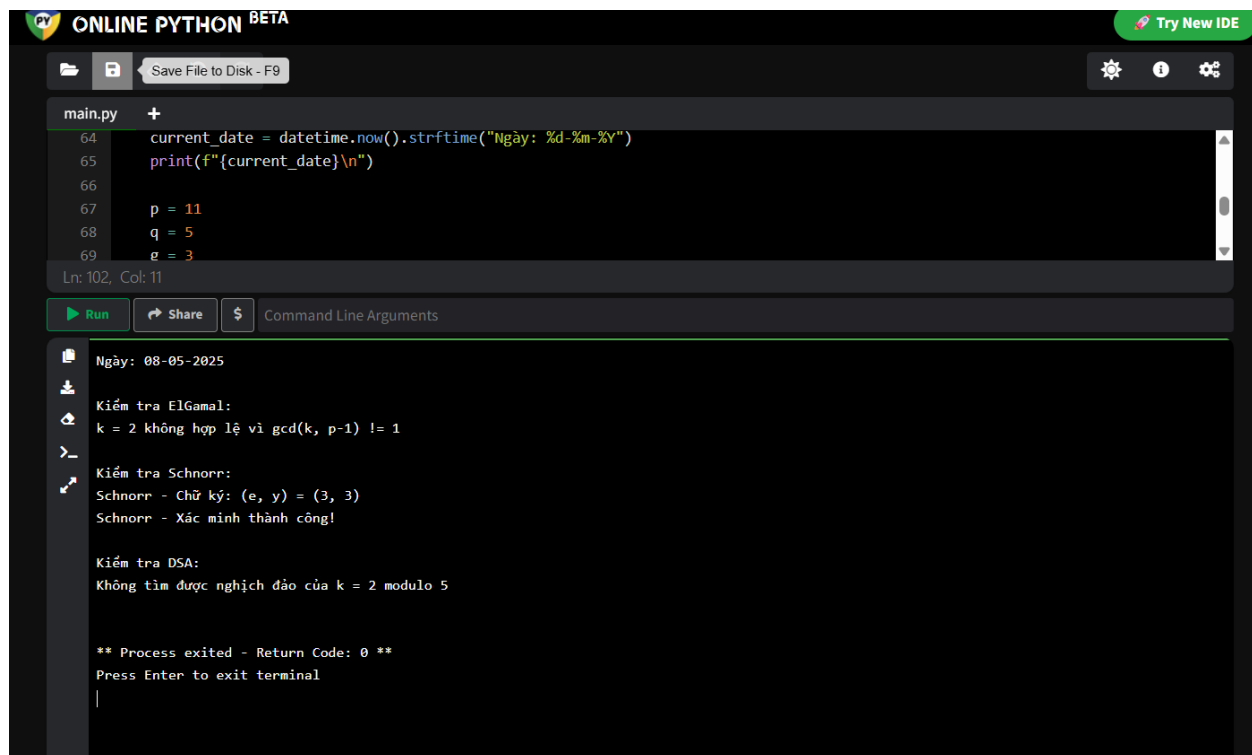
else:

    print("DSA - Xác minh thất bại!")


if __name__ == "__main__":

    main()
```

Kết quả thực thi chương trình:



The screenshot shows the ONLINE PYTHON BETA IDE interface. The top bar includes the logo, the text "ONLINE PYTHON BETA", and a "Try New IDE" button. Below the bar is a toolbar with icons for file operations and settings. The main editor area displays a Python script in a file named "main.py". The script contains the following code:

```
64 current_date = datetime.now().strftime("Ngày: %d-%m-%Y")
65 print(f"{current_date}\n")
66
67 p = 11
68 q = 5
69 e = 3
```

The status bar at the bottom of the editor indicates "Ln: 102, Col: 11". Below the editor is a toolbar with "Run", "Share", and "Command Line Arguments" buttons. The output area at the bottom shows the execution results:

```
Ngày: 08-05-2025

Kiểm tra ElGamal:
k = 2 không hợp lệ vì gcd(k, p-1) != 1

Kiểm tra Schnorr:
Schnorr - Chữ ký: (e, y) = (3, 3)
Schnorr - Xác minh thành công!

Kiểm tra DSA:
Không tìm được nghịch đảo của k = 2 modulo 5

** Process exited - Return Code: 0 **
Press Enter to exit terminal
```

Bài 6: băm chuỗi số "999" bằng thuật toán SHA-256

1. Chuẩn bị dữ liệu:

- Đầu vào: "999" → Mã hóa ASCII: [0x39, 0x39, 0x39] (3 byte).

2. Padding dữ liệu:

- Thêm byte 0x80 → [0x39, 0x39, 0x39, 0x80].
- Thêm 52 byte 0 để đạt 56 byte.
- Thêm 8 byte độ dài (24 bit = 0x18) → [0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x18].
- Kết quả: Khối 64 byte (512 bit).

3. Khởi tạo trạng thái:

- 8 giá trị trạng thái ban đầu (H0-H7):

$H0 = 0x6a09e667$, $H1 = 0xbb67ae85$, $H2 = 0x3c6ef372$, $H3 = 0xa54ff53a$,

$H4 = 0x510e527f$, $H5 = 0x9b05688c$, $H6 = 0x1f83d9ab$, $H7 = 0x5be0cd19$

4. Tạo lịch trình:

- Chia khối thành 16 word (4 byte/word):

- $W0 = 0x39393980$, $W1-W14 = 0x00000000$, $W15 = 0x00000018$.
- Tính 48 word tiếp theo ($W16-W63$) bằng công thức:

$$W[i] = \sigma_1(W[i-2]) + W[i-7] + \sigma_0(W[i-15]) + W[i-16]$$

5.Nén khối:

- Xử lý khối qua 64 vòng lặp, dùng 8 biến (A, B, C, D, E, F, G, H) khởi tạo từ $H0-H7$.
- Mỗi vòng:
 - Tính $T1 = H + \Sigma_1(E) + Ch(E,F,G) + K[i] + W[i]$
 - Tính $T2 = \Sigma_0(A) + Maj(A,B,C)$
 - Cập nhật: $H=G$, $G=F$, $F=E$, $E=D+T1$, $D=C$, $C=B$, $B=A$, $A=T1+T2$
- Sau 64 vòng, cập nhật: $H0 = H0+A$, ..., $H7 = H7+H$.

6.Tạo giá trị băm:

- Gộp $H0-H7$ thành chuỗi 32 byte (256 bit).
- Kết quả (hex):
83cf8b609de60036a8277bd0e96135751bbc07eb234256d4b65b893360651bf2
- Giá trị băm : g8+LYJ3mADaoJ3vQ6WE1dRu8B+sjQlbUtluJM2BIG/I=

Phần code:

```
import hashlib
```

```
import base64
```

```
def hash_sha256(data):
```

```
    """Băm dữ liệu bằng SHA-256 và trả về giá trị băm"""
```

```
    # Chuyển dữ liệu thành bytes (ASCII)
```

```
    data_bytes = data.encode('ascii')
```

```
    # Tạo đối tượng SHA-256 và băm
```

```
    sha256 = hashlib.sha256()
```

```
    sha256.update(data_bytes)
```

```
    # Lấy giá trị băm
```

```
return sha256.digest()
```

```
def main():
```

```
    # Nhập chuỗi số từ bàn phím
```

```
    plaintext = input("Nhập chuỗi số cần băm: ")
```

```
    # Kiểm tra input (chỉ chấp nhận số)
```

```
    if not plaintext.isdigit():
```

```
        print("Vui lòng chỉ nhập các chữ số (0-9)!")
```

```
        return
```

```
    try:
```

```
        # Băm bằng SHA-256
```

```
        hash_value = hash_sha256(plaintext)
```

```
        # Xuất kết quả
```

```
        print("Chuỗi gốc:", plaintext)
```

```
        print("Giá trị băm (hex):", hash_value.hex())
```

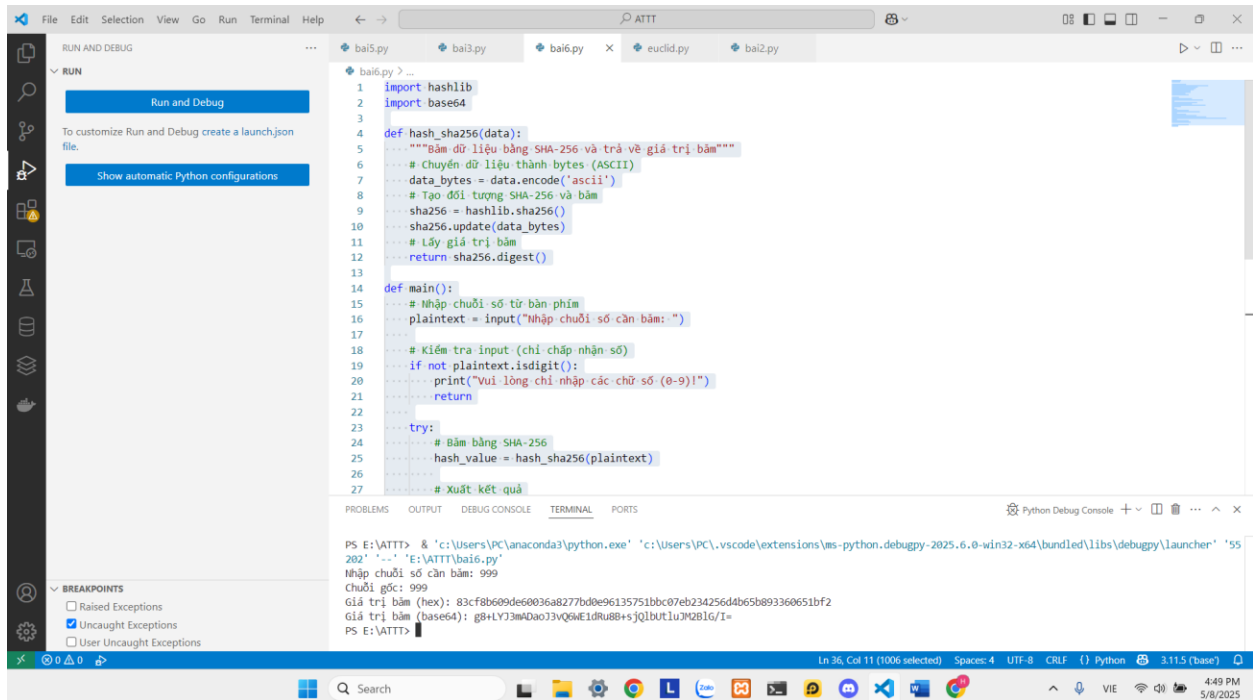
```
        print("Giá trị băm (base64):", base64.b64encode(hash_value).decode('ascii'))
```

```
    except Exception as e:
```

```
        print("Lỗi:", str(e))
```

```
if __name__ == "__main__":  
  
    main()
```

Kết quả thực thi chương trình:



The screenshot displays a Python IDE with a file named `bai6.py` open. The code implements a SHA-256 hashing function and a main routine that prompts the user for a string to hash. The IDE interface includes a sidebar with icons for Explorer, Search, Run and Debug, and Breakpoints. The Run and Debug panel on the left shows the 'Run and Debug' button and a link to 'Show automatic Python configurations'. The main editor area shows the following Python code:

```
1 import hashlib  
2 import base64  
3  
4 def hash_sha256(data):  
5     """Băm dữ liệu bằng SHA-256 và trả về giá trị băm"""  
6     # Chuyển dữ liệu thành bytes (ASCII)  
7     data_bytes = data.encode('ascii')  
8     # Tạo đối tượng SHA-256 và băm  
9     sha256 = hashlib.sha256()  
10    sha256.update(data_bytes)  
11    # Lấy giá trị băm  
12    return sha256.digest()  
13  
14 def main():  
15     # Nhập chuỗi số từ bàn phím  
16     plaintext = input("Nhập chuỗi số cần băm: ")  
17  
18     # Kiểm tra input (chỉ chấp nhận số)  
19     if not plaintext.isdigit():  
20         print("Vui lòng chỉ nhập các chữ số (0-9)!")  
21         return  
22  
23     try:  
24         # Băm bằng SHA-256  
25         hash_value = hash_sha256(plaintext)  
26  
27         # Xuất kết quả
```

The bottom panel shows the terminal output of the program's execution:

```
PS E:\ATT7> & 'c:\Users\PC\anaconda3\python.exe' 'c:\Users\PC\.vscode\extensions\ms-python.debugpy-2025.6.0-win32-x64\bundled\libs\debugpy\launcher' '55'  
202' '-' 'E:\ATT7\bai6.py'  
Nhập chuỗi số cần băm: 999  
Chuỗi gốc: 999  
Giá trị băm (hex): 83cf8b609de0036a8277bd0e96135751bbc07eb234256d4b65b893360651bf2  
Giá trị băm (base64): g8+LYJ3m4QaoJ3vQ6wE1dRu8B+sJQ1but1u7M2B1G/I=  
PS E:\ATT7>
```

The status bar at the bottom indicates the current line is 36, column 11, with 1006 characters selected. The encoding is UTF-8, and the interpreter is Python 3.11.5 (base).

