Họ và tên: Trần Hữu Đức
MSSV: 23520321

1. **Code your own chaotic-based stream cipher:**

- Key generation use chaotic map:

  Sử dụng Basin chaotic map với công thức $x_{n+1} = \mu \cdot x_n \cdot (1 - x_n^2)$

  + Source code:

```
def basin_map_key_hex(seed, length, mu=3):
    if not (0 < seed < 1):
        raise ValueError("Seed phải là số thực trong khoảng (0,1)")

    x = seed
    hex_key = ""
    for _ in range(length):
        x = mu * x * (1 - x * x)

        frac = abs(x - int(x))

        key_byte = int(frac * 256) % 256

        hex_key += "{:02x}".format(key_byte)
    return hex_key

def main():
    try:
        seed_input = input("Nhập seed (số thực trong khoảng (0,1)): ")
        seed = float(seed_input)
        length_input = input("Nhập độ dài chuỗi khóa (số byte): ")
        length = int(length_input)
    except ValueError:
        print("Sai định dạng đầu vào! Seed phải là số thực trong khoảng (0,1)
và độ dài là số nguyên.")
        return

    try:
        key_hex = basin_map_key_hex(seed, length)
        print("Chuỗi khóa dạng hex (Basin chaotic map):", key_hex)
    except Exception as e:
        print("Lỗi:", e)
```

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

- Test the Key (randomly and sensitive to input seeds)

```
PS D:\NT219\week03_code> python .\key_generation_BasinChaoticMap.py
Nhập seed (số thực trong khoảng (0,1)): 0.08062005
Nhập độ dài chuỗi khóa (số byte): 25
Chuỗi khóa dạng hex (Basin chaotic map): 3dad18ac1ab80a3fb312791ab8093eaf16982722fe0b226500
PS D:\NT219\week03_code> python .\key_generation_BasinChaoticMap.py
Nhập seed (số thực trong khoảng (0,1)): 0.08052006
Nhập độ dài chuỗi khóa (số byte): 25
Chuỗi khóa dạng hex (Basin chaotic map): 3dad19ae18a91dd1d0d3cae38f27240f68051f5cf14ed4c6ee
```

  + Với seed = 0.08062005 thì key = 3dad18ac1ab80a3fb312791ab8093eaf16982722fe0b226500
  + Với seed = 0.08062006 thì key = 3dad19ae18a91dd1d0d3cae38f27240f68051f5cf14ed4c6ee
    => đã có sự khác biệt rõ rệt giữa 2 seed gần giống nhau

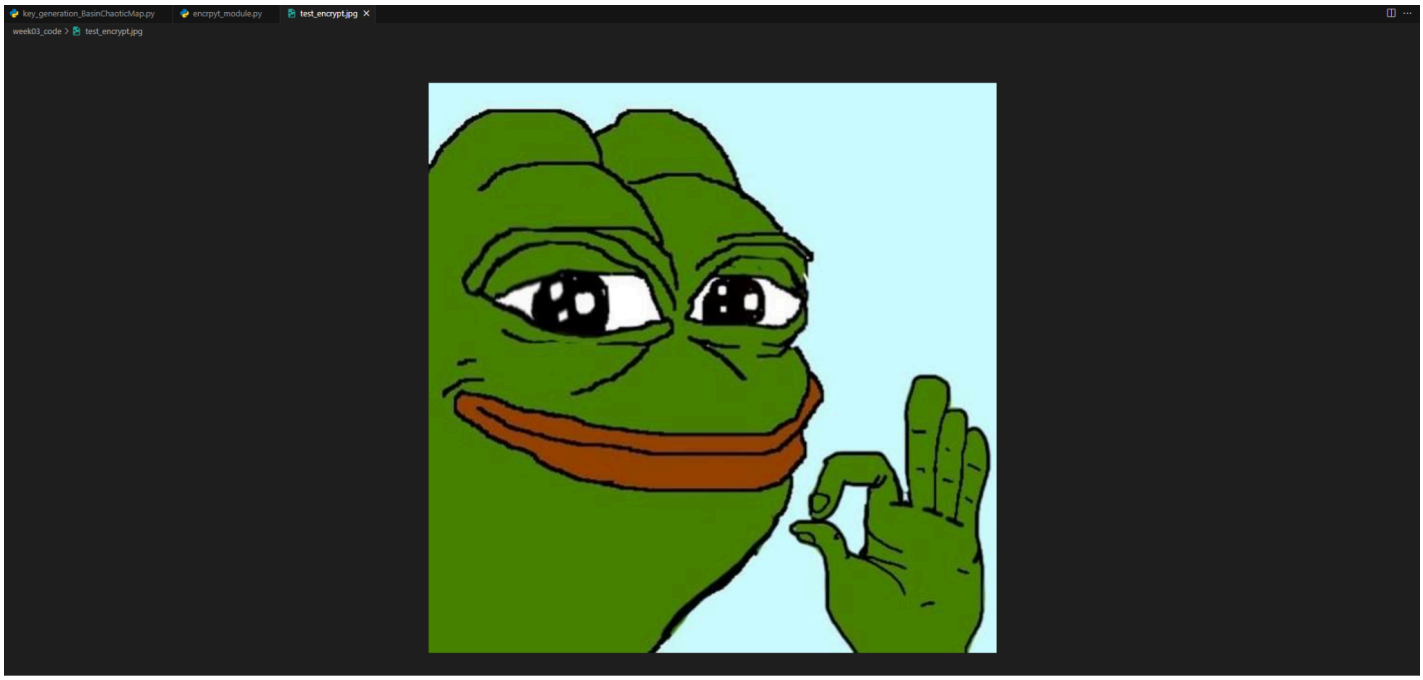
- Encryption Module
  + Source code:

```
import sys
import os

def encrypt_file_with_key(input_file, key_hex):
    # Chuyển chuỗi key hex thành bytes
    try:
        key_bytes = bytes.fromhex(key_hex)
    except ValueError:
        raise ValueError("Key không hợp lệ. Vui lòng nhập chuỗi hex hợp lệ.")

    if len(key_bytes) == 0:
        raise ValueError("Key không được rỗng.")

    # Đọc dữ liệu file ở dạng nhị phân
    with open(input_file, "rb") as f_in:
        data = f_in.read()

    # Thực hiện phép XOR: nếu key ngắn hơn dữ liệu, lặp lại key (cyclic)
    encrypted_data = bytes([data[i] ^ key_bytes[i % len(key_bytes)] for i in
range(len(data))])

    return encrypted_data
```

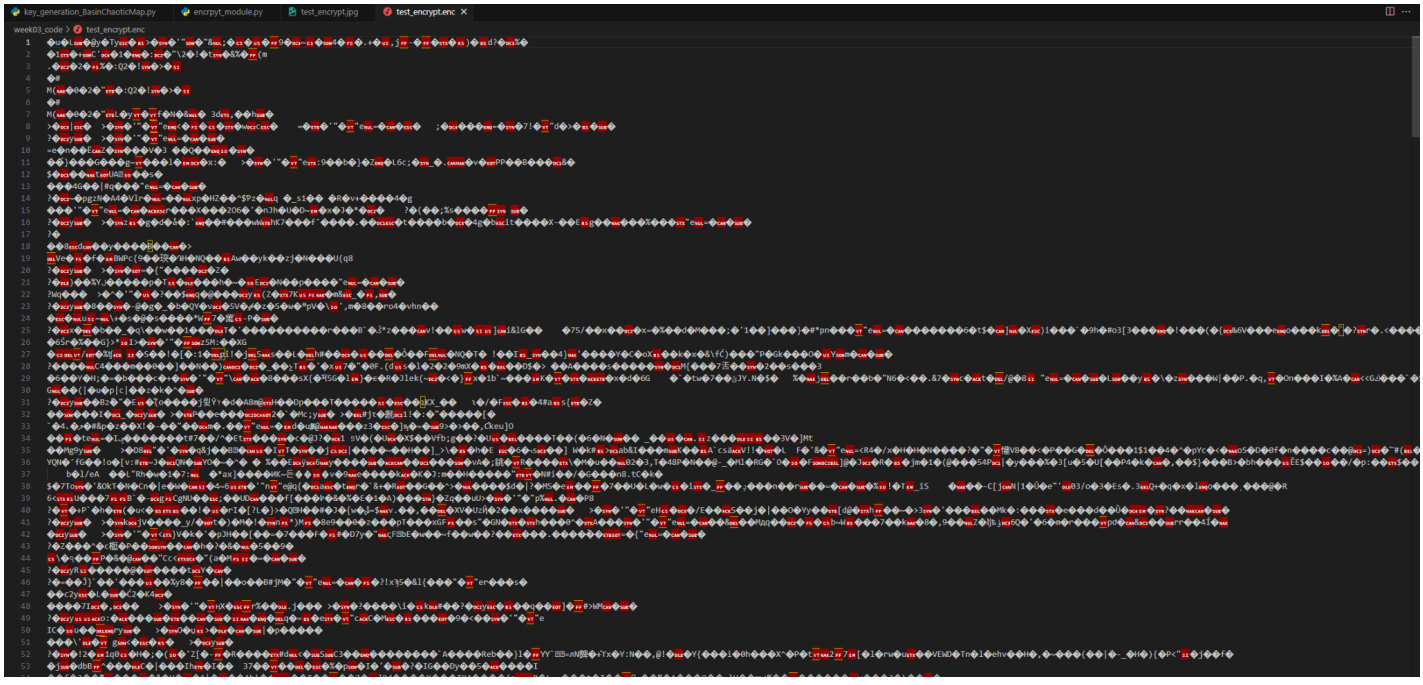
```python
def main():
    if len(sys.argv) != 3:
        sys.exit(1)

    input_file = sys.argv[1]
    key_hex = sys.argv[2]

    try:
        encrypted_data = encrypt_file_with_key(input_file, key_hex)

        output_file = input_file
        with open(output_file, "wb") as f_out:
            f_out.write(encrypted_data)

        print(f"Đã mã hóa file '{input_file}' thành '{output_file}' sử dụng key đã
cho.")
    except Exception as e:
        print("Lỗi:", e)

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

```
PS D:\NT219\week03_code> python .\encrpyt_module.py .\test_encrypt.jpg .\test_encrypt.enc 3dad18ac1ab80a3fb312791ab8093eaf16982722fe0b226500
Đã mã hóa file '.\test_encrypt.jpg' thành '.\test_encrypt.enc' sử dụng key đã cho.
```

+ test_encrypt.jpg



+ test_encrpyt.enc

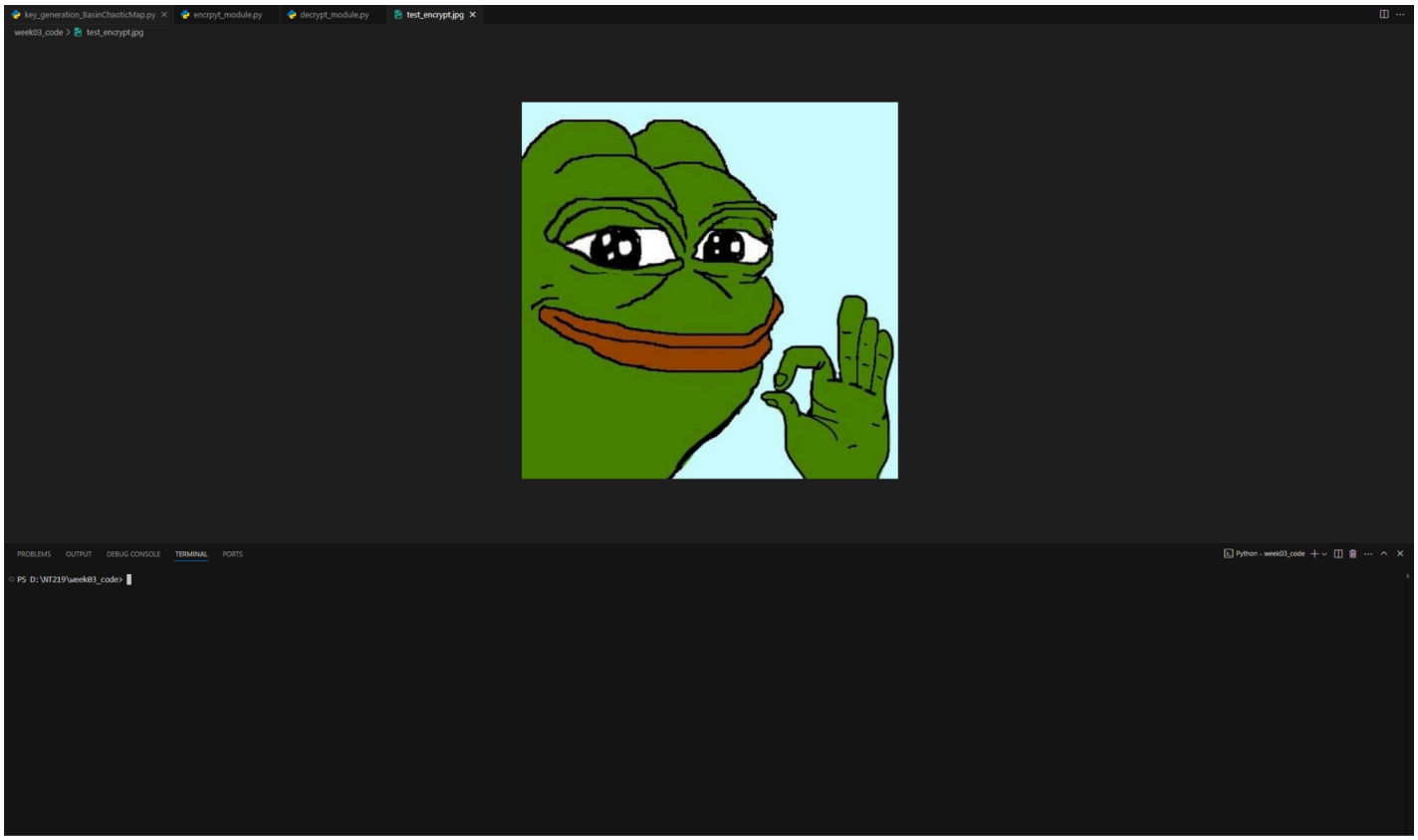


- Decryption Module

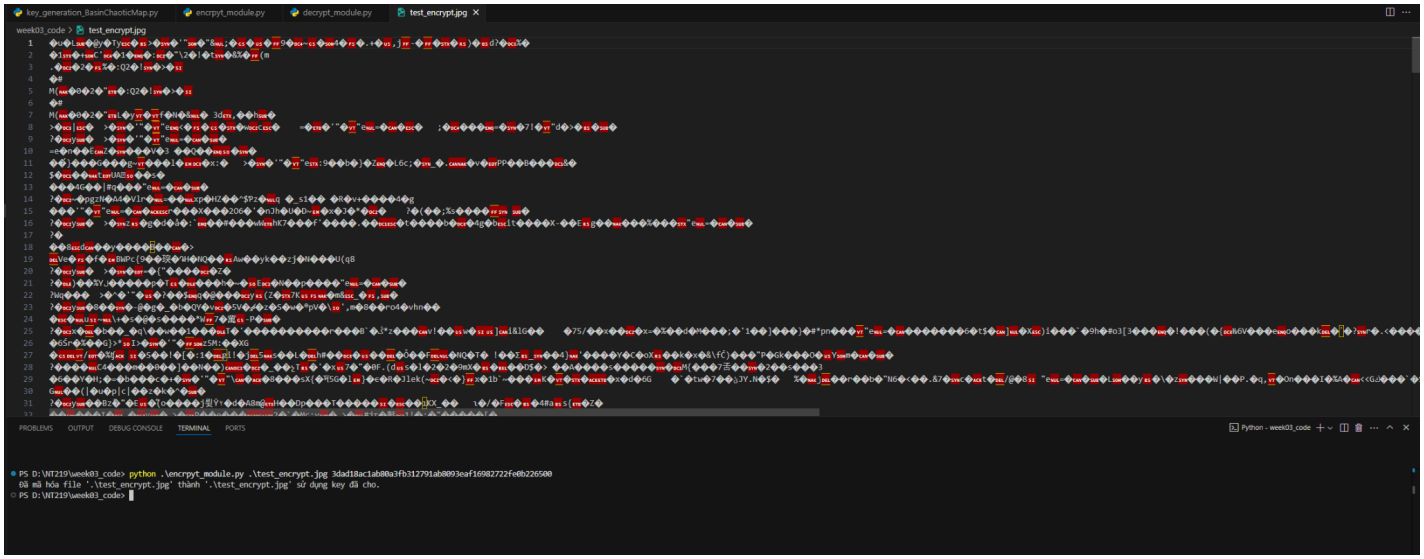
```
import sys
import os


def decrypt_file_with_key(input_file, key_hex):
```

```
    # Chuyển chuỗi key hex thành bytes
    try:
        key_bytes = bytes.fromhex(key_hex)
    except ValueError:
        raise ValueError("Key không hợp lệ. Vui lòng nhập chuỗi hex hợp lệ.")

    if len(key_bytes) == 0:
        raise ValueError("Key không được rỗng.")

    # Đọc dữ liệu file mã hóa ở dạng nhị phân
    with open(input_file, "rb") as f_in:
        encrypted_data = f_in.read()

    # Thực hiện phép XOR: nếu key ngắn hơn dữ liệu, lặp lại key (cyclic)
    decrypted_data = bytes([encrypted_data[i] ^ key_bytes[i % len(key_bytes)] for i
in range(len(encrypted_data))])

    return decrypted_data

def main():
    if len(sys.argv) != 3:
        sys.exit(1)

    input_file = sys.argv[1]
    key_hex = sys.argv[2]

    try:
        decrypted_data = decrypt_file_with_key(input_file, key_hex)

        output_file = input_file
        with open(output_file, "wb") as f_out:
            f_out.write(decrypted_data)

        print(f"Đã giải mã file '{input_file}' thành '{output_file}' sử dụng key đã
cho.")
    except Exception as e:
        print("Lỗi:", e)

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

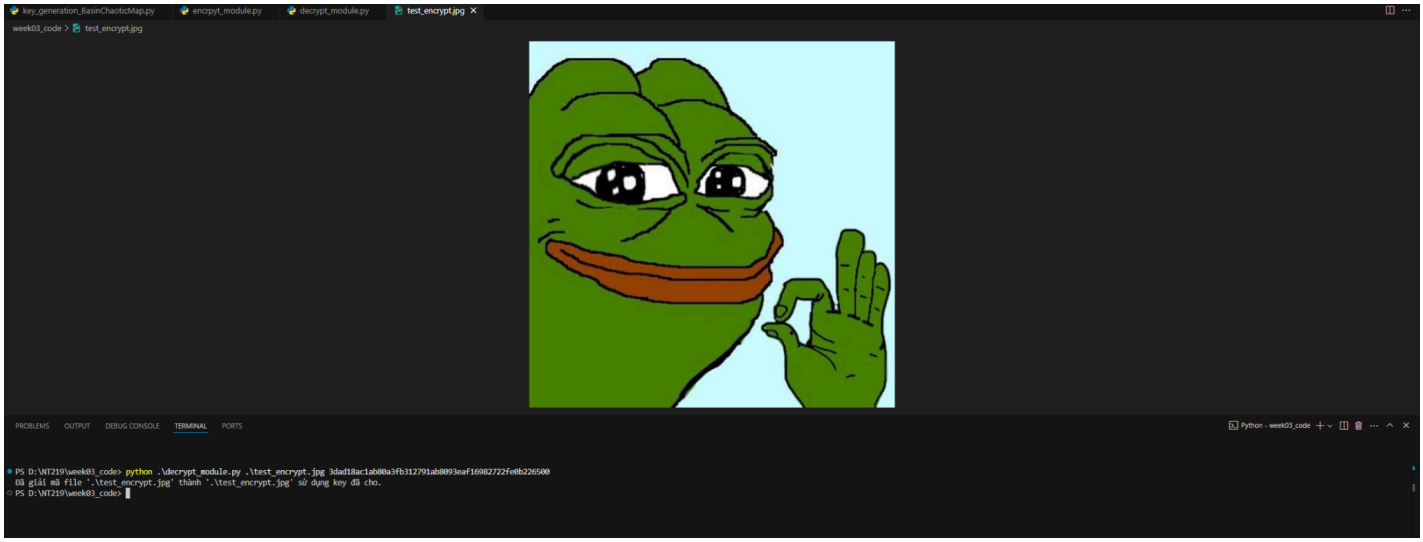
- Test Encryption/Decryption for any input files (pdf, mp3, image, etc.)
    + File img mẫu:



    + FIle sau khi encrypt:



    + File sau khi decrypt:



2. Cryptanalysis Stream Cipher

- Does your Stream Cipher is secure of one use the same key to encrypt many files?

Việc sử dụng cùng một khóa để mã hóa nhiều file trong một stream cipher (đặc biệt là khi áp dụng phép XOR như ví dụ của chúng ta) không được coi là an toàn. Vì:

+ Nếu cùng một khóa được dùng để mã hóa hai file khác nhau, kẻ tấn công có thể lấy XOR của hai ciphertext và loại bỏ hiệu ứng của khóa, từ đó có thể thu được mối quan hệ giữa hai plaintext ban đầu.

+ Khi sử dụng lại khóa, nếu một số phần của plaintext được biết hoặc dự đoán, kẻ tấn công có thể sử dụng thông tin đó để phục hồi các phần khác của dữ liệu.

+ Để một stream cipher đạt được mức an toàn tối đa, mỗi khóa chỉ nên được sử dụng một lần duy nhất. Việc tái sử dụng khóa dẫn đến các lỗ hổng bảo mật nghiêm trọng.