**Nhập môn An toàn thông tin - Assignment 2**

**Họ và tên:** Lê Hữu Đức Anh

**Mssv:** 20215304

1. **Giải thích quá trình**
2. **Lý thuyết**

Thuật toán mã hóa Trivium stream cipher thực hiện dựa trên việc tạo ra các keystream từ các giá trị Key( 80 bit) và IV( 80 bit) ban đầu, được truyền vào 2 bộ LFSR 1 và 2. Sau đó nó sẽ chạy các 4 vòng khởi tạo trước khi bắt đầu tạo ra 1 keystream sử dụng cho việc mã hóa.

A screenshot of a computer code

Description automatically generated

A screenshot of a math problem

Description automatically generatedCác giá trị **t1**, **t2**, **t3** là kết quả của phép **XOR** các bit như hình dưới, và kết quả của phép **XOR** **t1**, **t2**, **t3** sẽ là giá trị của 1 đơn vị thuộc **keystream**. Sau đó các giá trị **t1**, **t2**, **t3** sẽ được tính toán lại bằng phép **&** và **XOR** và được truyền vào vị trí đầu của các **Register**

A diagram of a machine

Description automatically generated

*Hình mô phỏng quá trình tính toán*

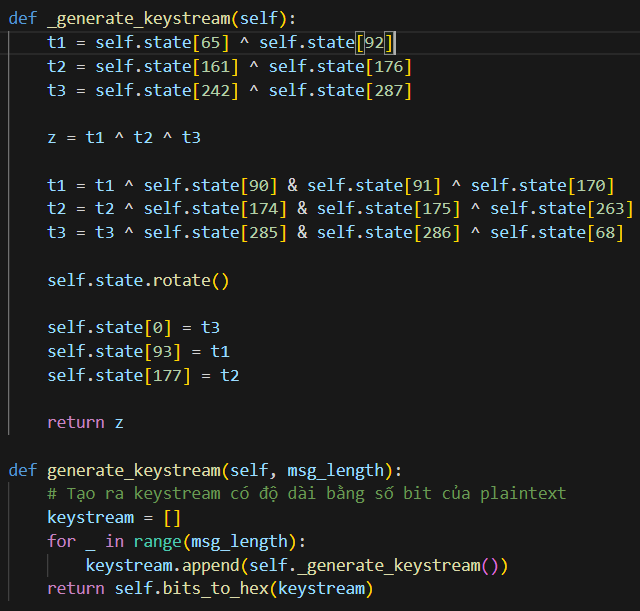
**KeyStream** thu được qua quá trình tạo ngẫu nhiên bằng 3 đơn vị **LFSR** sẽ được sử dụng để mã hóa các bit được đọc từ các file. Kết quả thu được dưới dạng các bit sẽ được lưu vào file txt kèm với **KeyStream** ở quá trình **Random KeyStream Generating**

1. **Cài đặt chương trình**

Tạo class **Trivium** với hàm khởi tạo ban đầu cho **Key** và **IV**, sẵn sàng cho quá trình tạo **KeyStream**

1. class Trivium:
2. def \_\_init\_\_(self, key, iv):
3. self.state = None
4. self.key = key
5. self.iv = iv
7. # Khởi tạo trạng thái ban đầu với key và iv
8. init\_state = list(map(int, list(self.key))) + [0] \* 13
9. init\_state += list(map(int, list(self.iv))) + [0] \* 4
10. init\_state += [0] \* 108 + [1, 1, 1]
11. self.state = deque(init\_state)
12. # Khởi tạo Trivium với 4 vòng lặp
13. for \_ in range(4 \* 288):
14. self.\_generate\_keystream()

Tạo các hàm tạo **KeyStream**: **\_generate\_keystream()** để tạo 1 giá trị ở mỗi vòng của bộ tạo số và **hàm generate\_keystream()** để tạo ra 1 chuỗi độ dài bằng số bit của file để mã hóa file.



Tạo ra các hàm mã hóa và giải mã, cùng với các hàm tiện ích khác.

A computer screen shot of text

Description automatically generated

***Hàm encypt() và decrypt()***

 @staticmethod

    def hex\_to\_bytes(hex\_string):

        byte\_list = []

        for i in range(0, len(hex\_string), 2):

            byte = int(hex\_string[i:i+2], 16)

            byte\_list.append(byte)

        return byte\_list

    @staticmethod

    def bits\_to\_hex(bits):

        hex\_string = ""

        for i in range(0, len(bits), 8):

            byte\_sum = sum([bits[i + j] << j for j in range(8)])

            hex\_string += "%02X" % byte\_sum

        return hex\_string

***Hàm hex\_to\_bytes() và bits\_to\_hex(***

Cuối cùng ta sẽ tạo hàm **encrypt()** nhận 2 đầu vào là đường dẫn đến file **Input** và **Output**. **Key** và **IV** sẽ được tạo ngẫu nhiên với độ dài 80-bit. **Plaintext** sẽ được đọc theo bytes từ file gốc. Sau đó dựa trên các hàm kể trên để mã hóa và in ra dữ liệu đã mã hóa vào file **Output** dưới dạng file txt chứa ***binary*** và ***hex.***

def encrypt(input\_file, output\_file):

    key = generate\_random\_bits(80)

    iv = generate\_random\_bits(80)

    plaintext = read\_bytes\_from\_file(input\_file)

    trivium = Trivium(key, iv)

    keystream = trivium.generate\_keystream(len(plaintext) \* 8)

    print("IV (in hex):", trivium.bits\_to\_hex(iv))

    print("Key (in hex):", trivium.bits\_to\_hex(key))

    print("Keystream (in hex):", keystream)

    ciphertext = trivium.encrypt(plaintext, keystream)

    print("Cipher (in hex):", ciphertext.hex())

    with open(output\_file + "\_binary.txt", "wb") as file:

        file.write(iv)

        file.write(ciphertext)

    with open(output\_file + "\_hex.txt", "w") as file:

        file.write(iv.hex()+'\n')

        file.write(ciphertext.hex())

    # Lặp qua các tệp trong thư mục đầu vào

   for filename in os.listdir(input\_directory):

        input\_file = os.path.join(input\_directory, filename)

        output\_file = os.path.join(output\_directory, filename +

                                   "\_encrypted")

        start\_time = time.time()

        encrypt\_no\_print(input\_file, output\_file)

        end\_time = time.time()

        file\_size = os.path.getsize(input\_file)

        print("File:", filename, "- Size:", file\_size, "bytes",

              "- Time:", round(end\_time - start\_time, 2), "seconds")

1. **Kết quả chạy chương trình**

Sau khi mã hóa hết các file trong TestData, em thu được kết quả là các bản mã, quá trình mã hóa có thời gian chạy tỉ lệ thuận với độ lớn của file

**A screenshot of a computer

Description automatically generated**

***Các file được mã hóa***

**A screenshot of a computer code

Description automatically generatedA close up of numbers

Description automatically generated**

***Kích thước và thòi gian chạy***