Báo cáo: Cài đặt hệ mã dòng Trivium

Họ tên: Trần Ngọc Bảo
MSSV: 20215529

1. Cách cài đặt:

1.1 Cài đặt hệ mã dòng Trivium:

Cài đặt hệ mã dòng Trivium như trong bài giảng, cụ thể như sau:

```
from collections import deque
from itertools import repeat

# Convert strings of hex to strings of bytes and back, little-endian style
_allbytes = dict([("%02X" % i, i) for i in range(256)])

def hex_to_bytes(s):
    return [_allbytes[s[i:i+2].upper()] for i in range(0, len(s), 2)]

def hex_to_bits(s):
    return [(b >> i) & 1 for b in hex_to_bytes(s)
            for i in range(8)]

def bits_to_hex(b):
    return "".join(["%02X" % sum([b[i + j] << j for j in range(8)])
                    for i in range(0, len(b), 8)])
```

Thư viện trong module trivium.py và các hàm chuyển đổi lẫn nhau giữa string, hex, bit

```python
class Trivium:
    def __init__(self, key, iv):
        """in the beginning we need to transform the key as well as the IV.
        Afterwards we initialize the state."""
        self.state = None
        self.counter = 0
        self.key = key
        self.iv = iv

        # Initialize state
        # len 93
        init_list = list(map(int, list(self.key)))
        init_list += list(repeat(0, 13))
        # len 84
        init_list += list(map(int, list(self.iv)))
        init_list += list(repeat(0, 4))
        # len 111
        init_list += list(repeat(0, 108))
        init_list += list([1, 1, 1])
        self.state = deque(init_list)

        # Do 4 full cycles, drop output
        for i in range(4*288):
            self._gen_keystream()
```

Định nghĩa lớp Trivium và khởi tạo các thanh ghi

```
def encrypt(self, message):
    next_key_bit = self.keystream().__next__
    cipher = deque([])
    for byte in bytearray(message, "utf8"):
        # Key for each byte
        key = 0
        for i in range(0, 8):
            k = next_key_bit()
            key = key | (k << i)

        c = [int(i, 2) for i in bin(byte ^ key)[2:].zfill(8)]

        # Little Endian
        cipher.extendleft(c[4:])
        cipher.extendleft(c[:4])

    return list(cipher)

def decrypt(self, cipher):
    next_key_bit = self.keystream().__next__

    cipher = deque(cipher)
    plain = deque([])

    for i in range(0, int((len(cipher) / 8))):
        temp = []
        for j in range(0, 8):
            temp.append(cipher.pop())

        # cipher
        c_list = []
        c_list[:4] = temp[4:]
        c_list[4:] = temp[:4]
        c = int("".join(str(i) for i in c_list), 2)

        # key
        key = 0
        for j in range(0, 8):
            k = next_key_bit()
            key = key | (k << j)

        # plain text
        plain.extendleft([c ^ key])

    return ''.join(chr(i) for i in list(plain)[::-1])
```

Hàm mã hóa và giải mã của lớp Trivium

```
def keystream(self):
    """output keystream
    only use this when you know what you are doing!!"""
    while self.counter < 2**64:
        self.counter += 1
        yield self._gen_keystream()

def _gen_keystream(self):
    """this method generates triviums keystream"""

    t_1 = self.state[65] ^ self.state[92]
    t_2 = self.state[161] ^ self.state[176]
    t_3 = self.state[242] ^ self.state[287]

    out = t_1 ^ t_2 ^ t_3

    a_1 = self.state[90] & self.state[91]
    a_2 = self.state[174] & self.state[175]
    a_3 = self.state[285] & self.state[286]

    s_1 = a_1 ^ self.state[170] ^ t_1
    s_2 = a_2 ^ self.state[263] ^ t_2
    s_3 = a_3 ^ self.state[68] ^ t_3

    self.state.rotate(1)

    self.state[0] = s_3
    self.state[93] = s_1
    self.state[177] = s_2

    return out
```

Hàm sinh dòng khóa của lớp Trivium

1.2. Xử lý dữ liệu:

Mã hóa dữ liệu trong thư mục TestData (Chỉ chứa file) và lưu trong thư mục EncryptData, đồng thời giải mã dữ liệu từ thư mục EncryptData và lưu trong thư mục DecryptData, cụ thể như sau:

```python
from trivium import Trivium
from trivium import hex_to_bits, bits_to_hex
import os
import random
import time

def randomIV():
    result = ""
    for i in range(20):
        result += hex[random.randint(0, 15)]
    return result
```

Thư viện trong module main.py và hàm sinh IV ngẫu nhiên

```python
def encrypt(file):
    rIV = randomIV()
    IV = hex_to_bits(rIV)[::-1]

    # Read plaintext file
    f = open(testData + file, "rb")
    lines = f.readlines()
    content = ""
    for line in lines:
        content += line.decode("latin-1")

    # Create ciphertext and save it
    trivium_encoder = Trivium(KEY, IV)
    cipher = trivium_encoder.encrypt(content)
    f = open(encryptData + file, "w")
    f.write(rIV + bits_to_hex(cipher))
    f.close()

def decrypt(file):
    f = open(encryptData + file, "r")
    content = f.readline()

    IV = hex_to_bits(content[:20])[::-1] # The first 20 hexa numbers are IV

    # Create plaintext and save it
    trivium_decoder = Trivium(KEY, IV)
    f = open(decryptData + file, "w")
    f.write(trivium_decoder.decrypt(hex_to_bits(content[20:])))
    f.close()
```

Hàm mã hóa và giải mã các file dữ liệu

```python
def solution(file):
    startTime = time.time()

    encrypt(file)
    decrypt(file)

    t = time.time() - startTime
    print(file, ":", t, "sec")
    return t

if __name__ == "__main__":
    hex = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'D', 'E', 'F']

    KEY = hex_to_bits("0F62B5085BAE0154A7FA")[::-1]

    testData = "TestData/"
    encryptData = "EncryptData/"
    decryptData = "DecryptData/"

    totalTime = 0.0
    listFile = os.listdir(os.path.expanduser(testData)) # TestData contains only file
    for file in listFile:
        totalTime += solution(file)

    print("TOTAL TIME:", totalTime, "sec")
```

Hàm chính để chạy hệ thống

2. Thời gian chạy:

Có tổng cộng 14 file được mã hóa và giải mã.

Thời gian chạy tổng cộng là: 305 giây ~ 5 phút

Thời gian chạy trung bình mỗi file là: 22s

File có thời gian chạy nhanh nhất: fields.c (0.2368 giây) nặng 11.2 KB

File có thời gian chạy lâu nhất: E.coli (97.5223 giây) nặng 4.6 MB

```
bao@bao-Latitude-3520:~/GitHub/Introduction_to_Cryptography_and_Security/Trivium$ /bin/python3 /home/bao/GitHub/Introduction_to_Cryptography_and_Security/Trivium/main.py
sum : 1.0141806602478027 sec
world192.txt : 55.50696015357971 sec
fields.c : 0.23680448532104492 sec
grammar.lsp : 0.08424901962280273 sec
asyoulik.txt : 2.5818605422973633 sec
plrabn12.txt : 9.940576314926147 sec
cp.html : 0.54270339012146 sec
kennedy.xls : 23.700982570648193 sec
ptt5 : 12.26832127571106 sec
lcet10.txt : 9.373458862304688 sec
alice29.txt : 3.3178937435150146 sec
xargs.1 : 0.12564659118652344 sec
bible.txt : 88.98756766319275 sec
E.coli : 97.52232074737549 sec
TOTAL TIME: 305.20352602005005 sec
bao@bao-Latitude-3520:~/GitHub/Introduction_to_Cryptography_and_Security/Trivium$
```