Nguyễn Trung Kiên_20205090

Mô tả thuật toán

Thuật toán được khởi tạo bằng cách tải khóa 80 bit và IV 80 bit vào trạng thái ban đầu gồm 288 bit và đặt tất cả các bit còn lại thành 0, ngoại trừ s_{286}, s_{287} và s_{288} thành 1. Sau đó, trạng thái được xoay qua 4 chu kỳ đầy đủ, nhưng không tạo ra các bit luồng khóa.

```
# bit 1 -> 93
init_list = list(map(int, list(self.key)))
init_list += list(repeat(0, 13))
# bit 94 -> 177
init_list += list(map(int, list(self.iv)))
init_list += list(repeat(0, 4))
# bit 178 -> 288
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()
```

Thuật toán được khởi tạo bằng cách tải khóa 80 bit và thiết kế được đề xuất chứa trạng thái bên trong 288 bit được ký hiệu là (s_1,..., s_{288}).
Các Key và IV được sinh ngẫu nhiên bằng hàm:

```
def get_random_bits(length):
    randbits = secrets.randbits(length)
    randstring = '{0:080b}'.format(randbits)
    return bytearray(map(int ,randstring))
```

Quá trình tạo dòng khóa bao gồm một quy trình lặp đi lặp lại để trích xuất giá trị của 15 bit trạng thái cụ thể và sử dụng cả hai để cập nhật 3 bit trạng thái và để tính toán 1 bit của dòng khóa zi. Các bit trạng thái sau đó được xoay và quá trình lặp lại chính nó cho đến khi $N \leq 2^{64}$ bit của dòng khóa đã được được tạo ra.

```
def keystream_1(self, number):
    keystream = []
    for i in range(number):
        keystream.append(self._gen_keystream())
    return bits_to_hex(keystream)

def _gen_keystream(self):
    '''
    <Key stream generation>
    for i = 1 to N do
        t1 <- s66 + s93
        t2 <- s162 + s177
        t3 <- s243 + s288
        zi <- t1 + t2 + t3
        t1 <- t1 + s91 · s92 + s171
        t2 <- t2 + s175 · s176 + s264
        t3 <- t3 + s286 · s287 + s69
        (s1 , s2 , . . . , s93 ) <- (t3, s1 , . . . , s92)
        (s94 , s95 , . . . , s177 ) <- (t1 , s94, . . . , s176 )
        (s178 , s279 , . . . , s288) <- (t2 , s178 , . . . , s287 )
    end for
    '''
    t_1 = self.state[65] ^ self.state[92]
    t_2 = self.state[161] ^ self.state[176]
    t_3 = self.state[242] ^ self.state[287]

    z = t_1 ^ t_2 ^ t_3

    t_1 = t_1 ^ self.state[90] & self.state[91] ^ self.state[170]
    t_2 = t_2 ^ self.state[174] & self.state[175] ^ self.state[263]
    t_3 = t_3 ^ self.state[285] & self.state[286] ^ self.state[68]

    self.state.rotate() #1 positive rotation

    self.state[0] = t_3
    self.state[93] = t_1
    self.state[177] = t_2

    return z
```

Plain được nhập vào từ file txt:

```
def get_bytes_from_file(filename):
    return open(filename, "rb").read()
```

Hàm mã hóa trivium:

```
def encrypt(input, output):
    key = get_random_bits(80)
    iv = get_random_bits(80)
    plain = get_bytes_from_file(input)
    print("Plain: ", plain)
    trivium = Trivium(key, iv)
    keystream = trivium.keystream_1(len(plain) * 8)
    print("IV in hex:  {}".format(bits_to_hex(iv)))
    print("Key in hex: {}".format(bits_to_hex(key)))
    print("Keystream in hex: {}".format(keystream))
    cipher = trivium.encrypt(plain, keystream)
    print("Cipher: {}".format(cipher.hex()))
    print(cipher)
    with open(output, "wb") as output_file:
        # 80 first bits of the output file is iv
        output_file.write(iv)
        output_file.write(cipher)
```

Chạy chương trình mã hóa Trivium ta có:

```
C:\Users\Admin\Python>python trivium2.py plain.txt -o cipher.txt
Plain:  b'Trivium Setup\r\n'
IV in hex:  224280E397E2A7C01A57
Key in hex: 5953A8CD7893F930CCFF
Keystream in hex: A3B87F6B74CD28BE9F2E41A4EB8E1D
Cipher: f7ca161d1db8459ecc4b35d19b8317
bytearray(b'\xf7\xca\x16\x1d\x1d\xb8E\x9e\xccK5\xd1\x9b\x83\x17')

C:\Users\Admin\Python>
```