A Report on

Cryptoware: Implementation of A5/1 Cipher

(Mini-Project of CSE1007-Introduction to Cryptography)

Submitted by

Vadlapati Hari Praneeth

(Registration-18BCE7147)

on

14 November 2020

School of Computer Science and Engineering

VIT-AP University, Andhra Pradesh

Abstract

This projects implements A5/1 Cipher using Python.

The code is uploaded on GitHub in the following link.

`https://github.com/vhpraneeth/python_A51`

1 Introduction

This projects implements A5/1 cipher using Python using Alice-Bob model. Ciphertext is transferred from Alice to Bob using TCP protocol in localhost.

1.1 About

A5/1 (a member of the A5 family of ciphers) is used in the Global System for Mobile Communication (GSM), a network for mobile telephone communication.
A5/1 uses three LFSRs with 19, 22, and 23 bits.
Registers are clocked in a stop/go fashion using a majority rule.

- Each register has an associated clocking bit.
- At each cycle, the clocking bit of all three registers is examined and the majority bit is determined.
- A register is clocked if the clocking bit agrees with the majority bit

Majority(x,y,z)=1, if majority number of bits is 1, else 0

1.2 Implementation Environment

Programming language used: Python
Input required: message to send

2 Procedure

```
# a part of my_a5_1.py
def get_keystream(length):
    'calculates the keystream by XOR-ing the appropriate indeces'
    reg_x_temp = reg_x.copy()
    reg_y_temp = reg_y.copy()
```

```
reg_z_temp = reg_z.copy()
keystream = []
i = 0
while i < length:
    majority = get_majority(reg_x_temp[8], reg_y_temp[10], \
                            reg_z_temp[10])
    if reg_x_temp[8] == majority:
        new = reg_x_temp[13] ^ reg_x_temp[16] ^ \
                reg_x_temp[17] ^ reg_x_temp[18]
        reg_x_temp_two = reg_x_temp.copy()
        j = 1
        while(j < len(reg_x_temp)):
            reg_x_temp[j] = reg_x_temp_two[j-1]
            j += 1
        reg_x_temp[0] = new

    if reg_y_temp[10] == majority:
        new_one = reg_y_temp[20] ^ reg_y_temp[21]
        reg_y_temp_two = reg_y_temp.copy()
        k = 1
        while(k < len(reg_y_temp)):
            reg_y_temp[k] = reg_y_temp_two[k-1]
            k += 1
        reg_y_temp[0] = new_one

    if reg_z_temp[10] == majority:
        new_two = reg_z_temp[7] ^ reg_z_temp[20] ^ \
                    reg_z_temp[21] ^ reg_z_temp[22]
        reg_z_temp_two = reg_z_temp.copy()
        m = 1
        while(m < len(reg_z_temp)):
            reg_z_temp[m] = reg_z_temp_two[m-1]
```

```
                m += 1
            reg_z_temp[0] = new_two

        keystream.insert(i, reg_x_temp[18] ^ reg_y_temp[21] ^ \
                         reg_z_temp[22])
        i += 1
    return keystream

def encrypt(plain):
    'encrypts plain text'
    ciphertext = ''
    binary = to_binary(plain)
    keystream = get_keystream(len(binary))
    i = 0
    while(i < len(binary)):
        ciphertext += str(binary[i] ^ keystream[i])
        i += 1
    return ciphertext

def decrypt(cipher):
    'decrypts the cipher and returns plain text'
    s = ''
    binary = []
    keystream = get_keystream(len(cipher))
    i = 0
    while(i < len(cipher)):
        binary.insert(i,int(cipher[i]))
        s += str(binary[i] ^ keystream[i])
        i += 1
    return convert_binary_to_str(str(s))  # plaintext
```

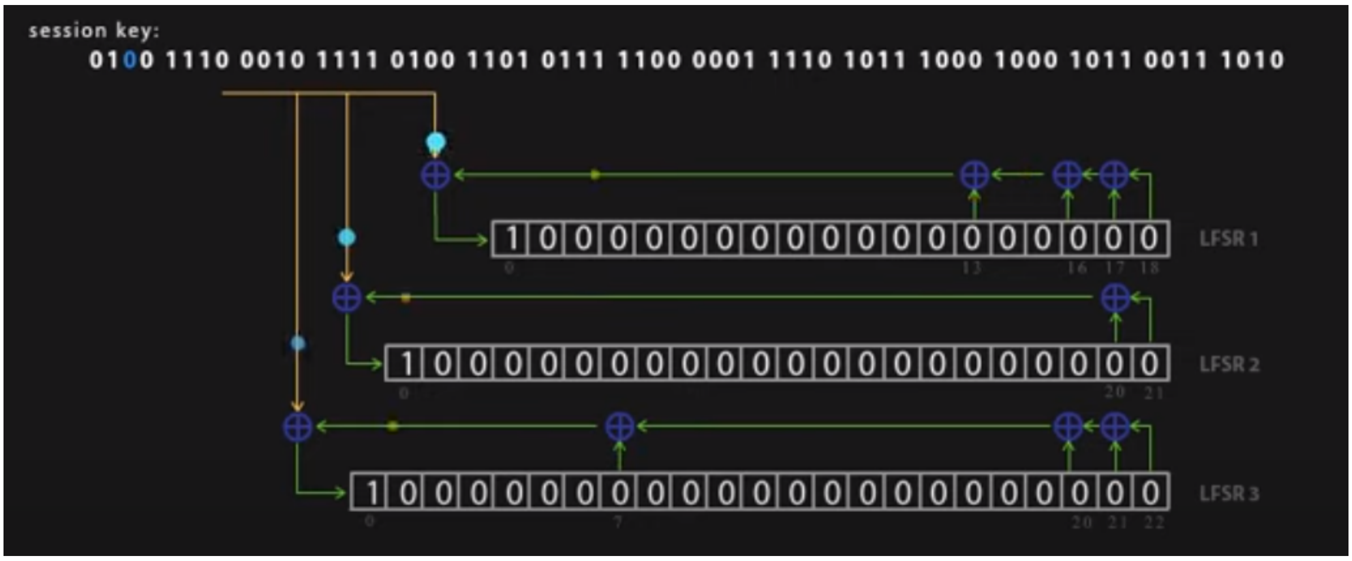


Figure 1: How A5/1 works.

3 Major Components

```
# Alice.py
import socket
from my_a5_1 import encrypt, port

message = input('Enter message to send: ')
message = encrypt(message)

s = socket.socket()
s.bind(('', port))
s.listen(5)
print('Socket is listening')

while True:
    c, addr = s.accept()
    c.send(message.encode())
    print('Message sent')
    c.close()
```

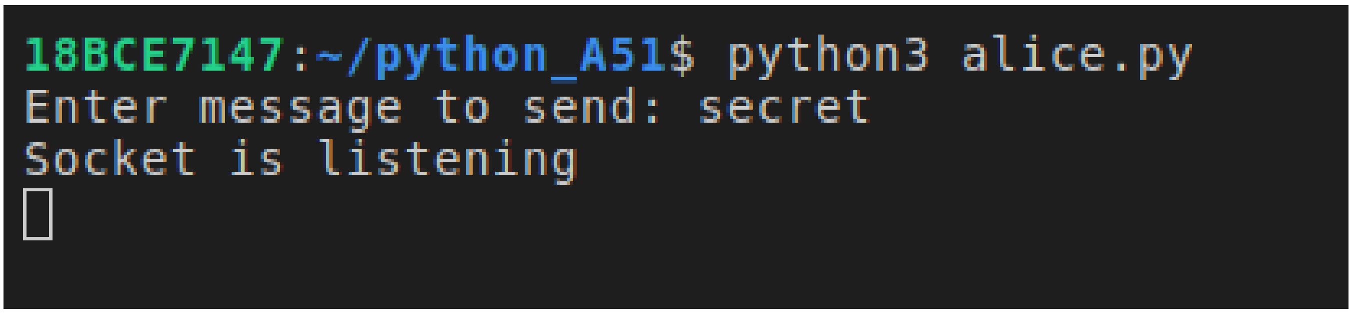
```
# Bob.py
import socket
from my_a5_1 import decrypt, port

s = socket.socket()
s.connect(('127.0.0.1', port))
message = s.recv(1024)
s.close()

message = decrypt(message.decode())
print('Received message: ' + message + '\n')
```

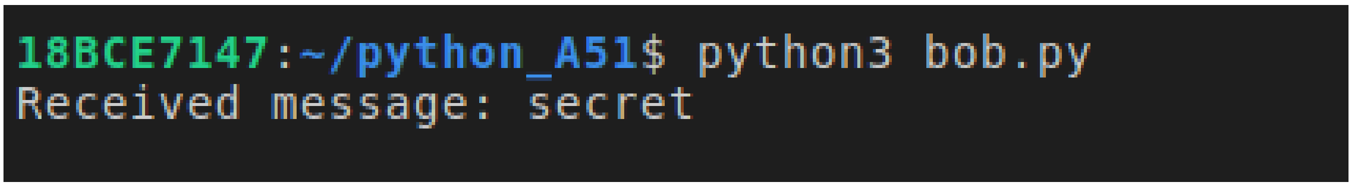
4 Results

Depicted in Figure 2 and 3.



```
18BCE7147:~/python_A51$ python3 alice.py
Enter message to send: secret
Socket is listening
```

Figure 2: Sender-side (Alice).



```
18BCE7147:~/python_A51$ python3 bob.py
Received message: secret
```

Figure 3: Receiver-side (Bob).

References

[1] Behrouz A Forouzan, Debdeep Mukhopadhyay, "Cryptography and Network Security", Mc Graw Hill, Third Edition, 2015.

[2] William Stallings, "Cryptography and Network Security: Principles and Practice", Pearson Education, Seventh Edition, 2017.