Experiment No. 3

Title: A5/1

Batch: B2 **Roll No.:16010421119** **Experiment No.: 3**

Aim: To implement stream cipher A5/1

Resources needed: Windows/Linux

Theory: Pre Lab/ Prior Concepts:

A5/1 employs three linear feedback shift registers , or LFSRs, which are labeled X, Y, and Z. Register X holds 19 bits, (x_0,x_1…x_{18}).The register Y holds 22 bits, (y_0,y_1…y_{21}) and Z holds 23 bits, (z_0,y_1…z_{22}) ·Of course, all computer geeks love powers of two, so it's no accident that the three LFSRs hold a total of 64 bits.

Not coincidentally, the A5/1 key K is also 64 bits. The key is used as the initial fill of the three registers, that is, the key is used as the initial values in the three registers. After these three registers are filled with the key,1 we are ready to generate the keystream. But before we can describe how the keystream is generated, we need to say a little more about the registers X, Y, and Z.

When register X steps, the following series of operations occur:

$$t = x_{13} \oplus x_{16} \oplus x_{17} \oplus x_{18}$$
$$x_i = x_{i-1} \text{ for } i = 18, 17, 16, \ldots, 1$$
$$x_0 = t$$

Similarly, for registers Y and Z, each step consists of

$$t = y_{20} \oplus y_{21}$$
$$y_i = y_{i-1} \text{ for } i = 21, 20, 19 \ldots, 1$$
$$y_0 = t$$

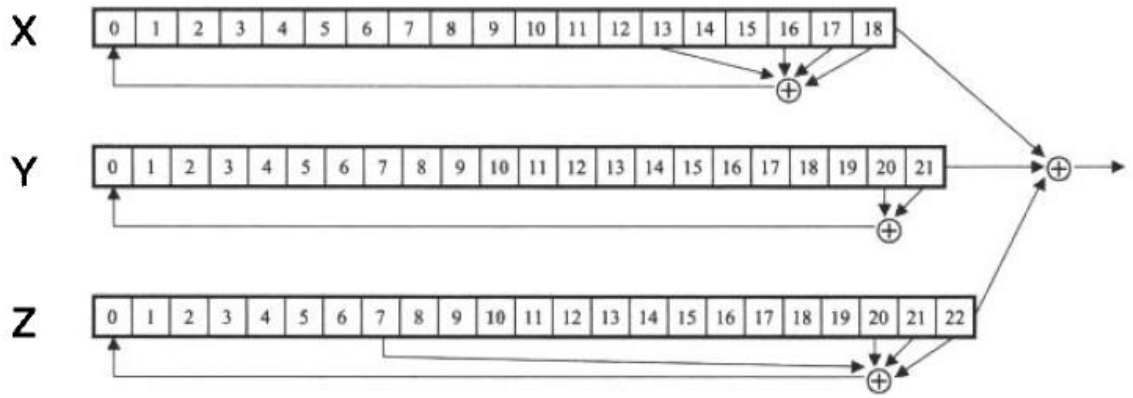
and

$$t = z_7 \oplus z_{20} \oplus z_{21} \oplus z_{22}$$
$$z_i = z_{i-1} \text{ for } i = 22, 21, 20, \ldots, 1$$
$$z_0 = t$$

respectively.

Given three bits x, y, and z, define ma,](x,y, z) to be the majority vote function, that is, if the majority of x, y, and z are 0, the function returns 0; otherwise it returns 1. Since there are an odd number of bits, there cannot be a tie, so this function is well defined.

The wiring diagram for the A5/1 algorithm is illustrated below:



A5/1 Keystream Generator

Procedure / Approach /Algorithm / Activity Diagram:

A. Key Stream generation Algorithm:

At each step: m = maj(x_8, y_{10}, z_{10})

-Examples: maj(0,1,0) = 0 and maj(1,1,0) = 1

If x_8 = m then X steps

-t = $x_{13} \oplus x_{16} \oplus x_{17} \oplus x_{18}$

-$x_i = x_{i-1}$ for i = 18,17,…,1 and x_0 = t

If y_{10} = m then Y steps

-t = $y_{20} \oplus y_{21}$

-$y_i = y_{i-1}$ for i = 21,20,…,1 and y_0 = t

If z_{10} = m then Z steps

-t = $z_7 \oplus z_{20} \oplus z_{21} \oplus z_{22}$

-$z_i = z_{i-1}$ for i = 22,21,…,1 and z_0 = t

Keystream bit is $x_{18} \oplus y_{21} \oplus z_{22}$

Implementation:

Implement the A5/1 algorithm. Encryption and decryption function should ask for key and a input and show the output to the user.

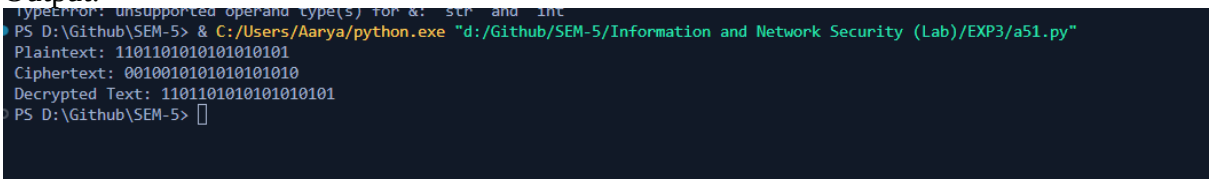
Results: (Program with output as per the format)

Code:

```
def majority(x, y, z):
    # Returns the majority bit
    return (x & y) | (x & z) | (y & z)


def clock(register, majority_bit):
    new_bit = register[18] ^ majority_bit
    register.pop()
    register.insert(0, new_bit)


def generate_keystream(key, num_bits):
```

```
    # Convert key to a list of integers
    R1 = [int(bit) for bit in key] + [0] * (19 - len(key))
    R2 = [0] * 22
    R3 = [0] * 23
    keystream = []

    for _ in range(num_bits):
        majority_bit = majority(R1[8], R2[10], R3[10])
        if R1[8] == majority_bit:
            clock(R1, majority_bit)
        if R2[10] == majority_bit:
            clock(R2, majority_bit)
        if R3[10] == majority_bit:
            clock(R3, majority_bit)
        keystream_bit = R1[18] ^ R2[21] ^ R3[22]
        keystream.append(keystream_bit)

    return keystream


def encrypt_or_decrypt(data, keystream):
    return [str(int(data[i]) ^ keystream[i]) for i in range(len(data))]


def a51_encrypt(key, plaintext):
    keystream = generate_keystream(key, len(plaintext))
    encrypted = encrypt_or_decrypt(plaintext, keystream)
    return ''.join(encrypted)


def a51_decrypt(key, ciphertext):
    # Encryption and decryption are the same in a stream cipher
    return a51_encrypt(key, ciphertext)


def main():
    key = "1010101010101010101"
    plaintext = "11011010101010101010"

    ciphertext = a51_encrypt(key, plaintext)
    decrypted_text = a51_decrypt(key, ciphertext)

    print("Plaintext:", plaintext)
    print("Ciphertext:", ciphertext)
    print("Decrypted Text:", decrypted_text)


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

Output:



Questions:

1) List the stream cipher used in current date along with the name of applications in which those are used.

1. **RC4 (Rivest Cipher 4):**
   - RC4 was widely used in various applications, but its security vulnerabilities have led to its disuse in many modern systems.
   - It was used in wireless networks (WEP), TLS/SSL (early versions), and other cryptographic protocols.
2. **Salsa20 / ChaCha20:**
   - Salsa20 and ChaCha20 are modern stream ciphers designed by Daniel Bernstein.
   - They are used in applications like Internet security protocols (e.g., OpenVPN), disk encryption (e.g., VeraCrypt), and in the construction of other cryptographic primitives.
3. **Rabbit:**
   - Rabbit is a stream cipher designed for low-resource environments.
   - It was used in the Bluetooth protocol and was considered in ISO/IEC standards for cryptography.

Outcomes:

CO2: Illustrate different cryptographic algorithms for security.

Conclusion: (Conclusion to be based on the objectives and outcomes achieved)
We can conclude that we have learnt about A 51 stream cipher.

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of faculty in-charge with date

References: Books/ Journals/ Websites:

1. Mark Stamp, “Information Security Principles and Practice”, Wiley.
2. Behrouz A. Forouzan, “Cryptography and Network Security”, Tata McGraw Hill
3. William Stalling, “Cryptography and Network Security”, Prentice Hall