

**Experiment No. 3**

**Title:** A5/1

**Batch: B1 Roll No.: 1914078 Experiment No.: 3**

**Aim:** To implement stream cipher A5/1

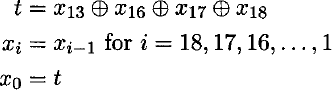
**Resources needed:** Windows/Linux, C or JAVA language

# Theory: Pre Lab/ Prior Concepts:

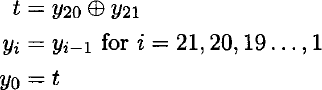
A5/1 employs three linear feedback shifl registers , or LFSRs, which are labeled X, Y, and Z. Register X holds 19 bits, (x0,x1…x18)-The register Y holds 22 bits, (y0,y1…y21) and Z holds 23 bits, (z0,y1…z22) ·Of course, all computer geeks love powers of two, so it's no accident that the three LFSRs hold 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.

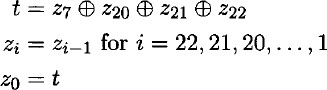
When register X steps, the following series of operations occur:



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



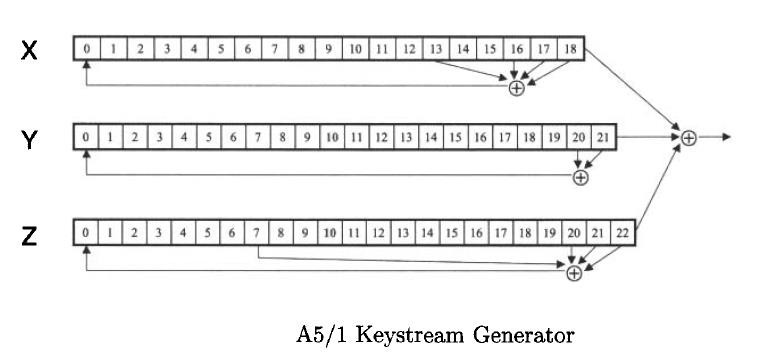
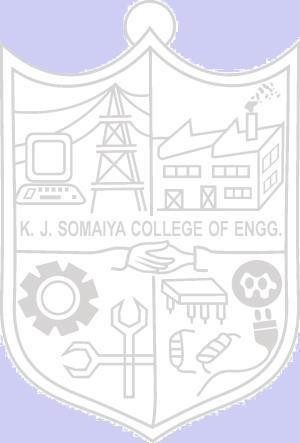
and



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:



# Procedure / Approach /Algorithm / Activity Diagram:

1. **Key Stream generation Algorithm:**

At each step: m = maj(x8, y10, z10)

-Examples: maj(0,1,0) = 0 and maj(1,1,0) = 1 If x8 = m then X steps

-t = x13  x16  x17  x18

-xi = xi1 for i = 18,17,…,1 and x0 = t If y10 = m then Y steps

-t = y20  y21

-yi = yi1 for i = 21,20,…,1 and y0 = t If z10 = m then Z steps

-t = z7  z20  z21  z22

-zi = zi1 for i = 22,21,…,1 and z0 = t

# Keystream bit is *x*18  *y*21  *z*22

**Implementation:**

Implement the A5/1 algorithm. Generate and print the next 32 keystream bits. Print the contents of *X, Y and Z after* the 32 keystream bits have been generated

**Results:** (Program printout with output / Document printout as per the format)

**Code:**

def majorityFn(bit1, bit2, bit3):

    list1 = [bit1, bit2, bit3]

    return max(list1, key=list1.count)

def returnList(string):

    list1 = []

    for x in string:

        list1.append(int(x))

    return list1

def XOR(a,b,c=0,d=0):

    return a^b^c^d

def A51CipherEncrypt(key, bits):

    X = returnList(key[0:19])

    Y = returnList(key[19:41])

    Z = returnList(key[41:64])

    keystream = []

    for bit in range(bits):

        maj = majorityFn(X[8], Y[10], Z[10])

        if X[8] == maj:

            Tx = XOR(X[13], X[16], X[17], X[18])

            X.insert(0, Tx)

            X.pop()

        if Y[10] == maj:

            Ty = XOR(Y[20], Y[21])

            Y.insert(0, Ty)

            Y.pop()

        if Z[10] == maj:

            Tz = XOR(Z[7], Z[20], Z[21], Z[22])

            Z.insert(0, Tz)

            Z.pop()

    keystream.append(XOR(X[18], Y[21], Z[22]))

    keystream = ''.join(str(x) for x in keystream)

    return X, Y, Z, keystream

key = input('Enter the 64 bit key: ')

bits = int(input('Enter the number of bits of keystream: '))

X, Y, Z, keystream = A51CipherEncrypt(key, bits)

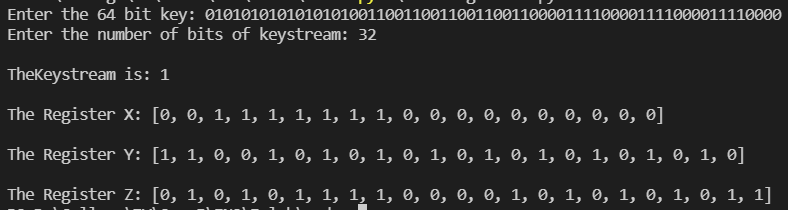
print(f'\nTheKeystream is: {keystream}')

print(f'\nThe Register X: {X}')

print(f'\nThe Register Y: {Y}')

print(f'\nThe Register Z: {Z}')

**Output:**



**Outcomes:**

**CO2:** Illustrate different cryptographic algorithms for security .

**Conclusion:** We learnt how to implement A5/1 Cipher and learnt how keystream bits are generated using Left Shift Registers in Python.

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

**Signature of faculty in-charge with date**

**References: Books/ Journals/ Websites:**

* 1. Charles P. Pfleeger, “Security in Computing”, Pearson Education
  2. Behrouz A. Forouzan, “Cryptography and Network Security”, Tata McGraw Hill
  3. William Stalling, “Cryptography and Network Security”, Prentice Hall