Department of Information Technology

Academic Term: Jan-May 2021

Class **:** T.E IT Sem -VI
Subject : Sensor Network Lab (ITL604)

Practical No:	**7**
Title:	Implementation of GSM security algorithms (A3/A5/A8)
Date of Performance:	
Date of Submission:	
Roll No:	8669
Name of the Student:	Mareena Mark Fernandes

Evaluation:

Sr. No	Rubric	Grade
1	**On time submission or competition (2)**	
2	**Preparedness (2)**	
3	**Skill (4)**	
4	**Output (2)**	

Signature of the Teacher **:**

PRACTICAL - 4

Title : Implementation of GSM security algorithms (A3/A5/A8)

Objective : To study about GSM security .

References :

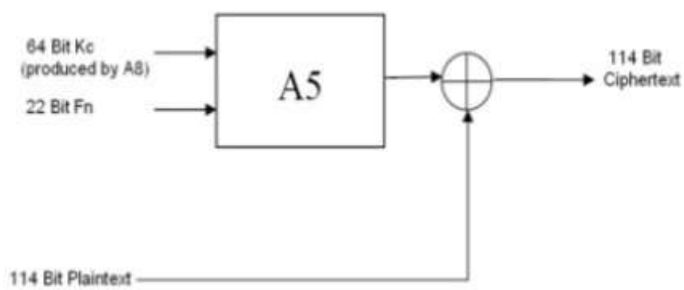
Prerequisite : knowledge of any Programming.

Theory:

A5 is a stream cipher which can be implemented very efficiently on any hardware. It is used by GSM cell phones for confidentiality.
There exist several implementations of this algorithm, the most commonly used ones are A5/0, A5/1 and A5/2 (A5/3 is used in 3G system).
As a stream cipher, A5 works on a bit by bit basis and not on blocks.

A5 Algorithm

Kc is the key which was produced by A8 algorithm. PT is the data which is transmitted. Fn is the frame bits which come from LFSR (Linear Feedback Shift Register) process.



- A5/1 employs 3 linear feedback shift registers, which will label as X,Y,Z
- Register X holds 19 bits, which we label (x0,x1,x2… .....x18)
- The register Y holds 22 bits (y0,y1,y2…................... y21) and Z holds 23 bits (z0,z1,z2,…............... z22)
- So three LFSRs hold 64 bits.
- Initialize key K of size 64 bits. The key is used as the initial fill of the three registers.
- After these 3 registers are filled with the key, we are ready to generate the keystream.

Keystream Generation

- When register X steps, the following occur

  $t = x_{13}$ xor x_{16} xor x_{17} xor x_{18}

  $x_i = x_{i-1}$ for i=18,17,16,………1

  $x_0 = t$

  Similarly for registers Y and Z , each step consists of

  $t = y_{20}$ xor y_{21}

  $y_i = y_{i-1}$ for i=21,20,19,………1

  $y_0 = t$

  and

  $t = z_7$ xor z_{20} xor z_{21} xor z_{22}

  $z_i = z_{i-1}$ for i=22,21,20,………1

  $z_0 = t$

- Given three bits x,y,z , define maj(x,y,z) to be the majority vote function; that is, if the majority of x,y,z are 0, then the function returns 0, otherwise it returns 1.

- A5/1 is implemented in hardware, and at each clock pulse the value

  m= maj(x8, y10, z10) is computed.

- Then the registers X,Y and Z step according to the following rules:

  If x_8 = m then X steps

  If y_{10} = m then Y steps

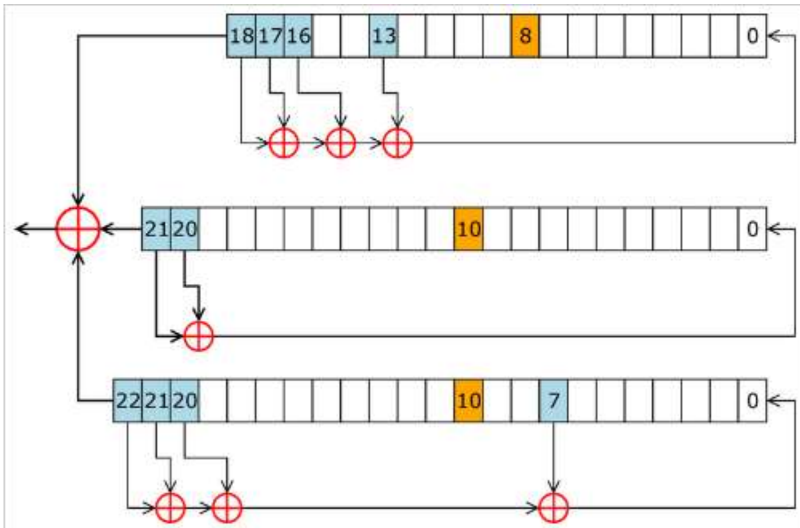
  If z_{10} = m then Z steps

- Finally , a keystream bit s is generated as

  $s = X_{18} \oplus y_{21} \oplus z_{22}$

which is then XORed with the plaintext (if encrypting) or XORed with the ciphertext (if decrypting)

Although this seem like a complicated way to generate a single keystream bit, A5/1 is easily implemented in hardware and can generate bits at a rate proportional to the clock speed .



A5/1 Keystream generator

CODE:

```
#include <bits/stdc++.h>#include <regex>
using namespace std;class A51
{
private:
string key;
static const int REG_X_LEN = 19;static const int REG_Y_LEN = 22;static const int REG_Z_LEN
= 23;

int regX[REG_X_LEN];int regY[REG_Y_LEN];int regZ[REG_Z_LEN];

void load_registers(string key)
{
for (int i = 0; i < REG_X_LEN; i++)regX[i] = key[i] - '0';
for (int i = 0; i < REG_Y_LEN; i++) regY[i] = key[REG_X_LEN + i] - '0';
for (int i = 0; i < REG_Z_LEN; i++)
regZ[i] = key[REG_X_LEN + REG_Y_LEN + i] - '0';
}
int getMejority(int x, int y, int z)
{
return x + y + z > 1 ? 1 : 0;
}
int toBin(string s){int tp = 0;
int no = 0;
for (int i = s.length() - 1; i >= 0;i--){
no += pow(int(s[i] - '0'), tp);
}
return no;
}
string toStr(string binary)
{
string s;
// if (binary.length() % 8 != 0)
// {
//                              binary = binary.substr(0, binary.length() - binary.length() % 8);
// }
for (int i = 0; i < binary.length() - 8; i += 8)
{
s.push_back((char)toBin(binary.substr(i, i + 8)));
}
return s;
}
```

```
public:
string setKey(string key)
{

if (key.length() == 64)
{
this->key = key; load_registers(key);return "Success";
}
return "Abort";
}
string getKey()
{
return key;
}
vector<int> getKeyStream(int length)
{
int regX[REG_X_LEN];
copy(this->regX, this->regX + this->REG_X_LEN, regX);int regY[REG_Y_LEN];
copy(this->regY, this->regY + this->REG_Y_LEN, regY);int regZ[REG_Z_LEN];
copy(this->regZ, this->regZ + this->REG_Z_LEN, regZ);vector<int> keyStream(length);

for (int i = 0; i < length; i++)
{
int maj = getMejority(regX[9], regY[10], regZ[10]);if (regX[8] == maj)
{
int newStart = regX[13] ^ regX[16] ^ regX[17] ^ regX[18];int temp[REG_X_LEN];
copy(begin(regX), regX + this->REG_X_LEN, begin(temp));for (int j = 1; j < REG_X_LEN; j++)
	regX[j] = temp[j - 1];regX[0] = newStart;
}
if (regY[10] == maj)
{
int newStart = regY[20] ^ regY[21];int temp[REG_Y_LEN];
copy(regY, regY + this->REG_Y_LEN, temp);

for (int j = 1; j < REG_Y_LEN; j++)regY[j] = temp[j - 1];
regY[0] = newStart;
}
if (regZ[10] == maj)
{
int newStart = regZ[7] ^ regZ[20] ^ regZ[21] ^ regZ[22];int temp[REG_Z_LEN];
copy(regZ, regZ + REG_Z_LEN, temp);for (int j = 1; j < REG_Z_LEN; j++)
```

```
        regZ[j] = temp[j - 1];regZ[0] = newStart;
}
keyStream[i] = regX[18] ^ regY[21] ^ regZ[22];
}
return keyStream;
}
vector<int> toBinary(string plaintext)
{
string s;
for (int i = 0; i < plaintext.length(); i++)
{
string temp = bitset<8>(plaintext[i]).to_string();for (int j = temp.length(); j < 8; j++)
        temp = "0" + temp;s += temp;
}
vector<int> binary(s.length());

for (int i = 0; i < s.size(); i++)
{
binary[i] = int(s[i] - 48);
}
return binary;
}
string encrypt(string plaintext)
{
vector<int> binary = toBinary(plaintext);vector<int>
        keystream = getKeyStream(binary.size());string s;
for (int i = 0; i < binary.size(); i++)
{
s.push_back((binary[i] ^ keystream[i]) + '0');
}
return s;
}
string decrypt(string ciphertext)
{
string s;
vector<int> binary(ciphertext.length());
vector<int> keyStream = getKeyStream(ciphertext.length());for (int i = 0; i < binary.size(); i++)
{
binary[i] = ciphertext[i];
s += binary[i] ^ keyStream[i];
}
return toStr(s);}};
```

```
int main()
{
A51 a51;
string key = "1101000111000111000010011010101010010110101001100111100101101001";
cout << a51.setKey(key) << endl;

string plaintext = "Hello World";
cout << "Plaintext:" << plaintext << endl;
cout << "CipherText: " << a51.encrypt(plaintext) << endl;
	string ciphertext =
"01100001110001010011111100110100001011101011101010100110110010110011
1110100001000000001 10";
cout << "Decrypted Text: " << a51.decrypt(ciphertext) << "|" << endl;vector<int> keystream =
a51.getKeyStream(64);
for (int i = 0; i < 64; i++)cout << keystream[i];
return 0;
}
```

OUTPUT: