DEPARTMENT OF COMPUTER ENGINEERING

Semester	T.E. Semester VI – Computer Engineering
Subject	Mobile Computing
Subject Professor In-charge	Prof. Sneha Annappanavar
Assisting Teachers	Prof. Sneha Annappanavar
Laboratory	M310A

Student Name	Deep Salunkhe
Roll Number	21102A0014
TE Division	A

Title:

GSM security

Explanation:

In GSM (Global System for Mobile Communications) security, A3, A8, and A5 are algorithms used to provide authentication, key generation, and encryption, respectively. Here's some theoretical background on each:

1. **A3 Algorithm (Authentication Algorithm)**:

    - The A3 algorithm is responsible for authenticating the mobile subscriber to the network. It generates a digital signature based on a secret key (Ki) stored on the SIM card and the subscriber's authentication challenge (RAND) received from the network.
    - When a mobile device attempts to connect to the network, the network sends a random challenge (RAND) to the device.
    - The device uses the A3 algorithm along with its secret key (Ki) to generate a response (SRES), which it sends back to the network.
    - The network also calculates the expected response using the same algorithm and compares it with the received response to authenticate the device.

2. **A8 Algorithm (Key Generating Algorithm)**:

    - The A8 algorithm is used to generate session keys (Kc) for encryption purposes. These session keys are unique to each call or data session and are used to encrypt the communication between the mobile device and the base station.

- A8 takes as input the secret key (Ki) and the random challenge (RAND) generated during the authentication process.
- It produces the session key (Kc) as its output, which is then used by both the mobile device and the network to encrypt and decrypt communication.

3. **A5 Algorithm (Encryption Algorithm)**:
   - The A5 algorithm is responsible for encrypting voice and data traffic between the mobile device and the base station. It ensures the confidentiality of the communication by scrambling the data using a stream cipher.
   - A5 generates a pseudo-random keystream based on a session key (Kc) and an initialization vector (IV). This keystream is XORed with the plaintext data to produce encrypted ciphertext.
   - Both the mobile device and the network use the same session key (Kc) and initialization vector (IV) to synchronize the encryption and decryption process.

Implementation:

```
#include <string>
#include <iostream>
#include <cstdlib> // For rand() and srand()
#include <ctime>   // For time()
using namespace std;

string Random128(){

   // Seed the random number generator
    srand(time(nullptr));

    // This function creates a random 128-bit string comprising of 0 and 1
    std::string s;
    for (int i = 0; i < 128; ++i) {
        s.push_back('0' + rand() % 2); // Convert 0 and 1 to characters '0' and
'1'
    }
    return s;

}

void printstring(string s){
```

```
    for(int i=0;i<s.length();i++){
        cout<<s[i];
    }
    cout<<endl;
}

string A3(string key, string random){
    //Logic for A3
    //Divide key and random in two 64 bits strings
    //say key_64_L, key_64_R, random_64_L, random_64_R
    //XOR key_64_L and random_64_R and store in A3_64_1
    //XOR key_64_R and random_64_L and store in A3_64_2
    //Divide A3_64_1 and A3_64_2 in two 32 bits strings
    //name them as A3_64_1_L, A3_64_1_R, A3_64_2_L, A3_64_2_R
    //XOR A3_64_1_L and A3_64_2_R and store in A3_32_1
    //XOR A3_64_1_R and A3_64_2_L and store in A3_32_2
    //Xor A3_32_1 and A3_32_2 and store in A3_32
    //return A3_32

    string key_64_L=key.substr(0,64);
    string key_64_R=key.substr(64,64);
    string random_64_L=random.substr(0,64);
    string random_64_R=random.substr(64,64);

    string A3_64_1;
    string A3_64_2;
    for(int i=0;i<64;i++){
        A3_64_1.push_back(key_64_L[i]^random_64_R[i]);
        A3_64_2.push_back(key_64_R[i]^random_64_L[i]);
    }

    string A3_64_1_L=A3_64_1.substr(0,32);
    string A3_64_1_R=A3_64_1.substr(32,32);
    string A3_64_2_L=A3_64_2.substr(0,32);
    string A3_64_2_R=A3_64_2.substr(32,32);

    string A3_32_1;
    string A3_32_2;
    for(int i=0;i<32;i++){
        A3_32_1.push_back(A3_64_1_L[i]^A3_64_2_R[i]);
        A3_32_2.push_back(A3_64_1_R[i]^A3_64_2_L[i]);
    }

    string A3_32;
```

```
    for(int i=0;i<32;i++){
        A3_32.push_back(A3_32_1[i]^A3_32_2[i]);
    }



    return A3_32;


}

string changebit(string s){
    int i=rand()%128;
    if(s[i]=='0'){
        s[i]='1';
    }
    else{
        s[i]='0';
    }

    return s;
}

string A8(string key ,string random){
  //logic for A8
  //Divide key and random in two 64 bits strings
  //say key_64_L, key_64_R, random_64_L, random_64_R
  //XOR key_64_L and random_64_R and store in A8_64_1
  //XOR key_64_R and random_64_L and store in A8_64_2
  //Xor A8_64_1 and A8_64_2 and store in A8_64
  //return A8_64

    string key_64_L=key.substr(0,64);
    string key_64_R=key.substr(64,64);

    string random_64_L=random.substr(0,64);
    string random_64_R=random.substr(64,64);

    string A8_64_1;
    string A8_64_2;

    for(int i=0;i<64;i++){
        A8_64_1.push_back(key_64_L[i]^random_64_R[i]);
        A8_64_2.push_back(key_64_R[i]^random_64_L[i]);
```

```
    }

    string A8_64;
    for(int i=0;i<64;i++){
        A8_64.push_back(A8_64_1[i]^A8_64_2[i]);
    }

    return A8_64;
}

int main(){
    string Key_AUC = Random128();
    string Key_SIM=Key_AUC;
    string Random_Number = Random128();

    string Random_Number_2=changebit(Random_Number);


    cout<<"Key_AUC: "<<endl;
    printstring(Key_AUC);
    cout<<"Key_SIM: "<<endl;
     printstring(Key_AUC);
    cout<<"Random_Number: "<<endl;
    printstring(Random_Number);

    string A3_AUC=A3(Key_AUC,Random_Number);
    string A3_SIM=A3(Key_SIM,Random_Number);




    if(A3_AUC==A3_SIM){
        cout<<"Valid SIM Card"<<endl;
    }
    else{
        cout<<"Invalid SIM Card"<<endl;
    }

    string ecrykey_AUC=A8(Key_AUC,Random_Number);
    string ecrykey_SIM=A8(Key_SIM,Random_Number);



    if(ecrykey_AUC==ecrykey_SIM){
```

```
        cout<<"Proper ecryption is possible"<<endl;
    }
    else{
        cout<<"Proper encryption is not possible"<<endl;
    }


    return 0;
}
```

End Result:

```
PS E:\GIt\SEM-6\MC> cd "e:\GIt\SEM-6\MC\" ; if ($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ; if ($?) { .\tem
pCodeRunnerFile }
Key_AUC:
01101101111110001000010111101000111101101000000011001011101110010011000011101110000100010011100110111011111000001101000000
001011
Key_SIM:
01101101111110001000010111101000111101101000000011001011101110010011000011101110000100010011100110111011111000001101000000
001011
Random_Number:
01101101111110001000010111101000111101101000000011001011101110010011000011101110000100010011100110111011111000001101000000
001011
Valid SIM Card
Proper ecryption is possible
PS E:\GIt\SEM-6\MC>
```

```
PS E:\GIt\SEM-6\MC> cd "e:\GIt\SEM-6\MC\" ; if ($?) { g++ GSMAuth.cpp -o GSMAuth } ; if ($?) { .\GSMAuth }
Key_AUC:
11011000101101110100111010000011111010011110001000011110010111101010111101010001001000101011111100110111100110000100100010
110011
Key_SIM:
11011000101101110100111010000011111010011110001000011110010111101010111101010001001000101011111100110111100110000100100010
110011
Random_Number:
11011000101101110100111010000011111010011110001000011110010111101010111101010001001000101011111100110111100110000100100010
110011
Invalid SIM Card
Proper encryption is not possible
```

Conclusion:

These algorithms collectively form the security framework of GSM networks, providing authentication, key management, and encryption to protect the privacy and integrity of mobile communication. However, it's worth noting that the A5 encryption algorithm has been subject to various security vulnerabilities over the years, leading to the development of more secure encryption algorithms in later generations of mobile networks