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:

DSSS

Explanation:

Introduction: Direct Sequence Spread Spectrum (DSSS) is a modulation technique used in wireless communication systems to spread the signal over a wider bandwidth than the minimum required. It involves multiplying the data signal by a chipping code sequence.

Chipping Code: The chipping code, also known as the spreading code, is a sequence of chips (binary digits) generated by a pseudorandom noise (PN) sequence generator. This code determines how the data signal is spread across the bandwidth.

Encryption Process:

1. Input Data: The original data is taken as input, usually in the form of binary digits.
2. Chipping Code Generation: Another binary sequence, called the chipping code, is generated. This sequence is typically longer than the original data sequence.
3. DSSS Encryption: Each bit of the original data is XORed with the corresponding bit of the chipping code sequence. This process spreads the data across a wider bandwidth.
4. Transmission: The encrypted signal, which now occupies a larger bandwidth due to the spreading effect, is transmitted over the channel.

Decryption Process:

1. Received Signal: The transmitted signal, possibly corrupted by noise and interference, is received.

2. Synchronization: Synchronization is achieved between the received signal and the chipping code sequence.
3. DSSS Decryption: The received signal is XORed with the synchronized chipping code sequence to recover the original data.
4. Output Data: The decrypted data is obtained, which ideally matches the original input data.

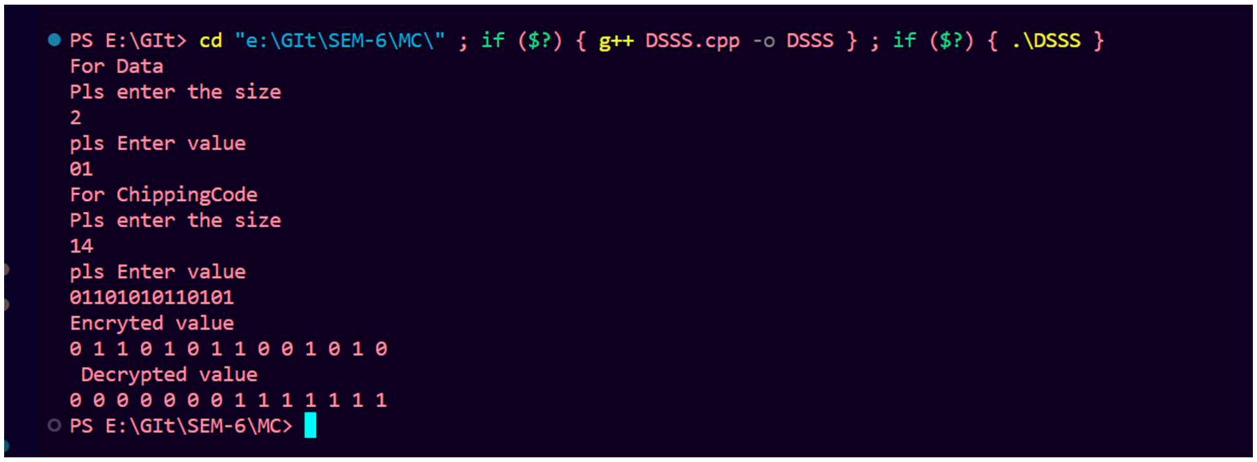
Implementation:

```cpp
#include<iostream>
#include<vector>
using namespace std;

bool Takeinput(vector<int>&input,int &size){

    cout<<"Pls enter the size"<<endl;
    cin>>size;

    string tempinput;

    cout<<"pls Enter value"<<endl;
    cin>>tempinput;

    if(tempinput.length()!=size){
        cout<<"Invalid size"<<endl;
        return false;
    }

    for(int i=0;i<size;i++){

        char temp=tempinput[i];

        if(temp=='1')
        input.push_back(1);
        else
        input.push_back(0);
    }

    return true;
```

```
}

void DSSS_encrypt(vector<int>data,int data_size,vector<int>chipping_code,int
chipping_size,vector<vector<int> >&division){

    int parts=data_size;
    int size_of_part=chipping_size/parts;

    int index=0;
    for(int i=0;i<chipping_size;i++){

        int parts=size_of_part;
        vector<int>temp;
        while(parts--){
            int xored=chipping_code[i]^data[index];
            temp.push_back(xored);
            i++;
        }
        i--;
        index++;
        division.push_back(temp);


    }

    cout<<"Encryted value"<<endl;
    for(int i=0;i<division.size();i++){
        for(int j=0;j<division[0].size();j++){
            cout<<division[i][j]<<" ";
        }
    }
    cout<<endl;


}

vector<int> DSSS_decode(vector<vector<int> >encoded,vector<int>chipping){

    int index=0;
    vector<int>ans;

    for(int i=0;i<encoded.size();i++){
        for(int j=0;j<encoded[0].size();j++){
            int temp=chipping[index]^encoded[i][j];
```

```
                ans.push_back(temp);
                index++;
            }

        }

        return ans;
}

void Print(vector<int>ans){
    cout<<" Decrypted value"<<endl;
    for(int i=0;i<ans.size();i++){
        cout<<ans[i]<<" ";
    }
}


int main(){
    vector<int>data;
    int data_size;
    vector<int>chippingCode;
    int chipping_code_size;
    vector<vector<int> >encoded;
    bool check=true;

    cout<<"For Data"<<endl;
    check=Takeinput(data,data_size);
    if(!check) return 0;

    cout<<"For ChippingCode"<<endl;
    check=Takeinput(chippingCode,chipping_code_size);
    if(!check) return 0;

    DSSS_encrypt(data,data_size,chippingCode,chipping_code_size,encoded);
    vector<int>ans=DSSS_decode(encoded,chippingCode);
    Print(ans);
    return 0;
}
```

End Result:



```
PS E:\GIt> cd "e:\GIt\SEM-6\MC\" ; if ($?) { g++ DSSS.cpp -o DSSS } ; if ($?) { .\DSSS }
For Data
Pls enter the size
2
pls Enter value
01
For ChippingCode
Pls enter the size
14
pls Enter value
01101010110101
Encryted value
0 1 1 0 1 0 1 1 0 0 1 0 1 0
 Decrypted value
0 0 0 0 0 0 0 1 1 1 1 1 1 1
PS E:\GIt\SEM-6\MC>
```

Conclusion:

In this report, we implemented a basic simulation of the DSSS encryption and decryption process using C++. The program takes input data and a chipping code, encrypts the data using DSSS, and then decrypts the encrypted signal to recover the original data.

DSSS offers several advantages in wireless communication systems, including increased resistance to interference and jamming, enhanced security through encryption, and the ability to coexist with other communication systems operating in the same frequency band. However, it also comes with some drawbacks, such as increased bandwidth requirements and complexity in implementation.

By understanding the principles of DSSS and implementing them in this lab exercise, we gain insights into the fundamental concepts of spread spectrum communication techniques and their practical applications in mobile computing and wireless networks. Further experimentation and study in this area can lead to a deeper understanding of advanced modulation and coding schemes used in modern wireless communication systems.