LAB-01

Aim: Implementing the Simple Password based Authentication

Code:

```
#include <iostream>
#include <fstream>
#include <string>
#include <map>
using namespace std;
//reading the file from the database
map<string, string> readUserPasswordsFromFile(const string &filename) {
    map<string, string> userPasswords;
    ifstream file(filename);
    if (file.is_open()) {
        std::string username, password;
        while (file >> username >> password) {
            userPasswords[username] = password;
        }
        file.close();}
    return userPasswords;
}
//storing the User in the system
void storeUserPassword(const string &filename, const string &username, const string
&password) {
    std::ofstream file(filename, ios::app);
    if (file.is_open()) {
        file << username << " " << password << endl;
        file.close();
        std::cout << "User created and stored successfully." << endl;
    } else {
        std::cerr << "Error opening file for writing." << endl;
```

```
    }}
//authenticating the user from the database
bool authenticate(const string &username, const string &password,
                const map<string, string> &userPasswords) {
    auto it = userPasswords.find(username);
    if (it != userPasswords.end() && it->second == password) {
        return true;
    }
    return false;}
int main() {
    string filename = "userpasswords.txt";
    cout << "Welcome to the Login System" << std::endl;
    string username, password;
    cout << "Do you want to create a new user? (yes/no): ";
    string createNewUser;
    cin >> createNewUser;
    if (createNewUser == "yes") {
        cout << "Enter a new username: ";
        cin >> username;
        cout << "Enter a new password: ";
        cin >> password;
        cout<<endl;
        storeUserPassword(filename, username, password);
    } else {
        cout << "Enter your username: ";
        cin >> username;
        cout << "Enter your password: ";
        cin >> password;
        map<string, std::string> userPasswords = readUserPasswordsFromFile(filename);
        if (authenticate(username, password, userPasswords)) {
```

```
        std::cout << "Login successful. Access granted." << std::endl;
    } else {
        std::cout << "Login failed. Access denied." << std::endl;
    } }
  cout<<endl;
  return 0;
}
```

Output:

```
Welcome to the Login System

Do you want to create a new user? (yes/no): no
Enter your username: lavish
Enter your password: 1234
Login failed. Access denied.

PS C:\Users\Asus\Desktop\sem 5th\crypto lab> cd "c:\Users\Asus\Desktop\sem

Welcome to the Login System

Do you want to create a new user? (yes/no): no
Enter your username: lavish
Enter your password: 1234
Login successful. Access granted.

PS C:\Users\Asus\Desktop\sem 5th\crypto lab>
```

Figure 1.1-User login

```
Welcome to the Login System

Do you want to create a new user? (yes/no): no
Enter your username: lavish
Enter your password: 1234
Login successful. Access granted.

PS C:\Users\Asus\Desktop\sem 5th\crypto lab> cd "c:\Users\Asus\Desktop\sem 5th\crypto lab

Welcome to the Login System

Do you want to create a new user? (yes/no): yes
Enter a new username: winter
Enter a new password: queen123

User created and stored successfully.

PS C:\Users\Asus\Desktop\sem 5th\crypto lab>
```

Figure 1.2-User Registration

LAB-02

Aim: Implementing the Hashed Password based Authentication

Code:

```
#include <iostream>
#include <fstream>
#include <string>
#include <map>
using namespace std;
//reading the file from the database
map<string, string> readUserPasswordsFromFile(const string &filename) {
    map<string, string> userPasswords;
    ifstream file(filename);
    if (file.is_open()) {
        std::string username, password;
        while (file >> username >> password) {
            userPasswords[username] = password;
        }
        file.close();
    }
    return userPasswords;
}

//storing the User in the system
void storeUserPassword(const string &filename, const string &username, const string
&password) {
    ofstream file(filename, ios::app);
    if (file.is_open()) {
        file << username << " " << password << endl;
        file.close();
    } else {
```

```
        std::cerr << "Error opening file for writing." << endl;
    }
}

//authenticating the user from the database
bool authenticate(const string &username, const string &password,const map<string, string>
&userPasswords) {
    auto it = userPasswords.find(username);
    if (it != userPasswords.end() && it->second == password) {
        return true;
    }
    return false;
}

int main() {
    std::hash<string> hash_obj;
    string dd;
    cout<<endl;
    cout << "     Welcome to the Registration System" << endl<<endl;
    cout<<"Want to Register yes/no"<<endl;
    cin>>dd;
    string filename = "Data.txt";
    if(dd=="yes")
    {
    string username,passwords;
    cout<<" Enter the data of the Users " <<endl<<endl;
       cout << "Enter a new username: ";
        cin >> username;
        cout << "Enter a new password: ";
        cin >> passwords;
        passwords=hash_obj(passwords);
```

```
        storeUserPassword(filename, username, passwords);
        cout<<"User successfully registered ";
    }
    cout<<endl;
    cout << "Welcome to the Login System" << endl<<endl;;
    string usernames, password;
        cout << "Enter your username: ";
        cin >> usernames;
        cout << "Enter your password: ";
        cin >> password;
        password=hash_obj(password);
        cout<<endl;
        map<string, string> userPasswords = readUserPasswordsFromFile(filename);
        if (authenticate(usernames, password, userPasswords)) {
            cout << "Login successful. Access granted." << endl;
        } else {
            cout << "Login failed. Please register." << endl;
        }
    return 0;
}
```

Output:

```
     Welcome to the Registration System

Want to Register yes/no
yes
 Enter the data of the Users

Enter a new username: bhaibhav
Enter a new password: great
User successfully registered
Welcome to the Login System

Enter your username: baibhav
Enter your password: great

Login failed. Please register.
PS C:\Users\Asus\Desktop\sem 5th\crypto lab>
```

Figure 2.1-User registration and login

LAB-03

Aim: Implementing the Hashed Password based Authentication with Salting

Code:

```
#include <iostream>
#include <fstream>
#include <string>
#include <map>
#include <cstdlib>
#include <ctime>
#include <sstream>
using namespace std;

string generateSalt() {
    srand(time(nullptr));
    int salt = rand() % 10000;
    stringstream ss;
    ss << salt;
    return ss.str();
}


map<string, pair<string, string>> readUserPasswordsFromFile(const string &filename) {
    map<string, pair<string, string>> userPasswords;
    ifstream file(filename);

    if (file.is_open()) {
        string username, salt, hashedPassword;
        while (file >> username >> salt >> hashedPassword) {
            userPasswords[username] = make_pair(salt, hashedPassword);
```

```
        }
        file.close();
    }

    return userPasswords;
}

void storeUserPassword(const string &filename, const string &username, const string &salt,
const string &hashedPassword) {
    ofstream file(filename, ios::app);
    if (file.is_open()) {
        file << username << " " << salt << " " << hashedPassword << endl;
        file.close();
    } else {
        cerr << "Error opening file for writing." << endl;
    }
}

bool authenticate(const string &username,  string &password, const map<string, pair<string,
string>> &userPasswords) {
    auto it = userPasswords.find(username);
    if (it != userPasswords.end()) {
        string salt = it->second.first;
        string storedHashedPassword = it->second.second;
        std::hash<string> hash_obj;
        password=password+salt;
        password =hash_obj(password);
        if (storedHashedPassword == password) {
            return true;
        }
    }
```

```
    return false;
}

int main() {
    std::hash<string> hash_obj;
    string filename = "salting.txt";
    string dd;
    cout << "Welcome to the Registration System" << endl << endl;
    cout << "Want to Register? (yes/no) ";
    cin >> dd;
    if (dd == "yes") {
        string username, password;
        cout << "Enter the data of the Users " << endl << endl;
        cout << "Enter a new username: ";
        cin >> username;
        cout << "Enter a new password: ";
        cin >> password;
        string salt = generateSalt();
         password=password+salt;
        password=hash_obj(password);
        storeUserPassword(filename, username, salt, password);
        cout << "User successfully registered" << endl;
    }
    cout << endl;
    cout << "Welcome to the Login System" << endl << endl;
    string username, password;
    cout << "Enter your username: ";
    cin >> username;
    cout << "Enter your password: ";
    cin >> password;
```

```
    map<string, pair<string, string>> userPasswords = readUserPasswordsFromFile(filename);
    if (authenticate(username, password, userPasswords)) {
        cout << "Login successful. Access granted." << endl;
    } else {
        cout << "Login failed. Please register." << endl;
    }
    return 0;
}
```

Output:

```
Welcome to the Login System

Enter your username: Lavish
Enter your password: 1234
Login successful. Access granted.
PS C:\Users\Asus\Desktop\sem 5th\crypto lab> cd "c:\Users\Asus\Desktop\sem 5th\cryp
Welcome to the Registration System

Want to Register? (yes/no) yes
Enter the data of the Users

Enter a new username: Shubham
Enter a new password: 111111
User successfully registered

Welcome to the Login System

Enter your username: Shubham
Enter your password: 111111
Login successful. Access granted.
PS C:\Users\Asus\Desktop\sem 5th\crypto lab>
```

Figure 3.1-User registration and login

LAB-04

Aim: To implement Challenge Response unidirectional using random nonce.

Code:

```
#include <iostream>
#include <fstream>
#include <string>
#include <map>
#include <cstdlib>
using namespace std;

map<string, string> readUserPasswordsFromFile1(const string &filename) {
    map<string, string> userPasswords;
    ifstream file(filename);
    if (file.is_open()) {
        std::string username;
        string key;
        while (file >> username >> key) {
            userPasswords[username] = key;
        }
        file.close();
    }
    return userPasswords;
}

string encryptXOR(const string &plaintext, const string &key)
{
string ciphertext = plaintext;
for (size_t i = 0; i < plaintext.size(); ++i)
{
```

```
ciphertext[i] ^= key[i % key.size()];
}
return ciphertext;}

string decryptXOR(const string &ciphertext, const string &key)
{
return encryptXOR(ciphertext, key);
}

string generateNonce()
{
string nonce(16, '\0');
for (int i = 0; i < 16; ++i)
{
nonce[i] = static_cast<char>(rand() % 256);
}
return nonce;
}

void storeUserPassword(const string &filename, const string &username, const string &key)
{
    std::ofstream file(filename, ios::app);
    if (file.is_open()) {
        file << username << " " << key << endl;
        file.close();
        std::cout << "User created and stored successfully." << endl;
    } else {
        std::cerr << "Error opening file for writing." << endl;
    }}
```

```
bool authenticate1(const string &username,const map<string, string>  &userPasswords) {
    auto it = userPasswords.find(username);
    if (it != userPasswords.end()) {
    string r=generateNonce();
    string key;
    cout<<"enter the key to encrypt the nonce :"<<r<<endl;
    cin>>key;
    string l= encryptXOR(r,key);
    string sharedkey=it->second;
    string m= decryptXOR(l,sharedkey);
    if(m==r)
    return true; }
    return false;
}

int main() {
    string filename = "challenge.txt";
    cout << "Welcome to the Login System" << std::endl;
    string username;
    string key;
    cout << "Do you want to create a new user? (yes/no): ";
    string createNewUser;
    cin >> createNewUser;
    if (createNewUser == "yes") {
        cout << "Enter a new username: ";
        cin >> username;
        cout<<"Enter the shared key";
        cin>>key;
        storeUserPassword(filename, username, key);
```

```
        cout<<endl;
    }
    else {
        cout<<endl;
        cout << "Enter your username: ";
        cin >> username;
        map<string, string> userPasswords = readUserPasswordsFromFile1(filename);
        if (authenticate1(username,userPasswords)) {
            std::cout << "Login successful. Access granted." << std::endl;
        } else {
            std::cout << "Login failed. Access denied." << std::endl;
        }
    }
   cout<<endl;
    return 0;
}
```

Output:

```
Welcome to the Login System
Do you want to create a new user? (yes/no): yes
Enter a new username: Shubham
Enter the shared key1221
User created and stored successfully.
```

Figure 4.1-User registration

```
Welcome to the Login System
Do you want to create a new user? (yes/no): no

Enter your username: Shubham
enter the key to encrypt the nonce :)#╝äßl╓«RÉI±╡╕θδ
1221
Login successful. Access granted.
```

Figure 4.2-User Authentication

LAB-05

Aim: To implement Challenge Response bidirectional using random nonce.

Code:

```
#include <iostream>
#include <fstream>
#include <string>
#include <map>
#include <cstdlib>
using namespace std;

map<string, string> readUserPasswordsFromFile1(const string &filename) {
    map<string, string> userPasswords;
    ifstream file(filename);

    if (file.is_open()) {
        std::string username;
        string key;
        while (file >> username >> key) {
            userPasswords[username] = key;
        }
        file.close();
    }




    return userPasswords;
}

string encryptXOR(const string &plaintext, const string &key)
{
```

```
string ciphertext = plaintext;
for (size_t i = 0; i < plaintext.size(); ++i)
{
ciphertext[i] ^= key[i % key.size()];
}
return ciphertext;
}

string decryptXOR(const string &ciphertext, const string &key)
{
return encryptXOR(ciphertext, key);
}



string generateNonce()
{
string nonce(16, '\0');
for (int i = 0; i < 16; ++i)
{
nonce[i] = static_cast<char>(rand() % 256);
}
return nonce;
}

void storeUserPassword(const string &filename, const string &username, const string &key)
{
    std::ofstream file(filename, ios::app);
    if (file.is_open()) {
        file << username << " " << key << endl;
        file.close();
        std::cout << "User created and stored successfully." << endl;
```

```
    } else {
        std::cerr << "Error opening file for writing." << endl;
    }
}



bool authenticate1(const string &username,const map<string, string>  &userPasswords) {

    auto it = userPasswords.find(username);
    if (it != userPasswords.end()) {
    string nonce1=generateNonce();
    string key;
    cout<<"enter the key to encrypt the nonce sent by the verifier :"<< nonce1 <<endl;
    cin>>key;

    string nonce2=generateNonce();
    cout<<"Sending a nonce from the claimant side :"<<nonce2<<endl;
    string mixednonce=nonce1+"/"+nonce2;
    string encrypted1= encryptXOR(mixednonce,key);



    string sharedkey=it->second;
    string m1= decryptXOR(encrypted1,sharedkey);
    string one="",two="";
    int l=0;
    for(int i=0;i<m1.size();i++)
    {   if(m1[i]=='/')
        {l=1;
         continue;
```

```
    }
    if(l==0)
    {
        one+=m1[i];
    }
    else{
      two+=m1[i];
    }
}

if(nonce1!=one)
return false;
string mixednonce2=two+"/"+one;
string encrypted2=encryptXOR(mixednonce2,sharedkey);
string m2= decryptXOR(encrypted2,key);
one="";
two="";
 l=0;
for(int i=0;i<m2.size();i++)
{   if(m2[i]=='/')
    {l=1;
     continue;
    }
    if(l==0)
    {
        one+=m2[i];
    }
    else{
      two+=m2[i];
    }
```

```
    }

    if(nonce2!=one)
    return false;



    return true;
    }
    return false;
}

int main() {
    string filename = "challenge2.txt";

    cout << "Welcome to the Login System" << std::endl;

    string username;
    string key;
    cout << "Do you want to create a new user? (yes/no): ";
    string createNewUser;
    cin >> createNewUser;

    if (createNewUser == "yes") {
        cout << "Enter a new username: ";
        cin >> username;
        cout<<"Enter the shared key";
        cin>>key;
        storeUserPassword(filename, username, key);
        cout<<endl;
    }
```

```
    else {
        cout<<endl;
        cout << "Enter your username: ";
        cin >> username;
        map<string, string> userPasswords = readUserPasswordsFromFile1(filename);
        if (authenticate1(username,userPasswords)) {
            std::cout << "Login successful. Access granted." << std::endl;
        } else {
            std::cout << "Login failed. Access denied." << std::endl;
        }
    }
   cout<<endl;
    return 0;
}
```

```
PS C:\Users\Asus\Desktop\sem 5th\crypto lab> cd "c:\Users\Asus\Desktop\sem 5th\crypto lab\" ; if ($?) { g++
Welcome to the Login System
Do you want to create a new user? (yes/no): yes
Enter a new username: lavish
Enter the shared key2222
User created and stored successfully.


PS C:\Users\Asus\Desktop\sem 5th\crypto lab> cd "c:\Users\Asus\Desktop\sem 5th\crypto lab\" ; if ($?) { g++
Welcome to the Login System
Do you want to create a new user? (yes/no): no

Enter your username: lavish
enter the key to encrypt the nonce sent by the verifier :)#╝äßl╓«RÉI╪╪╗θδ
2222
Sending a nonce from the claimant side :|ª<ç
L♠╗G
Login successful. Access granted.

PS C:\Users\Asus\Desktop\sem 5th\crypto lab>
```

Figure 5.1-User registration and login

LAB-05

Aim: To implement Challenge Response using Timestamp.

Code

```
#include <iostream>
#include <fstream>
#include <map>
#include <string>
#include <sstream>
#include <functional>
#include <ctime> // For timestamp
using namespace std;
string encryptXOR(const string &plaintext, const string &key)
{
string ciphertext = plaintext;
for (size_t i = 0; i < plaintext.size(); ++i)
{
ciphertext[i] ^= key[i % key.size()];
}
return ciphertext;
}
string decryptXOR(const string &ciphertext, const string &key)
{
return encryptXOR(ciphertext, key);
}
string generateTimestamp()
{
time_t now = time(0);
tm* timeInfo = localtime(&now);
char buffer[20]; // Format: "YYYYMMDDHHMMSS"
strftime(buffer, sizeof(buffer), "%Y%m%d%H%M%S", timeInfo);
```

```
return string(buffer);
}
int main()
{
string key, clientname;
ifstream inFile("challenge.txt");
if (!inFile.is_open()) {
cout << "Error opening file for reading." << endl;
return 1;
}
string line;
if (getline(inFile, line)) {
key = line;
} else {
cout << "Key not found in the file." << endl;
return 1;
}
cout << "Enter the claimant name: ";
cin >> clientname;
string timestamp = generateTimestamp();
cout << "The claimant sent its name and time stamp to the verifier: " <<
clientname <<
timestamp << endl;
string dataToEncrypt = clientname + timestamp;
string encryptedData = encryptXOR(dataToEncrypt, key);
cout << "Claimant: Encrypted name and timestamp sent to verifier using symmetric key: " <<
encryptedData << endl;
string receivedData = encryptedData;
string decryptedData = decryptXOR(receivedData, key);
cout << "Verifier: Received and decrypted name and timestamp: " <<
decryptedData <<
```

```
endl;
// Extract name and timestamp from decrypted data
string receivedName = decryptedData.substr(0, clientname.length());
string receivedTimestamp = decryptedData.substr(clientname.length(), 14);
string timestamp2 = generateTimestamp();
cout<<"verifier timestamp "<<timestamp2<<endl;
if ((receivedName == clientname) && (timestamp2==receivedTimestamp))
{
cout << "Name and time stamp verified" << endl;
}
else
{
cout << "Name and timestamp verification failed." << endl;
}
inFile.close();
return 0;
}
```

Output

```
Enter the claimant name: shivam
The claimant sent its name and time stamp to the verifier: shivam20230927181225
Claimant: Encrypted name and timestamp sent to verifier using symmetric key: ▼  ▼▼↕♠↕♥♥♠USAXKY↕♠
Verifier: Received and decrypted name and timestamp: shivam20230927181225
verifier timestamp 20230927181225
Name and time stamp verified
PS C:\Users\Asus\Desktop\sem 5th\crypto lab>
```