Semester	T.E. Semester VI – Computer Engineering
Subject	Cryptography and cyber security
Subject Professor In-charge	Prof. Amit Nerurkar
Assisting Teachers	Prof. Amit Nerurkar
Laboratory	M312B

Student Name	Deep Salunkhe
Roll Number	21102A0014
TE Division	A

Title:

Design and Implementation of HMAC

Explanation:

HMAC (Hash-based Message Authentication Code):

- **Definition**: HMAC is a mechanism for verifying the authenticity and integrity of a message using a cryptographic hash function and a secret key.
- **Algorithm**: HMAC involves a hash function (such as SHA-256) and a secret key shared between the sender and receiver.
- **Key Components**:
    - **Message**: The data that needs to be authenticated.
    - **Secret Key**: A shared secret known only to the sender and receiver.
    - **Hash Function**: A cryptographic hash function used to generate a fixed-size hash value from the input data.
- **Process**:

1. The sender computes a hash-based message authentication code using the message and the secret key.

2. The receiver independently computes the HMAC using the received message and the shared secret.

3. The receiver compares the computed HMAC with the received HMAC. If they match, the message is considered authentic and intact.

Advantages of HMAC:

- **Security**: HMAC provides strong security guarantees against message tampering and forgery.
- **Efficiency**: It offers efficient verification of message integrity without transmitting the entire message.
- **Flexibility**: HMAC can be implemented using various hash functions, allowing for flexibility in choosing the appropriate algorithm for the application.

Implementation:

```
#include <functional>
#include <iostream>
#include <string>
#include <vector>
using namespace std;

size_t stringHashing(string s)
{

    // Get the string
    // to get its hash value
    string hashing1 = s;

    // Instantiation of Object
    hash<string> mystdhash;

    // Using operator() to get hash value



    size_t ans=mystdhash(hashing1);
    return ans;

}

string encryption(string s){

    int n=s.size();
    string encry_s="";
    for(int i=0;i<n;i++){
        char temp=s[i]+1;
        encry_s= encry_s + (temp);
    }

    return encry_s;
}

string decryption(string s){
```

```
    int n=s.size();
    string decry_s="";
    for(int i=0;i<n;i++){
        char temp=s[i]-1;
        decry_s= decry_s + (temp);
    }

    return decry_s;
}

void manupulate(string &s){
    s[0]=s[0]+1;
}

int main(){

    string message;

    cin>>message;

    size_t hashed_message_int=stringHashing(message);
    string hashedMessage=to_string(hashed_message_int);
    string encrypt_hash=encryption(hashedMessage);
    string encrypt_message=encryption(message);

    cout<<"The sender side"<<endl;

    cout<<"The orignal message"<<endl;
    cout<<message<<endl;
    cout<<"the hashed message"<<endl;
    cout<<hashedMessage<<endl;
    cout<<"the encrypted hashed message "<<endl;
    cout<<encrypt_hash<<endl;
    cout<<"the encrypted  message "<<endl;
    cout<<encrypt_message<<endl;
    cout<<endl;
    cout<<endl;
    cout<<endl;


    cout<<"The receiver side"<<endl;
```

```
    cout<<"Do you want to manupulate the data"<<endl;
    cout<<"1=>Yes"<<endl;
    cout<<"2=>NO"<<endl;

    int t;
    cin>>t;

    if(t==1){
        manupulate(encrypt_message);
    }

    string decrypted_message= decryption(encrypt_message);
    string decrypted_hash=decryption(encrypt_hash);
    string hashed_decrypted_message=to_string(stringHashing(decrypted_message));
    cout<<"decrypted message"<<endl;
    cout<<decrypted_message<<endl;
    cout<<"decrypted hash"<<endl;
    cout<<decrypted_hash<<endl;
    cout<<"Hashed decrypted message"<<endl;
    cout<<hashed_decrypted_message<<endl;









    if(hashed_decrypted_message==decrypted_hash){
        cout<<"correct message"<<endl;
    }else{
        cout<<"incorrect message"<<endl;
    }



    return 0;
}
```

Conclusion:

In your lab work on HMAC (Hash-based Message Authentication Code), you've implemented a simple demonstration of how HMAC can be used for message integrity verification. Let's delve into some theory and then provide a conclusion you can include in your lab report.

Theory:

HMAC (Hash-based Message Authentication Code):

- **Definition**: HMAC is a mechanism for verifying the authenticity and integrity of a message using a cryptographic hash function and a secret key.
- **Algorithm**: HMAC involves a hash function (such as SHA-256) and a secret key shared between the sender and receiver.
- **Key Components**:
    - **Message**: The data that needs to be authenticated.
    - **Secret Key**: A shared secret known only to the sender and receiver.
    - **Hash Function**: A cryptographic hash function used to generate a fixed-size hash value from the input data.
- **Process**:

1. The sender computes a hash-based message authentication code using the message and the secret key.

2. The receiver independently computes the HMAC using the received message and the shared secret.

3. The receiver compares the computed HMAC with the received HMAC. If they match, the message is considered authentic and intact.

Advantages of HMAC:

- **Security**: HMAC provides strong security guarantees against message tampering and forgery.

- **Efficiency**: It offers efficient verification of message integrity without transmitting the entire message.
- **Flexibility**: HMAC can be implemented using various hash functions, allowing for flexibility in choosing the appropriate algorithm for the application.

Conclusion:

In conclusion, my implementation demonstrates the practical application of HMAC for ensuring message integrity in communication systems. By combining a hash function with a secret key, HMAC provides a reliable mechanism for verifying the authenticity of transmitted data. Through this lab work, I 've gained hands-on experience in implementing HMAC, understanding its key components, and evaluating its effectiveness in detecting message tampering. Overall, HMAC emerges as a valuable tool in maintaining the security and trustworthiness of communication protocols, offering robust protection against unauthorized alterations to transmitted information.