UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

ESCUELA DE CIENCIAS Y SISTEMAS

SISTEMAS OPERATIVOS 2

SECCIÓN A

A-5.1 - Encripción extremo a extremo

Nombre: Alvaro Emmanuel Socop Perez **Carne:** 202000194

Guatemala, 24 de Junio del 2023

PROGRAMA:

Dada las siguiente clase:

```
class Criptografo {

public:

  Criptografo (string llvPub, string llvPriv) ;

   /** genera el criptograma para enviar un mensaje certificado usando las llaves propias y la llave del destinatario llvDestinatario */

  string enviar (string llvDestinatario, string mensaje) ;

   /** Recibe un criptograma certificado usando las llaves propias y la llave del remitente llvRemitente y genera el mensaje recibido*/

  string recibir (string llvRemitente, string criptograma) ;

private:

  string llvPublica, llvPrivada ;

   /** devuelve el criptograma correspondiente a la encripción RSA del mensaje con la llave llv pública o privada */

  string encripta(string llv, string mensaje) ;

   /** devuelve el mensaje correspondiente a la desencripción RSA del criptograma con la llave llv pública o privada */

  string desencripta (string llv, string criptograma) ;

}
```

Implemente el proceso de encripción extremo a extremo (E2E) de forma que una persona X envíe un mensaje a una persona Y, de modo que X esté seguro que sólo Y podrá leer el mensaje y Y esté seguro que sólo X pudo haber enviado el mensaje, a través de desarrollar;

- ***a.*** El método ***enviar()***
- ***b.*** El método ***recibir()***
- c. El programa que usará X para encriptar el mensaje (programaX.cpp) utilizando la clase *Criptografo*.
- d. El programa que usará Y para desencriptar el mensaje (programaY.cpp) utilizando la clase *Criptografo*.

Solución

Programa X:

```cpp
#include <iostream>
#include "Cryptograph.h"
// Alvaro Emmanuel Socop Perez 202000194
using namespace std;
// This program is the sender
int main()
{
    // Create a Cryptograph object for X.
    Cryptograph x("llvPubX", "llvPrivX");

    // Get the recipient's public key.
    string llvReceiverPub = "llvPubY";
    cout << "————————— WELCOME PROGRAM X ————————————\n";
    // Create a message.
    string message = "Sistemas_Operativos_2_JUNIO_2023";

    // Encrypt the message and get the cryptogram.
    string cryptogram = x.send(llvReceiverPub, message);

    // Print the cryptogram.
    cout << "The cryptogram generated is: \n"
         << cryptogram << endl;

    return 0;
}
```

Genera un codigo de una palabra clave seteada:

```
┌──(droid㉿kali)-[~/Documents/Sistemas_Operativos_2/Tarea Criptografia 202000194]
└─$ ./programaX
------------- WELCOME PROGRAM X -----------------
The cryptogram generated is:
Vlvwhpdv_Rshudwlyrv_2_MXQLR_2023

┌──(droid㉿kali)-[~/Documents/Sistemas_Operativos_2/Tarea Criptogr https://amqbiq6t
└─$
```

Programa Y:

```cpp
#include <iostream>
#include "Cryptograph.h"
// Alvaro Emmanuel Socop Perez 202000194
using namespace std;

int main()
{
    // Create a Cryptograph object for Y.
    Cryptograph y("llvPubY", "llvPrivY");

    // Get the sender's public key.
    string llvSenderPub = "llvPubX";

    // Get the cryptogram.
    string cryptogram = "The cryptogram is: ";
    cout << "————————— WELCOME PROGRAM Y ————————————\n";
    cout << "Enter the cryptogram: ";
    cin >> cryptogram;

    // Decrypt the cryptogram and get the message.
    string message = y.receive(llvSenderPub, cryptogram);

    // Print the message.
    cout << "\nThe message is: \n"
         << message << endl;

    return 0;
}

```

Al tener esa encriptacion el programa Y lo desencripta:

```
┌──(droid㉿kali)-[~/Documents/Sistemas_Operativos_2/Tarea Criptografia 202000194]
└─$ ./programaY
------------ WELCOME PROGRAM Y -----------------
Enter the cryptogram: Vlvwhpdv_Rshudwlyrv_2_MXQLR_2023

The message is:
Sistemas_Operativos_2_JUNIO_2023
```

Cryptograph.h:

```cpp
Cryptograph::Cryptograph(std::string llvPub, std::string llvPriv)
{
  llvPublic = llvPub;
  llvPrivate = llvPriv;
}

std::string Cryptograph::send(std::string llvReceiver, std::string message)
{
  // Generate the certified cryptogram using own keys and recipient's key
  std::string criptograma = encrypt(llvReceiver, message);

  // Return the generated cryptogram
  return criptograma;
}

std::string Cryptograph::receive(std::string llvSender, std::string cryptogram)
{
  // Receive the certified cryptogram using own keys and sender's key
  std::string mensaje = decrypt(llvSender, cryptogram);

  // Return the received message
  return mensaje;
}

```

```cpp

std::string Cryptograph::encrypt(std::string llv, std::string message)
{
  // Implementation of RSA encryption using the specified llv (public or private key)
  // Here's a simple example using a basic Caesar cipher encryption

  std::string encryptedMessage = "";
  int shift = 3; // Number of positions to shift each character

  for (char c : message)
  {
    if (std::isalpha(c))
    {
      char base = std::isupper(c) ? 'A' : 'a';
      c = (c - base + shift) % 26 + base;
    }
    encryptedMessage += c;
  }

  // Return the encrypted cryptogram
  return encryptedMessage;
}

```

```cpp

std::string Cryptograph::decrypt(std::string llv, std::string cryptogram)
{
  // Implementation of RSA decryption using the specified llv (public or private key)
  // Here's the corresponding decryption for the Caesar cipher encryption used in encripta()

  std::string decryptedMessage = "";
  int shift = 3; // Number of positions to shift each character

  for (char c : cryptogram)
  {
    if (std::isalpha(c))
    {
      char base = std::isupper(c) ? 'A' : 'a';
      c = (c - base - shift + 26) % 26 + base;
    }
    decryptedMessage += c;
  }

  // Return the decrypted message
  return decryptedMessage;
}
```