ICB0009-UF1-PR01

Nombre del alumno

Practica entregable (Enunciado y entregable)

Guía para el alumno

El alumno debe de entregar la práctica enunciada en este documento antes del cierre programado en el calendario. La nota y corrección de la práctica se publicará en la plataforma en un plazo no superior a 10 días hábiles a contar desde la fecha límite de entrega.

Los entregables son:

- Este mismo documento incluyendo las respuestas solicitadas en el mismo:
    - [CSnn]: son capturas de pantalla que demuestran el resultado de una práctica. Para realizar la captura se utilizará la tecla "imp pant" o equivalente del teclado, y luego al final de documento y dentro de la página en blanco habilitada para cada captura, se realiza el "pegado" o "paste". Si fuera necesario se ajustará el tamaño.
    - [DIBnn]: Son tablas a completar o escribir texto.

  Utilizar los espacios habilitados al final de este documento para la inserción de las capturas y texto.

  El documento entregado tendrá el siguiente nombre:

ICB0009-UF1-PR01-"username".doc "username" = nombre de usuario del alumno en la plataforma Ejemplo: ICB0009-UF1-PR01-raulgarciaflores.doc

- Proyectos de Visual Studio Code con la solución programada, con el siguiente nombre. Se proporcionará mediante un fichero comprimido tipo .rar, .zip o similar.

ICB0009-UF1-PR01-"username" "username" = nombre de usuario del alumno en la plataforma Ejemplo: ICB0009-UF1-PR01-raulgarciaflores.rar

La práctica está formada por 2 partes.

Enunciado general

El objetivo de la práctica es simular los dos principales conceptos que se han trabajado a nivel teórico en la UF:

- Gestión de credenciales para autenticación (registro / login)
- Comunicación segura entre un emisor y un receptor.

Se proporciona un código previo que contiene la estructura del proyecto a desarrollar. Hay una gran cantidad de código ya programada y la práctica consiste en completarlo correctamente así como responder las preguntas que se formulan en este enunciado.

El código facilitado se describe a continuación:

Estructura de programa inicial

El programa proporcionado para realizar la práctica contiene ya diversos proyectos que os pueden ayudar a completar los diferentes ejercicios. Los explicamos:

- ClaveAsimetricaClass: Es una clase que contiene la definición ya programada de diversos métodos que nos van a permitir realizar las principales acciones que se pueden realizar con la criptografía asimétrica.
- ClaveSimetricaClass: Es una clase que contiene la definición ya programada de diversos métodos que nos van a permitir realizar las principales acciones que se pueden realizar con la criptografía simétrica.
- SimulacionEnvioRecepción. Es el programa principal donde realizaremos la gestión del registro / login así como la simulación de envío y recepción de un mensaje utilizando los criterios teóricos vistos en clase y que se explican a continuación.

  Este proyecto ya contiene indicaciones de los pasos a seguir y que debéis programar debidamente.

  Es en esta parte donde debemos acabar de realizar la programación de la práctica.

Parte #1 Registro y login con seguridad

La primera parte de la práctica consiste en completar correctamente un proceso de Registro / Login de los clientes.

Como es lógico un Registro / Login requiere de alguna forma de persistencia, normalmente una base de datos, para guardar los datos del registro de los usuarios y poderlos loguear posteriormente. En este caso, al no querer añadir complejidad a la práctica se ha optado por guardar la información de registro en variables.

Disponemos de la variable UserName donde guardaremos el nombre de usuario y la variable SecurePass donde guardaremos el password ya protegido. Si lo requieres puedes crear otras variables necesarias para el proceso de login / password.

Actualmente el sistema de Registro / Login ya está programado, en el lado del Cliente, pero no usa ningún tipo de seguridad, se guardan los datos de password en texto plano.

Los requisitos de esta parte a desarrollar serán:

- Modificar el registro para guardar los datos de password con les mecanismos de seguridad explicados en la teoría.
- Modificar el sistema de login para poder loguear correctamente a los usuarios según el proceso de registro anterior.

Etapa 1: Completar el registro

Para completar el registro el programa pregunta al usuario que entre su nombre de usuario y lo guarda en la variable UsenName. Posteriormente pregunta el password.

En este punto debemos realizar la programación para poder guardar el password de manera segura.

En caso de que lo necesites puedes generar nuevas variables globales para almacenar información que se necesite posteriormente para el login.

(2 puntos)

Etapa 2: Realizar login

En la etapa 2 modificaremos el método login para loguear correctamente a un usuario. Este método en primer lugar pregunta nombre de usuario y password.

Para el nombre de usuario simplemente compara con el que introdujo en el Registro.

Para el password deberemos realizar las acciones necesarias para comprobar que el password entrado es el correcto. En caso de que así sea se genera un boolean *true* y si el login no es correcto se genera un booleano *false*.

(2 puntos)

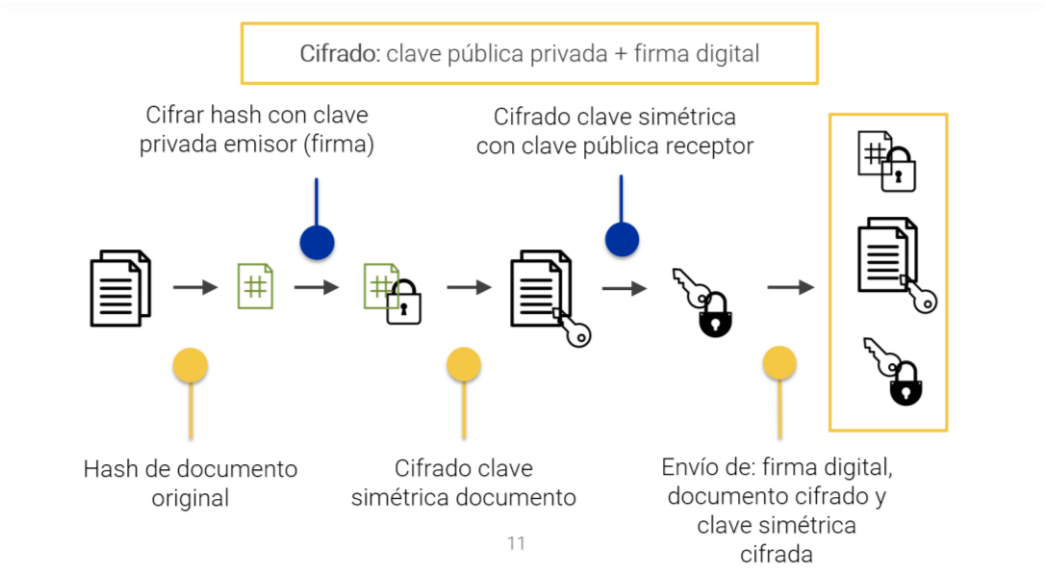
[DIB01] Explica el mecanismo de Registro / Login utilizado (máximo 5 líneas)

Durante el registro se guarda el usuario y la contraseña usando un hash SHA256. Esto evita guardar el password en texto plano. En el login se vuelve a calcular el hash de la contraseña introducida y se compara con el guardado. Si coinciden, el acceso es válido. El proceso evita fugas de información sensible.

Parte #2 Simulación de envío y recepción de información de forma segura

En la segunda parte de la práctica simularemos un envío y recepción de un mensaje entre emisor y receptor utilizando las técnicas vistas en teoría.

El envío de información se realizará mediante los siguientes pasos del lado del emisor:



Básicamente son tres pasos los que hay que hacer, aunque algunos estén divididos en más subpasos. Los tres pasos del lado del cliente son:

- Firma del mensaje original
- Cifrado del mensaje con clave simétrica
- Cifrado de la clave simétrica con criptografía asimétrica.

En este punto, en una comunicación real, el emisor enviaría al receptor los siguientes datos:

- Firma
- Mensaje cifrado
- Clave simétrica cifrada

Del lado del receptor se tendrán que hacer los siguientes pasos:

12

Básicamente también son tres:

- Descifrado de la clave simétrica
- Descifrado del mensaje
- Comprobación de la firma.

Todos estos pasos aquí descritos son los que ya se indican en el Main del proyecto, para desarrollar:

```
static void Main(string[] args)
{
 //LADO EMISOR
 //Firmar mensaje
 //Cifrar mensaje
 //Cifrar clave simétrica
 //Datos que el emisor "envía" al receptor
 Console.WriteLine("Firma: {0}", BytesToStringHex(Firma));
 Console.WriteLine("Texto cifrado: {0}", BytesToStringHex(TextoCifrado));
 Console.WriteLine("Clave simetrica cifrada (Key): {0}",
 BytesToStringHex(ClaveSimetricaKeyCifrada));
 Console.WriteLine("Clave simetrica cifrada (IV): {0}",
 BytesToStringHex(ClaveSimetricaIVCifrada));
 //LADO RECEPTOR
 //Descifrar clave simétrica
 //Descifrar mensaje
 //Comprobar firma y mostrar mensaje descifrado si la firma es correcta
}
```

ATENCION: Muy importante:

La programación de la simulación de envío y recepción se tiene que hacer siguiendo los pasos descritos anteriormente. Se podría dar el caso que se consiga cifrar y descifrar el menaje de manera aparentemente correcta, pero si no se siguen los pasos de manera lógica, no se considerará como correcta, aunque el resultado sea la visualización del mensaje descifrado correctamente.

Las classes ClaveAsimetricaClass y ClaveSimetricaClass contienen diferentes métodos que permiten realizar las acciones descritas anteriormente y ya están programados.

Por lo tanto el ejercicio consiste "solo" en utilizar estos métodos correctamente en el programa principal para realizar la simulación del envío y recepción de información correctamente, tal y como se indica en la teoría de la UF y en este enunciado.

Vamos a explicar con un poco más de detalle el programa principal:

Tal y como ya se explicó en el apartado de introducción, en esta parte utilizaremos las clases anteriores y sus métodos para realizar los diferentes pasos del envío y recepción de un mensaje de manera segura.

Existen algunas variables globales que ya se definen para su uso posterior y que son las siguientes:

```
static ClaveAsimetrica Emisor = new ClaveAsimetrica();
static ClaveAsimetrica Receptor = new ClaveAsimetrica();
static ClaveSimetrica ClaveSimetricaEmisor = new ClaveSimetrica();
static ClaveSimetrica ClaveSimetricaReceptor = new ClaveSimetrica();

static byte[] Firma;
static byte[] ClaveSimetricaKeyCifrada;
static byte[] ClaveSimetricaIVCifrada;
byte [] TextoCifrado;

static string TextoAEnviar = "Me he dado cuenta que incluso las personas que
dicen que todo está predestinado y que no podemos hacer nada para cambiar
nuestro destino igual miran antes de cruzar la calle. Stephen Hawking.";
```

ClavesAsimetrica Emisor: es la variable que contiene la RSA y clave pública del emisor.

ClavesAsimetrica Receptor: es la variable que contiene la RSA y clave pública del receptor.

ClaveSimetrica ClaveSimetricaEmisor: contiene la clave simétrica que "ve" el emisor y que "no ve" el receptor, es decir, esta variable no se puede utilizar en el LADO RECEPTOR.

ClaveSimetrica ClaveSimetricaReceptor: contiene la clave simétrica que "ve" el receptor y que "no ve" el emisor, es decir, esta variable no se puede utilizar en el LADO EMISOR.

Firma, ClaveSimetricaKeyCifrada, ClaveSimetricaIVCifrada, TextoCifrado son los datos que simulamos que se envían del emisor al receptor.

TextoAEnviar es el texto que queremos enviar de manera segura.

La programación de esta parte de la práctica consistirá en seis pasos, 3 del lado del emisor y 3 del lado del receptor:

LADO EMISOR

1. Firmar mensaje: El emisor firmará el mensaje original siguiendo el procedimiento explicado en la teoría, utilizando el objeto Emisor.

2. Cifrar mensaje: El emisor cifra el mensaje original utilizando criptografía simétrica.

3. Cifrar clave simétrica: El emisor cifrará la clave simétrica utilizando el procedimiento correcto de criptografía asimétrica.

LADO RECEPTOR

4. Descifrar clave simétrica: El receptor descifrará la clase simétrica "enviada" por el emisor.

5. Descifrar mensaje: Con la clave simétrica descifrada, descifrará el mensaje cifrado "enviado" por el emisor.

6. Comprobar firma y mostrar mensaje descifrado si la firma es correcta: Una vez descifrado el mensaje, se comprobará su validez utilizando la firma "enviada" por el emisor.

Para facilitar el desarrollo de esta parte, así como para facilitar la corrección se recomienda utilizar el método siguiente para poder visualizar en formato string las cadenas de bytes ya que, como sabéis, todal información que manejan las librerías criptográficas se trata en formato byte.

```
static string BytesToStringHex (byte[] result)
{
    StringBuilder stringBuilder = new StringBuilder();

    foreach (byte b in result)
        stringBuilder.AppendFormat("{0:x2}", b);

    return stringBuilder.ToString();
}
```

Realiza una pequeña explicación de cada uno de los pasos que has hecho especificando el procedimiento que empleas en cada uno de ellos.

1. El emisor firma el mensaje con su clave privada RSA.
2. El mensaje se cifra con clave simétrica (AES).
3. La clave y el IV del cifrado se cifran con la clave pública del receptor (RSA).
4. El receptor descifra la clave simétrica con su clave privada RSA.

5. Con esa clave, descifra el mensaje simétricamente.
6. Valida la firma con la clave pública del emisor.

Esta parte corresponde a 5 puntos de la nota global de 10 puntos.

Pregunta:

Una vez realizada la práctica, ¿crees que alguno de los métodos programado en la clase asimétrica se podría eliminar por carecer de una utilidad real?

No, todos los métodos de la clase asimétrica tienen sentido. Algunos permiten trabajar con claves públicas externas (como en una red real), mientras que otros son necesarios para cifrar, firmar o validar. Aunque algunos no se usen en esta simulación, son útiles en otros contextos reales.

Esta parte corresponde a 1 punto de la nota global de 10 puntos.

ENLACE GIT HUB: https://github.com/NuriaRodvin/M09-01.git