Práctica 2

Criptosistemas asimétricos

Introducción

Continuamos con el estudio de OpenSSL. En esta práctica nos centraremos en criptosistemas asimétricos. OpenSSL implementa de forma nativa RSA y criptosistemas basados en curvas elípticas.

RSA: Para realizar tareas con RSA disponemos de tres comandos. El primero de ellos,

```
$> openssl genrsa
```

se emplea para generar un par de claves pública/privada que se almacenan de forma conjunta. Las claves pueden estar protegidas por contraseña, en cuyo caso se cifran utilizando un criptosistema simétrico usando una clave derivada de la contraseña, o sin proteger. Dependiendo del uso, será conveniente el uso de la contraseña o no.
El segundo,

```
$> openssl rsa
```

se emplea para manipular claves generadas. Realiza conversiones de formato, extracción de parte pública y privada, adición o sustracción de contraseñas, etc.
El tercero,

```
$> openssl rsautl
```

permite utilizar varias acciones asociadas a las claves generadas, entre ellas cifrar, descrifrar, firmar y verificar. Estas últimas tratan sobre firma digital que será tratada en temas posteriores.
En la página manual, que también se encuentra en la dirección

`https://www.openssl.org/docs/man1.0.2/apps/`,

podéis leer todas las opciones de estos comandos.

CURVAS ELÍPTICAS: Para trabajar con curvas elípticas, disponemos de dos comandos concretos. El primero es

```
$> openssl ecparam
```

que se emplea para generar y manipular parámetros asociados a las curvas (como el tipo de curva y el punto base) y parejas de claves privadas/públicas. Los parámetros y las claves pueden generarse en una sola ejecución o en dos. Realizar la generación en dos pasos tiene utilidad cuando varios usuarios quieren emplear claves asociadas a los mismos parámetros, como ocurre en protocolos de intercambio de claves. El segundo comando es

```
$> openssl ec
```

cuya función es similar a `openssl rsa`. No está implementado en OpenSSL el cifrado y descifrado de ElGamal con aritmética modular o curvas elípticas.

NOTA: Para algunas de las tareas será necesario generar claves aleatorias de sesión. Esto puede hacerse de múltiples formas, pero OpenSSL también incluye un comando

```
$> openssl rand
```

que puede ser utilizado para ello.
Para observar los valores de las claves en forma "comprensible", hay que emplear las opciones `-text` y `-noout`. En esta práctica solo utilizaremos el formato `PEM`, que genera archivos texto fácilmente visualizables. El formato `DER` genera archivos binarios, y su uso es completamente análogo.
Cuando sea requerida una contraseña, recomiendo usar la cadena `'0123456789'`.

Tareas a realizar

- Generad, cada uno de vosotros, una clave RSA (que contiene el par de claves) de 768 bits. Para referirnos a ella supondré que se llama `nombreRSAkey.pem`. Esta clave no es necesario que esté protegida por contraseña.
- "Extraed" la clave privada contenida en el archivo `nombreRSAkey.pem` a otro archivo que tenga por nombre `nombreRSApriv.pem`. Este archivo deberá estar protegido por contraseña cifrándolo con `AES-128`. Mostrad sus valores.

- Extraed en `nombreRSApub.pem` la clave pública contenida en el archivo `nombreRSAkey.pem`. Evidentemente `nombreRSApub.pem` no debe estar cifrado ni protegido. Mostrad sus valores.
- Reutilizaremos el archivo binario `input.bin` de 1024 bits, todos ellos con valor 0, de la práctica anterior.
- Intentad cifrar `input.bin` con vuestras claves pública. Explicad el resultado.
- Diseñad un cifrado híbrido, con RSA como criptosistema asimétrico. El modo de proceder será el siguiente:
  1. El emisor debe seleccionar un sistema simétrico con su correspondiente modo de operación.
  2. El emisor generará un archivo de texto, llamado por ejemplo `sessionkey` con dos líneas. La primera línea contendrá una cadena aleatoria hexadecimal cuya longitud sea la requerida por la clave. OpenSSL permite generar cadenas aleatorias con el comando `openssl rand`. La segunda línea contendrá la información del criptosistema simétrico seleccionado. Por ejemplo, si hemos decidido emplear el algoritmo Blowfish en modo ECB, la segunda línea debería contener `-bf-ecb`.
  3. El archivo `sessionkey` se cifrará con la clave pública del receptor.

4. El mensaje se cifrará utilizando el criptosistema simétrico, la clave se generará a partir del archivo anterior mediante la opción `-pass file:sessionkey`.

- Utilizando el criptosistema híbrido diseñado, cada uno debe cifrar el archivo `input.bin` con su clave pública para, a continuación, descifrarlo con la clave privada. comparad el resultado con el archivo original.
- Generad un archivo `stdECparam.pem` que contenga los parámetros públicos de una de las curvas elípticas contenidas en las transparencias de teoría. Si no lográis localizarlas haced el resto de la práctica con una curva cualquiera a vuestra elección de las disponibles en OpenSSL. Mostrad los valores.
- Generad cada uno de vosotros una clave para los parámetros anteriores. La clave se almacenará en `nombreECkey.pem` y no es necesario protegerla por contraseña.
- "Extraed" la clave privada contenida en el archivo `nombreECkey.pem` a otro archivo que tenga por nombre `nombreECpriv.pem`. Este archivo deberá estar protegido por contraseña cifrándolo con 3DES. Mostrad sus valores.
- Extraed en `nombreECpub.pem` la clave pública contenida en el archivo `nombreECkey.pem`. Como antes

`nombreECpub.pem` no debe estar cifrado ni protegido. Mostrad sus valores.

NOTA: Debéis entregar un PDF describiendo todas las tareas realizadas, incluyendo en él los archivos empleados y generados. No es necesario enviar dichos archivos, pero debéis conservarlos hasta que salga la evaluación de la práctica por si os son requeridos. Las tareas asociadas a la manipulación de claves RSA valen 2 puntos. Las asociadas al cifrado y descifrado con RSA, 5 puntos. La generación de claves para curvas elípticas vale 3 puntos.