Práctica 3

Protocolos criptográficos

Introducción

En esta práctica vamos a introducir DSA y funciones Hash. Con ellas vamos a realizar tareas de firma digital y a implementar un intercambio de claves mediante el protocolo Estación a Estación.

DSA: De manera similar a las curvas elípticas, tenemos tres comandos para trabajar con claves DSA. El primero de ellos,

```
$> openssl dsaparam
```

se emplea para generar y manipular los parámetros asociados (primos p, q y el generador g) 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. La generación de claves a partir de los parámetros puede hacerse también con

```
$> openssl gendsa
```

Finalmente el comando

```
$> openssl dsa
```

se emplea para manipular claves generadas de forma completamente análoga a los comandos equivalentes para RSA y EC. Realiza conversiones de formato, extracción de parte pública y privada, adición o sustracción de contraseñas, etc.

FUNCIONES HASH: Para trabajar con estas funciones se emplea el comando

```
$> openssl dgst
```

Este comando, junto con sus opciones, permite generar valores hash en hexadecimal y binario, códigos de autenticación de mensajes (HMAC) y firmar digitalmente un valor hash. Observad que lo usual es que la salida de una función hash sea una cadena hexadecimal, pero para tareas de firma la salida debe ser binaria. Empleando las opciones correspondientes, el comando `dgst` produce la salida en el formato adecuado. Por ejemplo, la ejecución `openssl dgst -sign` produce una salida en binario del hash para a continuación firmarlo.

FIRMA Y VERIFICACIÓN: OpenSSL realiza tareas de firma y verificación mediante dos comandos, `openssl pkeyutl` y `openssl dgst` junto con las opciones `-sign` y `-verify`. Cada uno de ellos funciona con diferentes opciones y modos de introducir claves y archivos. Leed detenidamente la documentación para ello. La principal diferencia de funcionamiento está en el tipo de entrada. El comando `openssl`

dgst está diseñado para trabajar con un archivo (que actúa como mensaje a firmar) al que se le va a calcular un valor hash antes de proceder a su firma, tal y como se ha descrito en las transparencias teóricas. Por otra parte openssl pkeyutl espera que el archivo a firmar ya sea el hash (binario) de algún archivo original; por dicho motivo, si la entrada es mayor que el tamaño del hash simplemente trunca la entrada. Con esta última forma es fácil obtener una firma válida para dos archivos distintos si no emplea una función hash previamente.

ESTACIÓN A ESTACIÓN: Para este protocolo será necesario derivar una clave común a partir de dos parejas pública/privada compartiendo los parámetros. En OpenSSL se realiza con el comando

```
$> openssl pkeyutl
```

y la opción `-derive`. Las claves se introducen con `-inkey` y `-peerkey`.

Tareas a realizar

Para esta práctica, deberéis simular la presencia de dos usuarios, por lo que la generación de claves será doble, incluyendo las reutilizadas de la práctica anterior.

1. (0,5 puntos) Generad un archivo `sharedDSA.pem` que contenga los parámetros. Mostrad los valores.

2. (0,5 puntos) Generad dos parejas de claves para los parámetros anteriores. La claves se almacenarán en los archivos `<nombre>DSAkey.pem` y `<apellido>DSAkey.p` No es necesario protegerlas por contraseña.
3. (0,5 puntos) "Extraed" la clave privada contenida en el archivo `<nombre>DSAkey.pem` a otro archivo que tenga por nombre `<nombre>DSApriv.pem`. Este archivo deberá estar protegido por contraseña. Mostrad sus valores. Haced lo mismo para el archivo `<apellido>DSAkey.pem`.
4. (0,5 puntos) Extraed en `<nombre>DSApub.pem` la clave pública contenida en el archivo `<nombre>DSAkey.pem`. De nuevo `<nombre>DSApub.pem` no debe estar cifrado ni protegido. Mostrad sus valores. Lo mismo para el archivo `<apellido>DSAkey.pem`.
5. Coged un archivo cualquiera cualquiera, que actuará como entrada, con al menos 128 bytes. En adelante me referiré a él como `message`, pero podéis llamarlo como os parezca.
6. (0,5 puntos) Firmad directamente el archivo `message` empleando el comando `openssl pkeyutl` sin calcular valores hash, la firma deberá almacenarse en un archivo llamado, por ejemplo, `message.sign`. Mostrad el archivo con la firma.
7. (1 punto) Construid un archivo `message2` diferente de `message` tal que la verificación de la firma

`message.sign` sea correcta con respecto al archivo `message2`.

8. (0,5 puntos) Calculad el valor hash del archivo con la clave pública `nombreDSApub.pem` usando `sha384` con salida hexadecimal con bloques de dos caracteres separados por dos puntos. Mostrad los valores por salida estándar y guardadlo en `nombreDSApub.sha384`.
9. (0,5 puntos) Calculad el valor hash de `message2` usando una función hash de 160 bits con salida binaria. Guardad el hash en `message2.<algoritmo>` y mostrad su contenido.
10. (0,5 puntos) Firmad el archivo `message2` mediante el comando `openssl dgst` y la función hash del punto anterior. La firma deberá almacenarse en un archivo llamado, por ejemplo, `message2.sign`.
11. (1 punto) Verificad la firma `message2.sign` con los archivos `message` y `message2` empleando el comando `openssl dgst`.
12. (0,5 puntos) Verificad que `message2.sign` es una firma correcta para `message2` pero empleando el comando `openssl pkeyutl`
13. (0,5 puntos) Generad el valor HMAC del archivo `sharedDSA.pem` con clave `'12345'` mostrándolo por pantalla.
14. (3 puntos) Simulad una ejecución completa del protocolo Estación a Estación. Para ello emplearemos

como claves para firma/verificación las generadas en esta práctica, y para el protocolo DH emplearemos las claves asociadas a curvas elípticas de la práctica anterior junto con las de otro usuario simulado que deberéis generar nuevamente. Por ejemplo, si mi clave privada está en `javierECpriv.pem` y la clave pública del otro usuario está en `lobilloECpub.pem`, el comando para generar la clave derivada será

```
$> openssl pkeyutl -inkey javierECpriv.pem
-peerkey lobilloECpub.pem -derive -out
key.bin
```

El algoritmo simétrico a utilizar en el protocolo estación a estación será AES-128 en modo CFB8.

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.