Cifrado con curvas elípticas usando OpenSSL

Yabir García Benchakhtir
David Cabezas Berrido
Patricia Córdoba Hidalgo

1. Introducción

OpenSSL es una herramienta muy completa con la que hemos trabajado en la asignatura y que además nos da la facilidad para trabajar también con curvas elípticas. En esta práctica vamos a realizar pasos análogos a lo que hemos visto durante el curso pero aplicando la teoría de curvas elípticas.

2. Detalles a tener en cuenta

Durante nuestros intentos de cifrar y descifrar utilizando ECC hemos encontrado este comentario donde se nos indica que no podemos realizar estos pasos directamente usando ECC. Usamos lo que se ha llamado *Elliptic Curve Integrated Encryption Scheme* en esta práctica.

3. Taller de criptografía

En primer lugar comentamos que OpenSSL nos ofrece una seria de curvas ya definidas en su utilidad para terminal. Tenemos la posibilidad de listar las curvas disponibles utilizando:

```
openssl ecparam -list_curves
```

tras ejecutar esta instrucción se nos mostrará una lista con distintas curvas conocidas. Para continuar con el taller vamos a elegir una de estas curvas, en nuestro caso *secp256k1* que es una curva famosa y bastante segura.

Para crear un archivo de configuración para la curva utilizamos

```
openssl ecparam -name secp256k1 -out secp256k1.pem
```

que creará un achivo *secp256k1.pem* con información sobre la curva. Si ahora hacemos

```
openssl ecparam -in secp256k1.pem -text -param_enc explicit -noout
```

obtenemos información sobre los parametros elegidos como

- El número primo elegido para el cuerpo sobre el que se construye la curva.
- El punto base que se ha elegido en la curva.
- El orden del grupo que se genera.

Una vez elegida la curva procedemos ahora a crear la clave privada. Para ello utilizamos

```
openssl ecparam -in secp256k1.pem -genkey -noout -out userS-privkey.pem
```

Con esta orden le indicamos que genere en un archivo *userS-privkey.pem* una clave privada partiendo del archivo de configuración para la curva que hemos generado anteriormente. Si deseamos ver la información de la clave privada que hemos generado podemos usar

```
openssl ec -in userS-privkey.pem -text -noout
```

Si queremos guardar la clave pública en un archivo tenemos que hacer

```
openssl ec -in userS-privkey.pem -pubout -out userS-pubkey.pem
```

donde a partir de nuestra clave privada generamos un arvhivo de clave pública con la clave pública generada. Para ver la información almacenada en el nuevo archivo

```
openssl ec -in userS-pubkey.pem -pubin -text -noout
```

En este paso podemos comprobar que el contenido coincide con la información que obtuvimos al generar la clave privada.

Procedemos a firmar ahora un documento. Creamos un archivo para ello que hemos llamado *msg.txt* y procedemos a firmarlo utilizando la clave privada que hemos generado con anterioridad.

```
openssl dgst -sign userS-privkey.pem -out msg.txt.sign msg.txt
```

Para realizar el cifrado creamos un archivo generado a partir de nuestra clave privada. Esto lo hacemos, ya que OpenSSL no nos permite cifrar directamente, usando el archivo de la clave privada generada en los primeros pasos.

```
openssl pkeyutl -derive -inkey userS-privkey.pem \\
  -peerkey userR-pubkey.pem -out secret.txt
```

y ciframos ahora utilizando el archivo de clave generado como clave

```
openssl enc -aes-256-cbc -md sha512 -pbkdf2 -iter 100000 -salt \\
 -in msg.txt -out msg.txt.enc -pass file:secret.txt
```

además le hemos indicado que usamos *sha512* como función de resumen y *aes-256-cbc* como sistema de cifrado.

Ahora el potencial receptor del mensaje debe seguir los siguientes pasos. En primer lugar se crea un secreto

```
openssl pkeyutl -derive -inkey userR-privkey.pem -peerkey \\
  userS-pubkey.pem -out secret.txt
```

aquí el receptor usa su clave privada y la clave pública del emisor. Para descrifrar se usa

```
openssl enc -aes-256-cbc -md sha512 -pbkdf2 -iter 100000 -salt -d \\
  -in msg.txt.enc -out msg.txt -pass file:secret.txt
```

Y por último para verificar la firma del documento usamos la clave pública de la persona que envía y el documento de la firma sobre el documento que hemos obtenido al descifrar.

```
openssl dgst -verify userS-pubkey.pem -signature msg.txt.sign msg.txt
```