Ejemplo Real

Fase de Generación de Claves

Creamos directorios:

```
mkdir userR userS
```

Emisor

Usuario emisor:

```
cd userS
```

Generamos la llave privada del usuario S:

```
openssl genpkey -algorithm RSA -pkeyopt rsa_keygen_bits:2048
-pkeyopt rsa_keygen_pubexp:3 -out privkey-userS.pem
```

- `genpkey`: generar llave.
- `rsa_keygen_bits:2048`: tamaño de los objetos de la clave (los primos). Si queremos más seguridad, ponemos el doble.
- `rsa_keygen_pubexp`: exponente público. Se puede modificar. No es malo que sea bajo, el privado si tiene que ser alto.

Así se genera el archivo `privkey-userS.pem`, un código en base 64. Para verlo con más detalle:

```
openssl pkey -in privkey-userS.pem -text
```

Si le pones más primos a RSA, es menos seguro.

Generamos la llave pública a partir de la privada:

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

La clave pública tiene menos información que la privada:

```
openssl pkey -in pubkey-userS.pem -pubin -text
```

Receptor

```
cd ../userR/
```

Hacemos lo mismo.

```
openssl genpkey -algorithm RSA -pkeyopt rsa_keygen_bits:2048
-pkeyopt rsa_keygen_pubexp:5 -out privkey-userR.pem
```

```
openssl pkey -in privkey-userR.pem -out pubkey-userR.pem -pubout
```

Intercambio de llaves

```
mv pubkey-userR.pem ../userS
```

```
mv ../userS/pubkey-userS.pem .
```

Transmitir mensaje Emisor

Escribimos el mensaje en un fichero:

```
echo Le ruego que disponga de 15540.40 dólares de mi cuenta, a la
que tiene acceso, para adquirir para mí un Bitcoin > message-userS.txt
```

- No se puede superar la longitud de 256 bits.

Hace un resumen con -sha256 a modo de firma con la llave privada del usuario S. El resultado está en `message-userS.txt.sgn`, que se genera a partir de `message-userS.txt`.

```
openssl dgst -sha256 -sign privkey-userS.pem -out message-userS.txt.sgn
message-userS.txt
```

Puedo trasmitir el mensaje cifrado o no.

Para cifrarlo:

```
openssl pkeyutl -encrypt -in message-userS.txt -pubin -inkey
pubkey-userR.pem -out message-userS.txt.enc
```

Saca el mensaje cifrado con la llave pública del receptor.

Se los enviamos al receptor:

```
mv message-userS.txt.* ../userR/
```

Recibir mensaje Receptor

```
cd ../userR/
```

Primero descifra con su llave privada.

```
openssl pkeyutl -decrypt -in message-userS.txt.enc -inkey privkey-userR.pem
-out rec-message-userS.txt
```

Se genera `rec-message-userS.txt`.

Veamos que quien lo envía es quien dice ser con la verificación.

```
openssl dgst -sha256 -verify pubkey-userS.pem -signature message-userS.txt.sgn
rec-message-userS.txt
```

Como pone `Verified OK`, ahora si puedo abrir el mensaje y leerlo.

Pregunta

¿Cómo se ha llevado a cabo la firma? ¿De dónde sale `Verified OK`?

Se comparan los hash porque solo él tiene su llave privada.