Seguridad en Redes
Practica 2.2

David Antuña Rodríguez
Javier Carrión García

1 OpenSSL

1.1 Creacion de claves RSA, DSA, DH y EC

Hemos adjuntado una carpeta llamada keys que contiene las claves tanto públicas, xxpubkey.txt, como privadas, xxkey.txt.

RSA utiliza el teorema chino del resto para precomputar tres valores que aceleran el desencriptado del mensaje. Primero veamos cómo desencripta RSA.

- Sean **p** y **q** dos numeros primos diferentes escogidos aleatoriamente.
- Sea **n** la longitud de las claves rsa, privada y pública, $n = pq$.
- Sea **d** $\equiv e^{-1}$ mod λ(n), donde e es un entero tal que 1 < e < λ(n).

Sean m y c el mensaje desencriptado y encriptado, respectivamente.

$$m = c^d \ mod \ n$$

Dicho calculo puede resultar muy costoso debido al exponente, d, por lo que se aplica el teorema chino del resto para agilizarlo.

En primer lugar se precomputan d_p, d_q y q_{inv}, con la precondición de que p > q.

- $d_p = e^{-1}$ mod (p-1)
- $d_q = e^{-1}$ mod (q-1)
- $q_{inv} = q^{-1}$ mod p

Una vez se poseen esos valores se pueden realizar los siguientes computos para desencriptar.

- $m_1 = c^{d_p}$ mod p
- $m_2 = c^{d_q}$ mod q
- $h = q_{inv}(m_1$ - $m_2)$ mod p

Ahora el desencriptado no necesita resolver el exponente que incrementaba su coste.

$$m = m_2 + hq$$

1.2 Cifrado y descrifrado con RSA

En la carpeta rsa se encuentra el keyfile cifrado con la clave pública de rsa, keyfile.bin, y el resultado de cifrar /etc/services utilizando des3 y el keyfile, cipher.bin.

1.3 Firma y verificación con RSA, DSA y EC (ECDSA)

Para todas las firmas se ha empleado el algoritmo de hash sha256.

```
usuario_vms@ptol103:~/SeR$ openssl dgst -sha256 -sign rsakey.pem -out sigrsa /etc/services
usuario_vms@ptol103:~/SeR$ openssl dgst -sha256 -verify rsapubkey.pem -signature sigrsa /etc/services
Verified OK
usuario_vms@ptol103:~/SeR$ openssl dgst -sha256 -sign dsakey.pem -out sigdsa /etc/services
usuario_vms@ptol103:~/SeR$ openssl dgst -sha256 -verify dsapubkey.pem -signature sigdsa /etc/services
Verified OK
usuario_vms@ptol103:~/SeR$ openssl dgst -sha256 -sign eckey.pem -out sigec /etc/services
usuario_vms@ptol103:~/SeR$ openssl dgst -sha256 -verify ecpubkey.pem -signature sigec /etc/services
Verified OK
```

Figure 1.3.1 : Firmado y verificado con rsa, dsa y ec.

1.4 Acuerdo de claves con DH y EC (ECDH)

En la carpeta acuerdos estan los ficheros correspondientes a las nuevas keys y los secretos generados.

```
usuario_vms@ptol103:~/SeR$ openssl pkeyutl -derive -inkey dhkey.pem -peerkey dhpubkey2.pem -out secret1dh
usuario_vms@ptol103:~/SeR$ openssl pkeyutl -derive -inkey dhkey2.pem -peerkey dhpubkey.pem -out secret2dh
usuario_vms@ptol103:~/SeR$ cmp secret1dh secret2dh
```

Figure 1.4.1 : Acuerdo de claves con DH.

```
usuario_vms@ptol103:~/SeR$ openssl pkeyutl -derive -inkey eckey.pem -peerkey ecpubkey2.pem -out secret1ec
usuario_vms@ptol103:~/SeR$ openssl pkeyutl -derive -inkey eckey2.pem -peerkey ecpubkey.pem -out secret2ec
usuario_vms@ptol103:~/SeR$ cmp secret1ec secret2ec
```

Figure 1.4.2 : Acuerdo de claves con EC.

2 GnuPG

2.1 Creación y gestión de claves PGP

El id de la clave siempre corresponde al codigo que aparece despues de /, marcado en una de las claves de la imagen.

```
usuario_vms@ptol103:~/SeR$ gpg2 --list-keys
/home/usuario_vms/.gnupg/pubring.gpg
------------------------------------
pub   1024R/DF99D131 2018-02-27
uid    [  absoluta ] Bertoldo Perez Perez (No haré más comentarios señoria) <bertoldo@gmail.com>
sub   1024R/E071E8AC 2018-02-27

pub   2048R/1A8CD27E 2018-02-27
uid    [  absoluta ] Aparicio Álvarez-del Caño (Javier es muy lento) <alva@gmail.com>
sub   2048R/03324A2D 2018-02-27
```

Figure 2.1.1 : Claves públicas del anillo.

```
usuario_vms@pto1103:~/SeR$ gpg2 --list-keys
/home/usuario_vms/.gnupg/pubring.gpg
------------------------------------
pub   1024R/DF99D131 2018-02-27
uid    [  absoluta ] Bertoldo Perez Perez (No haré más comentarios señoria) <bertoldo@gmail.com>
sub   1024R/E071E8AC 2018-02-27

pub   2048R/1A8CD27E 2018-02-27
uid    [  absoluta ] Aparicio Álvarez-del Caño (Javier es muy lento) <alva@gmail.com>
sub   2048R/03324A2D 2018-02-27

pub   1024R/7254F82F 2017-11-07 [caduca: 2018-11-07]
uid    [desconocida] Inma Pardines
sub   1024R/0D9C8170 2017-11-07 [caduca: 2018-11-07]
```

Figure 2.1.2 : Claves públicas del anillo tras importar.

2.2 Cifrado y descifrado

Hemos cifrado el fichero con la clave pública que has subido al campus.

2.3 Firma y verificación

Como se puede ver en la captura hemos probado a cifrar con sha256, esta es la firma que hemos entregado, y no hay ningún problema pero si firmamos con md5 no se puede verificar porque es un algoritmo no seguro.

```
usuario_vms@pto1103:~/SeR$ gpg2 --verify signbertoldo
gpg: Firmado el mar 27 feb 2018 13:32:05 CET usando clave RSA ID DF99D131
gpg: Firma correcta de "Bertoldo Perez Perez (No haré más comentarios señoria) <bertoldo@gmail.com>" [absoluta]
usuario_vms@pto1103:~/SeR$ gpg2 --verify signaparicio
gpg: Firmado el mar 27 feb 2018 13:32:56 CET usando clave RSA ID 1A8CD27E
gpg: Note: signatures using the MD5 algorithm are rejected
gpg: Imposible comprobar la firma: Invalid digest algorithm
```

Figure 2.3.1 : Firma y verificacion para Bertoldo y Aparicio.

El fichero gpg contiene la clave pública de Bertoldo.