Seguridad en Redes
Practica 2.2

David Antuña Rodríguez
Javier Carrión García

1 OpenSSL

1.1 Creacion de claves RSA, DSA, DH y EC

Comandos
openssl genpkey -algorithm RSA -out rsakey.pem
openssl pkey -pubout -in rsakey.pem -out rsapubkey.pem
openssl pkey -in rsakey.pem -noout -text
openssl pkey -pubin rsapubkey.pem -noout -text

openssl genpkey -genparam -algorithm DSA -out dsaparam.pem
openssl genpkey -paramfile dsaparam.pem -out dsakey.pem
openssl pkey -pubout -in dsakey.pem -out dsapubkey.pem
openssl pkey -in dsakey.pem -noout -text
openssl pkey -pubin dsapubkey.pem -nout -text

openssl genpkey -genparam -algorithm DH -out dhparam.pem
openssl genpkey -paramfile dhparam.pem -out dhkey.pem
openssl pkey -pubout -in dhkey.pem -out dhpubkey.pem
openssl pkey -in dhkey.pem -noout -text
openssl pkey -pubin dhpubkey.pem -nout -text

openssl genpkey -genparam -algorithm EC -out ecparam.pem
-pkeyopt ec_paramgen_curve:prime192v1
openssl genpkey -paramfile ecparam.pem -out eckey.pem
openssl pkey -pubout -in eckey.pem -out ecpubkey.pem
openssl pkey -in eckey.pem -noout -text
openssl pkey -pubin ecpubkey.pem -nout -text

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

Comandos
openssl rand 32 -out keyfile
openssl pkeyutl -encrypt -pubin -inkey rsapubkey.pem -in keyfile -out keyfile.bin
openssl pkeyutl -decrypt -inkey rsakey.pem -in keyfile.bin > keyfile2
openssl enc -des3 -pass file:keyfile -in /etc/services -out cipher.bin
openssl enc -d -des3 -pass file:keyfile2 -in 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.

Figure 1.3.1 : Firmado con rsa.

Figure 1.3.2 : Firmado con dsa.

Figure 1.3.3 : Firmado con ec.

Comandos
openssl dgst -sha256 -sign rsakey.pem -out sigrsa /etc/services
openssl dgst -sha256 -verify pubkey.pem -signature sigrsa /etc/services

openssl dgst -sha256 -sign dsakey.pem -out sigdsa /etc/services
openssl dgst -sha256 -verify pubkey.pem -signature sigdsa /etc/services

openssl dgst -sha256 -sign eckey.pem -out sigec /etc/services
openssl dgst -sha256 -verify pubkey.pem -signature sigec /etc/services

1.4 Acuerdo de claves con DH y EC (ECDH)

Comandos
openssl genpkey -paramfile dhparam.pem -out dhkey2.pem
openssl pkey -pubout -in dhkey2.pem -out dhpubkey2.pem
openssl pkey -in dhkey2.pem -noout -text
openssl pkey -pubin dhpubkey2.pem -nout -text

openssl genpkey -paramfile ecparam.pem -out eckey2.pem
openssl pkey -pubout -in eckey2.pem -out ecpubkey2.pem
openssl pkey -in eckey2.pem -noout -text
openssl pkey -pubin ecpubkey2.pem -nout -text

openssl pkeyutl -derive -inkey dhkey.pem -peerkey dhpubkey2.pem -out secret1dh
openssl pkeyult -derive -inkey dhkey2.pem -peerkey dhpubkey.pem -out secret2dh
cmp secret1dh secret2dh
xxd secret1dh
xxd secret2dh

openssl pkeyutl -derive -inkey eckey.pem -peerkey ecpubkey2.pem -out secret1ec
openssl pkeyult -derive -inkey eckey2.pem -peerkey ecpubkey.pem -out secret2ec
cmp secret1ec secret2ec
xxd secret1ec
xxd secret2ec

2 GnuPG

2.1 Creación y gestión de claves PGP

Figure 2.1.1 : Claves públicas del anillo.

Figure 2.1.2 : Claves públicas del anillo tras importar.

Comandos
gpg2 −−list-keys Hacer captura(sring)

gpg2 −−gen-key Usar dos veces para crear dos claves, contraseña: seguridad
gpg2 −−output keys.gpg -a −−export id Listar las keys y coger el id de una de las creadas

gpg2 −−import pubInma.gpg
gpg2 −−list-keys Hacer captura(nring)

2.2 Cifrado y descifrado

Hemos cifrado el fichero con la clave pública que has subido al campus.
Comandos
cp /etc/services $HOME/SeR/services
gpg2 –output services.gpg −−encrypt −−recipient (id de la clave de inma) -a services

2.3 Firma y verificación

Comandos
gpg2 −−list-keys
Probar para dos claves, una de ellas ha de ser la que hemos exportado para subir

gpg2 −−sign –output sign(id1) -u (id1) -a −−digest-algo md5 /etc/services
gpg2 −−verify sign(id1)

gpg2 −−sign –output sign(id2) -u (id2) -a −−digest-algo sha256 /etc/services
gpg2 −−verify sign(id2)