Seguridad en Redes

Práctica 2.2. Criptografía de clave pública

Preparación del entorno

Las prácticas 2.1, 2.2 y 2.3 se pueden realizar en cualquier computador con S.O. Linux o en cualquier máquina virtual Linux. Si ya dispones de él, tan sólo es necesario instalar (en caso de que no estén instalados) los paquetes `openssl` y `gnupg`.

La segunda opción, si no dispones de un S.O. Linux propio, es descargar el servicio virtualizado `SER.ova` del campus virtual e importarlo con VirtualBox en tu PC. Esta máquina virtual (MV) ya cuenta con los paquetes `openssl` y `gnupg` instalados. Para esta práctica sólo se necesita una única MV (sin clones). Se recomienda disponer de conexión a Internet en la MV, por lo que el "Adaptador de red 1" (correspondiente a la interfaz `eth0` de la MV) debe configurarse en modo NAT.

NOTAS IMPORTANTES:
1. este documento está editado con MS Word y algunos caracteres especiales, especialmente el guion (–) o las comillas (″), a veces utilizan una codificación diferente a la que se usa en Linux. Por tanto, si haces un "copia y pega" del comando desde el documento a la consola de Linux, éste podría fallar. En tal caso, se recomienda revisar estos caracteres especiales o reescribir el comando completo.
2. Algunos comandos son demasiado largos y están separados en dos o más líneas dentro de este documento. Cuando se copien a la consola de Linux, deben escribirse en una única línea.

A. OpenSSL

A.1. Creación de claves RSA, DSA, DH y EC

Para crear una clave privada RSA, se usa el comando `genpkey`:

```
$ openssl genpkey -algorithm RSA -out rsakey.pem [-<cipher>]
  [-pkeyopt rsa_keygen_bits:<numbits>]
```

Si se indica un algoritmo de cifrado simétrico (como `-aes-256-cbc`), la clave privada se cifra con una contraseña. Opcionalmente también se puede indicar el número de bits para la clave RSA (si no se especifica, se usa por defecto una longitud de 2048 bits[1]).

Para crear claves DSA, DH y EC, primero hay que generar un fichero de parámetros:

```
$ openssl genpkey -genparam -algorithm DSA -out dsaparam.pem
  [-pkeyopt dsa_paramgen_bits:<numbits>]
$ openssl genpkey -genparam -algorithm DH -out dhparam.pem
  [-pkeyopt dh_paramgen_prime_len:<numbits>]
```

[1] Una pareja de claves RSA de 2048 bits (256 bytes), significa que se utilizan dos factores primos (p y q) de 1024 bits (128 bytes) cada uno, que dan lugar a un modulus (n) y un exponente privado (d) ambos de 2048 bits. El exponente público (e) suele tener el valor decimal 65537.

```
$ openssl genpkey -genparam -algorithm EC -out ecparam.pem
   -pkeyopt ec_paramgen_curve:<curve>
```

En caso de DSA y DH se puede especificar la longitud del número primo generado (en ambos casos, el valor por defecto 2048 bits). En el caso de EC, hay que indicar la curva a usar, por ejemplo `prime192v1`.

Después, se crean las claves privadas a partir de los ficheros de parámetros (sustituyendo `XX` por `dsa`, `dh` o `ec`):

```
$ openssl genpkey -paramfile XXparam.pem -out XXkey.pem
```

La clave pública se obtiene a partir de la clave privada usando el comando `pkey` con la opción `-pubout` (sustituyendo `XX` por `rsa`, `dsa`, `dh` o `ec`):

```
$ openssl pkey -pubout -in XXkey.pem -out XXpubkey.pem
```

Para ver las claves, se usa el comando `pkey` con la opción `-text`:

```
$ openssl pkey -in XXkey.pem -noout -text
$ openssl pkey -pubin -in XXpubkey.pem -noout -text
```

Ejercicio 1

- Crea una clave privada de cada tipo (sin cifrar o usando la contraseña “seguridad”) y obtén su correspondiente clave pública.
- Visualiza las claves privadas y públicas
- Identifica los valores de las claves públicas y privadas de RSA: números primos (p y q), modulus (n), exponente público (e) y exponente privado (d). Nota: Los últimos tres valores de las claves privadas de RSA son valores precalculados para acelerar las operaciones con la clave privada mediante la aplicación del teorema chino del resto.

Entrega #1: Entrega los archivos con las claves privadas creadas.

A.2. Cifrado con clave pública (con RSA)

Para cifrar con la clave pública, se usa el comando `pkeyutl` con la opción `-encrypt`:

```
$ openssl pkeyutl -encrypt -pubin -inkey XXpubkey.pem
  -in plain.txt -out enc.bin
```

Para descifrar con la clave privada, se usa el comando `pkeyutl` con la opción `-decrypt`:

```
$ openssl pkeyutl -decrypt -inkey XXkey.pem -in enc.bin
  -out plain-again.txt
```

Ejercicio 2

- Crea un archivo de texto llamado `datos1.txt`
- Cifra el archivo de texto con la clave pública y guarda el archivo cifrado con el nombre `datos1.bin`
- Descifra el archivo de texto con la clave privada y guarda el archivo descifrado con el nombre `datos2.txt`
- Compara los archivos `datos1.txt` y `datos2.txt`, comprueba que son idénticos

Entrega #2: Entrega el archivo `datos1.bin`.

A.3. Cifrado por clave de sesión: método de transporte de clave (con RSA)

El método de transporte de clave consiste en que el emisor genera aleatoriamente una clave simétrica (clave de sesión) para cifrar los datos. A su vez, la clave de sesión se cifra con la clave pública del receptor. Por ejemplo:

```
$ openssl rand 32 -out keyfile
$ openssl enc -des3 -pass file:keyfile -in plain.txt
  -out cipher.bin
$ openssl pkeyutl -encrypt -pubin -inkey rsapubkey.pem
  -in keyfile -out keyfile.bin
```

Los datos cifrados (`cipher.bin`), junto con la clave de sesión cifrada (`keyfile.bin`), se envían al receptor. A continuación, el receptor usa su clave privada para descifrar la clave de sesión y con ésta ya puede descifrar los datos:

```
$ openssl pkeyutl -decrypt -inkey rsakey.pem
  -in keyfile.bin > keyfile
$ openssl enc -d -des3 -pass file:keyfile
  -in cipher.bin -out plain.txt
```

Ejercicio 3

- Utiliza el método de transporte de clave descrito anteriormente para cifrar/descifrar un fichero de texto.

Entrega #3: Entrega los archivos `keyfile.bin` y `chipher.bin`.

A.4. Cifrado por clave de sesión: método de acuerdo de clave (con DH y EC)

El método de acuerdo de clave consiste en que las dos partes cooperan para acordar una clave secreta de forma segura. Para ello, cada una de las partes genera una clave simétrica a partir de su clave privada y la clave pública de la otra parte. Esto se puede realizar usando la opción `-derive` de `pkeyutl`:

```
$ openssl pkeyutl -derive -inkey XXkey1.pem
  -peerkey XXpubkey2.pem -out secretkey1
```

```
$ openssl pkeyutl -derive -inkey XXkey2.pem
  -peerkey XXpubkey1.pem -out secretkey2

$ cmp secretkey1 secretkey2
```

El primer comando simula la generación de la clave secreta compartida por una de las partes, usando su clave privada y la clave pública de la otra parte. El segundo comando simula la generación por la otra parte. Los secretos acordados han de ser idénticos.

Ejercicio 4

- Crea un segundo conjunto de claves DH y EC (usando el mismo fichero de parámetros que en el primer ejercicio) y obtén las claves secretas compartidas a partir de los dos pares de claves.

Entrega #4: Entrega los archivos con las claves secretas compartidas

A.5. Firma digital y verificación (con RSA, DSA y EC)

Para firmar un fichero y verificar su firma, se usan las opciones `-sign` y `-verify` del comando `dgst`:

```
$ openssl dgst [-<algoritmo>] -sign XXkey.pem
  -out sig.bin plain.txt
$ openssl dgst [-<algoritmo>] -verify XXpubkey.pem
  -signature sig.bin plain.txt
```

Por defecto, el archivo de firma (`sig.bin`) se guarda en formato binario, no en hexadecimal, ya que OpenSSL no puede realizar verificación de firmas en formato hexadecimal. Si queremos visualizar el resultado de la firma en hexadecimal, se puede realizar la conversión con el comando `xxd`:

```
$ xxd sig.bin sig.hex
```

Ejercicio 5

- Firma varios ficheros y verifica su firma con distintos algoritmos de clave pública y de *hash*.

Entrega #5: Copia y entrega al menos dos de las firmas generadas en el ejercicio anterior, pasadas a formato hexadecimal, usando algoritmos de clave pública distintos.

B. GnuPG

B.1. Creación y gestión de claves GPG

Para crear una clave, se usa el comando `--gen-key` y se responde a las distintas preguntas:

```
$ gpg --gen-key
```

(en las versiones más actuales de GPG, el comando `--gen-key` se sustituye por `--full-gen-key`)

Dado que el sistema tiene que generar muchos números aleatorios, es conveniente seleccionar un tamaño de clave de 1024 bits.

Nota:
Para generar las claves GPG, es necesario que haya suficiente entropía en el sistema. En caso contrario, la generación de claves puede llevar mucho tiempo. La entropía del sistema se puede consultar con el siguiente comando:

```
$ cat /proc/sys/kernel/random/entropy_avail
```

Para aumentar la entropía se puede instalar el paquete `rng-tools` para generación de números aleatorios y ejecutar el proceso `rngd`:

```
$ sudo apt-get install rng-tools
$ sudo rngd -r /dev/urandom
```

Para detener el proceso `rngd`, una vez generadas las claves necesarias:

```
$ sudo killall rngd
```

GPG soporta subclaves, que están vinculadas a la clave maestra. Una subclave se puede utilizar para firmar o para cifrar. La ventaja es que las subclaves pueden almacenarse de forma separada, e incluso invalidarse sin afectar a la clave maestra. GPG automáticamente crea una llave maestra sólo para firma, y una subclave para cifrado.

Para mostrar las claves públicas, se usa el comando `--list-keys`:

```
$ gpg --list-keys
```

Para mostrar las claves privadas, se usa el comando `--list-secret-keys`:

```
$ gpg --list-secret-keys
```

Ejemplo de las salidas de estas órdenes:

```
$ gpg --list-keys
/home/usuario/.gnupg/pubring.gpg
--------------------------------
pub   1024R/E78C2A25 2021-02-10 [caduca: 2021-03-12]
uid                  User1 (no comments) <user1@dominio.net>
sub   1024R/0A9A5B00 2021-02-10 [caduca: 2021-03-12]
```

```
$ gpg --list-secret-keys
/home/usuario/.gnupg/secring.gpg
--------------------------------
sec   1024R/E78C2A25 2021-02-10 [caduca: 2021-03-12]
uid                  User1 (no comments) <user1@dominio.net>
ssb   1024R/0A9A5B00 2021-02-10
```

Estas salidas muestran las siguientes claves:

- `pub`: clave pública maestra (PUBlic key)
- `sub`: subclave pública (public SUBkey), por defecto, igual que la maestra.
- `sec`: clave privada maestra (SECret key)
- `ssb`: subclave privada (Secret SuBkey), por defecto, igual que la maestra.

Los identificadores (ID) de estas claves aparecen detrás del campo `1024R` (por ejemplo: `E78C2A25`). El campo `1024R` significa que es una clave RSA de 1024 bits.

Para exportar las claves públicas a un archivo, de manera que se puedan para intercambiar con otros usuarios, se usa el comando `--export`:

```
$ gpg --output pubkeys.gpg -a --export [<id>]
```

El valor de `<id>` se puede sustituir por el identificador de la clave (ej. `E78C2A25`) o por el identificador del usuario (ej. `User1`). Si no se especifica un identificador tras `--export`, se exportarán todas las claves. La opción `-a` (o `--armor`) escribe el resultado en ASCII (usando Base64) en lugar de en binario.

Es posible exportar las claves privadas a un fichero. Para ello se usa el comando `--export-secret-keys`:

```
$ gpg --output keys.gpg [-a] --export-secret-keys [<id>]
```

Para importar claves, públicas o privadas, se usa el comando `--import`:

```
$ gpg --import keys.gpg
```

Ejercicio 6

- Crea varios pares de claves (con contraseña “seguridad”), realiza un listado de las mismas y exporta las claves públicas y privadas.

Entrega #6: Entrega los archivos con las claves privadas exportadas.

B.2. Cifrado y descifrado

Para cifrar un fichero con la clave pública y descifrarlo con la correspondiente clave privada, se usan las opciones `--encrypt` (similar a `-e`) y `--decrypt` (similar a `-d`), respectivamente:

```
$ gpg --encrypt --recipient <id> [-a] plain.txt
$ gpg --decrypt plain.txt.gpg
```

El último argumento de la orden debe ser siempre el fichero que queremos cifrar/descifrar. El valor de `<id>` se puede sustituir por el identificador de la clave (ej. `E78C2A25`) o por el

identificador del usuario (ej. `User1`). La opción `-a` (o `--armor`) escribe el resultado en ASCII (usando Base64) en lugar de en binario. Por defecto el fichero de salida tendrá el mismo nombre que le fichero de entrada con extensión `.gpg` o extensión `.asc` (si usamos la opción `-a`). Si queremos dar otro nombre al fichero de salida se puede usar la opción `--ouput <filename>`.

Ejercicio 7

- Cifra y descifra ficheros para con las distintas claves creadas anteriormente.

Entrega #7: Copia y entrega el resultado de cifrar el fichero `/etc/hosts` con una clave pública usando formato ASCII (copia primero el fichero `/etc/hosts` al directorio `$HOME`).

B.3. Firma digital y verificación

Para firmar y verificar ficheros, se usan los comandos `--sign` y `--verify`:

```
$ gpg --sign [-a] [--digest-algo <algoritmo>] plain.txt
$ gpg --verify plain.txt.gpg
```

Para indicar con qué clave se quiere firmar, se puede usar la opción `--local-user <id>`. La opción `-a` (o `--armor`) escribe el resultado en ASCII (usando Base64) en lugar de en binario. Por defecto el fichero de salida tendrá el mismo nombre que le fichero de entrada con extensión `.gpg` o extensión `.asc` (si usamos la opción `-a`). Si queremos dar otro nombre al fichero de salida se puede usar la opción `--ouput <filename>`.

Es importante resaltar que el comando `--sign` no sólo firma el fichero, sino que también lo cifra con la clave pública. Por tanto, la salida `plain.txt.gpg` contiene el fichero cifrado más la firma. Para descifrar el fichero se usa:

```
$ gpg --decrypt plain.txt.gpg
```

Si tan sólo queremos obtener la firma, sin incluir el fichero cifrado, se usa el comando `--detach-sign`:

```
$ gpg --detach-sign [-a] [--digest-algo <algoritmo>] plain.txt
$ gpg --verify plain.txt.sig plain.txt
```

Para indicar con qué clave se quiere firmar, se puede usar la opción `--local-user <id>`. La opción `-a` (o `--armor`) escribe el resultado en ASCII (usando Base64) en lugar de en binario. Por defecto el fichero de salida tendrá el mismo nombre que le fichero de entrada con extensión `.sig` o extensión `.asc` (si usamos la opción `-a`). Si queremos dar otro nombre al fichero de salida se puede usar la opción `--ouput <filename>`.

Ejercicio 8

- Firma ficheros y verifica su firma con distintas claves y distintos algoritmos *hash*.

Entrega #8: Copia y entrega al menos dos firmas del archivo `/etc/hosts`, generadas en formato ASCII, con algoritmos hash distintos.