Seguridad en Redes

Práctica 2.2. Criptografía de clave pública

OpenSSL

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. Si no se indica un número de bits para la clave, se usa 1024.

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>]

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

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 con la opción `-pubout` del comando `pkey` (sustituyendo `XX` por `rsa`, `dsa`, `dh` o `ec`):

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

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

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

- Crea una clave privada de cada tipo (sin cifrar o usando la contraseña “seguridad”) y obtén su correspondiente clave pública.

- **Entrega:** Copia para cada par de claves (privada y pública) generada los valores obtenidos al visualizar las claves con la opción `-text`. Especificar cuáles de estos valores son públicos y cuáles privados. Indicar también cuáles de estos valores son parámetros y cuáles constituyen la clave pública y la privada.

- Al identificar los valores de la clave privada RSA, podemos observar que los últimos tres valores de esta clave privada son valores precalculados para acelerar las operaciones de descifrado mediante la aplicación del teorema chino del resto.

  - **Entrega:** Explicar en qué consiste el teorema chino del resto y cómo se aplica.

Cifrado y descifrado con RSA

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

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

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

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

Para poder usar el método anterior, el contenido del fichero debe ser algo menor que el tamaño de la clave. Además, el texto cifrado de esta forma se podría criptoanalizar aprovechando propiedades conocidas de los mensajes (repeticiones, patrones, propiedades numéricas…). De hecho, **la criptografía de clave pública solo debe usarse para cifrar datos aleatorios (claves de sesión) o pseudoaleatorios (códigos *hash*)**. Por tanto, se debe utilizar el método que se describe a continuación, denominado método híbrido, que, además, es más eficiente. Si usamos como algoritmo de clave pública el RSA, el método híbrido consistiría en:

1. El emisor genera una clave de sesión (clave secreta) aleatoria ejecutando

   ```
   $ openssl rand 32 -out keyfile
   ```

   En realidad, en este ejemplo, lo que hemos generado es la contraseña que se va a utilizar junto con la *salt* para crear la clave secreta.

2. El emisor cifra el fichero `keyfile`, que contiene la contraseña, con la clave pública del receptor

   ```
   $ openssl pkeyutl -encrypt -pubin -inkey rsapubkey.pem -in keyfile -out keyfile.bin
   ```

3. El emisor envía la clave secreta cifrada al receptor y éste la descifra con su clave privada

   ```
   $ openssl pkeyutl -decrypt -inkey rsakey.pem -in keyfile.bin > keyfile2
   ```

4. Una vez compartida la clave secreta, toda la información transmitida entre las partes se cifrará con la clave secreta. Por ejemplo, si el emisor quiere enviar el fichero `plain.text`, lo cifrará con un algoritmo de cifrado simétrico (AES, 3DES, …) y lo que enviará será el fichero cifrado.

   ```
   $ openssl enc -des3 -pass file:keyfile -in plain.txt -out cipher.bin
   ```

5. Finalmente, el receptor descifrará el fichero cipher.bin con la clave secreta compartida

   ```
   $ openssl enc -d -des3 -pass file:keyfile2 -in cipher.bin
   ```

En nuestro caso, los ficheros `keyfile.bin` y `cipher.bin` no serán enviados porque la misma máquina virtual está haciendo de emisor y receptor, y por lo tanto, ambas partes (receptor y emisor) conocen los dos ficheros.

El comando `smime` usa este procedimiento, pero se necesita un certificado.

- Cifra y descifra ficheros siguiendo el procedimiento descrito.

  - **Entrega:** Copia un fichero con el resultado de cifrar el fichero `/etc/services` con este procedimiento. Copia también el fichero con la clave secreta cifrada.

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

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 plain.txt

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

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

  - **Entrega:** Copia las firmas de `/etc/services` usando las distintas claves generadas en el primer ejercicio. Especifica para cada firma el algoritmo de clave pública y el algoritmo de *hash* utilizado.

Acuerdo de claves con DH y EC (ECDH)

Para acordar una clave secreta compartida entre dos partes, se usa la opción `-derive` de `pkeyutl`:

```
$ openssl pkeyutl -derive -inkey XXkey1.pem -peerkey
XXpubkey2.pem -out secret1
```

```
$ openssl pkeyutl -derive -inkey XXkey2.pem -peerkey XXpubkey1.pem -out secret2

$ cmp secret1 secret2
```

El primer comando simula la generación del secreto compartido 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. Fíjaos que en los algoritmos que realizan acuerdo de clave, la clave no viaja a través de la red (como ocurre con el algoritmo RSA) sino que es generada en cada una de las partes que se van a comunicar.

Para poder acordar una clave secreta entre dos partes es necesario que cada una de las partes tenga un par de claves pública y privada. Si consideramos que las claves EH y ECDH generadas en el primer apartado de la práctica pertenecen a una de las partes, será necesario generar un nuevo par de claves de cada tipo para la otra parte.

- Crea otra clave DH y otra clave 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. Tener en cuenta que EH y ECDH son dos algoritmos de acuerdo de clave, de lo que se trata, por lo tanto, es de es de obtener dos claves secretas compartidas, una para EH y otra para ECDH.

  - **Entrega**: Copia los ficheros de las nuevas claves privadas (lo que se ve usando la opción `-text` del comando `pkey`) y los de las claves secretas compartidas obtenidas (lo que devuelve `xxd`).

GnuPG

Creación y gestión de claves PGP

Para crear una clave, se usa el comando `--gen-key` y se responde a las distintas preguntas que nos plantea la ejecución del comando:

```
$ gpg2 --gen-key
```

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

OpenPGP 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. GnuPG automáticamente crea una llave maestra sólo para firma, y una subclave para cifrado.

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

```
$ gpg2 --list-keys
```

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

```
$ gpg2 --list-secret-keys
```

- **Entrega**: Haz una captura de pantalla de las claves públicas que contiene el anillo. Indicar cuál es el identificador de cada clave.

Para exportar las claves públicas (para intercambiarlas), se usa el comando `--export`:

```
$ gpg2 --output pubkeys.gpg -a --export [<id>]
```

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.

Para exportar las claves privadas, se usa el comando `--export-secret-keys`:

```
$ gpg2 --output keys.gpg [-a] --export-secret-keys [<id>]
```

Recordad que, en principio, las claves privadas no deben exportarse, ya que la seguridad de este tipo de algoritmos reside, entre otras cosas, en que la clave privada solo sea conocida por su propietario.

- Crea varios pares de claves (con contraseña “seguridad”) y exporta una de las claves públicas creadas.

  - **Entrega:** Sube al Campus (tarea Clave pública, cifrado y firma digital) el fichero generado al exportar la clave pública.

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

```
$ gpg2 --import keys.gpg
```

- Importa la clave pública *pubInma.pgp* que puedes descargar del Campus Virtual.

  - **Entrega:** Añade la captura de pantalla de cómo esa clave se ha añadido a tu anillo de claves (para ello mostrar las claves públicas del anillo).

Cifrado y descifrado

Para cifrar y descifrar ficheros, se usan los comandos `--encrypt` y `--decrypt`:

```
$ gpg2 --encrypt --recipient <id> [-a] plain.txt

$ gpg2 --decrypt plain.txt.asc
```

- Cifra y descifra ficheros para distintos destinatarios.

- **Entrega:** Cifra el fichero `/etc/services` para enviármelo a mí (cópialo primero al directorio `$HOME`). Sube el fichero cifrado al Campus Virtual (tarea Clave pública, cifrado y firma digital). ¿Con qué clave habéis cifrado el fichero? Recordad que yo tengo que poder descifrar el fichero que me enviéis.

Firma y verificación

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

```
$ gpg2 --sign –u [<id>] [-a] [--digest-algo <algoritmo>] plain.txt

$ gpg2 --verify plain.txt.asc
```

Para indicar con qué clave se quiere firmar, se puede usar la opción `--local-user`.

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

  - **Entrega:** Copia varias firmas del fichero `/etc/services` realizadas con distintas claves privadas. Subid al Campus Virtual (tarea Clave pública, cifrado y firma digital) la firma de dicho fichero firmada con la clave que has exportado para que yo pueda verificar vuestra firma. ¿Con qué clave habéis firmado? ¿El fichero gpg de la firma contiene solo la firma del fichero?