5. CRIPTOGRAFÍA EN DISPOSITIVOS MÓVILES

5.1 Principales conceptos sobre criptografía

5.1.1 Algoritmos de cifrado

Actualmente cada vez mayor número de actividades se está trasladando al mundo electrónico a través de Internet. Se hace, por lo tanto, necesario trasladar también los sistemas de seguridad a este contexto en el que el principal problema reside en que no existe contacto directo entre las partes implicadas. Necesitamos un documento digital que ofrezca las mismas funcionalidades que los documentos físicos con el plus de ofrecer garantías aún sin presencia física. Este problema se resuelve gracias a mecanismos criptográficos, los cuales se basan en dos elementos fundamentales: el certificado digital y la firma electrónica.

El concepto de cifrado es muy sencillo: dado un mensaje *en claro*, es decir, mensaje reconocible, al que se le aplique un algoritmo de cifrado, se generará como resultado un mensaje *cifrado* que sólo podrá ser descifrado por aquellos que conozcan el algoritmo utilizado y la clave que se ha empleado. Actualmente existen dos tipos de cifrado:

- Cifrado de clave privada o simétrico. Se emplea una sola clave para cifrar y descifrar el mensaje. El **beneficio** más importante de las criptografía de clave simétrica es su velocidad lo cual hace que éste tipo de algoritmos sean los más apropiados para el cifrado de grandes cantidades de datos. El **problema** que presenta la criptografía de clave simétrica es la necesidad de distribuir la clave que se emplea para el cifrado por lo que si alguien consigue hacerse tanto con el mensaje como con la clave utilizada, podrá descifrar el mensaje. Por esta razón se plantea el uso de claves asimétricas.

- *Cifrado de clave pública o asimétrico*. En este caso, cada usuario del sistema criptográfico ha de poseer una pareja de claves: la clave privada, que será custodiada por su propietario y no se dará a conocer a ningún otro, y la clave pública, que será conocida por todos los usuarios. Esta pareja de claves es complementaria: lo que cifra una sólo lo puede descifrar la otra y viceversa. Estas claves se obtienen mediante métodos matemáticos complicados de forma que por razones de tiempo de cómputo, es imposible conocer una clave a partir de la otra. El **beneficio** obtenido consiste en la supresión de la necesidad del envío de la clave, siendo por lo tanto un sistema más seguro. El **inconveniente** es la lentitud de la operación. Para solventar dicho inconveniente, el procedimiento que suele seguirse para realizar el cifrado de un mensaje es utilizar un algoritmo de clave pública junto a uno de clave simétrica.

Para solventar los inconvenientes de ambos mecanismos de cifrado, se suele usar un mecanismo de cifrado que utiliza un algoritmo de clave pública junto a uno de clave simétrica. El cifrado del mensaje se produce del siguiente modo:

Cada parte emisora posee su par de claves respectivamente. Cuando una de las partes decide enviar un mensaje, lo cifra con el sistema de criptografía de clave simétrica. La clave que utiliza se llama clave de sesión, y se genera aleatoriamente. Para enviar la clave de sesión de forma segura, ésta se cifra con la clave pública de la otra parte receptora, utilizando por tanto, criptografía de clave

asimétrica.

La parte receptora recibe el mensaje cifrado con la clave de sesión, que a su vez está cifrada con su clave pública. Para poder leer el mensaje, realiza el proceso de descifrado inverso. De esta forma se consigue confidencialidad e integridad. Aún así, todavía no quedan resueltos los problemas de no repudio y de autenticación.

5.1.1.1 Algoritmos de clave secreta o simétricos

En estos algoritmos, como ya se ha comentado, es necesario que antes de empezar a transmitir datos de forma segura, el emisor y el receptor se pongan de acuerdo en la clave a utilizar. A estos algoritmos se les llama de **clave secreta**, y no de **clave privada**, porque la clave no la conoce un sólo individuo, sino dos.

Estos algoritmos son los más clásicos, y fáciles de entender, pero tienen una limitación importante, y es que el emisor y el receptor se tienen que haber puesto en contacto previamente para acordar la clave secreta, lo cual es posible en algunos casos.

A continuación, vamos a analizar los algoritmos de clave secreta más comunes.

1. DES

DES es un algoritmo de encriptación inventado inicialmente por IBM en 1970 bajo el nombre "Lucifer". En 1977 fue adoptado por el NIST (National Institute of Standards and Technology) y por la NSA (National Security Agency) como su estándar nacional. En 1981 este algoritmo fue adoptado también por ANSI.

DES es un cifrador de bloque con tamaño de bloque de 64 bits. Para cada bloque de 64 bits de entrada se obtienen 64 bits a la salida, es decir, el tamaño del fichero resultante no crece ni decrece. La clave es de 56 bits. Se suele expresar como un número de 64 bits, pero el último bit de cada byte es de paridad, y se quita antes de aplicar el algoritmo.

El punto débil de DES no está en el algoritmo, sino en el tamaño de la clave, que es sólo de 56 bits, lo cual da lugar a demasiadas pocas claves posibles y es susceptible de sufrir un ataque por fuerza bruta. Actualmente un algoritmo de clave secreta se considera fuerte si tiene una clave de 128 bits o mayor. Esto ha dado lugar a que desde hace años estén surgiendo sistemas que desencriptan DES en tiempo cada vez menor.

Para acabar con este problema han surgido varias variantes de DES que sí que son seguras: **TripeDES** (también llamado DESede), no es más que DES aplicado tres veces con tres claves distintas. La primera vez en modo encriptación con la primera clave, la segunda en modo de desencriptación con la segunda clave, y la tercera en modo de encriptación con la tercera clave. La siguiente figura muestra gráficamente este proceso.

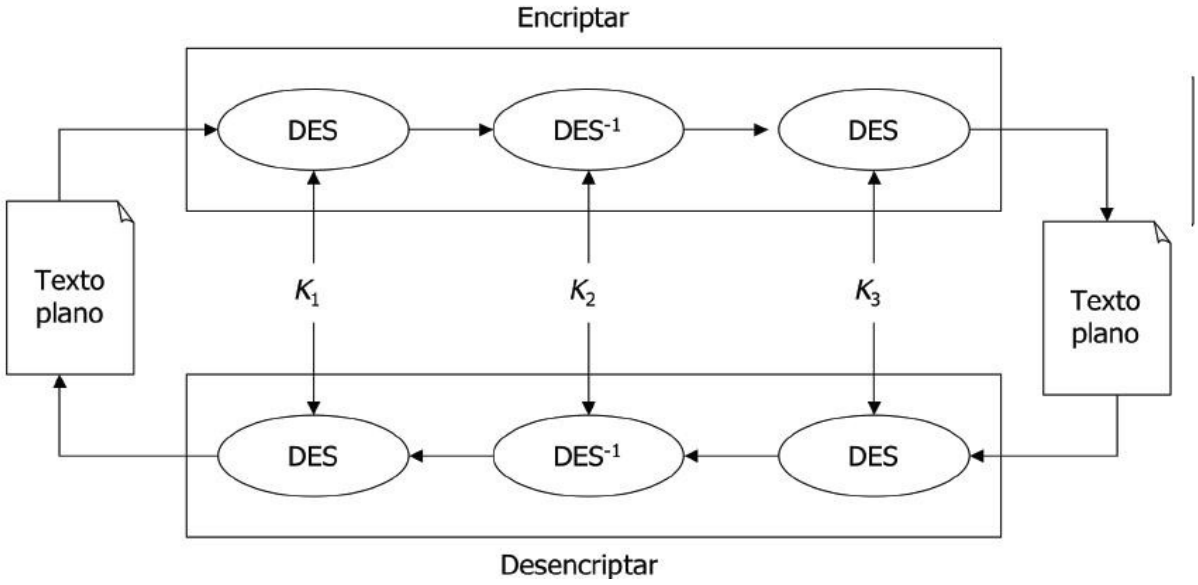

Figura 5.1. Proceso de encriptación y desencriptación en TripleDES

En TripeDES el tamaño de la clave es de 168 bits (3*56 bits) que es suficientemente segura como para eludir un ataque por fuerza bruta actual. Existe otra variante que usa claves de 112 bits, y donde la primera clave se usa la primera y tercera vez.

Otros algoritmos surgidos son DESede, cuyo nombre proviene de la forma en que se aplica el algoritmo, (Encryptation, Decryptation, Encryptation). También han surgido otras variantes de DES como són DESX, RDES o Generalized DES.

2. **Blowfish**. Es un algoritmo inventado por Bruce Schneier en 1993. Es de dominio público. Respecto a sus principales características, destaca que permite usar una clave variable de hasta 448 bits, y que está optimizado para microprocesadores de 32 bits y de 64 bits.

3. **IDEA** (International Data Encryption Algorithm) inventado en Zurich (Suiza) en 1990. Usa una clave de 128 bits, y es el que usa PGP para encriptar. Es seguro pero no de dominio público.

4. **RC2** (Rivest's Code). Fue desarrollado por Ronald Rivest de la empresa RSA Data Security, y el algoritmo es propiedad intelectual de la empresa RSA Data Security, que es vendida junto con su implementación. Posee claves de un tamaño de clave de 40 bits lo que lo hace inseguro.

5. **RC5**. Otra nueva implementación que actualmente vende RSA Data Security. Su principal característica es que permite usar claves de longitud variable.

6. **AES** (Advanced Encryption Standard), es el algoritmo elegido en Octubre del 2000 por el NIST como sustituto a DES. Utiliza un algoritmo llamado Rijndael, claves de 128, 192 y 256 bits, y tamaños de bloque de 128, 192 y 256 bytes respectivamente.

5.1.1.2 Algoritmos de clave pública o asimétricos

Esta nueva forma de criptografía fue propuesta por primera vez por Whitfield Diffie y Martin Hellman, dos profesores de la Universidad de Stanford en 1976. Propusieron un algoritmo de encriptación llamado algoritmo de la mochila, que sentó las bases de una nueva forma de pensar, en la que se basaron otros algoritmos, como por ejemplo el algoritmo RSA.

La gran novedad que introducen los algoritmos de clave pública es, como ya se ha comentado, que permiten que emisor y receptor se comuniquen de forma segura sin haberse puesto de acuerdo previamente en una clave secreta. Aquí, para que el emisor pueda enviar información al receptor de forma segura, el receptor debe disponer de dos claves: clave pública, que la debe conocer todo el mundo, y sólo vale para encriptar mensajes y la clave privada, que sólo la conoce el receptor (a diferencia de la clave secreta que la conocían emisor y receptor), y sólo vale para desencriptar mensajes.

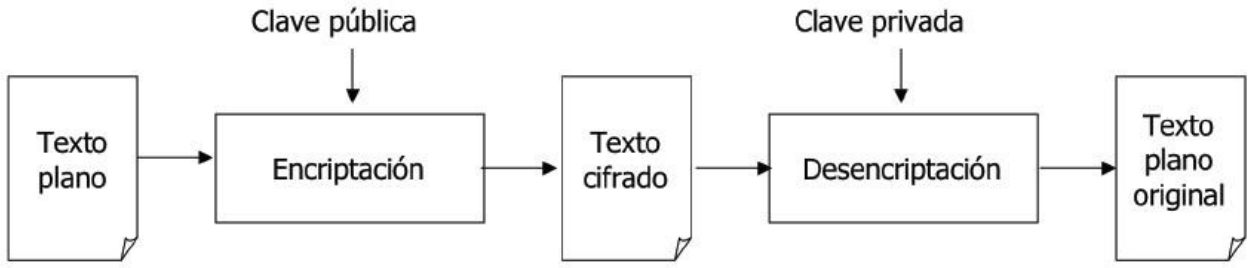

Figura 5.2. Algoritmos de clave pública

Tal como muestra la figura, el emisor puede encriptar mensajes y sólo el receptor puede desencriptarlos.

A continuación, vamos a analizar los algoritmos de clave pública más comunes.

1. RSA

RSA es seguramente el algoritmo de clave pública más conocido, más probado, y más sencillo que se conoce. Su nombre se debe a los tres inventores: Ron Rivest, Adi Shamir y Leonard Adleman.

Este algoritmo se basa en la dificultad de factorizar números grandes, es decir, sacar los primos que componen el número. Se cree que romper el algoritmo es equivalente a factorizar un número grande.

RSA se puede usar tanto para encriptación como para generar firmas digitales. RSA no tiene problemas de seguridad en el algoritmo, pero sí puede tener problemas de seguridad en el protocolo si éste no se sigue correctamente.

El principal problema es que al encriptar se encripta con las clave pública y se desencripta con la privada, y al firmar digitalmente se firma con la clave privada y se comprueba la firma con la clave pública. Si el atacante puede engañar al usuario para que firme un mensaje antes de encriptarlo, puede conseguir acceder al texto plano del mensaje cifrado.

El protocolo que seguiría el atacante sería:

- A quiere enviar un mensaje a B usando criptografía de clave pública RSA, pero hay un atacante escuchando la comunicación entre A y B.
- A encripta el mensaje usando la clave pública de B, y se lo envía a B. El atacante guarda el mensaje encriptado.
- El atacante manda a B un documento, dentro del cual está escrito el mensaje encriptado que captó en el paso anterior, para que B lo firme digitalmente.
- B manda el mensaje firmado al atacante, y donde antes estaba el mensaje cifrado ahora está el mensaje plano.

Para evitar este ataque se recomienda utilizar claves privadas distintas para encriptar y para firmar, o bien, usar para firmar otro algoritmo como por ejemplo DSA (Digital Signature Algorithm).

5.1.2 Firma digital

Una de las principales ventajas de la criptografía de clave pública es que ofrece un método para el desarrollo de *firmas digitales*. La firma digital permite al receptor de un mensaje verificar la autenticidad del origen de la información así como verificar que dicha información no ha sido modificada desde su generación. De este modo, la firma digital ofrece el soporte para la *autenticación* e *integridad* de los datos así como para el *no repudio* en origen, ya que el originador de un mensaje firmado digitalmente no puede argumentar que no lo es.

Una firma digital está destinada al mismo propósito que una manuscrita. Sin embargo, una firma manuscrita es sencilla de falsificar mientras que la digital es imposible mientras no se descubra la clave privada del firmante.

La firma digital se basa en la propiedad ya comentada sobre que un mensaje cifrado utilizando la clave privada de un usuario sólo puede ser descifrado utilizando la clave pública asociada. De tal manera, se tiene la seguridad de que el mensaje que ha podido descifrarse utilizando la clave pública sólo pudo cifrarse utilizando la privada. La firma digital, por tanto, es un cifrado del mensaje que se está firmando pero utilizando la clave privada en lugar de la pública.

Sin embargo ya se ha comentado el principal inconveniente de los algoritmos de clave pública: su lentitud que, además, crece con el tamaño del mensaje a cifrar. Para evitar éste problema, la firma digital hace uso de funciones *hash*. Una función hash es una operación que se realiza sobre un conjunto de datos de cualquier tamaño de tal forma que se obtiene como resultado otro conjunto de datos, en ocasiones denominado *resumen* de los datos originales, de tamaño fijo e independiente el tamaño original que, además, tiene la propiedad de estar asociado unívocamente a los datos iniciales, es decir, es prácticamente imposible encontrar dos mensajes distintos que tengan un resumen hash idéntico.

El proceso sería el siguiente:

Ambas partes que intervienen en la transmisión del mensaje tienen sus pares de claves respectivas.

La parte emisora escribe un mensaje a la parte receptora. Es necesario que la parte receptora pueda verificar que realmente es el emisor quien ha enviado el mensaje. Por lo tanto, el emisor debe enviarlo firmado:

1. Resume el mensaje mediante una función hash.
2. Cifra el resultado de la función hash con su clave privada. De esta forma obtiene su firma digital.
3. Envía a la parte receptora el mensaje original junto con la firma.

La parte receptora recibe el mensaje junto a la firma digital. Deberá comprobar la validez de ésta para dar por bueno el mensaje y reconocer al autor del mismo (integridad y autenticación).

4. Descifra el resumen del mensaje mediante la clave pública del emisor.
5. Aplica al mensaje la función hash para obtener el resumen.
6. Compara el resumen recibido con el obtenido a partir de la función hash. Si son iguales, el receptor puede estar seguro de que quien ha enviado el mensaje es el emisor y que éste no ha sido modificado.

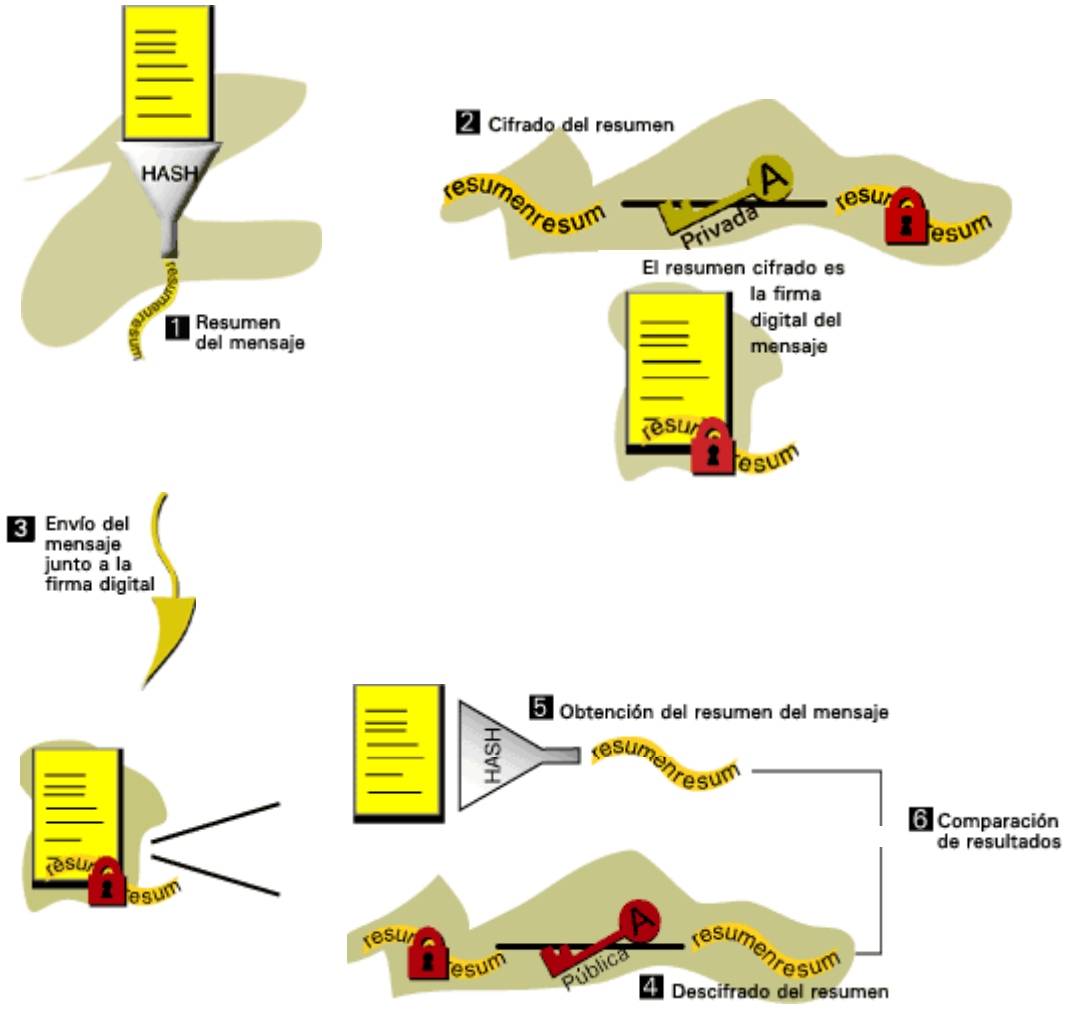

Figura 5.3. Proceso de firma digitalmente

Con este sistema conseguimos:

- **Autenticación**: la firma digital es equivalente a la firma física de un documento.
- **Integridad**: el mensaje no podrá ser modificado.
- **No repudio en origen**: el emisor no puede negar haber enviado el mensaje.

5.1.3 Certificados digitales

Según puede interpretarse de los apartados anteriores, la eficacia de las operaciones de cifrado y firma digital basadas en criptografía de clave pública sólo está garantizada si se tiene la certeza de que la clave privada de los usuarios sólo es conocida por dichos usuarios y que la pública puede ser dada a conocer a todos los demás usuarios con la seguridad de que no exista confusión entre las claves públicas de los distintos usuarios.

Para garantizar la unicidad de las claves privadas se suele recurrir a soportes físicos tales como tarjetas inteligentes o tarjetas PCMCIA que garantizan la imposibilidad de la duplicación de las claves. Además, las tarjetas criptográfica suelen estar protegidas por un número personal sólo conocido por su propietario que garantiza que, aunque se extravíe la tarjeta, nadie que no conozca dicho número podrá hacer uso de ella.

Por otra parte, para asegurar que una determinada clave pública pertenece a un usuario en concreto se utilizan los *certificados digitales*. Un certificado digital es un documento electrónico que asocia una clave pública con la identidad de su propietario.

Adicionalmente, además de la clave pública y la identidad de su propietario, un certificado digital puede contener otros atributos para, por ejemplo, concretar el ámbito de utilización de la clave pública, las fechas de inicio y fin de la validez del certificado, etc. El usuario que haga uso del certificado podrá, gracias a los distintos atributos que posee, conocer más detalles sobre las características del mismo.

La validez de un certificado es la confianza en que la clave pública contenida en el certificado pertenece al usuario indicado en el certificado. La validez del certificado en un entorno de clave pública es esencial ya que se debe conocer si se puede confiar o no en que el destinatario de un mensaje será o no realmente el que esperamos.

La manera en que se puede confiar en el certificado de un usuario con el que nunca hemos tenido ninguna relación previa es mediante la confianza en terceras partes. La forma en que esa tercera parte avalará que el certificado es de fiar es mediante su firma digital sobre el certificado. Por tanto, podremos confiar en cualquier certificado digital firmado por una tercera parte en la que confiamos. La TPC que se encarga de la firma digital de los certificados de los usuarios de un entorno de clave púbica se conoce con el nombre de Autoridad de Certificación (AC).

5.1.4 Infraestructura de clave pública

El modelo de confianza basado en Terceras Partes Confiables es la base de la definición de las Infraestructuras de Clave Pública (ICPs o PKIs, Public Key Infrastructures). Una infraestructura de

Clave Pública es un conjunto de protocolos, servicios y estándares que soportan aplicaciones basadas en criptografía de clave pública.

Algunos de los servicios ofrecidos por una ICP son los siguientes:

- Registro de claves: emisión de un nuevo certificado para una clave pública.
- Revocación de certificados: cancelación de un certificado previamente emitido.
- Selección de claves: publicación de la clave pública de los usuarios.
- Evaluación de la confianza: determinación sobre si un certificado es válido y qué operaciones están permitidas para dicho certificado.
- Recuperación de claves: posibilidad de recuperar las claves de un usuario.

5.2 X.509

5.2.1 Introducción

En criptografía, **X.509** es un estándar IUT-T para infraestructuras de claves públicas (en inglés, *Public Key Infrastructure* o PKI). X.509 especifica, entre otras cosas, formatos estándar para certificados de clave pública y un algoritmo de validación de la ruta de certificación.

X.509 fue publicado oficialmente en 1988 y comenzado conjuntamente con el estándar X.500 y asume un sistema jerárquico estricto de autoridades certificantes (ACs) para emisión de certificados. Esto contrasta con modelos de web de confianza, como PGP, donde cualquiera (no solo ACs) pueden firmar, y por ende avalar la validez de certificados de claves de otros. La versión 3 de X.509 incluye la flexibilidad de soporte de otras tecnologías como bridges y mallas. El sistema X. 500 nunca se implementó completamente, y el grupo de trabajo de la infraestructura de clave pública de la IETF, comúnmente conocido como PKIX para *infraestructura de clave pública (X. 509)* o PKIX, adaptó el estándar a la estructura más flexible de Internet. X.509 incluye también estándares para implementación de listas de certificados en revocación (CLRs), y con frecuencia aspectos de sistemas PKI. De hecho, el término *certificado X.509* se refiere comúnmente al Certificado PKIX y Perfil CRL del certificado estándar de X.509 v3 de la IETF, como se especifica en la RFC 3280.

5.2.2 Estructura de un certificado X.509 v3

La estructura de un certificado digital X.509 v3 es la siguiente:

- Certificado
    - Versión
    - Número de serie
    - ID del algoritmo
    - Emisor
    - Validez

        - No antes de
        - No después de
    - Sujeto
    - Información de clave pública del sujeto
        - Algoritmo de clave pública
        - Clave pública del sujeto
    - Identificador único de emisor (opcional)
    - Identificador único de sujeto (opcional)
    - Extensiones (optional)
        - ...
- Algoritmo usado para firmar el certificado
- Firma digital del certificado

Los identificadores únicos de emisor y sujeto fueron introducidos en la versión 2, y las extensiones en la versión 3. X.509 es la pieza central de la infraestructura de clave pública y es la estructura de datos que enlaza la clave pública con los datos que permiten identificar al titular. Su sintaxis se define empleando el lenguaje ASN.1 (*Abstract Syntax Notation One*) y los formatos de codificación más comunes son DER (*Distinguished Encoding Rules*) o PEM (*Privacy-enhanced Electronic Mail*). Siguiendo la notación de ASN.1, un certificado contiene diversos campos, agrupados en tres grandes grupos:

- El primer campo es el *subject* (sujeto), que contiene los datos que identifican al sujeto titular. Estos datos están expresados en notación DN (Distinguished Name), donde un DN se compone a su vez de diversos campos, siendo los más frecuentes los siguientes; CN (*Common Name*), OU (*Organizational Unit*), O (*Organization*) y C (*Country*). Además del nombre del sujeto titular (*subject*), el certificado, también contiene datos asociados al propio certificado digital, como la versión del certificado, su identificador (*serialNumber*), la CA firmante (issuer), el tiempo de validez (*validity*), etc. La versión X.509.v3 también permite utilizar campos opcionales (nombres alternativos, usos permitidos para la clave, ubicación de la CRL y de la CA,etc.).

- En segundo lugar, el certificado contiene la clave pública, que expresada en notación ASN. 1, consta de dos campos: el que muestra el algoritmo utilizado para crear la clave (ej. RSA), y la propia clave pública.

- Por último, la CA, ha añadido la secuencia de campos que identifican la firma de los campos previos. Esta secuencia contiene tres atributos, el algoritmo de firma utilizado, el hash de la firma, y la propia firma digital.

Un ejemplo real originado con el comando ssl es el siguiente:

```
openssl x509 -text -in eojeda.pem
```

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 1016556141 (0x3c976a6d)
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=ES, O=FNMT, OU=FNMT Clase 2 CA
        Validity
```

```
            Not Before: Nov 15 13:36:46 2007 GMT
            Not After : Nov 15 13:36:46 2010 GMT
        Subject: C=ES, O=FNMT, OU=FNMT Clase 2 CA, OU=500743266, CN=NOMBRE OJEDA
RODRIGUEZ ELENA - NIF 47206857R
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:c0:68:84:d8:39:69:77:3d:9b:26:46:cd:d5:51:
                    6a:a5:0c:8f:4f:e3:9f:fb:7e:c7:ee:b3:23:46:f3:
                    07:9b:dc:6b:9c:a8:e0:ae:37:d2:ad:83:fb:0a:93:
                    1f:a0:21:23:c1:82:1f:82:cf:57:ae:9d:a4:3d:46:
                    d1:01:73:f5:5a:d3:16:99:59:17:b1:8b:e1:63:ee:
                    33:8f:80:f3:34:ec:05:e6:d8:f1:71:34:b4:22:6b:
                    d2:d3:3b:f7:84:96:2c:53:bb:fe:29:c5:92:73:fa:
                    5f:2b:50:c1:cb:78:22:36:c9:ef:a5:7a:08:94:ba:
                    5f:34:b7:99:b7:f0:27:08:87
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Alternative Name:
                email:ELENA.OJEDA.RODRIGUEZ@GMAIL.COM,
DirName:/1.3.6.1.4.1.5734.1.4=47206857R/1.3.6.1.4.1.5734.1.3=RODRIGUEZ/1.3.6.1.4
.1.5734.1.2=OJEDA/1.3.6.1.4.1.5734.1.1=ELENA
            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 Private Key Usage Period:
                Not Before: Nov 15 13:36:46 2007 GMT, Not After: Nov 15 13:36:46
2010 GMT
            X509v3 Key Usage:
                Digital Signature, Key Encipherment
            Netscape Cert Type:
                SSL Client, S/MIME
            X509v3 Subject Key Identifier:
                77:F8:51:97:6B:26:5C:1E:EE:42:16:3D:35:34:62:C1:79:4A:27:7F
            X509v3 Authority Key Identifier:
                keyid:40:9A:76:44:97:74:07:C4:AC:14:CB:1E:8D:4F:3A:45:7C:
30:D7:61

            X509v3 Certificate Policies:
                Policy: 1.3.6.1.4.1.5734.3.5
                  CPS: http://www.cert.fnmt.es/convenio/dpc.pdf
                  User Notice:
                    Explicit Text: Certificado Reconocido expedido según
legislación vigente.Uso limitado a la Comunidad Electrónica por valor máximo de
100 e salvo excepciones en DPC.Contacto FNMT:C/Jorge Juan 106-28009-Madrid-
España.

            1.3.6.1.4.1.5734.1.33:
                ..PERSONA FISICA
            qcStatements:
                0!0......F..0......F..0...EUR..d...
            X509v3 CRL Distribution Points:
                DirName:/C=ES/O=FNMT/OU=FNMT Clase 2 CA/CN=CRL3850

    Signature Algorithm: sha1WithRSAEncryption
        4a:05:aa:8f:08:84:99:28:cf:df:d8:71:9f:99:a7:36:6d:b2:
        36:a9:34:58:e0:4e:dc:64:ef:66:12:c5:e8:aa:2f:b5:9f:ad:
        f1:d0:40:a0:67:93:1e:40:4f:1f:20:c3:86:1e:1d:cc:0d:82:
        5a:3c:97:da:9c:a2:c2:6f:b9:7c:3b:b6:60:8b:1b:3d:97:7f:
        20:91:18:59:8c:19:e1:3f:8d:ca:a9:db:fe:c4:f8:b1:f9:23:
```

```
25:1e:d5:87:9e:65:b3:b1:5a:89:c1:d5:94:11:42:a9:ea:83:
5d:b8:f0:76:f2:d2:e4:b2:bf:d7:83:d3:f9:d8:35:46:eb:bf:
82:2e
```

5.2.3 Extensión de certificados

Las extensiones de archivo de certificados X.509 son:

- .CER - Certificado codificado en CER, algunas veces es una secuencia de certificados
- .DER - Certificado codificado en DER
- .PEM - Certificado codificado en Base64, encerrado entre "-----BEGIN CERTIFICATE-----" y "-----END CERTIFICATE-----". Un archivo `.PEM` puede contener certificados o claves privadas, encerrados entre las líneas BEGIN/END apropiadas. Un ejemplo sería lo siguiente:

```
                    -----BEGIN CERTIFICATE-----
MIIFRDCCBK2gAwIBAgIEPJdqbTANBgkqhkiG9w0BAQUFADA2MQswCQYDVQQGEwJF
UzENMAsGA1UEChMERk5NVDEYMBYGA1UECxMPRk5NVCBDbGFzZSAyIENBMB4XDTA3
MTExNTEzMzY0NloXDTEwMTExNTEzMzY0NlowgYExCzAJBgNVBAYTAkVTMQ0wCwYD
VQQKEwRGTk1UMRgwFgYDVQQLEw9GTk1UIENsYXNlIDIgQ0ExEjAQBgNVBAsTCTUw
MDc0MzI2NjE1MDMGA1UEAxMsTk9NQlJFIE9KRURBIFJPRFJJR1VFWiBFTEVOQSAt
IE5JRiA0NzIwNjg1N1IwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMBohNg5
aXc9myZGzdVRaqUMj0/jn/t+x+6zI0bzB5vca5yo4K430q2D+wqTH6AhI8GCH4LP
V66dpD1G0QFz9VrTFplZF7GL4WPuM4+A8zTsBebY8XE0tCJr0tM794SWLFO7/inF
knP6XytQwct4IjbJ76V6CJS6XzS3mbfwJwiHAgMBAAGjggMRMIIDDTCBkAYDVR0R
BIGIMIGFgR9FTEVOQS5PSkVEQS5ST0RSSUdVRVpAR01BSUwuQ09NpGIwYDEYMBYG
CSsGAQQBrGYBBBMJNDcyMDY4NTdSMRgwFgYJKwYBBAGsZgEDEwlST0RSSUdVRVox
FDASBgkrBgEEAaxmAQITBU9KRURBMRQwEgYJKwYBBAGsZgEBEwVFTEVOQTAJBgNV
HRMEAjAAMCsGA1UdEAQkMCKADzIwMDcxMTE1MTMzNjQ2WoEPMjAxMDExMTUxMzM2
NDZaMAsGA1UdDwQEAwIFoDARBglghkgBhvhCAQEEBAMCBaAwHQYDVR0OBBYEFHf4
UZdrJlwe7kIWPTU0YsF5Sid/MB8GA1UdIwQYMBaAFECadkSXdAfErBTLHo1POkV8
MNdhMIIBMQYDVR0gBIIBKDCCASQwggEgBgkrBgEEAaxmAwUwggERMDQGCCsGAQUF
BwIBFihodHRwOi8vd3d3LmNlcnQuZm5tdC5lcy9jb252ZW5pby9kcGMucGRmMIHY
BggrBgEFBQcCAjCByxqByENlcnRpZmljYWRvIFJlY29ub2NpZG8gZXhwZWRpZG8g
c2Vn+m4gbGVnaXNsYWNp824gdmlnZW50ZS5Vc28gbGltaXRhZG8gYSBsYSBDb211
bmlkYWQgRWxlY3Ry825pY2EgcG9yIHZhbG9yIG3heGltbyBkZSAxMDAgZSBzYWx2
byBleGNlcGNpb25lcyBlbiBEUEMuQ29udGFjdG8gRk5NVDpDL0pvcmdlIEp1YW4g
MTA2LTI4MDA5LU1hZHJpZC1Fc3Bh8WEuMB0GCSsGAQQBrGYBIQQQFg5QRVJTT05B
IEZJU0lDQTAvBggrBgEFBQcBAwQjMCEwCAYGBACORgEBMBUGBgQAjkYBAjALEwNF
VVICAWQCAQAwWwYDVR0fBFQwUjBQoE6gTKRKMEgxCzAJBgNVBAYTAkVTMQ0wCwYD
VQQKEwRGTk1UMRgwFgYDVQQLEw9GTk1UIENsYXNlIDIgQ0ExEDAOBgNVBAMTB0NS
TDM4NTAwDQYJKoZIhvcNAQEFBQADgYEASgWqjwiEmSjP39hxn5mnNm2yNqk0WOBO
3GTvZhLF6KovtZ+t8dBAoGeTHkBPHyDDhh4dzA2CWjyX2pyiwm+5fDu2YIsbPZd/
IJEYWYwZ4T+Nyqnb/sT4sfkjJR7Vh55ls7FaicHVlBFCqeqDXbjwdvLS5LK/14PT
                          +dg1Ruu/gi4=
                    -----END CERTIFICATE-----
```

- .P7B y P7C - Estructura PKCS#7 SignedData sin datos, solo certificado(s) o CRL(s) . En la siguiente sección de explica con mayor detenimiento cada uno de los tipos de estructura PKCS7.
- .PFX y .P12 - PKCS#12, puede contener certificado(s) (público) y claves privadas

(protegido con clave). Evolucionó del estándar PFX (Personal inFormation eXchange) y se usa para intercambiar objetos públicos y privados dentro de un archivo.

5.3 Estándar PKCS7

En criptografía, **PKCS** se refiere a un grupo de estándares de criptografía de clave pública concebidos y publicados por los laboratorios de RSA en California. A RSA Security le asignaron los derechos de licenciamiento para la patente de algoritmos de clave asimétrica RSA y adquirió los derechos de licenciamiento para muchas otras patentes de claves.

El estándar PKCS7 #7 es un conjunto de normas para firmar y encriptar documentos. Podemos encontrar varios tipos de objetos PKCS #7:

- data: Sólo datos, usado para enviar datos sin encriptar.
- signed sata: datos firmados, usado para autentificación del remitente
- enveloped data: datos juntos - o enfundados - que pueden ser datos o datos firmados o datos encriptados o varios de ellos a la vez,utilizados para confidencialidad.
- signed-and-enveloped data: datos firmados y enfundados. Para autenticidad y confidencialidad.
- digested data: datos resumidos, para comprobar la integridad del mensaje.
- encrypted data: datos encriptados, utilizados para confidencialidad.

A continuación, analizaremos los tipos de objetos más comunes con mayor profundidad.

5.3.1 Tipo de contenido Data

El tipo data es únicamente la información en una cadena de octetos, sin ningún tipo de manipulación criptográfica.

5.3.2 Tipo de contenido SignedData

Representa cualquier tipo de datos que estén firmados por uno o varios firmantes mediante un resumen de los datos, y su posterior encriptación utilizando la clave privada de cada firmante sobre el resumen de los datos. A esta encriptación del resumen es a lo que se llama firma electrónica. Para crear un tipo signed data se siguen los siguientes pasos:

- Para cada firmante, se calcula un resumen con el algoritmo que use el firmante - si dos firmantes utilizan el mismo algoritmo, el resumen sólo se calcula para uno de ellos, el otro utiliza el del otro firmante - Si uno de los firmantes quiere autentificar cualquier otro tipo de información que quiera incluir adicionalmente, se le calculará el resumen del mensaje original junto con la información de más que ese firmante haya querido poner.

- Para cada firmante, el resumen y la información asociada a éste se encripta utilizando la

clave privada del firmante.

- Para cada firmante, el resumen del mensaje encriptado y otra información específica del remitente se agrupan en un campo del objeto llamdo `SignerInfo`. Otros datos como una lista de certificados revocados de cada firmante y sus certificados se agrupan en este paso.

- Los algoritmos que se han utilizado para resumir el mensaje - de todos los firmantes - y el campo `SignerInfo` se agrupan en otro campo llamado `SignedData` junto con el mensaje original.

Los pasos que debe realizar el destinatario son más simples:

- Verificar las firmas desenciptando cada resumen para cada firmante con la clave pública del mismo - las claves públicas de los firmantes están dentro del mismo objeto con el formato de certificados.
- Comparar los resúmenes utilizando el mismo algoritmo que utilizó el firmante al crearlo y compararlo con el extraído en el paso anterior.

Los tipos de datos en ASN.1 contenidos en un **SignedData** son:

```
SignedData::= SEQUENCE {
    version Version;
    digestAlgorithms DigestAlgorithmsIdentifiers,
    contentInfo ContentInfo,
    certificates
        [0] IMPLICIT ExtendedCertificatesAndCertificates
            OPTIONAL,
    crls
        [1] IMPLICIT CertificateRevocationLists
            OPTIONAL,
    signerInfos SignerInfos
    }
DigestAlgorithmIdentifiers ::= SET OF DigestAlgorithmIdentifier
SignerInfos ::= SET OF SignerInfo
```

Donde:

- `version` es la versión del estándar que se está utilizando
- `digestAlgorithms` son un conjunto de identificadores de algoritmos para resúmenes - inclusive ninguno -. Cada elemento identifica el algoritmo utilizado para resumir el contenido y los parámetros que se han utilizado.
- `contentInfo` es el objeto que queremos firmar.
- `certificates` es un conjunto de certificados extendidos tipo PKCS#6 y certificados X. 509. Se intenta que el conjunto sea suficiente para contener certificados hasta una CA raíz reconocida para todos los firmantes que están en el campo `SignersInfo`. Pueden haber también certificados suficientes para que la CA raíz no sea única, en el caso de que fuera única, los certificados podrán ser verificados por otros medios.
- `crls` es un conjunto de listas de certificados revocados.
- `signersInfo` es un conjunto de información sobre los firmantes, pueden haber cualquier número de elementos, incluido ninguno.

Los tipos de datos en ASN.1 contenido en SignerInfo son:

```
SignerInfo ::= SEQUENCE {
    version Version,
    issuerAndSerialNumber IssuerAndSerialNumber,
    digestAlgorithm DigestAlgorithmIdentifier,
    authenticatedAttributes
        [0] IMPLICIT Attributes OPTIONAL,
    digestEncryptionAlgorithm
        DigestEncryptionAlgorithmIdentifier,
    encryptedDigest EncryptedDigest,
    unautenticatedAttributes
        [1] IMPLICIT Attributes OPTIONAL
    }
EncryptedDigest ::= OCTET STRING
```

Donde:

- `version` es la version de la sintaxis que se está utilizando
- `issuerAndSerialNumber` precisa el certificado del firmante - el nombre del mismo y su clave pública - mediante el nombre de la CA y el número de serie del certificado.
- `digestAlgorithm` identifica el algoritmo utilizado para resumir el contenido del mensaje y los atributos que se han querido autenticar con los parámetros que se han utilizado al resumir.
- `authenticatedAttributes` representan un conjunto de atributos que han sido firmados. Este campo es opcional, pero debe estar presente si el tipo de contenido del campo `contentInfo` no es data. Si el campo está, entonces debe tener como mínimo dos atributos:
    1. Un objeto PKCS#9 del tipo contenido, el cual nos dice el tipo de contenido que lleva el mensaje.
    2. Otro objeto PKCS#9 del tipo resumen del mensaje, que nos dice el resumen del mensaje.
- `digestEncryptionAlgorithm` identifica el algoritmo utilizado para encriptar el mensaje y la información suplementaria con la clave privada del firmante.
- `encryptedDigest` es el resumen y la información adicional encriptados.

5.3.3 Tipo de contenido EnvelopedData

Este tipo de contenido consiste en un contenido cifrado de cualquier tipo junto con la llave de sesión cifrada con la/s clave/s pública/s de el/los destinatario/s. Ésta combinación es lo que se conoce como un sobre digital para cada destinatario. Es decir, cualquier tipo de datos se puede meter en un sobre y enviarlo a cualquier número de destinatarios. Normalmente, el uso que se dará a este tipo será el envío a varios destinatarios de contenido cifrado del tipo data, digested-data o signed-data.

Para crear un tipo envelope necesitamos hacer los siguientes pasos:

- Se genera de forma aleatoria una clave de sesión para un algoritmo de encriptación en

particular.

- Para cada destinatario, la clave de sesión se cifra con la clave pública del destinatario
- Para cada destinatario, la clave de sesión y alguna otra información propia del destinatario se agrupa en un valor del tipo RecipientInfo.
- El contenido se cifra con la llave de sesión
- Los valores del tipo RecipientInfo - uno por cada destinatario - se juntan con el contenido cifrado del mensaje original, creándose así el "sobre".

El proceso a seguir por el destinatario es el siguiente:

- El destinatario abre el "sobre" descifrando la única parte que puede descifrar - la que hace referencia a él ya que está cifrada con su clave pública - y descifra la clave de sesión.
- Se descifra el contenido utilizando esta clave de sesión que se ha descifrado antes. La llave privada del destinatario está referenciada mediante el nombre de la CA - issuer- y el número de serie del certificado, entonces únicamente distingue como válido ese certificado al que referencia.

La estructura de datos **EnvelopedData** en ASN.1 es la que sigue:

```
EnvelopedData ::= SEQUENCE {
    version Version,
    recipientInfos RecipientInfos,
    encryptedContentInfo EncriptedContentInfo
    }
RecipientInfos ::= SET OF RecipientInfo
EncryptedContentInfo ::= SEQUENCE {
    contentType ContentType,
    contentEncryptionAlgorithm ContentEncryptionAlgorithmIdentifier,
    encryptedContent
        [0] IMPLICIT EncryptedContent OPTIONAL
    }
EncryptedContent ::= OCTET STRING
```

5.3.4 Tipo de contenido SignedEnvelopedData

Este tipo consisten en un contenido cifrado de cualquier tipo, una llave de sesión cifrada para uno o más destinatarios y los resúmenes doblemente cifrados del mensaje para uno o más firmantes - doblemente cifrados, ya que se cifran primero para firmar con la clave privada del firmante y luego, se cifran con la llave de sesión del cifrado. Así creamos un sobre - ya que el uso de la combinación del cifrado del mensaje y el cifrado de la llave de sesión es un sobre digital - firmado o autenticado. El resumen encriptado del mensaje con la clave privada del remitente es una firma digital. Cualquier tipo de contenido puede ser metido en un sobre y además firmado por cualquier número de firmantes en paralelo.

El proceso con el que se crea este tipo de datos es el siguiente:

- Se genera una llave de sesión aleatoria para un algoritmo de encriptación en particular
- Para cada destinatario se cifra la llave de sesión con su clave pública.

- Para cada destinatario, se une la llave de sesión encriptada junto con información adicional del destinatario en una estructura RecipientInfo.

- Para cada firmante, se calcula un resumen del mensaje utilizando el algoritmo que desee el firmante, si dos firmantes utilizan un mismo algoritmo, sólo se calcula una sola vez.

- Para cada firmante, el resumen y otra información adicional se cifra con la clave pública propia de cada firmante para después ser cifrada de nuevo con la llave de sesión utilizada.

- Para cada firmante, el resumen doblemente cifrado se fusiona con otra información adicional del firmante en una estructura de datos `SignerInfo`.

- El contenido del mensaje se cifra con la llave de sesión.

- Los resúmenes del mensaje de todos los firmantes, las estructuras `SignerInfo` y `RecipientInfo` de todos los destinatarios y firmantes se añaden al mensaje encriptado en una estructura `SignedAndEnvelopedData`.

La estructura de datos **SignedAndEnvelopedData** en ASN.1 es la que sigue:

```
SignedAndEnvelopedData ::= SEQUENCE {
    version Version,
    recipientInfos RecipientInfos,
    digestAlgorithms DigestAlgorithmsIdentifiers,
    encryptedContentInfo EncriptedContenInfo,
    certificates
        [0] IMPLICIT ExtendedCertificatesAndCertificates OPTIONAL,
    crls
        [1] IMPLICIT CertificateRevocationLists OPTIONAL,
    signerInfos SignerInfos
    }
```

Hasta aquí, los distintos tipos de estructuras de datos PKCS#7.

5.4 Librería Bouncy Castle

La librería criptográfica Bouncy Castle es una implementación Java de algoritmos criptográficos. El paquete está organizado de tal forma que contiene una API adecuada para ser usada en cualquier entorno, incluido J2ME.

El software es distribuido bajo licencia basada en MIT X Consortium. La librería OpenPGP también

incluye una librería BZIP2 modificada cuya licencia está bajo la licencia de Apache Software, Versión 1.1.

El paquete completo posee los siguientes .jar: bcprov.jar y jce.jar, que contiene el JCE y bcmail.jar, que contiene el API mail.

Algunos de los algoritmos que proporciona son los siguientes:

- Algoritmos simétricos:

Algorimo	Constructor
BufferedBlockCipher	BlockCipher
CBCBlockCipher	BlockCipher
CFBBlockCipher	BlockCipher, block size (in bits)
CCMBlockCipher	BlockCipher
EAXBlockCipher	BlockCipher
OFBBlockCipher	BlockCipher, block size (in bits)
SICBlockCipher	BlockCipher, block size (in bits)
OpenPGPCFBBlockCipher	BlockCipher
GOFBBlockCipher	BlockCipher

Algunos de los cifrados usados por los algoritmos anteriores son los siguientes:

Nombre	Tamaño de claves (en bits)
AESEngine	0 .. 256
BlowfishEngine	0 .. 448
CamelliaEngine	128, 192, 256
DESEngine	64
DESedeEngine	128, 192
IDEAEngine	128
NoekeonEngine	128
RC2Engine	0 .. 1024
RC532Engine	0 .. 128
RC564Engine	0 .. 128
RC6Engine	0 .. 256

- Algoritmos asimétricos:

Algoritmo	Constructor
BufferedAsymmetricBlockCipher	AsymmetricBlockCipher
OAEPEncoding	AsymmetricBlockCipher
PKCS1Encoding	AsymmetricBlockCipher
ISO9796d1Encoding	AsymmetricBlockCipher

Los cifrados más comunes usados por los algoritmos anteriores son:

Nombre	Tamaño de claves (en bits)
RSAEngine	Cualquier múltiplo de 18 suficientemente grande para codificar
ElGamalEngine	Cualquier múltiplo de 18 suficientemente grande para codificar

La librería BouncyCastle implementa JCE 1.2.1 con generadores/procesadores de S/MIME y CMS. Para usarlo, hay que especificarlo en las llamadas al JCE:

```
Cipher.getInstance("Blowfish/ECB/NoPadding", "BC");
```

porque sino, utiliza librerías propias de Sun. Estas precedencias están en un fichero llamado jre/lib/security.

Los demás especificaciones usadas relativas a la librería Bouncy Castle se especificarán en un capítulo posterior, donde se detallarán las clases usadas.