XUNTA DE GALICIA

ESCOLA GALEGA DE ADMINISTRACIÓN PÚBLICA

38. CERTIFICADOS DIGITALES. TARJETAS CRIPTOGRÁFICAS. FIRMA DIGITAL. TÉCNICAS DE CIFRADO. INFRAESTRUCTURA DE CLAVE PÚBLICA (PKI).

Fecha de actualización 26 de julio de 2011

XUNTA DE GALICIA

ESCOLA GALEGA DE ADMINISTRACIÓN PÚBLICA

Tema 38: Certificados digitales. Tarjetas criptográficas. Firma digital. Técnicas de cifrado. Infraestructura de clave pública (PKI).

INDICE

1. TECNICAS DE CIFRADO

La palabra **criptografía** es una palabra de origen griego (krypto –oculto- y graphos –escribir-) y se define como el arte de escribir con clave secreta o de un modo enigmático.

1.1. Criptografía y criptoanálisis

La historia de la criptografía se remonta a miles de años atrás y tiene una larga tradición en las escrituras religiosas que podrían ofender a la cultura dominante o a las autoridades políticas. La finalidad de esta técnica ha sido siempre enviar mensajes confidenciales con la garantía de que sólo el destinatario de los mismos pudiera acceder a la información contenida en el mensaje.

El método consiste en la aplicación de una transformación al mensaje conocida como **cifrado**, con el objetivo de que las personas que desconozcan la transformación realizada sean incapaces de acceder a la información contenida en el mensaje.

El estudio de técnicas destinadas a encontrar el sentido de una información cifrada, sin tener acceso a la información secreta requerida, es el **criptoanálisis**. La finalidad del criptoanálisis es, por tanto, descubrir la clave de cifrado. La vulnerabilidad de los algoritmos de cifrado dependerá de la dificultad de la tarea de descubrimiento de la clave. Una ataque por fuerza bruta consiste en buscar la clave de cifrado probando uno a uno todos los posibles valores de la.

La **criptología** es la disciplina que abarca la criptografía y el criptoanálisis.

Otro concepto relacionado es la **esteganografía**. Al igual que la criptografía lo que busca es ocultar un mensaje ante un posible atacante, pero la diferencia estriba en cómo ocultan la información ambas técnicas:

mientras que la criptografía pretende que la información no sea descifrada, la esteganografía lo que pretende es que la información pase desapercibida (por ejemplo un código secreto tatuado en el cuero cabelludo y oculto por el pelo)

Las técnicas criptográficas se pueden clasificar siguiendo varios criterios. Siguiendo un **criterio temporal** se pueden clasificar en clásicas o extemporáneas y modernas o contemporáneas.

1.2. Técnicas criptográficas clásicas

Las técnicas criptográficas clásicas**,** realizan el cifrado en base a la sustitución y transposición de los caracteres del mensaje. El secreto está en el algoritmo aplicado al mensaje, por lo que tienen el inconveniente de que si un atacante lo descubre, será capaz de interpretar todos los mensajes cifrados que capture. Como ejemplos podemos citar:

- **Sustitución monoalfabeto**: consiste en la sustitución de símbolos uno a uno. Como ejemplo se puede citar el algoritmo de César.
- **Sustitución polialfabeto**: consiste en la sustitución de un símbolo por uno de un conjunto. Como ejemplo se puede citar el cifrado de Vigenère.
- **Transposición**: consiste en cambiar el orden de los símbolos.
- **Combinación de sustitución y transposición** (máquinas rotoras).

1.3. Técnicas criptográficas modernas

Las técnicas criptográficas modernas, a diferencia de las clásicas, utilizan claves de cifrado para cifrar la información. Una premisa fundamental de la criptografía moderna es que la seguridad del método debe depender únicamente de la clave de cifrado, debiendo ser los algoritmos conocidos.

Esta premisa hace que estas técnicas resulten mucho más seguras y efectivas ya que resulta más sencillo mantener el secreto de la clave y además cambiar la clave de cifrado siempre será menos costoso que idear un nuevo algoritmo resultando frecuente que la clave de cifrado se genere de forma automática.

Podemos clasificar los algoritmos de cifrado atendiendo a las claves que utilizan o al modo en que procesan la información.

Si nos fijamos en el tipo de claves que utilizan tenemos dos tipos de algoritmos:

- Algoritmos de **cifrado simétrico o de clave privada**: Utilizan la misma clave para el cifrado y el descifrado por lo que debe ser secreta y compartida por el emisor y el receptor. Ejemplos de algoritmos de este tipo son DES, 3DES, AES, IDEA, RC5, etc.
- Sistemas de cifrado **asimétrico o de clave pública**: Utilizan un par de claves generadas por el emisor. Una de las claves es pública, es decir, conocida por todo el mundo y la otra es privada o secreta de forma que lo que se cifra con una clave es descifrado por la otra y viceversa. Ejemplos de algoritmos de este tipo son RSA, DSA, Diffie-Hellman, ElGamal, etc.

Si nos fijamos en el modo en que procesan tenemos 3 tipos de algoritmos:

- Técnicas criptográficas de cifrado en **modo flujo** (*stream cipher*): estos algoritmos de cifrado se basan en la combinación de un texto en claro con un texto de cifrado obtenido a partir de una clave. La característica fundamental es que se va cifrando un flujo de datos bit a bit. Ejemplos: RC4, SEAL.
- Técnicas criptográficas de cifrado en **modo bloque** (*block cipher*): se caracterizan porque el algoritmo de cifrado o descifrado se aplica separadamente a bloques de longitud *l*, y para cada uno de ellos el

resultado es un bloque de la misma longitud: Ejemplos: DES, 3DES, AES.

- Técnicas criptográficas basadas en funciones resumen (**hash functions**): la característica principal de estos algoritmos es que permiten obtener una cadena de bits de longitud fija a partir de un mensaje de longitud arbitraria: Ejemplos: MD5, familia SHA.

1.4. Criptografía de clave privada o simétrica

La criptografía de clave simétrica, se caracteriza porque la clave de descifrado ***k,*** es idéntica a la clave de cifrado o se puede obtener a partir de esta, residiendo de este modo la fortaleza del algoritmo en el secreto de la misma.

Si **M** es el mensaje en claro que se quiere proteger, al cifrarlo con un algoritmo en base a una clave privada $\mathbf{E_k(M)}$ se obtiene otro mensaje llamado texto cifrado **C**. Para que este cifrado sea útil, existe otra función $\mathbf{D_k(C)}$ que a partir del texto cifrado por el emisor permite obtener de nuevo el mensaje en claro **M**.

$$\mathbf{C = E_k(M)}$$

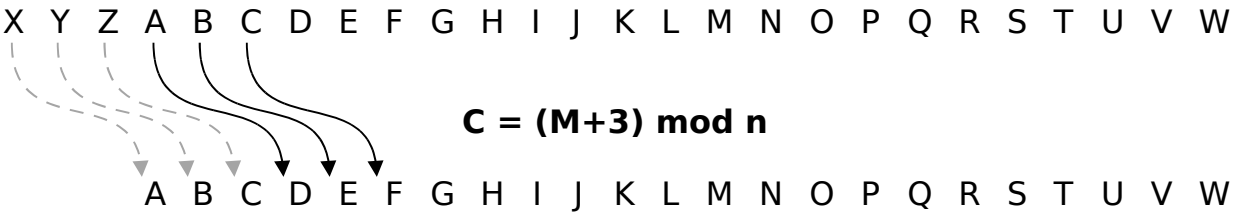
$$\mathbf{M = D_k(C) = D_k(E_k(M))}$$

La seguridad del sistema recae pues en mantener en secreto la clave ***k.*** El principal inconveniente de la criptografía simétrica, es el **intercambio de claves**. Este problema se soluciona con ayuda de la criptografía asimétrica.

1.4.1. Sustitución monoalfabeto

Un ejemplo de algoritmo de cifrado por sustitución monoalfabeto es el cifrado César. Es un tipo de cifrado por sustitución en el que cada letra en el texto original es reemplazada por otra letra que se encuentra un número

fijo de posiciones más adelante en el alfabeto. Por ejemplo, con un desplazamiento de 3, la A sería sustituida por la D (situada 3 lugares a la derecha de la A), la B sería reemplazada por la E, etc. Este método debe su nombre a Julio César, que lo usaba para comunicarse con sus generales.



Ejemplo:

Mensaje en claro:	**H O L A C E S A R**
Mensaje cifrado:	**K R O D G H V D U**

El descifrado de un mensaje consistiría en sustituir cada letra del texto por la que hay tres posiciones delante en el alfabeto.

La principal condición que debe cumplir la clave es que ha de ser una permutación del alfabeto, es decir, no puede haber letras repetidas ni faltar ninguna. Si no, la transformación no sería invertible en general.

1.4.2. Sustitución polialfabeto

El inconveniente de los algoritmos de sustitución monoalfabeto, es que el texto cifrado mantiene la misma distribución de frecuencia de caracteres que tiene el texto claro original, lo que hace que sean criptoanalizables por métodos estadísticos sencillos. Una posible mejora de los cifrados por sustitución es intentar métodos que destruyan esa correspondencia de frecuencias entre el mensaje en claro y el criptograma. Por ejemplo, utilizando varios alfabetos a la vez para el cifrado. En los cifrados

polialfabéticos la sustitución aplicada a cada carácter varía en función de la posición que ocupe este dentro del texto claro. En realidad corresponde a una aplicación cíclica de n cifrados de sustitución monoalfabeto. Un ejemplo típico de cifrado polialfabético es el Cifrado **de Vigenère**.

1.4.3. Cifrado en bloque

Un algoritmo de cifrado en bloque toma como entrada un bloque de longitud fija y una clave y genera un nuevo bloque cifrado de la misma longitud que el bloque de entrada.

La técnica consiste en dividir el texto a cifrar (con longitud *L*) en bloques de tamaño *b* y a continuación cifrar cada uno de los bloques. Si *L* no es múltiplo de *b*, se agregan bits adicionales para conseguir que todos los bloques estén completos. Para descifrar el mensaje se procede de manera análoga.

Muchos de los algoritmos de cifrado en bloque se basan en la combinación de dos operaciones básicas: sustitución y transposición.

- La **sustitución** consiste en traducir cada bloque de bits que llegan como entrada a otro de salida siguiendo una permutación determinada. El cifrado César sería un ejemplo simple de sustitución en el que cada grupo de bits correspondería a una letra.
- La **transposición** consiste en reordenar la información del texto en claro según un patrón determinado. Un ejemplo podría ser la formación de grupos de cinco letras, incluidos los espacios en blanco, y rescribir cada grupo (1, 2, 3, 4, 5) en el orden (3, 1, 5, 2, 4). Por ejemplo:

Texto en claro: “HOLA MUNDO”

Texto cifrado: “LH OANMOUD”

La transposición no dificulta extraordinariamente el criptoanálisis, pero puede combinarse con otras operaciones para añadir complejidad a los algoritmos de cifrado.

El **producto de cifras**, o combinación en cascada de distintas transformaciones criptográficas es una técnica muy efectiva para implementar algoritmos bastante seguros de forma sencilla. Por ejemplo, muchos algoritmos de cifrado en bloque se basan en una serie de iteraciones de productos sustitución-transposición.

Dos propiedades deseables en un algoritmo criptográfico son la **confusión y la difusión**. La confusión consiste en esconder la relación entre la clave y las propiedades estadísticas del texto cifrado. La difusión propaga la redundancia del texto en claro a lo largo del texto cifrado para que no sea fácilmente reconocible.

La confusión consigue que, cambiando un solo bit de la clave, cambien muchos bits del texto cifrado, y la difusión implica que el cambio de un solo bit del texto en claro afecte también a muchos bits del texto cifrado.

Modos de operación del cifrado en bloque

Un aspecto que hay que tener en cuenta cuando se utiliza el cifrado es que, aunque se puede conseguir que un atacante no descubra directamente los datos transmitidos, en ocasiones es posible que se pueda deducir información indirectamente. Por ejemplo, en un protocolo que utilice mensajes con una cabecera fija, la aparición de los mismos datos cifrados varias veces en una transmisión puede indicar dónde empiezan los mensajes. Para intentar contrarrestar esto, el cifrado bloque opera en varios modos:

- El modo **ECB** (Electronic Codeobook): consiste en dividir el texto en bloques y cifrar cada uno de ellos de forma separada. El

inconveniente de este método es que bloques idénticos de mensaje sin cifrar producirán idénticos textos cifrados.

- En el modo **CBC** (Cipher Block Chaining), antes de ser cifrado, a cada bloque de texto se le aplica una operación XOR bit a bit, con el previo bloque ya cifrado. De este modo, cada bloque es dependiente de todos los bloques de texto previos hasta ese punto. Además, para hacer cada mensaje único se puede usar un vector de inicialización. CBC es el modo usado más a menudo. Su principal contrapartida es que es secuencial y no puede funcionar en paralelo.
- En el modo **CFB** (*Cipher Feedback*), el algoritmo de cifrado no se aplica directamente al texto en claro sino a un vector auxiliar (inicialmente igual al IV). Del resultado del cifrado se toman *n* bits que se suman a *n* bits del texto en claro para obtener *n* bits de texto cifrado. Estos bits cifrados se utilizan también para actualizar el vector auxiliar. El número *n* de bits generados en cada iteración puede ser menor o igual que la longitud de bloque *b*. Tomando como ejemplo $n=8$, tenemos un cifrado que genera un byte cada vez sin que sea necesario esperar a tener un bloque entero para poderlo descifrar.
- El modo **OFB** (Output Feedback) opera como el CFB pero en lugar de actualizar el vector auxiliar con el texto cifrado, se actualiza con el resultado obtenido del algoritmo de cifrado. La propiedad que distingue este modo de los demás consiste en que un error en la recuperación de un bit cifrado afecta solamente al descifrado de este bit.

Ejemplos de algoritmos de cifrado en bloque: DES

DES[1] es uno de los algoritmos de cifrado más usados en el mundo. Fue publicado en 1977 en el documento **FIPS[2]** PUB 46 del Instituto Nacional de Estándares y Tecnología (NIST).

El algoritmo fue controvertido al principio, con algunos elementos de diseño clasificados, una longitud de clave relativamente corta, y las continuas sospechas sobre la existencia de alguna puerta trasera para la NSA[3]. Posteriormente DES fue sometido a un intenso análisis académico y motivó el concepto moderno del cifrado por bloques y su criptoanálisis.

Hoy en día, DES se considera inseguro para muchas aplicaciones. Esto se debe principalmente a que el tamaño de clave de 56 bits es corto. A finales de 2001 el algoritmo ha sido sustituido por el nuevo AES[4].

DES es el prototipo de algoritmo de cifrado por bloques: toma un texto en claro de una longitud fija de bits y lo transforma mediante una serie de operaciones básicas en otro texto cifrado de la misma longitud, dividiendo para ello el mensaje, en bloques de 64 bits. El algoritmo DES utiliza una clave criptográfica para modificar la transformación, de modo que el descifrado sólo puede ser realizado por aquellos que conozcan la clave concreta utilizada en el cifrado. La longitud de la clave es de 64 bits, aunque en realidad, sólo 56 de ellos son empleados por el algoritmo. Los ocho bits restantes se utilizan únicamente para comprobar la paridad, y después son descartados.

La parte central del algoritmo consiste en dividir el mensaje de entrada en grupos de bits, hacer una sustitución distinta sobre cada grupo y, a continuación una transposición de todos los bits. Esta transformación se repite dieciséis veces: en cada iteración, la entrada es una transposición

[1] Siglas en inglés de Data Encryption Standard.

[2] Siglas en inglés de *Federal Information Processing Standard*.

[3] Siglas en inglés de *National Scurity Agency*.

[4] Siglas en inglés de *Advanced Encryption Standard.*

distinta de los bits de la clave sumada bit a bit (XOR) con la salida de la iteración anterior. Este entrecruzamiento se conoce como esquema *Feistel*.

La estructura de *Feistel* asegura que el cifrado y el descifrado sean procesos muy similares. La única diferencia es que las subclaves se aplican en orden inverso cuando desciframos. Esto simplifica enormemente la implementación, en especial sobre hardware, al no haber necesidad de algoritmos distintos para el cifrado y el descifrado.

Ejemplos de algoritmos de cifrado en bloque: 3DES

Aunque a lo largo de los años el algoritmo DES se ha mostrado muy resistente al criptoanálisis, su principal problema es actualmente la vulnerabilidad a los ataques de fuerza bruta, a causa de la longitud de la clave, de sólo 56 bits. En los años 70 era muy costoso realizar una búsqueda entre las 2^{56} combinaciones posibles, pero la tecnología actual permite romper el algoritmo en un tiempo cada vez más corto. Por este motivo, en 1999 el NIST cambió el algoritmo DES por el "Triple DES" como estándar oficial, mientras no estuviera disponible el nuevo estándar AES. El Triple DES, como su nombre indica, consiste en aplicar el DES tres veces consecutivas. Esto se puede realizar con tres claves (k1, k2, k3), o bien con sólo dos (k1, k2, y otra vez k1). La longitud total de la clave con la segunda opción es de 112 bits (dos claves de 56 bits). La primera opción proporciona más seguridad, pero a costa de utilizar una clave total de 168 bits (3 claves de 56 bits), que puede ser un poco más difícil de gestionar e intercambiar. Para conseguir que el sistema sea adaptable al estándar antiguo, en el Triple DES se aplica una secuencia cifrado-descifrado-cifrado (E-D-E) en lugar de tres cifrados:

$$\mathbf{C = E(k_3, D(k_2, E(k_1, M)))}$$

o bien: $\mathbf{C = E(k_1, D(k_2, E(k_1, M)))}$

De este modo, para cifrar un mensaje M, primero se cifra con k_1, luego se descifra con k_2 y finalmente se vuelve a cifrar con k_3 (o k_1). Nótese que en el caso de usar 2 claves si hacemos que $k_1=k_2$ tenemos un sistema equivalente al DES simple.

Ejemplos de algoritmos de cifrado en bloque: AES

La longitud de la clave del algoritmo DES se ha ido convirtiendo en un problema a medida que iban aumentando las capacidades de procesamiento y el algoritmo se hací cada vez más vulnerable a un ataque por fuerza bruta.

En 1977, a la vista de que el Triple DES no es excesivamente eficiente cuando se implementa en software, el NIST convocó a la comunidad criptográfica a presentar propuestas para un nuevo estándar que sustituyera al DES. De los quince algoritmos candidatos que se aceptaron, se escogieron cinco como finalistas (MARS, RC6, RIJNDAEL, SERPENT y TWOFISH), y en octubre de 2000 se dio a conocer el ganador: el algoritmo Rijndael, propuesto por los criptógrafos belgas Joan Daemen y Vincent Rijmen. En noviembre de 2001 se publicó el documento FIPS 197 donde AES se asumía oficialmente.

El Rijndael puede trabajar en bloques de 128, 192 o 256 bits, y la longitud de la clave también puede ser de 128, 192 o 256 bits. Dependiendo de esta última longitud, el número de iteraciones del algoritmo es 10, 12 ó 14, respectivamente. Cada iteración incluye una sustitución fija byte a byte, una transposición, una transformación consistente en desplazamientos de bits y XORs, y una suma binaria (XOR) con bits obtenidos a partir de la clave.

1.4.4. Cifrado en flujo

Para algunas aplicaciones, tales como el cifrado de conversaciones telefónicas, el cifrado en bloques es inapropiada porque los flujos de datos se producen en tiempo real en pequeños fragmentos. Las muestras de datos pueden ser tan pequeñas como 8 bits o incluso de 1 bit, y sería un desperdicio rellenar el resto de los 64 bits antes de cifrar y transmitirlos.

El funcionamiento de un cifrado en flujo consiste en la combinación de un texto claro ***M*** con un texto de cifrado ***S*** que se obtiene a partir la clave simétrica ***k*** obteniendo un texto cifrado **C**. Para descifrar, sólo se requiere realizar la operación inversa con el texto cifrado **C** y el mismo texto de cifrado ***S*.**

La operación de combinación que se utiliza normalmente es la suma y como operación inversa la resta. Si el texto está formado por caracteres, el algoritmo sería como un cifrado César en que la clave va cambiando de un carácter a otro. La clave que corresponde viene dada por el texto de cifrado **S** (llamado *keystream* en inglés).

Considerando el texto formado por bits, la suma y la resta son equivalentes. Cuando se aplican bit a bit, ambas son idénticas a la operación lógica “o exclusiva”, denotada con el operador XOR (eXclusive OR). Así pues:

$$C = M \text{ XOR } S(k)$$
$$M = C \text{ XOR } S(k)$$

En los esquemas de cifrado en flujo, el texto claro ***M*** tiene una longitud variable y el texto de cifrado **S** ha de ser como mínimo igual de largo. No es necesario disponer del mensaje entero antes de empezar a cifrarlo o descifrarlo, ya que se puede implementar el algoritmo para que trabaje con un “flujo de datos” que se va generando a partir de la clave (el texto cifrado). De ahí procede el nombre de este tipo de algoritmos.

Existen varias formas de obtener el texto cifrado ***S*** en función de la clave ***k***:

- Si se escoge una secuencia ***k*** más corta que el mensaje ***M***, una posibilidad sería repetirla cíclicamente tantas veces como sea necesario para ir sumándola al texto en claro. El inconveniente de este método es que se puede romper fácilmente, sobre todo cuanto más corta sea la clave.

- En el otro extremo, se podría tomar directamente ***S(k) = k***. Esto quiere decir que la propia clave debe ser tan larga como el mensaje que hay que cifrar. Este es el principio del conocido **cifrado de Vernam**. Si ***k*** es una secuencia totalmente aleatoria que no se repite cíclicamente, estamos ante un ejemplo de cifrado incondicionalmente seguro. Este método de cifrado se llama en inglés *one-time-pad* ("cuaderno de un solo uso"). Un ejemplo de uso del cifrado de Vernam ocurre a veces entre los portaaviones y los aviones. En este caso, se aprovecha que en un instante dado (antes del despegue) tanto el avión como el portaaviones están en el mismo sitio, con lo cual, intercambiarse un disco duro de 20GB con una secuencia pseudoaleatoria no es ningún problema. Posteriormente cuando el avión despega puede establecerse una comunicación segura con el portaaviones utilizando un cifrado de Vernam con la clave aleatoria que ambos comparten.

- Lo que en la práctica se utiliza son funciones que generan **secuencias pseudoaleatorias** a partir de una **semilla** (un número que actúa como parámetro del generador), y lo que se intercambia como clave secreta ***k*** es solamente esta semilla. En cada paso el algoritmo se encontrará en un determinado estado, que vendrá dado por sus variables internas. Dado que las variables serán finitas, habrá un número máximo de posibles estados distintos. Esto significa que al cabo de un cierto período, los datos generados se volverán a repetir. Para que el algoritmo sea seguro, interesa que el período de

repetición sea cuanto más largo mejor (con relación al mensaje que hay que cifrar), con el fin de dificultar el criptoanálisis.

Las características de este tipo de cifrado lo hacen apropiado para entornos en los que se necesita un rendimiento alto y los recursos (capacidad de cálculo, consumo de energía) sean limitados. Por ello se suelen utilizar en comunicaciones móviles: redes sin hilos, telefonía móvil, etc.

Un ejemplo clásico de cifrado en flujo es **RC4** (Ron's Code 4). Fue diseñado por Ronald Rivest en 1987 y publicado en Internet por un remitente anónimo en 1994. Es conocido por ser el algoritmo de cifrado empleado en el sistema de seguridad **WEP** (*Wired Equivalent Privacy*) reconocido en el estándar **IEEE 802.11**. RC4 utiliza claves de 64 bits (40 bits más 24 bits del vector de iniciación IV) o de 128 bits (104 bits más 24 bits del IV).

1.4.5. Cifrado en base a funciones resumen

A veces los algoritmos de cifrado no sólo se usan para cifrar datos, sino que son utilizados para garantizar la autenticidad de los mismos. Como ejemplo de algoritmo de estas características se puede citar las llamadas **funciones *hash***, también conocidas como **funciones de resumen** de mensaje[5].

En general, podemos decir que una función resumen nos permite obtener una cadena de bits de longitud fija, relativamente corta, a partir de un mensaje de longitud arbitraria:

$$H = h(M)$$

Para mensajes ***M*** iguales, la función ***h*** debe dar resúmenes ***H*** iguales. Pero si dos mensajes dan el mismo resumen ***H*** no deben ser necesariamente iguales. Esto es así porque sólo existe un conjunto limitado de posibles valores ***H***, ya que su longitud es fija.

[5] *Message Digest*, en inglés.

Para que una función ***h*** se pueda aplicar en sistemas de autenticación, debe cumplir una serie de condiciones que le permitan ser considerada una **función *resumen* segura**. Entre ellas destacan la **unidireccionalidad y la resistencia a colisiones.**

Para dificultar los ataques contra las funciones de resumen, por un lado los algoritmos tienen que definir una relación compleja entre los bits de entrada y cada bit de salida. Por otro lado, los ataques por fuerza bruta se contrarrestan alargando lo suficiente la longitud del resumen.

Hasta hace poco, el algoritmo de resumen más usado era el ***MD5*** *(Message Digest 5)*. Pero como el resumen que obtiene es de sólo 128 bits, y aparte se han encontrado otras formas de generar colisiones parciales en el algoritmo, actualmente se recomienda utilizar algoritmos más seguros, como el ***SHA-1***[6]. El algoritmo ***SHA-1***, publicado el 1995 en un estándar del NIST (como revisión de un algoritmo anterior llamado simplemente SHA), obtiene resúmenes de 160 bits. El año 2002 el NIST publicó variantes de este algoritmo que generan resúmenes de 256, 384 y 512 bits.

1.5. Criptografía de clave pública o asimétrica

Los **sistemas de cifrado de clave pública** o **sistemas de cifrado asimétricos** se inventaron con el fin de evitar por completo el problema del intercambio de claves de los sistemas de cifrado simétricos.

En un algoritmo criptográfico de clave pública se utilizan claves distintas para el cifrado y el descifrado. Una de ellas, la **clave pública**, se puede obtener fácilmente a partir de la otra, la **clave privada**, pero por el contrario es prácticamente imposible. Los algoritmos de clave pública típicos permiten cifrar con la clave pública (k_{pub}) y descifrar con la clave privada (k_{pr}):

[6] Siglas en inglés de *Secure Hash Algorithm-1*

$$C = e(k_{pub}, M)$$

$$M = d(k_{pr}, C)$$

Pero también puede haber algoritmos que permitan cifrar con la clave privada y descifrar con la pública:

$$C = e(k_{pr}, M)$$

$$M = d(k_{pub}, C)$$

En la práctica, los algoritmos utilizados permiten cifrar y descifrar fácilmente, pero todos ellos **son considerablemente más lentos que los equivalentes con criptografía simétrica**. Por eso, la criptografía de clave pública se suele utilizar solo en los problemas que la criptografía simétrica no puede resolver: el intercambio de claves y la autenticación con no repudio (firmas digitales).

Los mecanismos de **intercambio de claves** permiten que dos partes se pongan de acuerdo en les claves simétricas que utilizaran para comunicarse, sin que un tercero que esté escuchando el diálogo pueda deducir cuales son estas claves.

La **autenticación** basada en clave pública se puede utilizar si el algoritmo permite utilizar las claves a la inversa: la clave privada para cifrar y la clave pública para descifrar. Si A envía un mensaje cifrado con su clave privada, todo el mundo podrá descifrarlo con la clave pública de *A*, y al mismo tiempo todo el mundo sabrá que el mensaje sólo lo puede haber generado quien conozca la clave privada asociada (que debería ser *A*). Ésta es la base de las **firmas digitales.**

1.5.1. Ejemplos de algoritmos de clave pública: Diffie-Hellman

Es un mecanismo que permite que dos partes se pongan de acuerdo de forma segura sobre una clave secreta utilizando un canal inseguro. El algoritmo se basa en la dificultad de calcular logaritmos discretos y se usa generalmente como medio para acordar claves simétricas que serán empleadas para el cifrado de una sesión.

1.5.2. Ejemplos de algoritmos de clave pública: RSA

Es el algoritmo más utilizado en la historia de la criptografía de clave pública. Su nombre procede de las iniciales de quienes lo diseñaron en 1977: Ronald Rivest, Adi Shamir y Leonard Adleman. La clave pública está formada por un número *n*, calculado como producto de dos factores primos muy grandes ($n = p * q$) y un exponente *e*. La clave privada es otro exponente *d* calculado a partir de *p*, *q* y *e*, de tal forma que el cifrado y el descifrado se puede realizar de la siguiente forma:

Cifrado: $\boldsymbol{C = M^e \bmod n}$

Descifrado: $\boldsymbol{M = C^d \bmod n}$

Como se puede ver, la clave pública y la privada son intercambiables: si se usa cualquiera de ellas para cifrar, se deberá utilizar la otra para descifrar. La fortaleza del algoritmo RSA se basa, por un lado, en la dificultad de obtener *M* a partir de *C* sin conocer *d* (problema del logaritmo discreto), y por otro lado, en la dificultad de obtener *p* y *q* (y, por tanto, *d*) a partir de *n* (problema de la factorización de números grandes, que es otro de los problemas considerados difíciles).

2. FIRMA DIGITAL

Una firma digital es, básicamente, un mensaje cifrado con la clave privada del firmante. Pero, por cuestiones de eficiencia, lo que se cifra no es

directamente el mensaje a firmar, sino solamente su resumen calculado con una función *resumen* segura.

La firma digital está basada en algoritmos criptográficos asimétricos, en los que son necesarias un par de claves para el intercambio de la información: una clave pública y una clave privada. La clave privada está bajo custodia del emisor y solamente es conocida por él. La clave pública es distribuida entre todos los posibles destinatarios de los mensajes o documentos firmados. El proceso para realizar una firma digital se resume a continuación:

- El emisor obtiene un resumen del mensaje a través de una función resumen (*Hash)*. La propiedad más importante de ese resumen o *hash* es que dos documentos diferentes siempre deben producir resúmenes diferentes.
- El resumen obtenido se cifra con la clave privada del firmante y se obtiene la firma digital del documento.
- El receptor del mensaje firmado utiliza la clave pública para descifrar la firma, obtiene el resumen del documento recibido y comprueba que es igual que el resumen que le ha llegado cifrado en la firma digital. De esta forma se garantiza que el contenido del mensaje no ha sido manipulado.

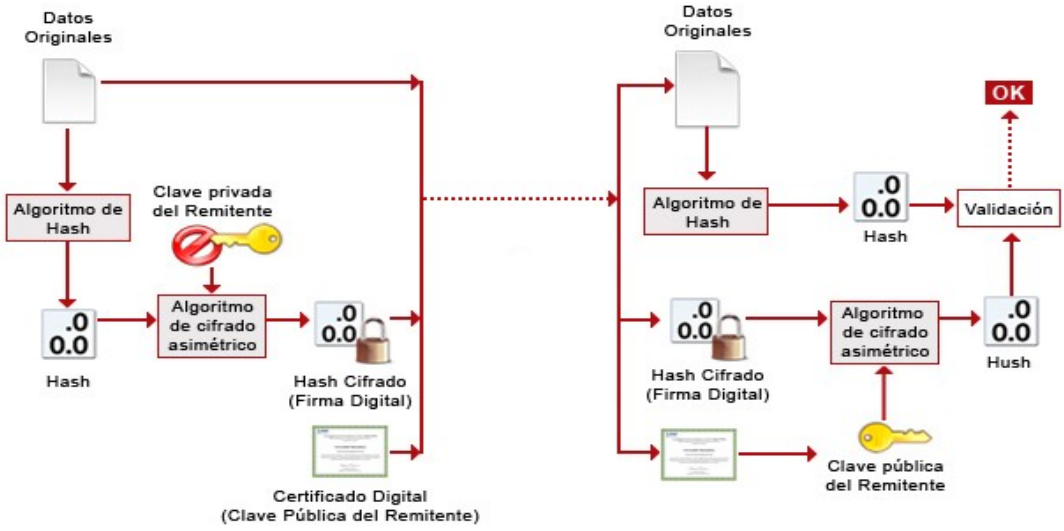


Figura 1: Proceso de creación de una firma digital (fuente: INTECO)

La firma digital, por si misma, no aporta confidencialidad al mensaje pero es habitual que los mensajes firmados electrónicamente se suelan enviar cifrados con la misma clave privada utilizada para mayor seguridad. La firma digital aporta:

- Identificación del firmante: la firma identifica al firmante de forma única igual que su firma manuscrita.
- Integridad del contenido firmado: es posible verificar que los documentos firmados no hayan sido alterados por terceras partes.
- No repudio del firmante: un documento firmado electrónicamente no puede repudiarse por parte de su firmante.

3. INFRAESTRUCTURA DE CLAVE PÚBLICA (PKI)

Como se ha visto hasta ahora, la criptografía de clave pública permite resolver el problema del intercambio de claves, utilizando las claves

públicas de los participantes. Pero se plantea otro problema: si alguien afirma ser *A* y su clave pública es k_{pub},

¿cómo podemos saber que realmente k_{pub} es la clave pública de A?

Porque es perfectamente posible que un atacante *Z* genere su par de claves (k'_{pr}, k'_{pub}) y afirme "yo soy *A*, y mi clave pública es k'_{pub}".

Una posible solución a este problema es que exista una entidad de confianza que nos asegure que, efectivamente, las claves públicas pertenecen a sus supuestos propietarios. Esta entidad puede firmar un documento que afirme "la clave pública de A es k_{pub}", y publicarlo para que todos los usuarios lo sepan. Este tipo de documento se llama **certificado de clave pública** o **certificado digital**, y es la base de lo que se conoce como **infraestructura de clave pública o PKI.**

Una **PKI** está formada entre otros por los siguientes elementos:

- **Certificados digitales**: Son documentos firmados electrónicamente por las autoridades de certificación que certifican que una clave pública pertenece a un determinado usuario.
- **Autoridades de certificación (AC)**: Son entidades de confianza que se encargan de emitir y revocar los certificados digitales.
- **Autoridades de registro (RA)**: Son entidades que registran las peticiones que hagan los usuarios para obtener un certificado, comprueban la veracidad y corrección de los datos que aportan los usuarios en dichas peticiones y las envían a una AC para que sean procesadas.
- **Autoridades de validación (VA):** suministran información sobre la vigencia de los certificados electrónicos que, a su vez, hayan sido registrados por una RA y certificados por la AC.

- **Autoridades de sellado de tiempos (TSA):** proporcionan certeza sobre la preexistencia de determinados documentos electrónicos en un momento dado, cuya indicación temporal junto con el hash del documento se firma por la Autoridad de sellado de tiempo.
- **Directorios de certificados**: proporcionan almacenamiento y distribución de certificados y listas de revocación (*CRL*s).
- **Hardware criptográfico (HSM):** dispositivos criptográficos basados en hardware que generan, almacenan y protegen claves criptográficas y suelen aportar aceleración hardware para operaciones criptográficas
- **Tarjetas criptográficas (TI)**: son tarjetas que incluyen un chip con un microprocesador con módulos hardware específicos para realizar operaciones criptográficas.

3.1. Certificados Digitales

Un certificado digital es un documento emitido y firmado electrónicamente por una autoridad certificadora en el que certifica la asociación entre una clave pública y un participante.

El certificado garantiza que la clave pública pertenece al participante identificado y que el participante posee la correspondiente clave privada.

Los certificados digitales sólo son útiles si existe una *Autoridad de Certificación* (*CA*), de confianza para las dos partes, que los valide

Los certificados digitales proporcionan un **mecanismo criptográfico** para implementar la **autenticación**. También proporcionan un mecanismo seguro y escalable para **distribuir claves públicas** en comunidades con gran número de participantes.

El formato de los certificados *X.509* es una recomendación del *ITU*[7] que se publicó por primera vez en 1988. La revisión actual del estándar fue publicada en 1996 y se conoce con el nombre de *X.509 v3*. Los elementos que componen un certificado *X.509 v3* son:

- **Versión**. Es el número de versión del certificado codificado. Los valores aceptables son 1, 2 y 3.
- **Número de serie del certificado**. Es un entero asignado por la autoridad certificadora. Cada certificado emitido por una CA debe tener un número de serie único.
- **Identificador del algoritmo de firmado**. Especifica el algoritmo empleado para firmar el certificado (ej: *sha1withRSAEncryption*).
- **Nombre del emisor**. identifica la CA que ha firmado y emitido el certificado.
- **Periodo de validez**. Es el periodo de tiempo durante el cual el certificado es válido y la CA está obligada a mantener información sobre el estado del mismo.
- **Nombre del sujeto**. Identifica al sujeto cuya clave pública está certificada en el campo siguiente. El nombre debe ser único para cada entidad certificada por una CA dada, aunque puede emitir más de un certificado con el mismo nombre si es para la misma entidad.
- **Información de clave pública del sujeto**. Almacena la clave pública, sus parámetros y el identificador del algoritmo con el que se emplea la clave.
- **Identificador único del emisor**. Este es un campo opcional que permite reutilizar nombres de emisor.
- **Identificador único del sujeto**. Este es un campo opcional que permite reutilizar nombres de sujeto.
- **Extensiones**: Las extensiones del *X.509 v3* proporcionan una manera de asociar información adicional a sujetos, claves públicas, etc.

[7]Siglas en inglés de *International Telecommunication Union.*

- **Firma de la *AC***: En este campo se almacena la firma digital del certificado por parte de la *AC*.

Los certificados digitales se diferencian **según la finalidad** para la que son solicitados. Así podemos tener certificados para **personas físicas**, certificados de **servidor**, certificados para la **firma de código**, certificados de **entidad**, etc.

3.1.1. Autoridad de Certificación

Una Autoridad de Certificación, es una entidad de confianza, encargada de emitir y revocar los certificados digitales que garantizan de forma unívoca y segura la identidad asociada a una clave pública.

La Autoridad de Certificación, por sí misma o por mediación de una Autoridad de Registro, verifica la identidad del solicitante de un certificado antes de su expedición o, en caso de certificados expedidos con la condición de revocados, elimina la revocación de los certificados al comprobar dicha identidad.

Los certificados son documentos que recogen ciertos datos de su titular y su clave pública y están firmados electrónicamente por la Autoridad de Certificación utilizando su clave privada.

La Autoridad de Certificación es un tipo particular de Prestador de Servicios de Certificación que legitima ante los terceros que confían en sus certificados la relación entre la identidad de un usuario y su clave pública. La confianza de los usuarios en la CA es importante para el funcionamiento del servicio y justifica la filosofía de su empleo, pero no existe un procedimiento normalizado para demostrar que una CA merece dicha confianza.

La autoridad de certificación se encarga de renovar los certificados, proporcionar servicios de backup y archivo de claves de cifrado. También

crea la infraestructura de seguridad para la confianza de los participantes, establece políticas de operación segura y genera información de auditoría.

El mecanismo habitual **de solicitud de un certificado** de servidor web a una CA consiste en que la entidad solicitante, utilizando ciertas funciones del software de servidor web, completa ciertos datos identificativos (entre los que se incluye el localizador URL del servidor) y genera una pareja de claves pública/privada. Con esa información el software de servidor compone un fichero que contiene una petición CSR[8] en formato *PKCS#10* que contiene la clave pública y que se hace llegar a la CA elegida. Esta, tras verificar por sí o mediante los servicios de una Autoridad de Registro la información de identificación aportada y la realización del pago, envía el certificado firmado al solicitante, que lo instala en el servidor web con la misma herramienta con la que generó la petición CSR.

Las CA disponen de sus propios certificados públicos, cuyas claves privadas asociadas son empleadas por las CA para firmar los certificados que emiten. Un certificado de CA estará firmado por otra CA de rango superior estableciéndose así una jerarquía de certificación.

Existen **certificados de CA raíz** que están auto-firmados por la propia CA que los emite y que constituyen el elemento inicial de la jerarquía de certificación.

Una **jerarquía de certificación** consiste en una estructura jerárquica de CAs en la que se parte de una CA auto-firmada, y en cada nivel, existe una o más CAs que pueden firmar certificados de entidad final (servidor web, persona, aplicación de software) o bien certificados de otras CA subordinadas plenamente identificadas y cuya Política de Certificación sea compatible con las CAs de rango superior.

Una de las formas por las que se **establece la confianza en una CA** por parte de un usuario, consiste en la "instalación" en el ordenador del usuario

[8] *Certificate Signing Request*

(tercero que confía), del certificado autofirmado de la CA raíz de la jerarquía en la que se desea confiar.

Cuando el modelo de CA incluye una **jerarquía**, es preciso **establecer explícitamente la confianza en los certificados de todas las cadenas de certificación** en las que se confíe. Para ello, se puede localizar sus certificados mediante distintos medios de publicación en internet, pero también es posible que un certificado contenga toda la cadena de certificación necesaria para ser instalado con confianza.

Un **certificado revocado** es un certificado que no es válido aunque se emplee dentro de su período de vigencia. Un certificado revocado tiene la condición de suspendido si su vigencia puede restablecerse en determinadas condiciones.

Es necesario establecer un mecanismo que permita revocar un certificado antes de que este caduque para los casos de sustracción, errores, cambios de derechos, ruptura de la CA, etc.

Para comprobar si un certificado está revocado generalmente se utilizan las ***CRL* (*Certificate Revocation List*)**. De este modo, cuando se quiere verificar la firma de un documento, el usuario no sólo ha de verificar el certificado y su validez, sino que también ha de comprobar que el certificado no ha sido revocado consultando para ello la versión más reciente de la *CRL* .

Con las *CRL* se opera siguiendo dos modelos:

- **Modelo pull:** el cliente que tiene que hacer la verificación obtiene la CRL de la CA cuando lo necesita.
- **Modelo push:** una vez que la CA actualiza la CRL, la información es enviada a los clientes que necesitan verificar certificados.

Otro método alternativo de comprobación es el *protocolo de estado de certificado en línea* ***OCSP***[9]. Este método permite a los clientes desprenderse de la gestión del estado de los certificados y obtener una confirmación online del estado. Para ello la *CA* debe poner a disposición de todos los usuarios potenciales un servicio seguro online de alta disponibilidad. Este protocolo está definido por el IETF en el RFC 2560.
Los mensajes *OCSP* se codifican en *ASN.1* y habitualmente se transmiten sobre el protocolo *HTTP*. La naturaleza de las peticiones y respuestas de *OCSP* hace que a los servidores *OCSP* se les conozca como "*OCSP* ***responders***". Las *CA*s delegan la responsabilidad de proporcionar información de revocaciones en los *responders* creando así una arquitectura distribuida. Los **clientes envían una petición de estado** a un *responder* y suspende su aceptación hasta recibir la respuesta. Este modo de funcionamiento evita el uso de *CRLs*, reduciendo así el ancho de banda consumido, el uso de CPU y se evitan los problemas asociados a la gestión de información sensible que contienen las *CRLs.*

3.1.2. Autoridad de Registro

La Autoridad de Registro **gestiona el registro de usuarios y sus peticiones de certificación/revocación**, así como los certificados respuesta a dichas peticiones. Indica a la *CA* si debe emitir un certificado. La Autoridad de Registro es la que autoriza la asociación entre una clave pública y el titular de un certificado. Durante el ciclo de vida de un certificado, la Autoridad de Registro, es la que se encarga de las siguientes operaciones:

- Revocación.
- Expiración.

[9] Siglas en inglés de *Online Certificate Status Protocol*.

- Renovación (extensión del período de validez del certificado, respetando el plan de claves).
- Reemisión del par de claves del usuario.
- Actualización de datos del certificado.

3.1.3. Autoridad de Validación

La Autoridad de Validación **suministra información de forma online acerca del estado de un certificado**. La Autoridad de Validación suele proporcionar dos servicios de validación: el tradicional, permitiendo la descarga de ***CRLs*** para que el usuario las interprete él mismo, o a través del protocolo ***OCSP*** (*Online Certification Status Protocol*).

Los usuarios y las aplicaciones que deseen obtener el estado de un certificado, sólo tienen que realizar una petición *OCSP* contra la Autoridad de Validación para obtener dicho estado. La *CA* actualiza la información de la Autoridad de Validación cada vez que se modifica el estado de un certificado, con lo que, usando *OCSP*, se dispone de información en tiempo real.

3.1.4. Autoridad de Sellado de Tiempos

La Autoridad de Sellado de Tiempos (*TSA*) permite **firmar documentos con sellos de tiempo**, de manera que permite obtener una prueba de que un determinado dato existía en una fecha concreta. El sello de tiempo es uno de los servicios más importantes de la firma electrónica. Con el sello de tiempo se puede demostrar que una serie de datos han existido y no han sido alterados desde un instante específico en el tiempo. Este protocolo se describe en el RFC 3161 y está en el registro de estándares de Internet. Una autoridad de sellado de tiempo actúa como tercera parte de confianza testificando la existencia de dichos datos electrónicos en una fecha y hora

concretas. Los pasos que se siguen para generar un sello de tiempo son los siguientes:

- Un usuario quiere obtener un sello de tiempo para un documento electrónico que él posee.
- Un resumen digital (técnicamente un *hash*) se genera para el documento en el ordenador del usuario.
- Este resumen forma la solicitud que se envía a la autoridad de sellado de tiempo (*TSA*).
- La *TSA* genera un sello de tiempo con esta huella, la fecha y hora obtenida de una fuente fiable y la firma electrónica de la *TSA*.
- El sello de tiempo se envía de vuelta al usuario.
- La *TSA* mantiene un registro de los sellos emitidos para su futura verificación.

Las aplicaciones del sellado de tiempo son innumerables, ya que los certificados digitales se emiten con un período de validez determinado y es fundamental por ejemplo, poder verificar que una firma de un documento realizada hace *X* años atrás efectivamente se hizo con un certificado que no estaba revocado en ese instante. Ejemplos de uso: factura electrónica, voto electrónico, protección de la propiedad intelectual, etc.

3.1.5. Directorio de Certificados

Los directorios proporcionan almacenamiento y distribución de certificados y listas de revocación (*CRL*s). Cuando una Autoridad de Certificación emite un certificado o *CRL*, lo envía al Directorio y además, guarda el certificado o *CRL* en su base de datos local. Generalmente se utiliza *LDAP* (*Light-weight Directory Access Protocol*) para acceder a los directorios. El usuario puede obtener certificados de otros usuarios y comprobar el estado de los mismos.

3.2. Hardware Criptográfico

Los **Módulos de Seguridad Hardware** (*HSM*) son dispositivos especializados en realizar labores criptográficas. Proporcionan almacenamiento seguro de claves y/o realización de funciones criptográficas básicas como cifrado, firma, generación de claves, etc. Para ello usan interfaces estándar como *PKCS#11* y *CryptoAPI*. Este tipo de dispositivos aumentan significativamente la seguridad en comparación con los certificados basados en disco por lo siguiente:

- La clave privada y las firmas digitales se generan dentro del HSM.
- La clave privada se almacena cifrada dentro del HSM.

Si se compara el hardware criptográfico con las tecnologías de cifrado basado en software se puede decir que el hardware criptográfico es mucho más rápido a la hora de realizar el proceso. Dependiendo del tipo de hardware *HSM*, los ratios de trabajo oscilan de las 600 a 4000 operaciones de firma RSA/segundo. Además proporcionan seguridad física al no poder modificar los algoritmos de cifrado y limitando el acceso al almacenamiento seguro de claves. Esto permite que estas soluciones puedan ser certificadas por un tercero en jerarquías de certificación.

3.3. Tarjetas y chip criptográficos

Una tarjeta inteligente (***smart card***), o tarjeta con circuito integrado (*TCI*), es cualquier tarjeta de tipo bolsillo con circuitos integrados que permiten la ejecución de cierta lógica programada. Aunque existe un diverso rango de aplicaciones, hay varias categorías de *TCI*: **Las tarjetas de memoria** contienen sólo componentes de memoria no volátil y posiblemente alguna lógica de seguridad. **Las tarjetas microprocesadoras** contienen memoria y tienen capacidad de procesamiento limitada. Las **tarjetas con chip criptográfico** son tarjetas microprocesadas avanzadas en las que

hay módulos hardware para la ejecución de algoritmos usados en cifrado y firmas digitales.

Una **tarjeta inteligente con un chip criptográfico** se puede definir como una llave muy segura, no duplicable e inviolable, que contiene las claves y certificados necesarios para la firma electrónica grabados en la tarjeta y que además está protegida por un PIN secreto y/o biometría. El **chip criptográfico** contiene un microprocesador que realiza las operaciones criptográficas con la clave privada, con la característica adicional de que no es posible el acceso a la clave desde el exterior. Las características principales son las siguientes:

- Doble seguridad: posesión de la tarjeta y PIN de acceso (o mecanismos biométricos).
- Puede ser multipropósito: Tarjeta de identificación gráfica, tarjeta de control de acceso/horario mediante banda magnética o chip de radiofrecuencia, tarjeta monedero, tarjeta generadora de contraseñas de un solo uso (*OTP*).
- Se precisa de un middleware (*CSP*) específico para utilizar la tarjeta, así como de un lector (*USB*, integrado en teclado o *PCMCIA*)
- El número de certificados que se pueden cargar depende del perfil de certificado, de la capacidad del chip y del espacio que se reserve para los certificados.

3.4. 3.3. Marco legal y estándares

Legislación Española

- **Ley 59/2003**, de 19 de diciembre, de firma electrónica (BOE nº 304, 20/12/2003)

Directiva Europea

- **Directiva 1999/93/CE del parlamento europeo** y del consejo de 13 de diciembre de 1999 por la que se establece un marco comunitario para la firma electrónica

Estándares europeos

- ETSI TS 102 042: Policy requirements for certification authorities issuing public key certificates
- ETSI TS 102 023: Policy requirements for time-stamping authorities
- ETSI TS 101 862: Qualified Certificate profile
- ETSI TS 101 456: Policy requirements for certification authorities issuing qualified certificates
- CWA 14167-2 Security Req. for Trustworthy Systems Managing Certificates for Electronic Signatures
- CWA 14172 EESSI Conformity Assessment Guidance (Guía para aplicar los estándares de firma electrónica de acuerdo con la iniciativa de estandarización europea)

Internet Engineering Task Force (IETF) – Request For Comment

- RFC 3280: Certificate and Certificate Revocation List (CRL) Profile
- RFC 3739: Qualified Certificates Profile
- RFC 3647: Certificate Policy and Certification Practices Framework (Obsoletes RFC2527)

PKCS (Public Key Cryptography Standards): Familia de estándares para los sistemas de criptografía de clave pública definidos por los Laboratorios RSA:

- PKCS#1,#2,#4: RSA Cryptography Standard
- PKCS#3: Diffie-Hellman Key Agreement Standard
- PCKS#5: Password-Based Encryption Standard
- PKCS#6: Extended-Certificate Syntax Standard
- PKCS#7: Cryptographic Message Syntax Standard
- PKCS#8: Private Key Information Syntax Standard
- PKCS#9: Selected Attribute Types

- PKCS#10: Certification Request Standard
- PKCS#11: Cryptographic Token Interface
- PKCS#12: Personal Information Exchange Syntax Standard
- PKCS#13: Elliptic Curve Cryptography Standard

4. REFERENCIAS

- Universidad de Vigo. Asignatura Seguridad en sistemas de información. (http://ccia.ei.uvigo.es/docencia/SSI/)

- Centro Criptológico Nacional. (https://www.ccn.es/)

- Instituto Nacional de Tecnologías de la Comunicación - DNI Electrónico. (http://cert.inteco.es)

- Universidad Politécnica de Madrid - Departamento de Matemática Aplicada de la Facultad de Informática. (http://www.dma.fi.upm.es/java/matematicadiscreta/aritmeticamodular/)

- Universidad Pontificia Comillas (Madrid) - Asignatura de Seguridad Informática. (http://www.iit.upcomillas.es/seguridad)

Autor: Juan Otero Pombo

Ingeniero en Informática en el Concello de Ourense

Colegiado del CPEIG

(Todos los enlaces fueron verificados en noviembre de 2011)