UNIVERSIDAD POLITÉCNICA DE CARTAGENA

INGENIERÍA DE TELECOMUNICACIONES.

PROYECTO FIN DE CARRERA.

PORTAL DE CERTIFICADOS DE SEGURIDAD BASADO EN OPENWRT.

ALUMNO: Fuensanta Blanco Pascual.

DIRECTOR: Francesc Burrull i Mestres.

MARZO/2015.

ÍNDICE.

CAPÍTULO 1.
Presentación del proyecto.

1.1. Introducción.

En la actualidad, desde la aparición, y más aún desde la globalización de Internet, la **seguridad** en los sistemas de información se ha convertido en uno de los problemas más importantes al que debemos hacer frente.

Por ello, se ha tenido la necesidad de crear políticas de seguridad consistentes en realizar conexiones seguras, enviar y recibir información **codificada**, filtrar accesos e información, etc.

Además, el aumento del uso de Internet ha dirigido la atención del mundo entero a un problema crucial: la **privacidad**. Hasta hace relativamente poco, no existía una protección real que garantizara que los mensajes que se enviaban o recibían no fueran interceptados, leídos o incluso alterados por algún desconocido, ya que nadie en realidad dirige o controla la red Internet.

Es por ello que, para que la **privacidad** y **seguridad** sean unos aspectos importantes a tener en cuenta en el uso de Internet, cada una de las entidades necesita contar con una manera de verificar la **identidad** de la otra y establecer un nivel de confianza.

Y es aquí donde entra nuestro objeto de estudio: las **autoridades de certificación**. Éstas son unas entidades de confianza, responsables de emitir y revocar los certificados digitales, utilizados en la firma electrónica, para lo cual se emplea la criptografía de clave pública. Es por ello que este proyecto se basa en la creación de una **Autoridad de Certificación** para ser usada en las Administraciones Públicas, en concreto para la **UPCT**. Dicha Autoridad residirá en una máquina autónoma por razones de seguridad y permitirá la gestión de certificados como se explicará más adelante.

1.2. Motivación.

El principal motivo que impulsa el desarrollo de este proyecto es la creación de un software creado exclusivamente para la **UPCT** para la gestión de **certificados personales** de sus empleados, dado que actualmente esta entidad depende de la **Autoridad Certificadora de la Generalitat Valenciana** para conseguir sus certificados. Por lo que este software va a permitir la creación de la **Autoridad de Certificación** propia de la **UPCT**, y que ésta pueda ser usada para la solicitud, expedición y revocación de **certificados digitales**.

Este software se crea por tanto, para que el personal administrativo (en este caso del departamento de **certificación**) pueda crear su propia **Autoridad de Certificación** de manera **rápida** y **sencilla** y que así los empleados de esta entidad (en este caso personal de la UPCT) puedan obtener sus propios **certificados digitales personales** sin necesidad de depender de otras autoridades de mayor rango.

Además este software es relativamente **novedoso** y ofrece las posibilidades anteriormente mencionadas que se explicarán más detalladamente a lo largo de esta memoria.

1.3. Objetivos.

Como ya se ha mencionado el objetivo final de este proyecto es el desarrollo de un **software** que permita crear de manera **sencilla** una **Autoridad Certificadora interna para la UPCT** para que cada empleado de dicha institución que requiera obtener su propio certificado personale, pueda conseguirlo sin necesidad de que entre en juego una **Autoridad Certificadora confiable de terceros**.

La creación de este proyecto se ve impulsado por la necesidad de que surja una **Autoridad Certificadora** que permita **acreditar** a los **trabajadores** de la **UPCT** de manera independiente a la Autoridad Certificadora de la Generalitat Valenciana (la que se emplea actualmente), evitando así el trabajo que ello conlleva y creando de este modo un método más sencillo de acreditar a los empleados de la **UPCT**.

La **Autoridad Certificadora** que aquí se desarrolla residirá, por el momento, en una máquina autónoma por razones de seguridad. La plataforma de trabajo consiste en un sistema operativo Linux, en concreto la distribución **OpenWRT**. Dicha plataforma se instalará en un **router TP-Link TL-WR-1043ND**. De este modo se consigue **bajo coste**, **facilidad de instalación** y **acceso** a través de **red convencional e inalámbrica**. Una vez que se haya instalado todo se desarrollará un software en **php** para la gestión de certificados, basado en **php-ca**.

Por lo tanto, como ya he mencionado, el objetivo final de este proyecto es la creación de una **Autoridad Certificadora para la UPCT**, para lo cual se fijan una serie de objetivos secundarios aunque necesarios para el desarrollo del proyecto:

- Búsqueda y estudio para adquirir conocimientos acerca del ámbito de **seguridad de redes**, en concreto **certificados digitales** y **autoridades de certificación**.
- Estudio del sistema operativo **OpenWrt** para conocer las oportunidades que nos ofrece y posterior instalación en el router empleado para el desarrollo del proyecto.
- Estudio del software utilizado como base para la creación de este proyecto, así como el lenguaje empleado para el desarrollo del mismo **php,** empleando las herramientas del mismo para crear una **Autoridad Certificadora**, es por eso que lo designaremos como **php-ca**. Y posterior instalación de éste para poder así comenzar a trabajar sobre él.

Finalmente, para la aplicación que se va a proceder a desarrollar en este proyecto serán necesarios principalmente, entre otros, los siguientes elementos:

- Router TP-Link TL-WR-1043ND.
- Sistema Operativo OpenWrt.
- Software basado en php-ca.

Con todos estos objetivos fijados y llevados a cabo comprobaremos finalmente que el resultado obtenido es el software para la creación de la **Autoridad Certificadora para la UPCT**.

1.4. Recursos utilizados.

Emplearemos una serie de recursos para llevar a cabo el proyecto, tanto hardware como software algunos de los cuales detallaremos más en profundidad en capítulos posteriores.

1.4.1. Hardware.

- **PC portátil.** Necesario para el desarrollo y las pruebas del software creado.
- **Router TP-Link TL-WR-1043ND.** Máquina en la que se instalará el software creado por razones de seguridad.

1.4.2. Software.

- **Sistema Operativo Windows 7 (PC portátil).** Se ha empleado este sistema operativo para el desarrollo del proyecto aunque para facilitar ciertas acciones hubiera sido preferible emplear **SO Linux.**
- **Sistema Operativo OpenWrt (router).** Instalado en el router para soportar el software creado. En capítulos posteriores se explicará con más detalle su funcionamiento y utilidades.
- **Software basado en php-ca.** Empleado para la creación del software que aquí se plantea.

1.5. Distribución de la memoria.

En este subapartado se va a detallar cómo está dividida la memoria y cuáles son los aspectos más significativos que se van a detallar en cada capítulo, para tener de esta manera una visión global de lo que se explica en el presente documento.

La memoria está dividida en **6 capítulos** y dos **anexos**. A continuación se va a detallar lo que contiene cada uno de ellos.

- En el **primer capítulo**, como se ha visto se da una breve introducción de las bases del proyecto y lo que ha llevado a desarrollar el mismo, así como los objetivos fijados.
- En el **capítulo 2** se realiza un estudio sobre la **seguridad en redes** y más concretamente sobre lo que en este proyecto más nos interesa como son las **firmas digitales**, los **certificados digitales** y las **autoridades de certificación**. Se hablará de los ámbitos relacionados con la seguridad así como la necesidad de un buen grado de seguridad en lo que a Internet de refiere.
- En el **tercer capítulo** se explicará en detalle el software utilizado, el sistema operativo **OpenWrt** instalado en el router, así como el lenguaje empleado para el desarrollo de la aplicación. Además se dará una breve explicación de las funcionalidades ofrecidas por el router empleado (**TP-Link**).
- En el **capítulo 4** se explica en detalle el **software creado**, así como la **aplicación de usuario,** detallando cada una de las partes desarrolladas.
- En el **capítulo 5** se detallan las **conclusiones** y los **trabajos futuros**.
- En el **capítulo 6** la **biografía** usada.
- Y finalmente en el **anexo 1** se explicará cómo se ha instalado el **SO OpenWrt** en el router y como se ha puesto en marcha el software creado. Y en el **anexo 2** los diferentes formatos y extensiones de un certificado digital.

CAPÍTULO 2.
Marco teórico.

En este segundo capítulo se ofrecerá un análisis de la base teórica que ha impulsado el desarrollo de este proyecto. Se comenzará explicando **qué es la seguridad en redes** y la importancia de este ámbito y profundizaremos más en los **certificados digitales** y **autoridades de certificación**.

Todo lo que aquí se explica es necesario conocerlo tanto para el desarrollo de la aplicación como para entender cada uno de los objetivos fijados anteriormente y por tanto llegar a comprender el cómo y el porqué del desarrollo del proyecto aquí presentado.

Se ofrece por tanto, en este apartado, una introducción a alguno de los componentes que posteriormente trataremos y explicaremos más en detalle como es el caso de los certificados.

2.1. Introducción a la seguridad en redes.

2.1.1. ¿Qué es seguridad?

Seguridad es una característica de cualquier sistema, telemático o no, que indica si ese sistema está libre de todo peligro, daño o riesgo, y que es en cierta manera infalible. Por tanto, un sistema se considera seguro cuando ofrece **fiabilidad** (probabilidad de que un sistema se comporte tal y como se espera de él).

Para que un sistema sea seguro debe cumplir las siguientes **características:**

- **Confidencialidad.** Que el sistema sea accesible por aquellos elementos que estén autorizados para ello. Se obtiene mediante algoritmos de cifrado (**criptografía**).
- **Integridad.** La información sólo puede ser creada, modificada o destruida por los elementos del sistema autorizados para ello.
- **Disponibilidad.** Los componentes del sistema deben estar disponibles para aquellos elementos autorizados para usarlos (detectores de intrusismo).
- **Autenticación.** Garantiza que eres quien dice ser.
- **No repudio.** Consiste en no poder negar la transmisión de un mensaje por un emisor y la recepción del mismo por un receptor.
- **Control de acceso.** Funciones que se pueden hacer en el sistema una vez dentro.

Además un sistema seguro debe proteger el software, hardware y datos de

- **Vulnerabilidad.** Existencia de un punto débil de la red de comunicaciones o de sus equipos.
- **Amenaza.** Cualquier circunstancia o evento que potencialmente pude causar un daño a una organización mediante la exposición, modificación o destrucción de información o mediante la denegación de servicios críticos.
- **Ataque.** Poner en práctica una amenaza aprovechando las vulnerabilidades del sistema o red de comunicaciones.

Así un sistema que sea seguro debe evitar la posibilidad de que ocurra un ataque (**riesgo**).

2.1.1.1. Tipos de amenazas.

- **Interrupción.** Un objeto del sistema se pierde o queda no disponible, existe una amenaza a la **disponibilidad.**
- **Interceptación.** Cuando un elemento no autorizado consigue acceder a un objeto del sistema, ocurre una amenaza a la **confidencialidad.**
- **Modificación.** Además de conseguir acceso a un elemento no autorizado del sistema, lo modifica. Amenaza a la **integridad.**
- **Generación.** Cuando un elemento no autorizado consigue crear un objeto falso pero idéntico y lo introduce en el sistema. Estamos aquí ante una amenaza a la **integridad.**

2.1.1.2. Tipos de ataques.

- **Ataques pasivos.** En los ataques pasivos el atacante no altera la comunicación, sino que únicamente la escucha o monitoriza, para obtener información que está siendo transmitida. Sus objetivos son la intercepción de datos y el análisis de tráfico, una técnica más sutil para obtener información de la comunicación. Por tanto estos ataques se quedan en amenaza puesto que no hacen ningún daño aunque pueden suponer un problema.
- **Ataques activos.** Con estos ataques se producen daños a la red:
  - **Suplantación** (intercepción).
  - **Réplica** (generación).
  - **Alteración** (modificación).
  - **Denegación de servicio** (interrupción).

2.1.1.3. Procedencia de las amenazas.

- **Personas** (80% de las amenazas).
  - **Piratas.**
  - **Ingeniería social, shoulder surfing o basureo.** Consiste en aprovecharse de los escasos conocimientos que tienen los usuarios de internet o de la reducida protección que tiene la gente de su información personal (claves personales).
- **Programas o amenazas lógicas.** Existen multitud de programas o herramientas creadas para debilitar cualquier sistema y obtener así la información que se necesita, por ejemplo las **herramientas de seguridad** empleadas para analizar la red conllevan beneficios tanto para el que protege como para el que ataca o los **caballos de troya** los cuales consisten en líneas de código que están dentro de programas (que funcionan correctamente) y que de manera transparente al usuario intentan acceder a sus datos (por ejemplo datos bancarios).
- **Catástrofes naturales.**

2.1.1.4. Métodos de defensa.

Es decir, cómo se protegen los sistemas de sufrir amenazas y ataques.

- **Política de seguridad (gestión).** En primer lugar el sistema debe tener una gestión de la seguridad adecuada, la cual se define como el conjunto de requisitos definidos por

los responsables del sistema indicando qué está permitido (**política permitiva**) y qué no está permitido (**política prohibitiva**). Así se deberá producir un análisis de las amenazas, las pérdidas que éstas podrían originar y la probabilidad de ocurrencia de dichas amenazas, es decir la probabilidad de que las amenazas se puedan convertir en ataques. Por tanto, que una empresa tenga una correcta **política de seguridad** consiste en definir los objetivos de seguridad que ésta tendría que llevar a cabo, es decir definir **responsabilidades** y **reglas**.

- **Mecanismos de seguridad.** Del mismo modo existen métodos que se deberán ejecutar tanto para **prevenir** ataques (autenticación/identificación, control de acceso, cifrado, etc…), como para **detectarlos** (firewalls, detectores de intrusismos, etc…), así como para **recuperar** los daños provocados, los cuales se ejecutan cuando se ha producido el ataque (por ejemplo análisis forense, una vez que se ha producido un ataque ver cómo se ha hecho y quién lo ha hecho).
- **Protección del hardware.** Del mismo modo se deberá proteger el hardware de:
  - Un **acceso físico** previniendo dichos ataques a través de la autenticación, protegiendo el cableado o tarjetas electrónicas de acceso y detectándolos con cámaras de vigilancia o alarmas.
  - **Desastres naturales.** De los cuales se previene de una posible ocurrencia, por ejemplo, situando los equipos en superficies bajas (terremotos).
  - **Desastres del entorno.** Para prevenir éstos, por ejemplo ante un problema con la red eléctrica, se hace necesario el uso de SAI o apagar los equipos ante riesgo de una tormenta eléctrica.
- **Protección de datos.** Ante una posible **interceptación** se prevendrá usando segmentos de red que no sean de fácil acceso, aplicaciones de cifrado, etc… y ante **pérdidas de información** creando copias de seguridad o respaldo.

2.1.2. Criptografía.

Para poder entender mejor cómo funciona de manera interna lo que en este proyecto se desarrolla es necesario tener unos conocimientos básicos de lo que es la **criptografía** y de los diferentes algoritmos de cifrado que existen, y es lo que se va a proporcionar en este subapartado.

Se define la **criptografía** como el arte de escribir con clave secreta o de un modo enigmático. Aportando una visión más específica, la **criptografía** es la creación de técnicas para el cifrado de datos, teniendo como objetivo conseguir la confidencialidad de los mensajes. Si la **criptografía** es la creación de mecanismos para cifrar datos, el **criptoanálisis** son los métodos para "romper" estos mecanismos y obtener la información. Una vez que nuestros datos han pasado un **proceso criptográfico** decimos que la **información** se encuentra **cifrada**.

La **criptografía** cuenta con tres usos principales: **cifrar**, **autenticar** y **firmar.**

- **Cifrar:** siempre hay cierta información que no queremos que sea conocida más que por las personas que nosotros queramos. En esto nos ayuda el cifrado. Cifrando un mensaje hacemos que este no pueda ser leído por terceras personas consiguiendo así la tan deseada privacidad.

- **Autenticar:** Otra de las necesidades que surgen con la aparición de internet es la necesidad de demostrar que somos nosotros y que el emisor es quien dice ser. Un método de autenticación puede ser el propio cifrado. Si ciframos un mensaje con una clave solo conocida por nosotros, demostrando que somos quien decimos ser, el receptor podrá constatar nuestra identidad descifrándolo. Esto se puede conseguir mediante **clave simétrica** (el receptor tiene que estar en posesión de la clave empleada) o usando **clave asimétrica** en su modo de autenticación.
- **Firmar:** Dados los trámites que podemos realizar hoy en día a través de internet se hace necesaria la aparición de la **firma digital**. Igual que firmamos un documento, la firma digital nos ofrece la posibilidad de **asociar una identidad a un mensaje**. Este uso tiene una gran importancia en el proyecto aquí desarrollado y posteriormente se explicará con más detalle.

Los **sistemas criptográficos** se pueden clasificar en función de:

- **Tipos de operación.**
  - Sustitución.
  - Transposición.
- **El número de claves.**
  - **Simétrico/convencional/clásico**. Con este cifrado se dispone de una única clave entre emisor y receptor para cifrar y descifrar.
  - **Asimétrico/de clave pública**. En este caso disponemos de dos claves, una privada (Kp) y una pública (KP).
- **Modo de procesar el texto plano.**
  - Bloque.
  - Flujo.

En este caso atenderemos a la clasificación en función del modo de procesar el texto plano y dentro de esta en función del número de claves.

2.1.2.1.Cifrado en bloque.

Es un método de cifrado en el que se opera en **grupos de bits de longitud fija**, llamados **bloques**, aplicándoles una transformación invariante.

2.1.2.1.1. Algoritmos simétricos.

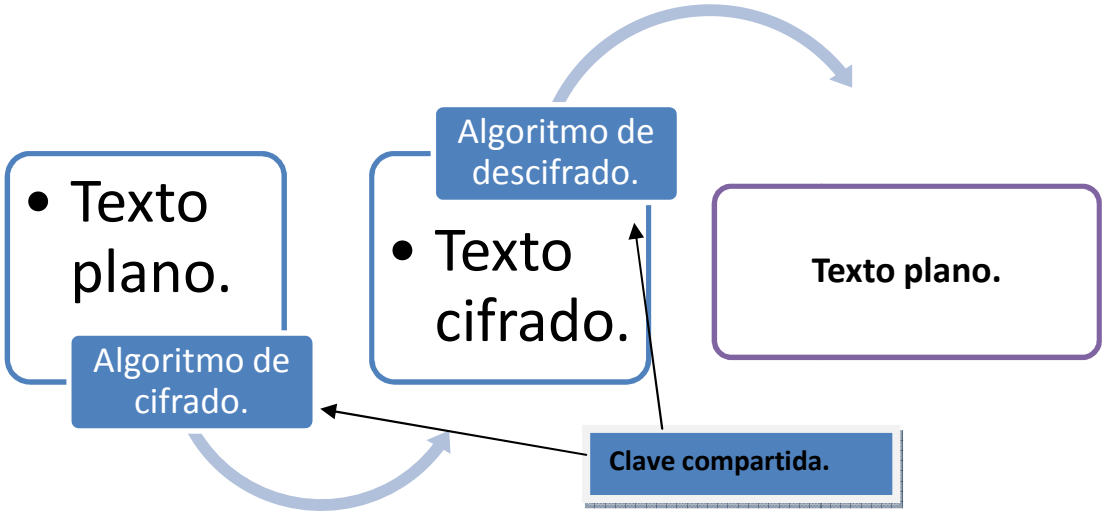

Figura 2.1. Esquema de funcionamiento de algoritmos simétricos.

En estos tipos de algoritmos, como ya se ha mencionado, existe **una única clave** entre emisor y receptor para **cifrar** y **descifrar**.
Donde mejor funcionan estos algoritmos es en hardware.
Se dispone de una **clave** independiente del texto plano, **en la cual radica la seguridad** dado que aunque se conozca el algoritmo, si no se conoce dicha clave no se puede obtener nada. Cualquier sistema de seguridad en el que no se conozca el algoritmo está condenado a fracasar pues éste se terminará descubriendo, cosa que no ocurre con la clave.
Dentro de estos algoritmos los más importantes son:

DES.

Dicho algoritmo **codifica** grupos de 64 bits, la clave es de 56 bits donde los 8 bits restantes son de paridad y rellena con ceros si el texto plano no es múltiplo de 64 bits.

Cuando **DES** queda obsoleto, debido a que la clave de 56 bits se queda pequeña surge **triple DES** el cual consiste en aplicar cifrado y descifrado tres veces pero con dos claves.

AES.

Este algoritmo es más complejo que el anterior, en él un byte es un elemento de campo finito. Se trabaja con dichos bytes y se interpretan como si fueran **polinomios de grado 7**.

2.1.2.1.2. Algoritmos asimétricos.

Estos algoritmos se implementan en software, al contrario que lo simétricos y están basados en **funciones matemáticas** (sustituciones simétricas y combinaciones). Además, como ya

hemos dicho, emplea **dos claves**, una pública y otra privada. Debido a estas dos razones, este tipo de algoritmos son **más lentos** que los simétricos. Además las claves son de longitud considerable para tener el mismo nivel de seguridad que los algoritmos simétricos.
Se emplean básicamente para cifrar la clave de sesión simétrica de cada mensaje o transacción particular (transportar claves de un lado para otro). Dado que la distribución de claves y la **firma digital** son los dos principales problemas a solucionar en la criptografía simétrica.

Si lo que queremos es una **comunicación cifrada de A a B** (**criptografía asimétrica**) tendremos una **clave para cifrar** y **otra para descifrar** (caso particular del algoritmo **RSA** que posteriormente explicaremos en más detalle). Tendremos una pareja de claves, pública y privada de B. Con la clave pública aplicaremos el algoritmo de **cifrado** mientras que con la privada se aplicará el algoritmo de **descifrado**. Únicamente B podrá descifrar el texto cifrado dado que **únicamente él conocerá la clave privada**.

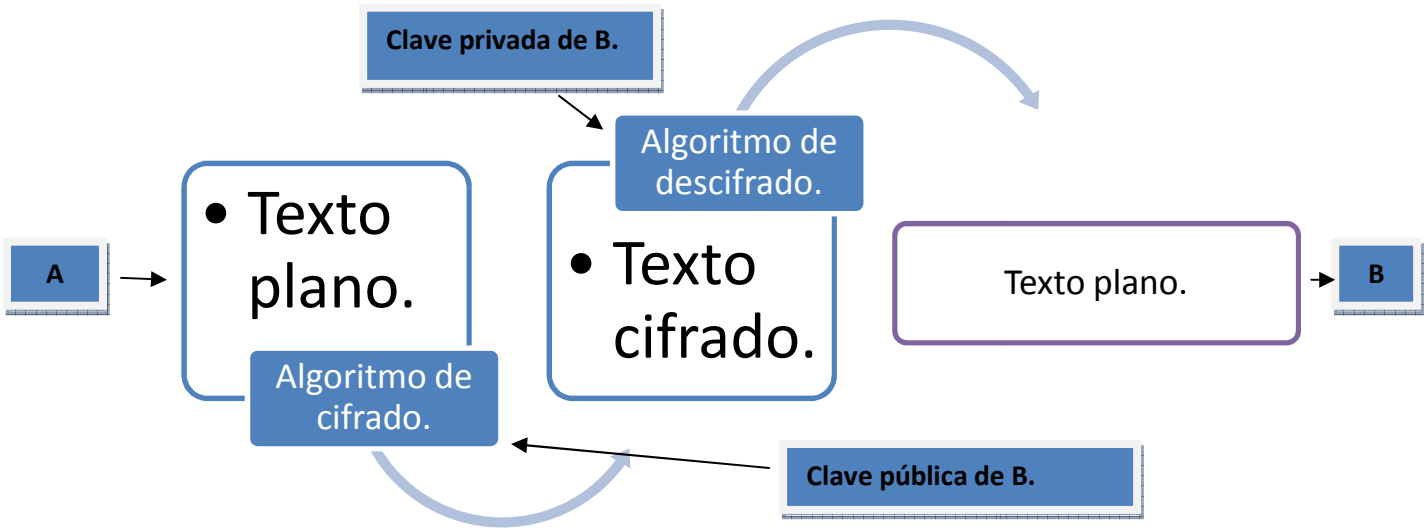

Figura 2.2. Esquema de funcionamiento de algoritmos asimétricos. Comunicación cifrada.

Sin embargo, si lo que queremos conseguir es **autenticación**, en tal caso se dispone también de una pareja de claves pública y privada pero en este caso del emisor. De manera que, se aplica el algoritmo de **cifrado** con la **clave privada de A** y en recepción el algoritmo de **descifrado** con la **clave pública de A**.

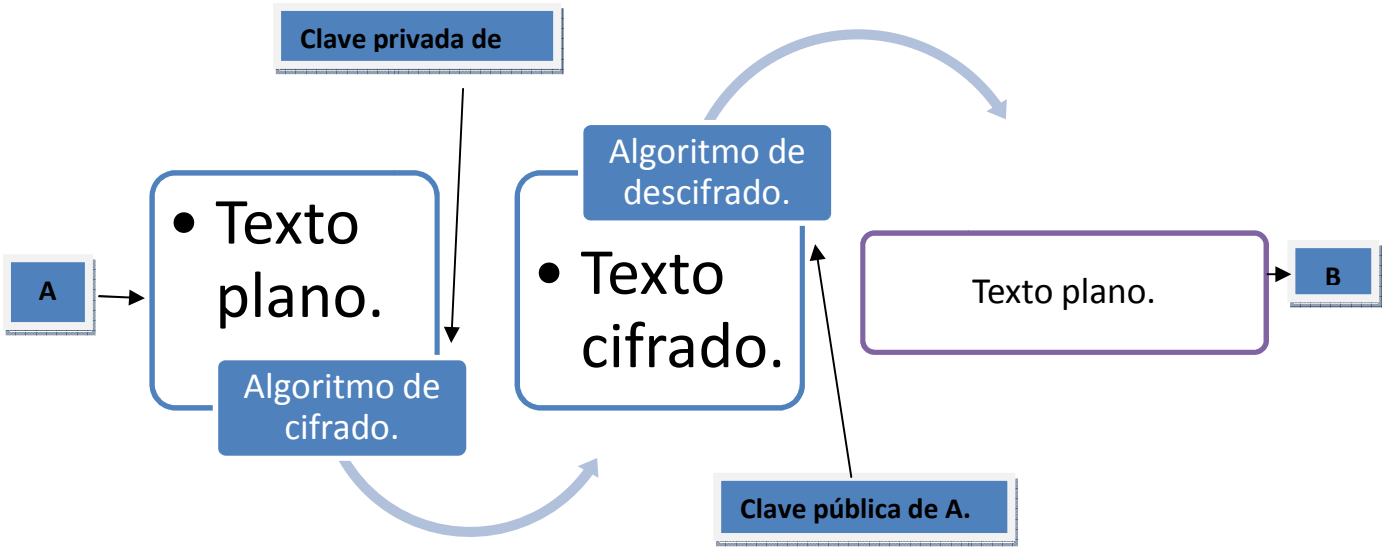

Figura 2.3. Esquema de funcionamiento de algoritmos asimétricos. Autenticación.

A pesar de que de esta manera **conseguimos autenticación e integridad de datos**, no conseguimos cifrado, por lo que para conseguir **autenticación y cifrado** se deberá: cifrar con la clave **privada de A** y volver a cifrar con la **pública de B**, de manera que en B se descifrará con la **pública de A** y posteriormente con la **privada de B**.

Por tanto las aplicaciones para las cuales empleamos los **algoritmos asimétricos** serán: **cifrado/descifrado**, **firma digital** e **intercambio de claves**.

Los **algoritmos asimétricos** más importantes son los siguientes:

RSA.

Este algoritmo se desarrolla en 1977 y es empleado tanto para cifrar como para **firmar digitalmente**, es decir para crear **certificados digitales**.
Su seguridad radica en el problema de la factorización de números enteros. Los mensajes enviados se representan mediante números, y el funcionamiento se basa en el producto, conocido, de dos números primos grandes elegidos al azar y mantenidos en secreto. Actualmente estos primos son del orden de 10^{200}, y se prevé que su tamaño crezca con el aumento de la capacidad de cálculo de los ordenadores.
Como en todo sistema de **clave pública** o **sistema asimétrico**, cada usuario posee dos claves de cifrado: una **pública** y otra **privada**. Cuando se quiere enviar un mensaje, el emisor busca la clave pública del receptor, cifra su mensaje con esa clave, y una vez que el mensaje cifrado llega al receptor, este se ocupa de descifrarlo usando su clave privada.
A continuación vamos a explicar brevemente el funcionamiento de dicho algoritmo. Se dispone de:

- Clave pública KP = {e,n}.
- Clave privada Kp = {d,n}.

Estas claves se emplean indistintamente para **cifrar**, **descifrar** y/o **autenticar**.

Se puede:

- Cifrar → $Y = X^e \bmod n$
- Descifrar → $X = Y^d \bmod n$

De manera que se puedan combinar como se quiera estas dos claves, mientras se cumpla lo siguiente para que las operaciones de cifrado y descifrado funcionen:

$$X = Y^d \bmod n = (X^e \bmod n)^d \bmod n = X^{ed} \bmod n = X$$

Se deben cumplir los siguientes requisitos:

- Tiene que ser posible encontrar valores de *e,d* y *n* tales que se cumpla la relación anterior.
- Debe ser relativamente fácil calcular X^e e Y^d para todos los valores de X <n.
- Aunque conozca *d* ó *e* debe ser imposible calcular la otra clave.

El algoritmo va a constar de tres partes descritas a continuación:

1. **Generación de claves.**

1.1. Cada usuario elige dos números primos distintos *p* y *q*.
Por motivos de seguridad, estos números deben escogerse de forma aleatoria y deben tener una longitud en bits parecida. Se pueden hallar primos fácilmente mediante test de primalidad.

1.2. Se calcula *n = p*q*.
Como ya hemos visto, *n* se usa como el módulo para ambas claves, pública y privada.

1.3. Se calcula φ (n) = (p-1)(q-1), donde φ es la **función ϕ de Euler**.

1.4. Se escoge un entero positivo *e* menor que φ (n), que sea coprimo con φ (n).
Como ya se ha mencionado, *e* se da a conocer como el exponente de la clave pública.
Si se escoge un *e* con una suma encadenada corta, el cifrado será más efectivo. Un exponente *e* muy pequeño podría suponer un riesgo para la seguridad.

1.5. Se determina un *d* (mediante **aritmética modular**) que satisfaga la congruencia *ed = 1 mod(*φ(n)*)*, es decir, que *d* sea el multiplicador modular inverso de *e mod (*φ(n)*).*
Esto suele calcularse mediante el **algoritmo de Euclides extendido**.
D se guarda como el exponente de la clave privada.
Por tanto se obtienen las claves que hemos visto anteriormente, la **clave pública** es *(n,e)*, esto es, el módulo y el exponente de cifrado. La **clave privada** es *(n,d)*, esto es, el módulo y el exponente de descifrado, que debe mantenerse en secreto.

2. **Cifrado.**

   Una vez que tenemos las claves, el emisor (A) desea enviar un mensaje al receptor (B), de manera que A convierte el texto plano (X) en un número entero menor que *n* mediante un protocolo reversible acordado de antemano, luego calcula el texto cifrado (Y) mediante la siguiente operación:

   $$Y = X^e \bmod n$$

   Una vez que se tiene Y el emisor envía el mensaje al receptor.

3. **Descifrado.**

   Cuando el receptor recibe el mensaje puedo obtener el número entero (del texto plano) mediante la siguiente operación:

   $$X = Y^d \bmod n$$

   Y una vez que lo tiene puede recuperar el mensaje original invierto el protocolo que ha aplicado el emisor.

El algoritmo **RSA** es el más importante de los algoritmos **asimétricos**. Para que éste sea seguro se recomienda que n sea aproximadamente de 2048 bits (700 dígitos) y que *p* y *q* sean aproximadamente de unos 300 dígitos de longitud cada uno.

Se cree que RSA será seguro mientras no se conozcan formas rápidas de descomponer un número grande en producto de primos. La computación cuántica podría proveer de una solución a este problema de factorización.

Los ataques más conocidos a este algoritmo son por **fuerza bruta** (excesivo coste computacional) y **man in the middle**, el cual se soluciona con un **intermediario de confianza (Autoridad Certificadora)** en el que emisor y receptor confíen (a continuación se explicará su funcionamiento en detalle).

Diffie-Hellman.

Este algoritmo surge en 1976 y es empleado principalmente para el intercambio de claves.
Se emplea generalmente como medio para acordar claves simétricas que serán empleadas para el cifrado de una sesión (establecer clave de sesión). Siendo no autenticado, sin embargo, provee las bases para varios protocolos autenticados.
Su seguridad radica en la dificultad de calcular logaritmos discretos.
Este algoritmo se emplea en la práctica para la red para anonimato **Tor** la cual usa el protocolo Diffie Hellman, sobre una conexión TLS de una capa inferior previamente establecida, para procurarse claves de sesión entre el cliente y los nodos de enrutamiento de la red. Esas claves son usadas para cifrar las capas de cebolla de los paquetes que transitan por la red. Además también se emplea para el protocolo **Off-the-record messaging** para comunicación de mensajería instantánea, el cual se apoya en el protocolo Diffie-Hellman para ir cambiando de clave de cifrado según se van intercambiando los mensajes.

Curvas elípticas.

Este algoritmo pretende ser la sustitución de **RSA**, dado que este emplea cada vez claves más largas y por tanto tiene un tiempo de procesado muy elevado. De manera que este método soluciona ese problema, dado que ofrece igual seguridad pero con longitudes de clave mucho menores.
Este tipo de sistemas se basa en la dificultad de encontrar la solución a ciertos problemas matemáticos, siendo uno de estos problemas el llamado logaritmo discreto.

2.1.2.2.Cifrado en flujo.

Para algunas aplicaciones, tales como el cifrado de conversaciones telefónicas, el cifrado en bloques es inapropiado 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.

Los algoritmos de **cifrado en flujo** realizan el cifrado incrementalmente, convirtiendo el texto en claro en texto cifrado bit a bit. Esto se logra construyendo un generador de flujo de clave. Un **flujo de clave** es una secuencia de bits de tamaño arbitrario que puede emplearse para oscurecer los contenidos de un flujo de datos combinando el flujo de clave con el flujo de datos mediante la **función XOR**. Si el flujo de clave es seguro, el flujo de datos cifrados también lo será. Se puede construir un generador de flujo de clave iterando una función matemática sobre un rango de valores de entrada para producir un flujo continuo de valores de salida. Los valores de salida se concatenan entonces para construir bloques de texto en claro, y los bloques se cifran empleando una clave compartida por el emisor y el receptor.

Para conservar la calidad de servicio del **flujo de datos**, los bloques del flujo de clave deberían producirse con un poco de antelación sobre el momento en que vayan a ser empleados, además el proceso que los produce no debiera exigir demasiado esfuerzo de procesamiento como para retrasar el flujo de datos.

Los algoritmos más importantes que emplean este tipo de cifrado son:

RC4.

Es el sistema de cifrado de flujo más utilizado y se usa en algunos de los protocolos más populares como **Transport Layer Security (TLS/SSL)** (para proteger el tráfico de Internet) y **Wired Equivalent Privacy (WEP)** (para añadir seguridad en las redes inalámbricas). RC4 fue excluido enseguida de los estándares de alta seguridad por los criptógrafos y algunos modos de usar el algoritmo de criptografía RC4 lo han llevado a ser un sistema de criptografía muy inseguro, incluyendo su uso WEP. No está recomendado su uso en los nuevos sistemas, sin embargo, algunos sistemas basados en RC4 son lo suficientemente seguros para un uso común.

A5.

Es usado para proporcionar privacidad en la comunicación al aire libre en el **estándar GSM**, es decir, el algoritmo que cifra la conversación entre 2 terminales GSM cuando el mensaje viaja por el aire. Inicialmente fue mantenido en secreto pero llegó al dominio público debido a sus debilidades y a la ingeniería inversa. Varias debilidades serias han sido identificadas en el algoritmo.

Como hemos visto existen multitud de ataques a los que se les hacen frente con diferentes **técnicas de seguridad**. Por ejemplo, para un ataque de revelación o en el que se pretenda analizar el tráfico se emplea el **cifrado** (evitando que la información quede al descubierto). Los ataques de enmascaramiento (insertar mensajes en la red que no existen), modificación de contenidos, de la secuencia o temporal se emplea la **autenticación** e **integridad**. Y para evitar el **repudio** se emplea la **firma digital**, la cual se basa en técnicas de autenticación como se va a explicar a continuación.

2.1.3. Firma Digital

Para la firma digital se utiliza **criptografía** de **clave pública** o de **clave asimétrica** (dos claves una privada y otra pública). Lo que se cifra con la clave privada (que solo nosotros conocemos) sólo se puede descifrar con la pública. De esta forma al cifrar con nuestra clave privada demostramos que somos nosotros (**no repudio**).

Para que una **firma digital** se considere equivalente a una firma manuscrita debe tener las siguientes propiedades:

- Poder verificar **autor**, **fecha** y **hora** de la firma.
- Poder autenticar el contenido del mensaje a la hora en la que se firmó (no puede haber tachones en un documento oficial).
- Debe estar verificada por un tercero para evitar disputas (lo que en una firma manuscrita se verifica con el DNI).

Además cualquier **firma digital** cumple lo siguiente:

- Se corresponde con un patrón de bits dependientes del mensaje firmado.
- Utilizará información única del emisor (clave secreta de la firma) para evitar denegación y falsificación.
- Es sencilla de crear.
- Es sencilla de reconocer y verificar.
- Falsificarla debe ser computacionalmente no factible.

El esquema de funcionamiento es el siguiente:

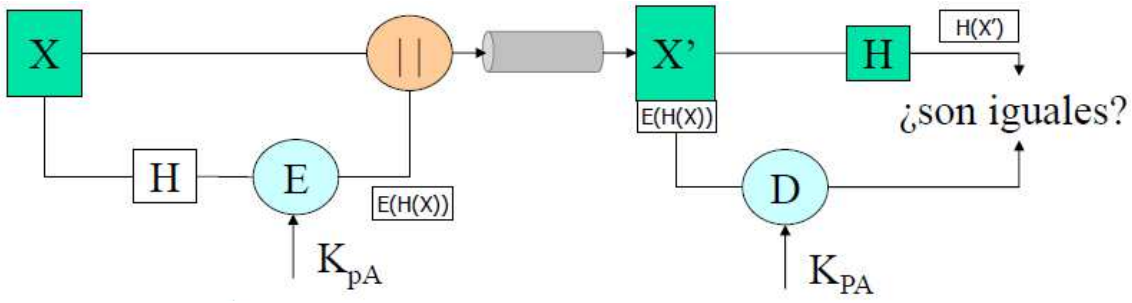

Figura 2.4. Esquema de funcionamiento de la firma digital.

Donde:

- **X** es el texto plano.
- **H** es la función **HASH**, algoritmos que consiguen crear a partir de una entrada (ya sea un texto, una contraseña o un archivo, por ejemplo) una salida alfanumérica de longitud normalmente fija que representa un resumen de toda la información que se le ha dado (es decir, a partir de los datos de la entrada crea una cadena que solo puede volverse a crear con esos mismos datos).
- **E** es el algoritmo de cifrado aplicado con **KpA** (clave privada del emisor).
- **D** el algoritmo de descifrado aplicado con **KPA** (clave pública del emisor).

Así el texto plano se pasa por una función **HASH** y posteriormente se le aplica el algoritmo de cifrado **E** con la clave privada del emisor **KpA** obteniendo así la firma digital **E(H(X))**, de modo que se envía por el canal el texto plano concatenado con la firma digital.

En recepción para realizar la verificación del mensaje el receptor generará el **HASH o huella digital** del mensaje recibido obteniendo **H(X')**, luego descifrará la firma digital del mensaje utilizando la clave pública del emisor **KPA** y obtendrá de esa forma el **HASH** del mensaje original; si ambas **huellas digitales** coinciden, significa que no hubo alteración y que el firmante es quien dice ser.

Para la firma digital existen dos modos de funcionamiento:

- **Firma digital directa.** En este modo sólo intervienen dos comunicantes, además el destino conoce la clave pública del emisor y se firma el mensaje completo con la clave privada del emisor **Kp**. El problema de este método reside en la seguridad de la clave secreta (problema de **Man in the middle**).
- **Firma digital Arbitrada.** En este método una tercera entidad (**autoridad certificadora**) actúa como árbitro, de modo que todos los mensajes pasan a través de esta entidad que comprueba la veracidad de origen y contenido. Por tanto existe una confiabilidad total en el árbitro. El problema de este método reside en que **no es escalable**.

2.1.3.1. Certificados.

Un **certificado digital** o certificado electrónico es un fichero informático generado por una entidad de servicios de certificación que **asocia unos datos de identidad a una persona física, organismo o empresa** confirmando de esta manera su **identidad digital en Internet**. El **certificado digital** es válido principalmente para autenticar a un usuario o sitio web en internet por lo que es necesaria la colaboración de un tercero que sea de confianza para cualquiera de las partes que participe en la comunicación. El nombre asociado a esta entidad de confianza es **Autoridad Certificadora** pudiendo ser un organismo público o empresa reconocida en Internet.

El **certificado digital** tiene como función principal **autenticar al poseedor** pero puede servir también para cifrar las comunicaciones y firmar digitalmente. En algunas administraciones públicas y empresas privadas es requerido para poder realizar ciertos trámites que involucren intercambio de información sensible entre las partes.

2.1.3.1.1. Creación de un certificado digital.

A continuación se va a explicar cómo se crea un **certificado digital**:

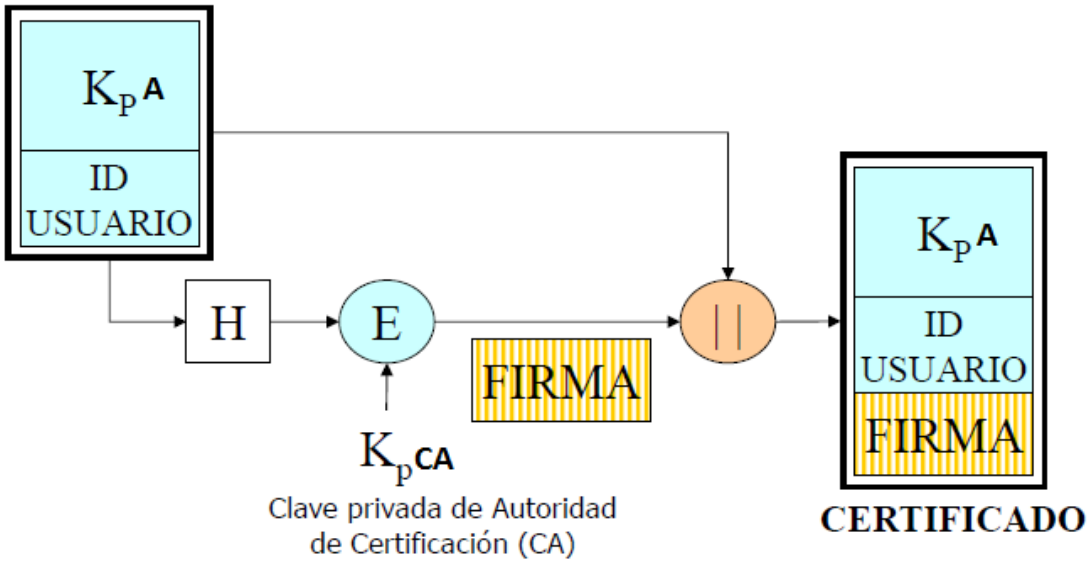

Figura 2.5. Esquema de creación de un certificado digital.

La sucesión de eventos es la siguiente:

1. Al inicio el navegador crea una pareja de claves para los usuarios, una **clave privada KpA** y una **clave pública KPA**.
2. **A** solicita a la **Autoridad Certificadora** un certificado digital, mandándole su **KPA** y su **ID de usuario**.
3. La **autoridad certificadora** aplica la función **HASH** sobre estos datos y posteriormente el **algoritmo de cifrado** con su clave privada **KpCA**. Garantizándose así que la clave pública es de quien dice ser y solucionándose de esta manera el problema de **Man in the middle**.

Una vez que se realiza este proceso se obtiene la **firma** que se concatena con la **KPA** y el **ID de usuario** (obteniendo así el **certificado digital**) como vemos en el esquema.

4. Una vez que se tiene el **certificado digital** se manda a **A** que es el usuario que lo ha solicitado. Éste le sirve a **A** para que a cualquiera que le pase su clave pública pueda garantizar que es de **A.**

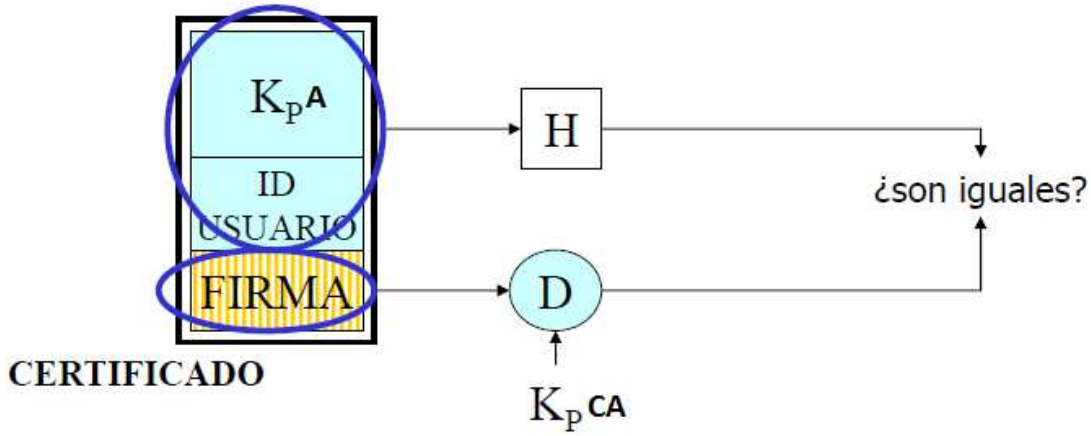

Figura 2.6. Comprobación de la firma digital.

5. **A** le manda a **B** su **certificado.**
6. **B** comprueba el certificado, es decir, que la clave pública de **A** es de **A**.
   Para ello aplica la función **HASH** a la pareja de clave pública e ID de usuario por un lado y el algoritmo de descifrado con la clave pública de la **CA** por otro, sin ambos coinciden entonces **B** sabe que la clave pública de **A** es de **A**.
7. Una vez que **B** confía en **A** comprueba su **firma** para **autenticar**. Dicha **firma** será la que esté firmada con la **clave privada** de **A** (como ya se ha explicado en el apartado anterior).

Así mismo sabemos que los certificados van a tener un cierto periodo de validez dependiendo de la **Autoridad de certificación**. Además estos certificados se pueden **revocar** por diversas razones: la clave privada del usuario se ve comprometida, la **CA** emite el certificado a una entidad incorrecta, el usuario cambia de **CA** o se ve violada la seguridad de la **CA**. De manera que va a existir una **lista de revocación de certificados ó CRL** en la **CA** la cual será consultada por **B** cuando recibe un certificado para comprobar si éste está en la **CRL**.

2.1.3.1.2. Tipos de certificados digitales.

Certificados de autoridades certificadoras.

Son los certificados creados para confiar en la propia **autoridad certificadora**, por lo que se dispone del nombre y la clave pública de la **CA**. Éstos pueden ser auto firmados. Además emplean la tecnología **PKI** la cual permite a los usuarios autenticarse frente a otros usuarios y usar la información de los certificados de identidad (por ejemplo, las

claves públicas de otros usuarios) para cifrar y descifrar mensajes, firmar digitalmente información, garantizar el no repudio de un envío, y otros usos.

Certificados de servidores.

Estos tipos de certificados deben estar siempre en el extremo del servidor, por ejemplo cada **servidor SSL** debe tener un certificado de **servidor SSL**. Estos certificados deben contener la longitud de la clave firmada, el número de serie del certificado, el algoritmo de firma y el nombre del servidor. Son fundamentales para comercio electrónico.

Certificados personales.

Este tipo de certificados garantizan que la clave pública pertenece a una persona física, es decir, están diseñados para comprobar la **identidad** de un individuo emitido por una **CA**. Además también se pueden emplear para hacer un **intercambio de claves** entre dos usuarios. Proporciona diversas ventajas como eliminar la necesidad de recordar login y password, tener una prueba de pertenecer a una organización, proporcionar comunicaciones cifradas y restringir accesos a sitios web.

A partir de la **versión 3** de Navigator Netscape e Internet Explorer se permite la creación de claves, la obtención de certificados, reto/respuesta y un almacenamiento seguro, lo cual es importante para disponer de estos certificados y poder acceder a diferentes recursos de la **administración Central**, **Autonómica** o **Local**.

Certificados de editor de software.

Estos certificados son empleados por las compañías que crean programas para verificar el creador del mismo. Se firman los programas ejecutables mediante la firma electrónica. De esta manera se mejora la **confiabilidad** del software distribuido por Internet.

2.1.3.1.3. Formato de certificados digitales.

El **formato** empleado en los navegadores para los certificados es actualmente el **X.509v3** el cual es un estándar UIT-T para infraestructuras de claves públicas (PKI). **X.509** específica, entre otras cosas, formatos estándar para certificados de claves públicas y un algoritmo de validación de la ruta de certificación. Su sintaxis, se define empleando el lenguaje ASN.1 (Abstract Syntax Notation One), y los formatos de codificación más comunes son **DER** (Distinguish Encoding Rules) o **PEM** (Privacy Enhanced Mail).

Dichos certificados contienen los siguientes datos:

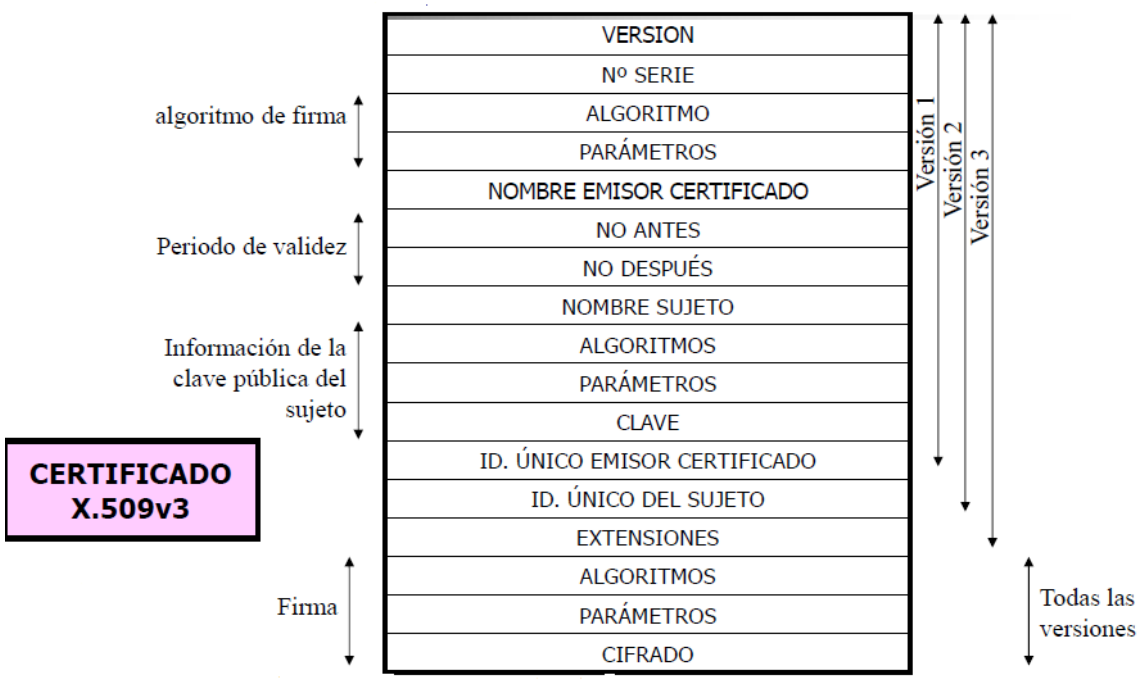

Figura 2.7. Formato X509 del certificado digital.

Donde como vemos a medida que aumenta la versión del certificado se van introduciendo nuevos campos.
En el **anexo 2** se explicarán con más detalles los diferentes formatos y extensiones de los **certificados digitales**.

Como podemos observar los **certificados digitales** pueden proporcionar numerosas ventajas pero también tienen aún ciertos inconvenientes, lo que sin duda propicia el desarrollo y la mejora de estos usos.

2.1.3.2.Autoridad de certificación.

Como ya se ha mencionado, en criptografía una **Autoridad de Certificación** (**AC** o **CA** por sus siglas en inglés Certification Authority) es una entidad de confianza, responsable de **emitir** y **revocar** los certificados digitales, utilizados en la firma electrónica, para lo cual se emplea la criptografía de **clave pública**. Jurídicamente es un caso particular de Prestador de Servicios de Certificación.

La **Autoridad de Certificación**, por sí misma o mediante la intervención 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.

2.1.3.2.1. Modo de funcionamiento.

Solicitud de un certificado.

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** (Certificate Signing Request) 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 RA (Registration Authority, 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**.

En este contexto, PKCS corresponde a un conjunto de especificaciones que son estándares de facto denominadas Public-Key Cryptography Standards.

Jerarquía de certificación.

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 puede estar **auto-firmado** cuando no hay ninguna CA de rango superior que lo firme. Este es el caso de los certificados de **CA raíz**, el elemento inicial de cualquier 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 (titular de certificado: 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.

Así van a existir diferentes servicios ofrecidos por las **CA** en función de su jerarquía:

- **CA interna.** Se emplea para certificar a sus propios empleados, puestos y niveles de autoridad.
- **CA externa de empleados.** La empresa contrata a otra para certificar a sus empleados.

- **CA externa de clientes.** La empresa contrata a otra para que certifique a sus clientes.
- **CA confiable de terceros.** Una compañía o gobierno opera una **CA** que relaciona claves públicas con nombres legales de individuos o empresas. Son aquellas en las que todos confían.

Además sabemos que existen **CA públicas** y **privadas**. Los certificados de **CA (certificados raíz)** de las **CAs públicas** pueden o no estar instalados en los navegadores pero son reconocidos como **entidades confiables**, frecuentemente en función de la normativa del país en el que operan. Las **CAs públicas** emiten los certificados para la población en general (aunque a veces están focalizadas hacia algún colectivo en concreto) y además **firman CAs de otras organizaciones**.

Confianza de la CA.

Una de las formas por las que se establece la confianza en una CA para un usuario consiste en la "**instalación**" en el ordenador del usuario (tercero que confía) del **certificado auto firmado de la CA raíz** de la jerarquía en la que se desea confiar. El proceso de instalación puede hacerse, en sistemas operativos de tipo Windows, haciendo doble clic en el fichero que contiene el certificado (con la extensión "crt") e iniciando así el "**asistente para la importación de certificados**". Por regla general el proceso hay que repetirlo por cada uno de los navegadores que existan en el sistema, tales como Opera, Firefox o Internet Explorer, y en cada caso con sus funciones específicas de importación de certificados.

Si está instalada una **CA** en el repositorio de **CAs de confianza** de cada navegador, **cualquier certificado firmado por dicha CA se podrá validar**, ya que se dispone de la **clave pública** con la que verificar la firma que lleva el certificado. 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.

2.1.3.2.2. Normativa.

Normativa europea.

La Directiva 93/19991 ha establecido un marco común aplicable a todos los países de la **Unión Europea** por el que el nivel de exigencia que supone la **normativa firma electrónica** implica que los **Prestadores de Servicios de Certificación** que emiten certificados cualificados son merecedores de confianza por cualquier **tercero que**

confía y sus certificados otorgan a la **firma electrónica avanzada** a la que acompañan el mismo valor que tiene la "**firma manuscrita**" o "**firma hológrafa**".

Normativa española.

La **Ley 59/2003 de Firma Electrónica** ha derogado el Real Decreto Ley 14/1999, de 17 de septiembre, sobre firma electrónica haciendo **más efectiva la actividad de certificación en España**.

2.1.3.2.3. Misión de la CA.

Finalmente, las **CA** también se encargan de la **gestión de los certificados firmados**. Esto incluye las tareas de **revocación de certificados** que puede instar el titular del certificado o cualquier tercero con interés legítimo ante la **CA** por correo electrónico, teléfono o intervención presencial.

Como ya se ha comentado, la lista denominada **CRL** (Certificate Revocation List) contiene los certificados que entran en esta categoría, por lo que es **responsabilidad de la CA publicarla y actualizarla** debidamente. Por otra parte, otra tarea que debe realizar una **CA** es la **gestión** asociada a la **renovación de certificados por caducidad o revocación**. Si la **CA** emite muchos certificados, corre el riesgo de que sus **CRL** sean de **gran tamaño**, lo que hace poco práctica su descarga para los terceros que confían. Por ese motivo desarrollan mecanismos alternativos de consulta de validez de los certificados, como servidores basados en los **protocolos OCSP** y **SCVP**.

Debido al gran número de entidades certificadoras existentes, y a pesar de las medidas de seguridad que emplean para la **correcta verificación de personas físicas y jurídicas**, existe el riesgo de que una **CA** emita un certificado a un **defraudador con una identidad falsa**. Como no existe un registro de qué certificados han sido emitidos por cada CA, no es fácil detectar qué casos se han filtrado de manera fraudulenta o qué certificados refieren a un nombre igual o muy parecido al de otra entidad. Para evitar este tipo de problemas, Google ha lanzado la **iniciativa certificate transparency**, que registra todos los certificados emitidos por las CA a través de unos ficheros de auditoría criptográficamente infalsificables, lo que puede ayudar a combatir el **phishing**.

CAPÍTULO 3. Descripción del Software y Hardware utilizado.

Una vez presentado el marco teórico del proyecto para que se pueda entender un poco mejor lo que se ha desarrollado, vamos a describir más en profundidad el software y hardware utilizado, mencionado en el primer capítulo (presentación del proyecto).

3.1. Software.

A continuación se va a explicar el sistema operativo que soporta el software creado así como el lenguaje empleado y el software usado como base para desarrollar el proyecto.

3.1.1. Sistema Operativo OpenWrt.

Es un sistema operativo utilizado principalmente en dispositivos embebidos para encaminar el tráfico de red. Los componentes principales son el núcleo de Linux, uClibc y BusyBox. Todos los componentes están optimizados en cuanto al tamaño se refiere, para que este SO sea lo suficientemente pequeño para encajar en un almacenamiento limitado disponible en los routers domésticos.

OpenWrt se puede configurar a través de una interfaz de línea de comandos (**ash**) o una interfaz web (**LUCI**). Hay alrededor de 2000 paquetes de software opcionales disponibles para la instalación a través del sistema de gestión de paquetes **opkg**. En nuestro caso no se llegó a instalar la interfaz **LUCI** (puerto 80) dado que se va a acceder a nuestro software empleando el servidor web **uhttpd** a través de este puerto, por tanto trabajaremos con el router a través de línea de comandos.

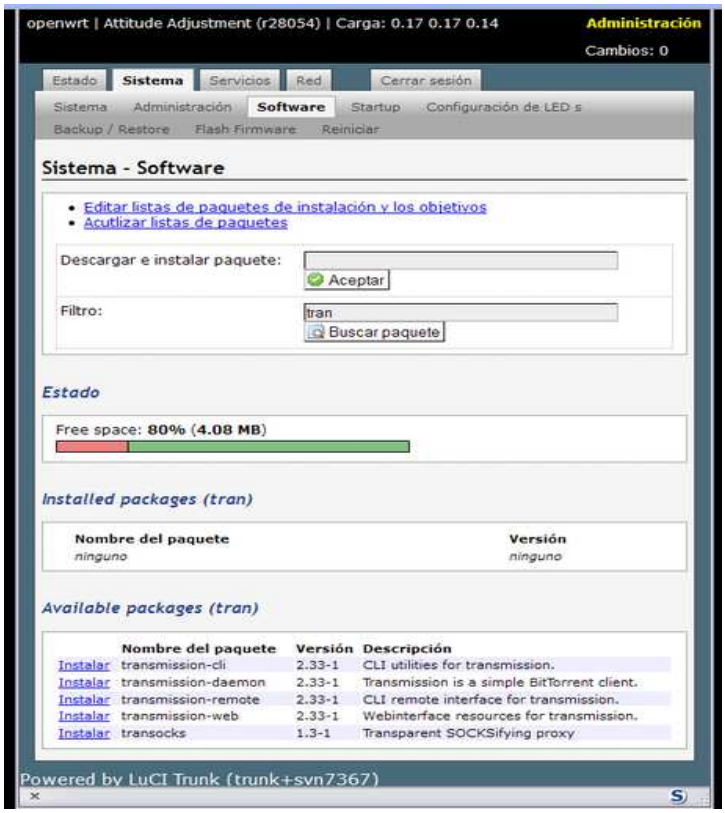

Figura 3.1. Interfaz Web (LuCi) OpenWrt.

A diferencia de muchas otras distribuciones para routers, **OpenWrt** está construido desde cero para ser completamente funcional y ser un **sistema operativo** fácilmente modificable para cualquier router.

En lugar de intentar crear un **firmware estático**, **OpenWrt** provee un sistema de ficheros totalmente modificable con la gestión de paquetes opcionales. Esto le libera de las restricciones de aplicaciones, funciones y configuraciones proporcionadas por el vendedor y le permite utilizar los paquetes para personalizar un dispositivo integrado, para que pueda **adaptarse** a cualquier aplicación. Para los **desarrolladores**, **OpenWrt** proporciona un marco para crear una aplicación sin tener que crear una imagen de firmware completa y la distribución que ello conlleva. Para los **usuarios**, esto significa la **libertad** de **personalización** completa, lo que permite el uso de un dispositivo embebido de muy diversas formas.

Además presenta las siguientes características:

- **Gratuito y de código abierto (Free and open-source)**. El proyecto es completamente gratis y de código abierto, licenciado bajo GPL. El proyecto tiene la intención de estar siempre alojado en un sitio de fácil acceso, con el código fuente completo disponible y fácil de construir.
- **Fácil y libre acceso.** El proyecto siempre estará abierto a nuevos desarrolladores y además promueve la participación de éstos. Cualquier persona deberá ser capaz de contribuir en el desarrollo, dado que los desarrolladores actuales, activamente conceden acceso de escritura para poder modificarlo a cualquier persona interesada en tenerlo.
- **Impulsada por la Comunidad.** Este trabajo consiste en un conjunto de personas uniéndose para trabajar y colaborar y poder lograr así un objetivo común.
- **Raíz de escritura del sistema de archivos**, lo que permite a los usuarios agregar, quitar o modificar cualquier archivo. Esto se logra mediante el uso de overlays.
- **El paquete opkg**, similar a dpkg o pacman, que permite a los usuarios instalar y quitar software. Esto contrasta con firmware basado en Linux, basado en sistemas de archivos de sólo lectura que ofrecen compresión eficiente, pero no hay manera de modificar el software instalado sin necesidad de recompilar y mostrando una imagen de framework completo.
- Un conjunto de scripts llamado **UCI** (interfaz unificada configuración) destinada a unificar y simplificar la configuración de todo el sistema.
- Configuración extensible de la red VLAN con la participación de las posibilidades exhaustivas para configurar el enrutamiento de sí mismo.
- Métodos personalizables para filtrar, manipular, retrasar y reorganizarlos paquetes de red como:
    - Firewall.
    - Calidad de servicio para el uso simultáneo de aplicaciones tales como VoIP, juegos en línea y medios de transmisión.
    - Gestión de cola Activa (AQM) con muchos programadores de paquetes disponibles. CODEL ha sido portado a Kernel 3.3.
    - Asignación de tráfico para garantizar una distribución justa de ancho de banda entre los usuarios.

    - Balanceo de carga para su uso con múltiples ISPs.
    - Supervisión de la red en tiempo real y estadísticas.
- Una amplia interfaz web Ajax, gracias al proyecto LUCI.
- Configuración del dispositivo como un repetidor inalámbrico, punto de acceso inalámbrico, puente inalámbrico, o una combinación de estos.
- La creación de redes de malla.
- Configurables por el usuario los botones de hardware.
- Correcciones de errores y actualizaciones regulares, incluso para los dispositivos ya no se admite por sus fabricantes.

OpenWrt se establece desde hace tiempo como la mejor solución de firmware de su clase. Es muy **superior** a otras soluciones incorporadas en el **rendimiento**, la **estabilidad**, la **expansibilidad**, **robustez** y **diseño**. La meta de los desarrolladores de **OpenWrt** es continuar expandiendo el desarrollo y asegurar que **OpenWrt** es el principal framework de soluciones innovadoras e ingeniosas.

El proyecto **OpenWrt** comienza a desarrollarse en enero de 2004. Las primeras versiones **OpenWrt** se basaron en fuentes Linksys WRT54G y GPL para un **buildroot** del proyecto **uclibc**. Esta versión se conoce como **OpenWrt "versión estable"** y su uso fue muy extendido. Todavía existen muchas aplicaciones **OpenWrt**, que se basan en esta versión.

A principios de 2005 algunos nuevos desarrolladores se unieron al equipo. Después de algunos meses de desarrollo cerrado el equipo decidió publicar las primeras versiones **"experimentales" de OpenWrt**. Las versiones experimentales utilizan un sistema de construcción muy personalizada basada en buildroot 2 del proyecto **uclibc**. **OpenWrt** utiliza fuentes oficiales del núcleo de GNU/Linux y sólo añade parches para el sistema en chip y controladores para las interfaces de red. El equipo desarrollador intenta volver a implementar la mayor parte del código propietario dentro de los archivos de código GPL de los diferentes proveedores. Hay herramientas gratuitas para escribir nuevas imágenes de firmware directamente en el **flash (MTD)**, para configurar el chip de **LAN inalámbrica** y programar el interruptor **VLAN** a través del sistema de ficheros proc.

El nombre en clave de la primera versión **OpenWrt** es **"White Russian"**. El desarrollo de esta primera versión finaliza con el lanzamiento de **OpenWrt 0.9**. Las versiones posteriores siguen el esquema de la versión sin el prefijo '0.0', y con el número de versión indican aproximadamente el año en el que la versión se convierte en una versión libre. En consecuencia, **OpenWrt 7 y 8**, fueron “puestos en libertad” a lo largo de 2007-2008. En 2010 **OpenWrt 10** estaba listo, con el nombre **"Backfire"**. En 2012 el lanzamiento fue el de la versión de **OpenWrt 12 "Attitude Adjustment"**.

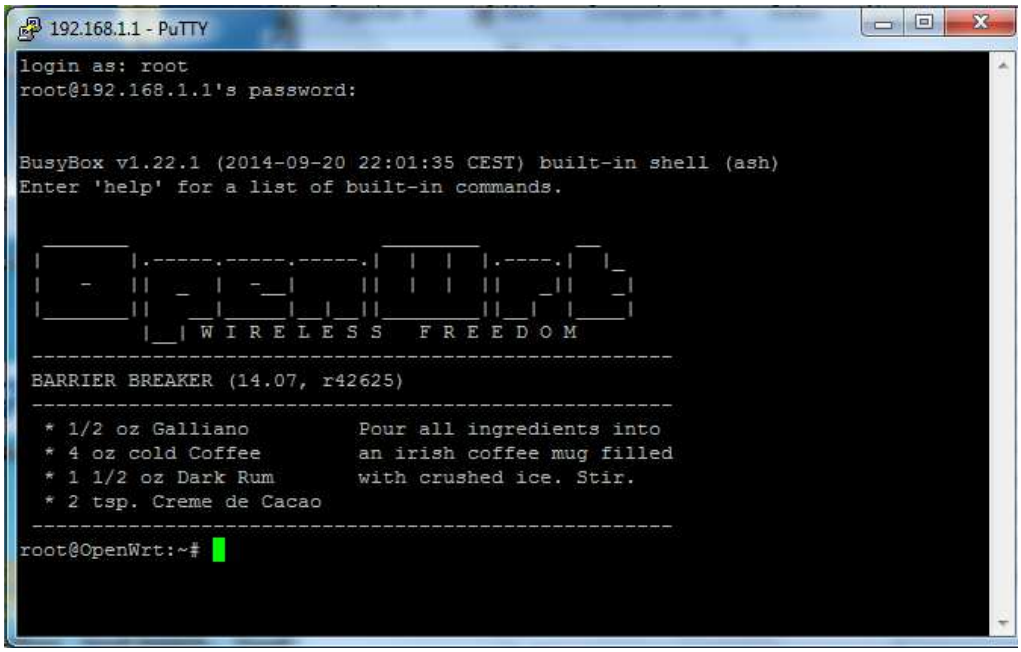

Figura 3.2. Conexión con OpenWrt línea de comandos. Información del Sistema Operativo.

Como vemos en la imagen la versión que se ha instalado en nuestro router es la **14.07** la cual presenta prácticamente las mismas novedades que la versión anterior release candidate 3. Es decir, con respecto a la **Release Candidate 3** no son demasiadas las modificaciones que podemos encontrar en la versión final, aunque se ha **optimizado el rendimiento global** del software para las tareas que le ocupan y, por supuesto, con respecto a versiones anteriores sí podemos encontrar una gran cantidad de novedades.

Con esta última versión se amplía el potencial de este firmware para ofrecer a los vendedores la posibilidad de personalizar sus dispositivos. Y lo más importante de todo es que se puede alcanzar un nivel de personalización completo sin necesidad de crear un firmware completo, sino haciendo "**ligeras modificaciones**" sobre el propio **OpenWrt**.

Con la actualización del kernel de Linux utilizado en **OpenWrt** a la versión 3.10, este sistema ha recibido soporte **IPv6** ahora de forma nativa, además de compatibilidad con el sistema de archivos **NAND – flash** y una importante optimización del **DHCP** y **DHCPv6**. Por otra parte, también hemos podido ver cambios que afectan al "**corta fuegos**" del sistema, lo que también es fundamental para garantizar no sólo conexiones veloces y estables, sino también seguras frente a amenazas que puedan tener lugar sobre un router y, por tanto, los usuarios que hacen uso del mismo. En cuanto al **soporte de hardware** que abarca OpenWrt, nos encontramos con un amplio listado de fabricantes conocidos como Asus, Belkin, D-Link y Netgear, de los cuales varios de sus más conocidos dispositivos están contemplados.

Así mismo para poder llevar a cabo el desarrollo del proyecto ha sido necesario instalar una serie de **paquetes** y hacer uso de algún **software** para poder llevar a cabo las funciones del software creado. Dichas herramientas se van a explicar brevemente a continuación:

3.1.2. Lenguaje de programación php.

PHP es un lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de **contenido dinámico**. Fue uno de los primeros lenguajes de programación del lado del servidor que se podían incorporar directamente en el documento **HTML** en lugar de llamar a un archivo externo que procese los datos. El código es interpretado por un servidor web con un módulo de **procesador de PHP** que genera la página Web resultante. **PHP** ha evolucionado por lo que ahora incluye también una interfaz de línea de comandos que puede ser usada en aplicaciones gráficas independientes. Puede ser usado en la mayoría de los servidores web al igual que en casi todos los sistemas operativos y plataformas sin ningún costo.

Se considera uno de los lenguajes más **flexibles**, **potentes** y de **alto rendimiento** conocidos hasta el día de hoy. Lo que ha atraído el interés de múltiples sitios con gran demanda de tráfico como **Facebook**, para optar por **PHP** como tecnología de servidor.

Fue creado originalmente por Rasmus Lerdorf en 1995. Actualmente el lenguaje sigue siendo desarrollado con nuevas funciones por el grupo **PHP**. Este lenguaje forma parte del software libre publicado bajo la licencia **PHP**, que es incompatible con la Licencia Pública General de GNU debido a las restricciones del uso del término **PHP**.

Fue originalmente diseñado en **Perl**, con base en la escritura de un grupo de CGI binarios escritos en el lenguaje C por el programador danés-canadiense Rasmus Lerdorf en el año 1994 para mostrar su currículum vítae y guardar ciertos datos, como la cantidad de tráfico que su página web recibía. El 8 de junio de 1995 fue publicado "Personal Home Page Tools" después de que Lerdorf lo combinara con su propio Form Interpreter para crear **PHP/FI**.

En mayo de 2000 **PHP 4** fue lanzado bajo el poder del motor Zend 1.0. El día 13 de julio de 2007 se anunció la suspensión del soporte y desarrollo de la **versión 4 de PHP**, a pesar de lo anunciado se ha liberado una nueva versión con mejoras de seguridad, la 4.4.8 publicada el 13 de enero del 2008 y posteriormente la versión 4.4.9 publicada el 7 de agosto de 2008. Según esta noticia se le dio soporte a fallos críticos hasta el 9 de agosto de 2008.

El 13 de julio de 2004, fue lanzado **PHP 5** (empleado en el desarrollo de este proyecto). Las **características** y **ventajas** más importantes de **php** son las siguientes:

- Orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una base de datos.
- Es considerado un lenguaje fácil de aprender, ya que en su desarrollo se simplificaron distintas especificaciones, como es el caso de la definición de las variables primitivas, ejemplo que se hace evidente en el uso de php arrays.
- El código fuente escrito en PHP es invisible al navegador web y al cliente, ya que es el servidor el que se encarga de ejecutar el código y enviar su resultado HTML al navegador. Esto hace que la programación en PHP sea segura y confiable.
- Capacidad de conexión con la mayoría de los motores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL y PostgreSQL.
- Capacidad de expandir su potencial utilizando módulos (llamados ext's o extensiones).
- Posee una amplia documentación en su sitio web oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.

- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Permite aplicar técnicas de programación orientada a objetos.
- No requiere definición de tipos de variables aunque sus variables se pueden evaluar también por el tipo que estén manejando en tiempo de ejecución.
- Tiene manejo de excepciones (desde PHP5).
- Si bien PHP no obliga a quien lo usa a seguir una determinada metodología a la hora de programar, aún haciéndolo, el programador puede aplicar en su trabajo cualquier técnica de programación o de desarrollo que le permita escribir código ordenado, estructurado y manejable.
- Debido a su flexibilidad ha tenido una gran acogida como lenguaje base para las aplicaciones WEB de manejo de contenido, y es su uso principal.
  Aunque también presenta una serie de **inconvenientes**:
- Como es un lenguaje que se interpreta en ejecución, para ciertos usos puede resultar un inconveniente que el código fuente no pueda ser ocultado. La ofuscación es una técnica que puede dificultar la lectura del código pero no necesariamente impide que el código sea examinado.
- Debido a que es un lenguaje interpretado, un script en PHP suele funcionar considerablemente más lento que su equivalente en un lenguaje de bajo nivel, sin embargo este inconveniente se puede minimizar con técnicas de caché tanto en archivos como en memoria.
- Las variables al no ser tipificadas dificulta a los diferentes IDEs para ofrecer asistencias para el tipificado del código, aunque esto no es realmente un inconveniente del lenguaje en sí. Esto es solventado por Zend Studio añadiendo un comentario con el tipo a la declaración de la variable.

La versión que aquí se emplea (**php5**) incluye las siguientes mejoras respecto a versiones anteriores:

- Mejor soporte para la programación orientada a objetos, que en versiones anteriores era extremadamente rudimentario.
- Mejoras de rendimiento.
- Mejor soporte para MySQL con extensión completamente reescrita.
- Mejor soporte a XML (XPath, DOM, etc.).
- Soporte nativo para SQLite.
- Soporte integrado para SOAP.
- Iteradores de datos.
- Manejo de excepciones.
- Mejoras con la implementación con Oracle.

3.1.3. Software OpenSSL.

OpenSSL es un proyecto de **software libre** basado en **SSLeay**, desarrollado por Eric Young y Tim Hudson. Consiste en un robusto paquete de **herramientas** de administración y bibliotecas relacionadas con la **criptografía**, que suministran funciones criptográficas a otros paquetes como **OpenSSH** y **navegadores web** (para acceso seguro a sitios HTTPS).

Estas herramientas ayudan al sistema a implementar el **Secure Sockets Layer** (**SSL**), así como otros protocolos relacionados con la seguridad, como el Transport Layer Security (TLS). **OpenSSL** también permite crear **certificados digitales** que pueden aplicarse a un servidor, por ejemplo Apache o como es nuestro caso **uhttpd**.

A principios de abril del 2014 se da a conocer un agujero de seguridad (conocido como **Heartbleed**) que afecta a las versiones 1.0.1 y 1.0.1f de este protocolo afectando a dos tercios de las comunicaciones seguras que se efectúan en Internet. El agujero está en el código de **OpenSSL** desde diciembre de 2011. Neel Mehta, del equipo de seguridad de Google, lo habría descubierto en diciembre de 2013, fecha de la inclusión del "bug" en la base de datos 'Common Vulnerabilities and Exposures'.

3.1.4. PHP5 + OpenSSL.

Así para poder hacer uso del software **OpenSSL** y poder así crear nuestros certificados (como se explicará en capítulos posteriores) será necesario instalar una serie de paquetes haciendo uso del gestor de paquetes de **OpenWrt** explicado anteriormente: **opkg**. Estos paquetes son:

- **Php5_mod_openssl.**
- **Libopenssl.**
- **Openssl_util.**

De esta manera quedan instaladas las herramientas necesarias para desarrollar nuestro trabajo.

3.1.5. Software basado en php-ca.

Consiste en un software desarrollado en **php**, empleando las herramientas de este lenguaje para la creación de una **Autoridad Certificadora** (**php-ca**), que ha sido usado como base para la creación de este proyecto, modificando algunas de las cosas que en este software se hacen e introduciendo **ciertas mejoras**. Además aún se puede trabajar más sobre este proyecto y seguir mejorándolo, pero eso será explicado en el capítulo dedicado a trabajos futuros.

3.2. Hardware.

A continuación se van a explicar las características más importantes del router empleado **hardware** donde ha sido instalado nuestro **software.**

3.2.1. Router TP-Link TL-WR-1043ND.

El **Router** Gigabit Inalámbrico N de 300Mbps, el TL- WR1043ND es un dispositivo de conexión a red conectado por cable/inalámbrico combinado integrado con **router** de compartición de Internet y **switch** de 4 puertos. Crea redes inalámbricas con **velocidades** increíblemente **altas**

de hasta 300Mbps, lo cual asegura que disfrute libremente múltiples aplicaciones simultáneamente sensibles a interrupciones y con alto consumo de banda ancha como el **streaming de video en HD**, hacer llamadas **VoIP**, compartir archivos grandes y jugar juegos en línea. Especialmente está equipado con un puerto de almacenamiento USB en la parte trasera del router para conectar dispositivos de almacenamiento USB a la red para la **compartición** conveniente de **recursos** para todas las personas que se encuentran en la red.

Figura 3.3. Router TP-Link-TL-WE1043ND.

Sus características más destacadas son las siguientes:

- La Velocidad Inalámbrica N hasta de 300Mbps lo hace ideal para aquellas aplicaciones que consumen banda ancha o que son sensibles a interrupción como el streaming de video, los juegos en línea y VoIP.
- Todos los puertos gigabit aseguran velocidades máximas de transferencia.
- Almacenamiento central y compartición del contenido conectando memorias USB.
- El enlace inalámbrico de WDS proporciona una conexión en puente sin interrupción para expandir la red inalámbrica.
- Fácil codificación de seguridad inalámbrica con sólo presionar el Botón WPS.
- El modo de mejora de velocidad incrementa las velocidades inalámbricas hasta 450Mbps.
- Configure fácilmente una conexión segura con codificación WPA con sólo presionar el botón WPS.
- El Asistente de fácil configuración proporciona una instalación rápida y libre de problemas.
- El control de banda ancha basada en la IP permite a los administradores determinar cuánto ancho de banda se asigna a cada PC.
- Compatible con versiones anteriores de productos 802.11b/g.
- Las antenas desmontables externas permiten un mejor ajuste y mayores mejoras al desempeño de la misma.

Para el desarrollo de este proyecto esta máquina va a actuar, como hemos dicho, como servidor para albergar el software desarrollado, el cual se va a explicar en el próximo capítulo.

CAPÍTULO 4. Descripción del software creado. Aplicación de usuario.

Una vez que se conoce con detalle todos los dispositivos y todo el software usado, en este capítulo se va a proceder a explicar de manera detallada **cómo se ha creado y cómo funciona** el **software** desarrollado en el proyecto, así como la **aplicación de usuario** obtenida. Por tanto este capítulo va a quedar dividido en dos subapartados, de los cuales el primero de ellos se dedica a explicar cómo se ha creado el software así como las partes del código más relevantes a la hora de crear la **Autoridad Certificadora** y los **certificados digitales**. Y el segundo se dedicará a explicar la **aplicación de usuario**, por tanto se explicará tanto lo que deberá realizar el personal de administración para poner en marcha este software como los pasos que deberá realizar el usuario para obtener su **certificado personal**.

4.1. Software creado.

Como se ha explicado anteriormente el lenguaje usado para desarrollar el código ha sido **php** junto con **html** dado que como sabemos estos dos lenguajes van de la mano a la hora de desarrollar **aplicaciones web**.

Así existen una serie de archivos con la extensión '**.php**' que se van a llamar unos a otros para ir realizando las labores requeridas por la aplicación. Por lo que a continuación se va a ir explicando los diferentes archivos creados y el contenido de las diferentes carpetas, así como los servicios que proporcionan y posteriormente en el siguiente subapartado veremos el resultado obtenido.

- **Index.php.** Este archivo simplemente sirve de enlace entre las diferentes opciones que se proporcionan. Empleando el método **switch** con sus diferentes **case** se proporciona la ruta hacia el archivo que se ha de seguir en función de la opción elegida.

- **./include/common.php.** En este archivo se incluyen las funciones básicas que se irán usando a lo largo del código, como por ejemplo la que chequea si ha ocurrido algún error con alguna función. Además también incluye el código **html** para determinar la distribución de las opciones en la página principal.

- **./modules/setup/intro.php.** Este archivo incluye el código de la página principal en **html**, en el que se incluye el primer formulario que el usuario (personal administrativo de la **UPCT**) tendrá que rellenar con los diferentes datos requeridos para crear la **Autoridad Certificadora**, así como una breve explicación del funcionamiento del software.

- **./modules/setup/create.php.** A esta parte del código se accede de manera transparente al usuario cuando éste en el primer formulario confirma para la creación de la **Autoridad Certificadora**. Por tanto, este es el punto donde se genera el **certificado de la AC**, de modo que el software genera un par de claves (una clave pública y una clave privada correspondiente), después se firma el par de claves, lo que crea un **certificado auto firmado**, como se ha explicado en el **capítulo 2** de la memoria.

Para ello se hace uso de una serie de funciones **OpenSSL**, que como hemos comentado anteriormente proporciona las **herramientas criptográficas** para la creación de **certificados digitales**. Así, se emplean las siguientes:

- ***openssl_pkey_new****($config)*. Esta función generará una **pareja de claves pública y privada** pasándole como parámetro **$config** el cual corresponde al archivo de configuración de **OpenSSL** (***openssl.conf***) en el que se especifica una serie de parámetros de configuración a la hora de crear las claves:
  - ***x509_extensions = usr_cert*** → selecciona qué extensiones deberían usarse cuando se crea un certificado ***x509***.
  - ***req_extensions = v3_req***→ selecciona qué extensiones deberían usarse cuando se crea una ***CSR***.

  Esta función devuelve un **identificador de recurso para la clave privada** si tuvo éxito o **FALSE** si se produjo un error. Dicho resultado se guardará en la variable **$privkey**.

- ***openssl_csr_new****($_REQUEST['dn'], $privkey)*. Esta función genera una nueva ***CSR*** (Certificate Signing Request - ***Petición de Firma de Certificado***) basada en la información proporcionada por ***dn***, el cual representa el Nombre Distinguido que se va a usar en el certificado (en este caso **UPCT**). Además se le pasa como parámetro la **clave privada** que ha sido generada anteriormente. Devuelve la ***Petición de Firma de Certificado*** la cual se guarda en la variable **$csr**.

- ***openssl_csr_export_to_file****($csr,'openssl/crypto/requests/solicitud.csr',FALSE)*. Exporta una ***CSR*** como texto ascii blindado en el archivo especificado por el segundo parámetro donde, en este caso, se especifica la ruta de dicho archivo. El último parámetro especificado afecta al nivel de detalle de la salida, como es FALSE la información legible adicional se incluye en la salida. Devuelve un valor booleano **TRUE** en caso de éxito o **FALSE** en caso de fallo. De esta manera se consigue conservar la ***Petición de Firma de Certificado*** por si fuera necesario refirmarla, como veremos en el siguiente subapartado.

- ***openssl_csr_sign****($csr, null, $privkey, 3650, array(), getSerial())*. Con esta función se firma una ***CSR*** con otro certificado (o **autofirmar**) y se genera un recurso de ***certificado x509*** desde la ***CSR*** dada. Como vemos se le pasa como primer parámetro **$csr** que corresponderá a la ***Petición de Firma de Certificado*** generada anteriormente. Como segundo parámetro se le pasa **NULL** lo cual quiere decir que el certificado generado será un **certificado auto firmado**. Posteriormente se pasa el parámetro **$privkey** que corresponde a la **clave privada** generada anteriormente. Como cuarto parámetro se pasa el **tiempo de validez** de dicho certificado, en este caso establecida en 10 años. Con el parámetro **array()** se ajusta la firma de la ***CSR***. Finalmente se especifica el **número de serie** opcional del certificado emitido, si no se especifica será 0 por defecto, en este caso se obtendrá con la función **getSerial()**. Por último

devuelve un recurso de **certificado x509** si se tuvo éxito, el cual se guarda en la variable **$sscert** o **FALSE** si falló.

- ***openssl_x509_export****($sscert, $myCert).* Exporta un certificado como una cadena, siendo **$sscert** el certificado devuelto por la función anterior y **$myCert** la cadena que contendrá el **PEM**. Una vez que obtenemos **$myCert** escribiremos esta variable en el archivo ***cacert.pem*** guardando así el certificado de la **Autoridad Certificadora** que después el usuario deberá instalar en su navegador para convertir a esta **Autoridad Certificadora** en una confiable. Esta función devuelve **TRUE** en caso de éxito y **FALSE** en caso de error.

- ***openssl_pkey_export****($privkey, $myKey, $passPhrase).* Con esta función conseguimos exportar la clave **$privkey** como una cadena **PEM** codificada y se almacena en **$myKey** (que es pasado por referencia), además se pasa como tercer parámetro **$passPhrase** siendo éste el **password** que se solicita al usuario y el cual es usado para **codificar la clave**. Del mismo modo que con el certificado ésta clave es guardada una vez que ha sido exportada en el archivo ***cakey.pem****.* Devuelve **TRUE** en caso de éxito o **FALSE** en caso de error.

Por tanto cuando se ejecuta este código obtenemos el certificado de la **Autoridad Certificadora**, así el usuario final de este software, en este caso los empleados de la **UPCT** que deseen conseguir su propio **certificado personal**, deberán instalarlo para establecer esta **Autoridad de la UPCT** como autoridad de confianza y que por tanto pueda expedir un **certificado digital**.

- **./modules/main/welcome.php.** Una vez que ha sido creada la **Autoridad Certificadora** aparecerá la página creada por el código de este archivo en **html**. Esta es la primera página que aparecerá una vez que el usuario, es decir, el empleado en cuestión, haga uso de este software. En ella el usuario tendrá dos opciones: podrá instalar la **Autoridad Certificadora UPCT** como una de **confianza** o bien **solicitar** su **certificado**. Posteriormente se explicarán cada una de estas dos opciones.

- **./modules/main/trust.php.** Este archivo contiene el código para tratar la solicitud de instalación del certificado de la **Autoridad Certificador** puesto que contiene la siguiente orden: ***application/x-x509-ca-cert*** la cual llevará a cabo la labor de instalar el certificado creado anteriormente ***cacert.pem***. Se ejecutará este código cuando el usuario seleccione la opción de '**instalar la Autoridad Certificadora'**.

- **./modules/main/about.php.** Esta opción de la página principal únicamente contiene información acerca del software creado, así como el enlace para descargar la LICENCIA y otro para descargar el propio software creado ya que se ha desarrollado un **software libre**.

- **./modules/main/help.php.** Simplemente se ha creado una página de ayuda en la que se explican los pasos a seguir por parte del personal administrativo para conseguir un correcto funcionamiento del software y por parte del usuario para obtener su propio **certificado digital.**

- **./modules/apply/emailConfirm.php.** Si en la página principal de ***'welcome.php'*** el usuario selecciona la opción de **'solicitar su certificado personal'** automáticamente se ejecutará el código correspondiente a este archivo. Por tanto, aquí el usuario tendrá que introducir su **usuario** y **contraseña** de la **UPCT**, puesto que antes de obtener su certificado el usuario deberá **autenticarse** como empleado de dicha entidad. Aunque esta parte se dejará como trabajos futuros y para poder continuar con los servicios que se ofrecen se ha creado una base de datos de manera local de momento para poder así comprobar la identidad del usuario.

- **./modules/apply/issueCert.php.** Una vez que el usuario ha introducido su usuario y contraseña en este archivo se **comprueba** que efectivamente es empleado de la **UPCT** de manera transparente al usuario. Una vez hecha dicha comprobación se crea otro formulario en el que el empleado deberá introducir unos **datos** adicionales y otros que ya se habían introducido en el primer formulario solicitado. El principal **dato** que el usuario deberá introducir y sin el cual no se podrá **solicitar** su **certificado cliente** es el **password** que se empleará para **codificar** la **clave privada** como veremos a continuación.

- **./modules/apply/signCert.php.** Una vez que se han rellenado los datos del formulario previo es en el código de este archivo donde, de manera transparente al usuario, se crea el **certificado personal** del cliente en cuestión, el cual en este caso tendrá una extensión '***.p12***' siendo este formato el más usado por los navegadores actualmente. Para la creación de este **certificado** se han seguido los siguientes pasos:
    - En primer lugar guardaremos el **password** solicitado en el formulario en un archivo puesto que cuando este empleado desee renovar su **certificado cliente** bien porque ha expirado, o bien porque ha sido revocado, se solicitará este **password** y será necesario comprobar que es el mismo.

    - ***openssl_x509_read****($certData)*. Con esta función se analiza el certificado ***x509*** guardado en la variable **$certData** en la cual se carga el contenido del fichero generado al crear la **Autoridad Certificadora**: ***cacert.pem***, y se devuelve un identificador de recurso para dicho **certificado** el cual se guarda en **$caCert**.

    - ***openssl_get_privatekey****($privKey,$config['passPhrase']).* Del mismo modo que ocurre con el certificado ***cacert.pem*** también se deberá cargar la **clave privada** de la **Autoridad Certificadora**, para lo cual se emplea esta función donde **$privKey** es la clave leída del archivo ***cakey.pem*** y ***passPhrase*** será el **password** que se emplea para codificar la clave privada de la **Autoridad Certificadora.** Es decir esta función analiza la clave **$privKey** y la prepara para

usarla con otras funciones de manera que devuelve un identificador de clave positivo si se tuvo éxito, el cual se guarda en **$caKey** o **FALSE** si se produjo un error.

- ***openssl_pkey_new()*.**Con esta función, la cual se ha empleado anteriormente, se genera un nuevo par de claves **pública** y **privada**, en este caso empleadas para codificar el **certificado cliente**. El recurso para la clave privada que se devuelve se guarda en la variable **$res**.

- ***openssl_pkey_export**($res, $privKeyClient).*Esta función, como hemos comentado también anteriormente exporta la clave **privada $res** como una cadena **PEM** codificada y la almacena en **$privKeyClient** (que es pasado por referencia).

- ***openssl_pkey_get_details** ($res).* Como su propio nombre indica esta función devolverá una matriz con los **detalles de la clave** si se tuvo éxito o **FALSE** si falló. La matriz devuelta tiene indexados los **bits** (número de bits), **key** (representación de cadena de la **clave pública**) y **type** (el tipo de la clave que es OPENSSL_KEYTYPE_RSA, OPENSSL_KEYTYPE_DSA, OPENSSL_KEYTYPE_DH, OPENSSL_KEYTYPE_EC o -1 significa desconocido). El resultado de esta función se guardará en la variable **$pubKeyClient**, de manera que empleando la siguiente sentencia: ***$pubKeyClient=$pubKeyClient["key"]*** guardará finalmente únicamente la **clave pública**.

- ***openssl_pkey_export**($privKeyClient, $pkeyout, $_REQUEST['clave']).* Anteriormente se ha exportado la clave privada a **$privKetClient** y es con esta sentencia del código donde ésta se exporta a **$pkeyout** codificada ahora sí con el **password** (en este caso llamado **'clave'**) solicitado al usuario en el formulario creado para solicitar su **certificado cliente**.

- ***openssl_csr_new**($_REQUEST['dn'], $privKeyClient).* Como en el caso anterior se crea una nueva ***CSR*** en este caso pasando como parámetro la **clave privada del cliente**. El resultado se guarda en la variable **$csrCert**.

- ***openssl_csr_export_to_file**($csrCert,'openssl/crypto/requests/solicitudcert.csr' ,FALSE)*. Al igual que antes esta ***Petición de Firma de Certificado*** queda almacenada en un archivo.

- ***openssl_csr_sign**($csrCert, $caCert, $caKey, 365*2, $config, getSerial()).* Finalmente se firma esta ***Petición de Firma de Certificado*** pero en este caso con el certificado **$caCert** el cual corresponde al de la **Autoridad Certificadora** creado anteriormente. Como vemos en este caso el tiempo de vigencia del **certificado cliente** será de dos años. El valor devuelto que será el recurso del certificado ***x509*** si se tuvo éxito, quedará guardado en **$signedCert**.

- ***openssl_pkcs12_export****($signedCert,$certout,$privKeyClient,$_REQUEST['clave'])*. Como hemos mencionado anteriormente la extensión del **certificado cliente** será ***'.p12'***, así esta función exporta el certificado ***x509*** contenido en **$signedCert** en una cadena nombrada por **$certout** en un formato de archivo ***PKCS#12***, siendo **$privKeyClient** el componente **clave privada** del archivo ***PKCS#12*** y **'clave'** la contraseña de codificación para **desbloquear** el archivo ***PKCS#12***. Es decir, finalmente el **certificado cliente** lo tendremos en la variable **$certout**.

- Finalmente ya tenemos el **certificado cliente** guardado en **$certout** como la **clave privada** del certificado cliente guardada en **$pkeyout**. Dichos datos serán almacenados en los archivos ***certClient.p12*** y ***keyClient.key*** respectivamente.

- Una vez que se ha generado el **certificado cliente**, como hemos dicho, de manera transparente al usuario, aparecerá una página donde el usuario podrá instalar la **Autoridad Certificadora** (de nuevo, por si no lo ha hecho anteriormente) e instalar su nuevo **certificado personal**, siguiendo los pasos guiados por su navegador web.

- **./modules/admin/options.php.** El código de este archivo desarrollado en **html** va a corresponder a la página que aparecerá cuando el usuario o el personal administrativo de la **UPCT** seleccione la opción '**Administración**' en la página principal. Aquí, si el que accede es el personal administrativo podrá: **renovar el certificado de la Autoridad Certificadora,** o bien **solicitar que se vuelva a firmar el *CSR*** del certificado de la **Autoridad Certificadora**. En cambio si el que accede es el empleado podrá o bien de nuevo **instalar el certificado de la Autoridad Certificadora** (se proporciona otro enlace) o bien **renovar su certificado personal.** A continuación se va a explicar el código de cada uno de estos servicios.

- **./modules/admin/IntroducirPassw.php** y **./modules/admin/IntroducirPasswClient.php.** Si el usuario (ya sea empleado o personal administrativo) solicita renovar cualquiera de los dos certificados el código lo llevará al archivo ***IntroducirPassw.php*** si lo que quiere es renovar el **certificado de la Autoridad Certificadora** donde deberá introducir el **password** que ha usado para codificar dicho certificado o al archivo ***IntroducirPasswclient.php*** si lo que desea es renovar el **certificado cliente** donde introducirá el **password** usado para codificar tal certificado. Si cualquiera de estas dos contraseñas son erróneas se redirige al usuario para que vuelva a introducirlas correctamente.

- **./modules/admin/renewCert.php.** Este es el código que se ejecuta cuando el usuario solicita **renovar** el certificado de la **Autoridad Certificadora**. Aquí simplemente se seguirán los mismos pasos que cuando se **crea** este **certificado** pero en lugar de crear una nueva clave privada se recuperará la que se ha creado anteriormente y que se

había guardado en el fichero ***cakey.pem***. Por lo que al final obtendremos un **certificado renovado** cuya duración en este caso será de cinco años.

- **./modules/admin/renweCertClient.php**. Al igual que antes este es el código que se ejecutará cuando el usuario solicite **renovar** el **certificado cliente** y de la misma manera en lugar de crear un nuevo par de claves, éstas se recuperarán (puesto que habían sido guardadas en ***keyClient.key***) pudiendo así renovar el **certificado personal** de la misma manera que se crea al principio y siguiendo los mismos pasos.

- **./modules/admin/certSign.php.** En esta parte del código el usuario podrá solicitar que se vuelva a firmar la ***Petición de Firma de Certificado*** cargando ésta (únicamente la de la **Autoridad Certificadora**) dado que ha sido guardad con anterioridad, obteniendo así el nuevo **certificado** pero que en este caso no estará **auto firmado**, si no que se firmará con el **certificado creado inicialmente**.

- **./css/basic.php.** Este archivo contiene la **hoja de estilo** (**css**), es decir, el código que contiene las líneas que se ocupan de los aspectos de **formato** y de presentación de los contenidos.

- **./certcontrol.** Esta carpeta contiene la biblioteca binaria ***xenroll.dll*** y contiene el **control** de inscripción de certificados.

- **./config.** En esta carpeta se guarda la base de datos que se usa para la **autenticación**, además aquí se guardará el archivo de configuración '***configuration.php***' que contendrá la información que se requiere al usuario a la hora de crear la **Autoridad Certificadora** y que posteriormente será necesaria a la hora de **crear** el **certificado cliente**.

- **./images.** Aquí se guardan las imágenes que irán apareciendo en las diferentes páginas como por ejemplo el **logo** de la **UPCT**.

- **./openssl/openssl.conf.** Como ya se ha mencionado antes este archivo es el archivo de configuración **OpenSSL** usado para la generación de la solicitud de certificado, donde se indican los diferentes parámetros de configuración que se deberán a tener en cuenta a la hora de crear los **certificados**.

- **./openssl/crypto/cacerts.** En esta carpeta es donde quedan almacenados ambos **certificados**.

- **./openssl/crypto/keys.** Del mismo modo en esta carpeta quedan almacenadas las **claves**.

- **./openssl/crypto/requests.** Esta carpeta ha sido creada para almacenar las ***Peticiones de Firma de Certificado***.

Ahora que ya sabemos cómo y para qué sirve cada parte del código que se ha desarrollado vamos a proceder a explicar la **aplicación de usuario**, es decir, el aspecto y formato que toma el código explicado anteriormente y que es lo que en definitiva el usuario va a ver y por tanto es con lo que va a tener que interactuar.

4.2. Aplicación de usuario.

Por tanto se va a proceder a explicar, en primer lugar, cada uno de los pasos que el personal administrativo de la **UPCT** va a tener que seguir para crear esta **Autoridad Certificadora**. Y en segundo lugar los pasos que deberán seguir los empleados para solicitar su **certificado personal**. Por tanto se va a mostrar la apariencia que toman cada uno de los archivos del código creado y explicado anteriormente.

1. En **primer lugar**, como hemos dicho, el personal administrativo deberá poner en marcha el software creado para obtener así la **Autoridad Certificadora** y que posteriormente los empleados puedan solicitar su **certificado personal**. Así, el usuario se encontrará con la página de **creación** y **configuración** de la **Autoridad Certificadora** donde deberá rellenar los datos requeridos.
   Esta página corresponde al código creado en el archivo ***'intro.php'*** y como vemos en la siguiente imagen en ella se explica el **servicio** que pretende ofrecer este software y se proporciona el **formulario** donde se debe introducir entre otras cosas la **contraseña** que posteriormente se usará para codificar la **clave privada** de esta **Autoridad Certificadora**.

Figura 4.1. Página de inicio.

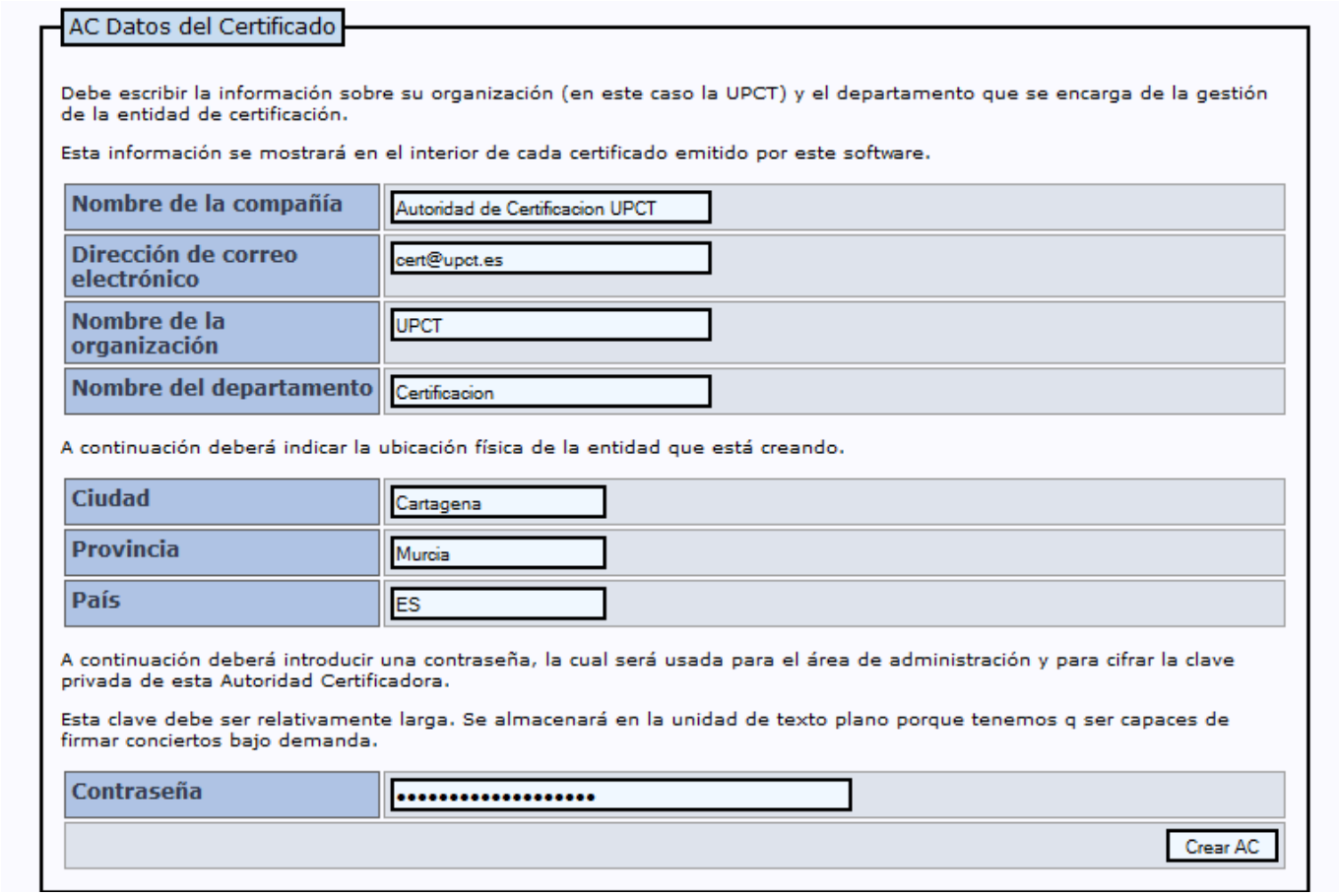

Figura 4.2. Página de inicio. Formulario.

2. Una vez que se rellena el formulario se ejecuta, de manera transparente al usuario, el código del archivo ***'create.php'*** en el cual, como ya se ha explicado, se crea el **certificado auto firmado** de la **Autoridad Certificadora** que posteriormente el empleado en cuestión deberá **instalar**, convirtiéndose así esta **Autoridad Certificadora** en una **confiable** para el usuario. Así una vez que se ejecuta este código aparece la siguiente imagen donde haciendo clic en el enlace del final de la página nos lleva al siguiente paso.

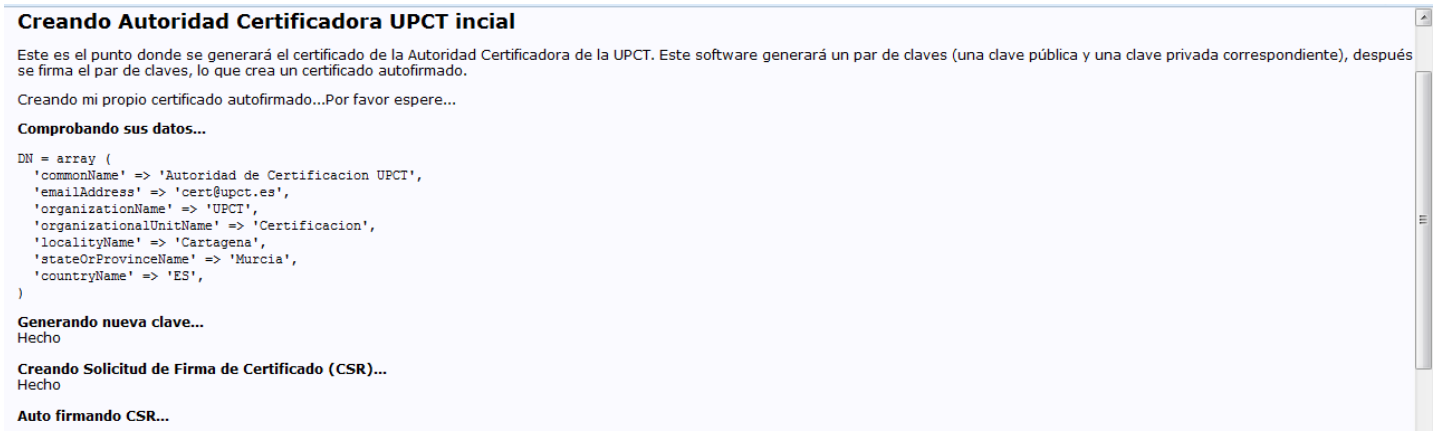

Figura 4.3. Creando Autoridad Certificadora UPCT inicial.

Como vemos en la siguiente imagen, se muestra de manera codificada tanto el certificado como la clave.

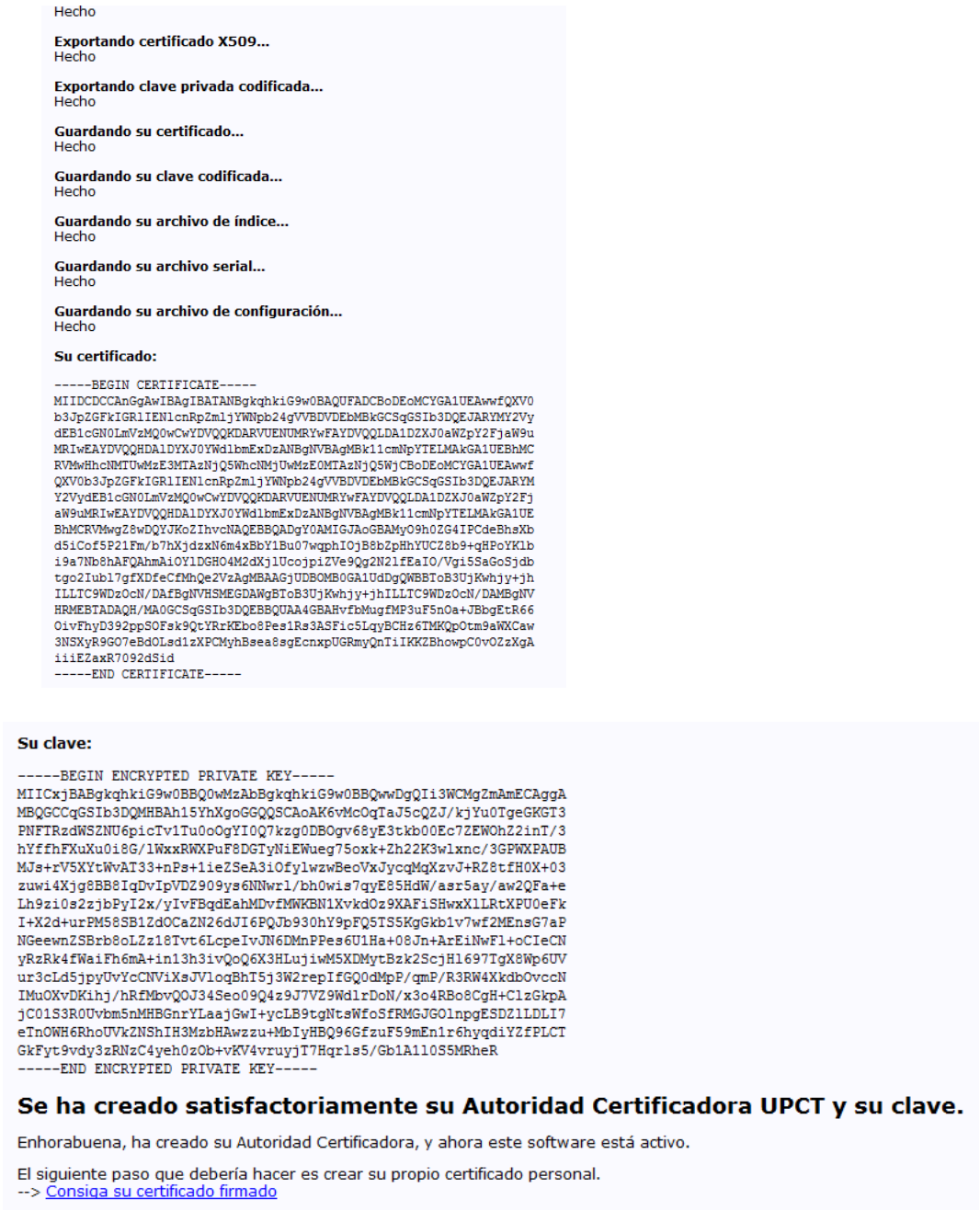

Figura 4.4. Creando Autoridad Certificadora UPCT inicial. Certificado y clave codificadas.

Una vez que esté creado el certificado se deberá hacer clic en el enlace ***'Consiga su certificado firmado'***, y nos aparecerá así la página principal que verá el empleado cuyo contenido se explica a continuación.

3. Una vez que se ha creado dicha **Autoridad**, esta será la primera página que aparezca para el empleado en cuestión y que corresponde al código del archivo ***'welcome.php'*** donde se ofrecen dos posibilidades al usuario, o bien **solicita** su propio **certificado personal** o bien **instala** la **Autoridad Certificadora** como una de **confianza** (siendo ésta segunda opción necesaria para que el **certificado** tenga validez).

Figura 4.5. Bienvenido a la Autoridad Certificadora de la UPCT.

3.1. Si el usuario hace clic en el enlace de **solicitar** su propio **certificado personal** automáticamente aparecerá la página donde deberá **autenticarse** como empleado de la **UPCT** (código correspondiente al archivo ***'emailConfirm.php'***). Si su usuario o contraseña no son válidos aparecerá una página de error y se dará oportunidad al usuario de volver a intentarlo.

Figura 4.6. Página de confirmación de Identidad.

Figura 4.7. Página de error.

Figura 4.8. Página de confirmación de Identidad. Usuario y Contraseña.

Una vez que se **autentica** aparecerá otro formulario similar al del inicio pero que ahora deberá rellenar el empleado, donde deberá introducir además de sus datos personales y del grado de seguridad que se otorgará al **certificado personal**, una **contraseña** la cual se empleará para codificar la **clave privada** y que posteriormente será solicitada para instalar tal certificado (el código de esta página corresponde al archivo ***'issueCert.php'***). Del mismo modo que antes habrá una página de **error** si el usuario no introduce su **correo electrónico** o la **contraseña** solicitándole, de nuevo, que lo vuelva a intentar.

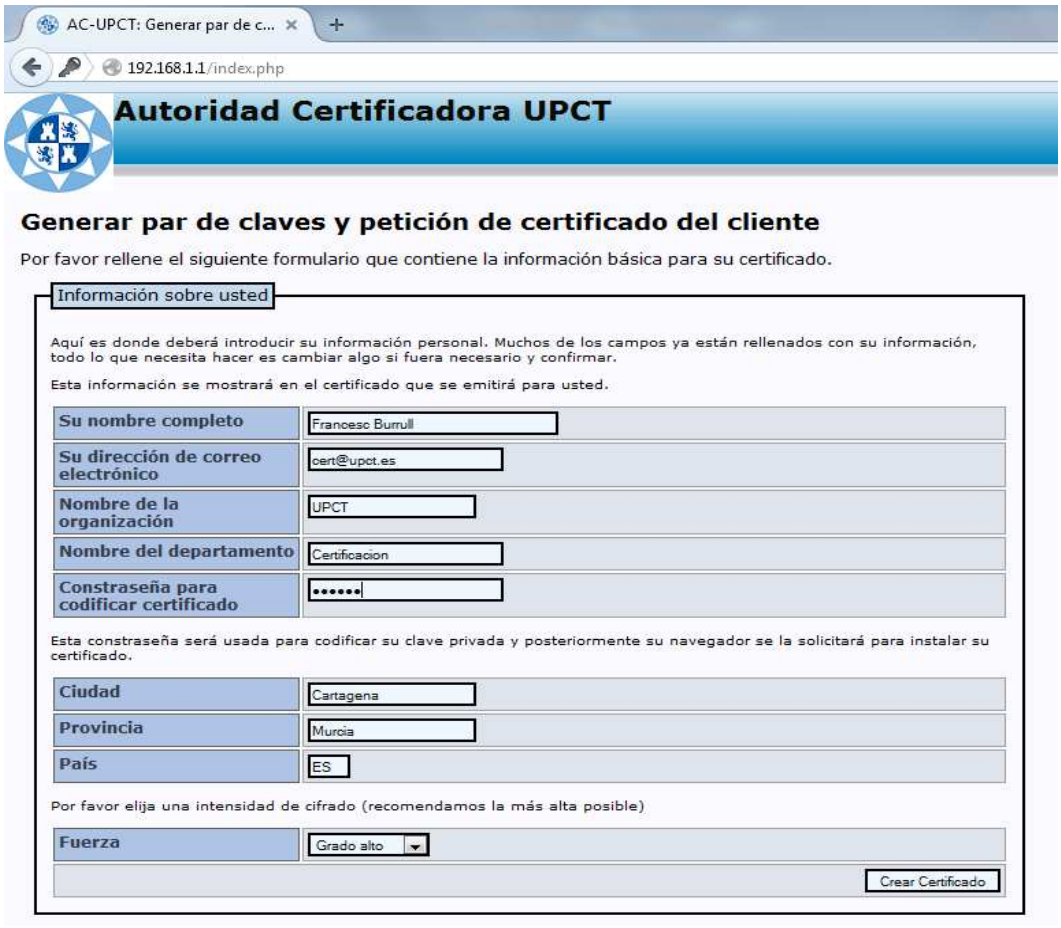

Figura 4.9. Formulario Cliente.

Figura 4.10. Página de error.

Una vez que el usuario selecciona el botón ***Crear Certificado*** aparece la siguiente ventana que indica que el **par de claves** se está generando.

Figura 4.11. Generación de claves.

Finalmente una vez que se ha creado el **certificado personal** aparecerá la página cuyo código corresponde a ***'signCert.php'***, la cual además, de manera transparente es la que se encarga de crear tal certificado, como ya se ha explicado en el desarrollo del código. En esta página el usuario podrá o bien **instalar** el **certificado personal** o bien **instalar** el **certificado** de la **Autoridad Certificadora** (al igual que en la página principal), lo cual, como hemos dicho, será necesario para que nuestro **certificado personal** tenga validez.

Figura 4.12. Instalación del Certificado Cliente.

3.1.1. Si el usuario selecciona, una vez que se ha instalado el **certificado de la AC** (en el punto 3.2 se explica cómo), la opción de instalar su **certificado personal** (enlace ***'Instálelo ahora'***), aparecerá la siguiente ventana donde, como vemos se da la opción de descargar dicho **certificado**. Si seleccionamos la opción de ***Guardar archivo*** y aceptamos se descargará el certificado y el usuario podrá instalarlo siguiendo los pasos

que se describen a continuación.

Figura 4.13. Descargar Certificado.

Como vemos en la **Figura 4.13.** y además hemos mencionado en el subapartado 4.1 donde se explica el código desarrollado, el certificado cliente se crea con la extensión ***.p12*** lo cual simplemente significa que se crea una copia de seguridad con la clave privada de un certificado (exportado desde Firefox). En el **anexo 2** se explica más en detalle los diferentes formatos y extensiones de los certificados.

Una vez que se ha descargado el certificado en **primer lugar** deberá importarlo, para ello deberá:

- Seleccionar la opción de configuración en la barra de herramientas y seleccionar ahí la opción ***'Opciones'*** como vemos en la siguiente figura.

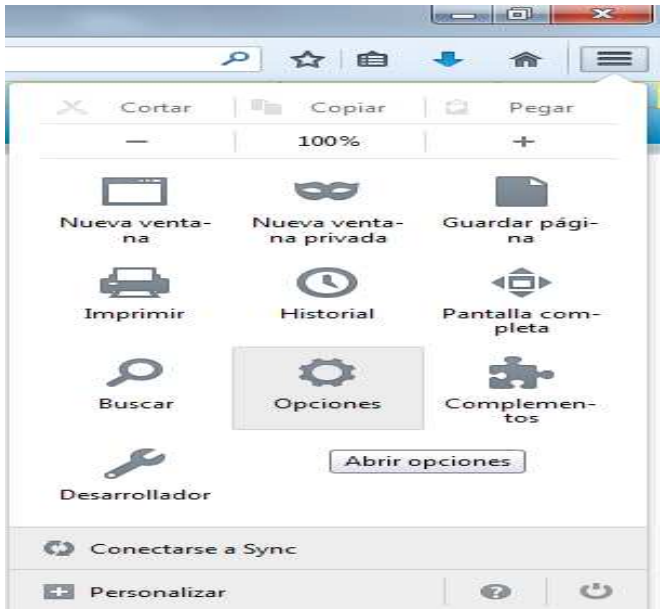

Figura 4.14. Instalación del certificado cliente (1).

- A continuación se selecciona la pestaña ***'Avanzado'***, dentro de ésta la pestaña ***'Certificados'*** y ahí se pulsa el botón ***'Ver certificados'***.

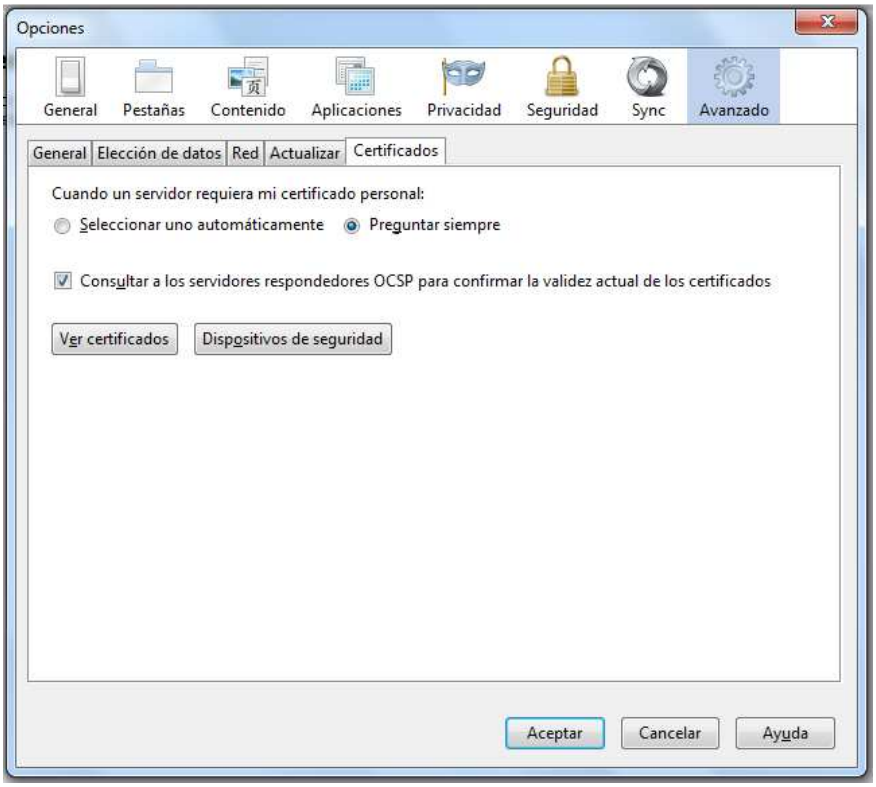

Figura 4.15. Instalación del certificado cliente (2).

- Una vez que entramos ahí, como vemos en la siguiente imagen hay diferentes pestañas, si seleccionamos la pestaña ***'Sus certificados'*** podemos ver los certificados que tenemos instalados, en este caso de momento no tenemos ningún certificado instalado, para instalarlo le damos al botón ***'Importar'***.

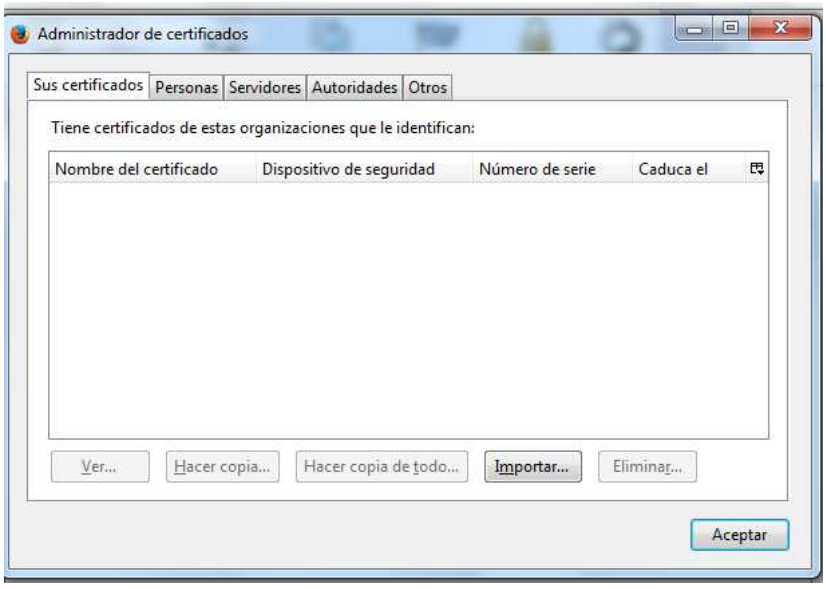

Figura 4.16. Instalación del certificado cliente (3).

- A continuación seleccionamos el certificado que deseamos importar, en este caso ***'certClient.p12'***.

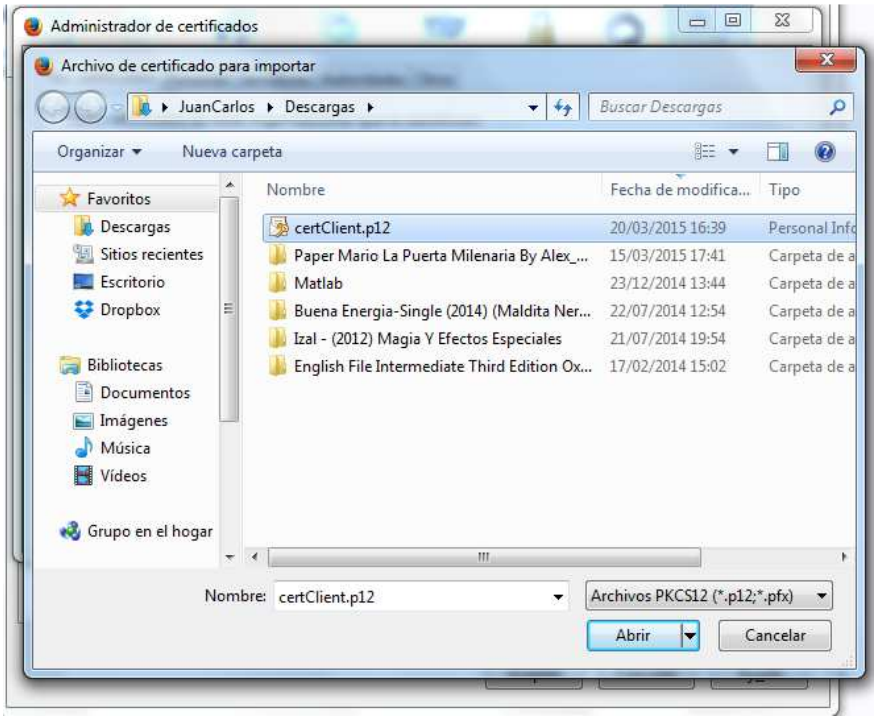

Figura 4.17. Instalación del certificado cliente (4).

- Una vez que pulsamos el botón ***'Abrir'***, el navegador nos solicitará la contraseña que se ha empleado para cifrar la clave (la cual ha introducido el usuario en el formulario inicial) para llevar a cabo la activación del certificado, por lo que deberemos introducirla, como vemos en la siguiente imagen.

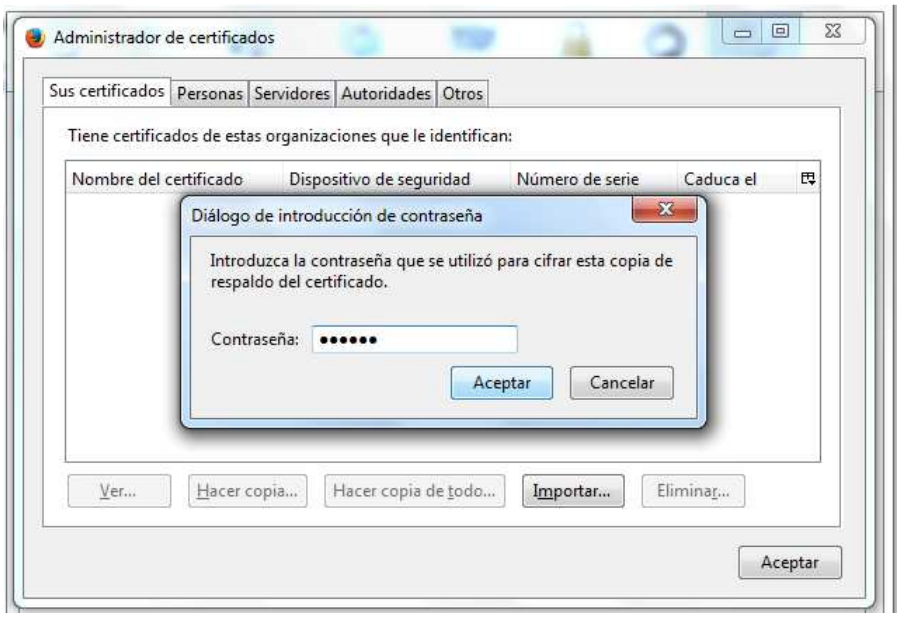

Figura 4.18. Instalación del certificado cliente (5).

- Una vez que se pulsa el botón ***'Aceptar'*** apareceré el siguiente mensaje y por tanto significará que ya se ha importado el certificado y como vemos en la **Figura 4.20** ya aparecerá éste en nuestra lista de certificados instalados**.**

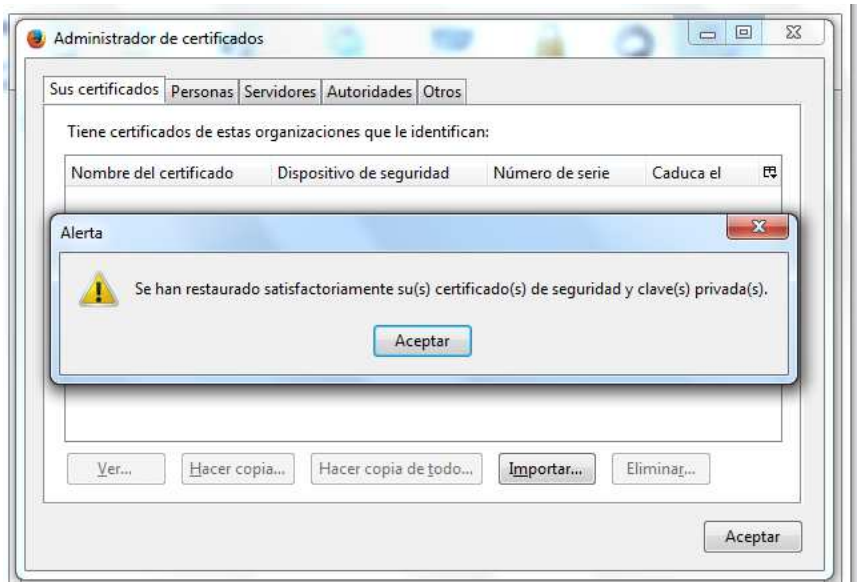

Figura 4.19. Instalación del certificado cliente (6).

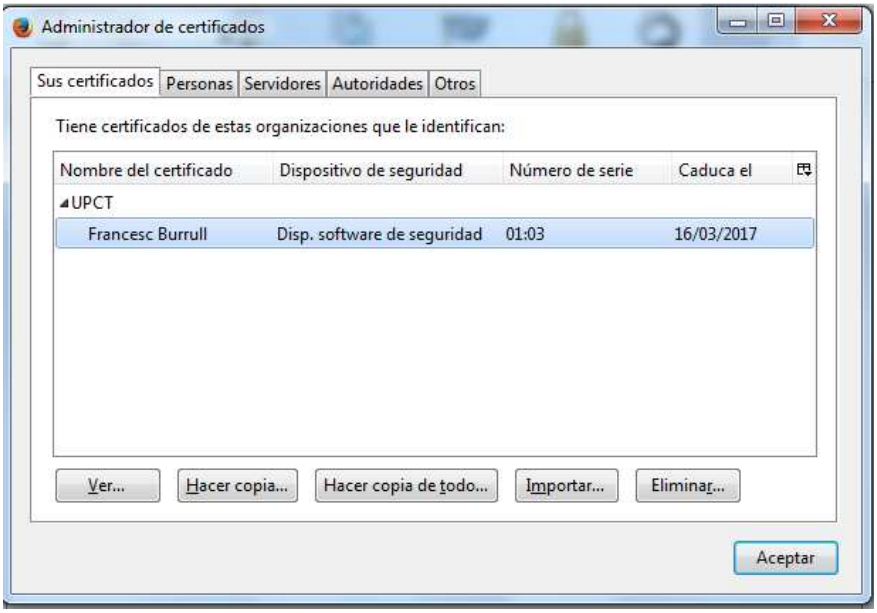

Figura 4.20. Instalación del certificado cliente (7).

- Así, si pulsamos el botón ***'Ver…'*** podremos visualizar los diferentes parámetros y características del **certificado cliente instalado**.
  Como vemos en la imagen podemos ver sus diferentes parámetros de manera más general (en la pestaña ***'General'***) o de forma más detallada (pestaña ***'Detalles'***).
  Al haber instalado ya la **Autoridad Certificadora de la UPCT** como un servidor de **confianza** vemos que en el certificado aparece que éste ha sido **verificado** (señalado con un círculo en la imagen), si no se hubiera instalado la **Autoridad Certificadora** aparecería que este certificado no ha podido ser verificado pero se instalaría de igual modo aunque no tendría validez a la hora de hacer uso del mismo.

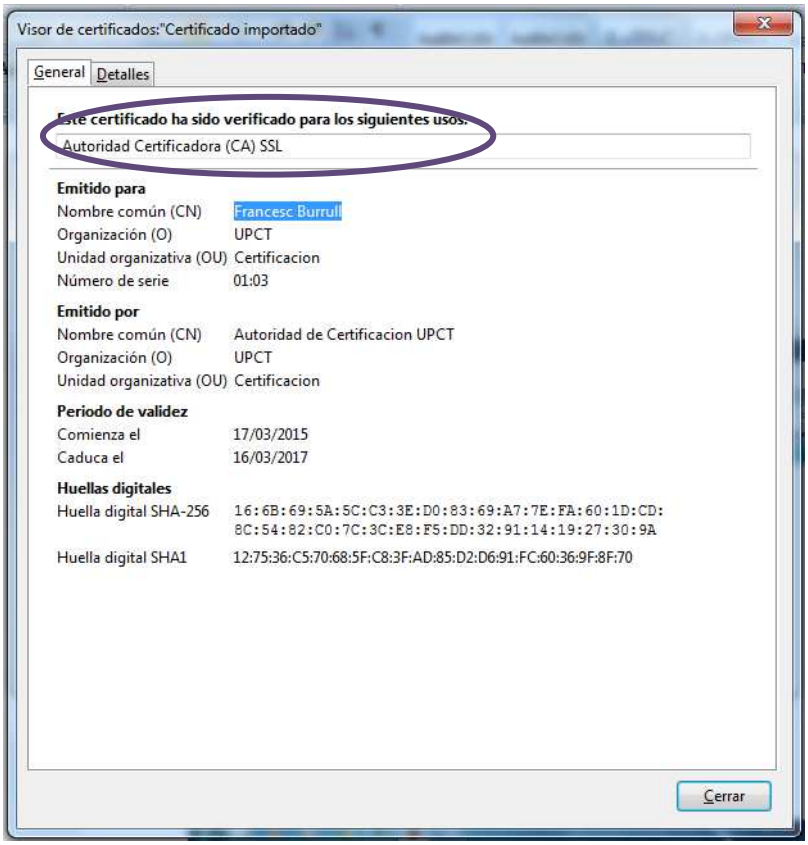

Figura 4.21. Características del certificado cliente instalado (1).

En cuanto a los detalles que se muestran en esta pestaña (en el **anexo 2** se explican con más detalle), son los siguientes:

- Persona para la que ha sido emitido el certificado cliente en cuestión, con los datos que ha proporcionado el usuario así como el número de serie que se da al certificado.
- Entidad que ha emitido el certificado cliente, en este caso la entidad creada: **Autoridad de Certificación UPCT**.
- Periodo de validez del certificado, es decir, el periodo durante el cual el certificado será válido para usarlo por el empleado. En este caso el certificado ha sido creado para que ese periodo sea de dos años.

- Y finalmente los valores que toman las huellas digitales (o funciones HASH) una vez que se realiza la operación matemática sobre el conjunto de datos, cuyo funcionamiento se explica en el **capítulo 2** (como hemos visto sirven para realizar la firma digital), siendo el **HASH SHA1** un tipo de los múltiples que hay de huella digital que corresponde con la segunda versión del sistema y que produce una salida resumen de 160 bits (20 bytes) de un mensaje que puede tener un tamaño máximo de 264 bits, y se basa en principios similares a los usados por el profesor Ronald L. Rivest del MIT en el diseño de los algoritmos de resumen de mensaje MD4 y MD5. Y siendo **SHA256** una de las funciones de la tercera versión del sistema **SHA2** cuya diferencia principal con **SHA1** radica en su diseño y que los rangos de salida han sido incrementados.

  Se emplean estas dos huellas digitales dado que por el momento se está realizando la transición de la segunda versión (**SHA1**) a la que se le han descubierto algunos fallos a la tercera (**SHA2**) más concretamente a **SHA256**. Si bien el Foro de Navegadores y Autoridades de Certificación aún no ha especificado el uso de **SHA256** en sus Requisitos básicos, Microsoft está dirigiendo al sector hacia la fecha de enero de 2017, cuando dejará de confiar en todos los certificados SHA-1 emitidos a partir de una raíz pública.

Del mismo modo en la pestaña ***'Detalles'*** se muestran estas mismas características y algunas más detalladas:

- Versión del certificado, en este caso la **versión 3** como se ha explicado en el **capítulo 2**.

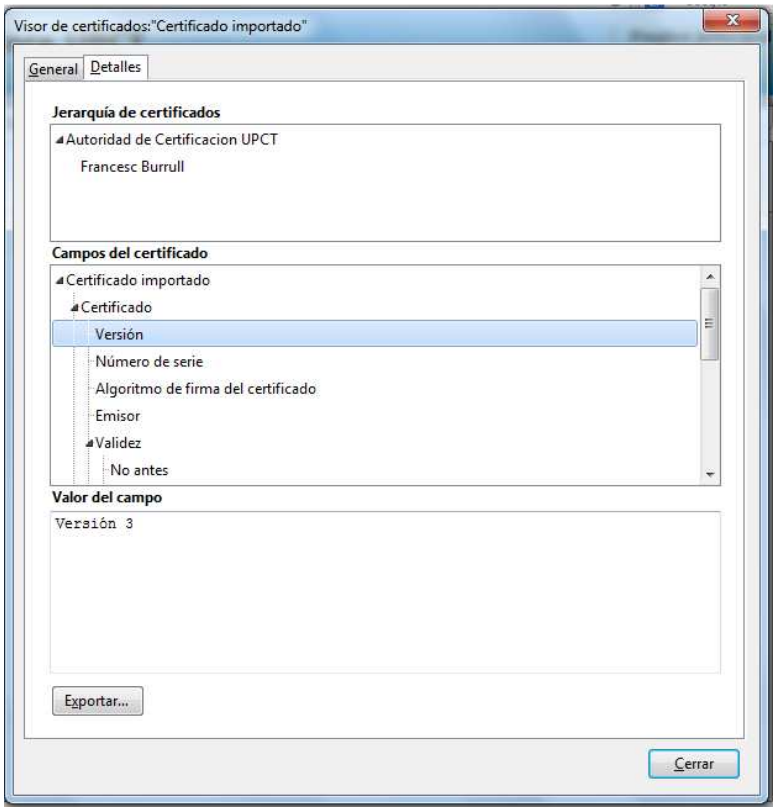

Figura 4.22. Características del certificado cliente instalado (2). Versión del certificado.

- El algoritmo de firma del certificado que como vemos en la siguiente imagen emplea la función **HASH SHA1** (como ya se ha explicado) y el algoritmo de cifrado **RSA** explicado con detalle en el **capítulo 2**.
  Como vemos en la imagen se emplea el **PKCS#1** el cual define el formato del cifrado **RSA**, siendo **PKCS** un grupo de estándares de criptografía de clave pública concebidos y publicados por los laboratorios de RSA en California. A RSA Security se le asignaron los derechos de licenciamiento para la patente de algoritmo de clave asimétrica RSA y adquirió los derechos de licenciamiento para muchas otras patentes de claves.

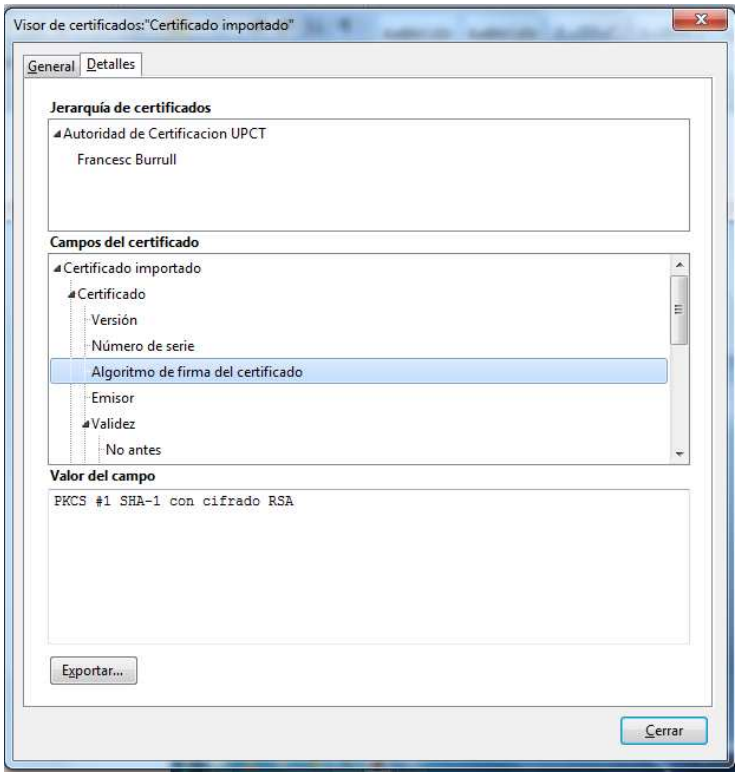

Figura 4.23. Características del certificado cliente instalado (3). Algoritmo de firma.

- Asunto, donde se muestran de nuevo los datos proporcionados por el empleado en cuestión.

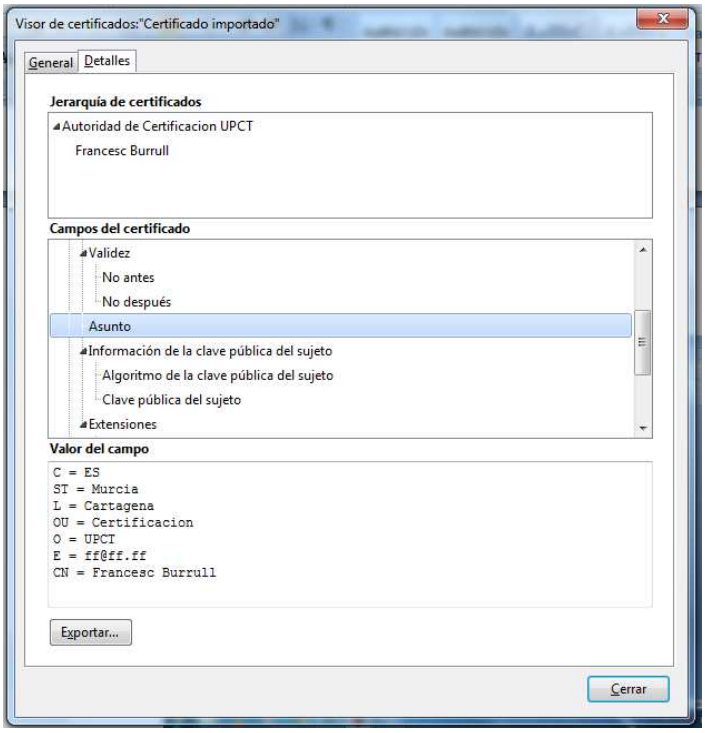

Figura 4.24. Características del certificado cliente instalado (4). Poseedor del certificado.

- Algoritmo de la clave pública del sujeto, que como hemos dicho es con cifrado **RSA**.

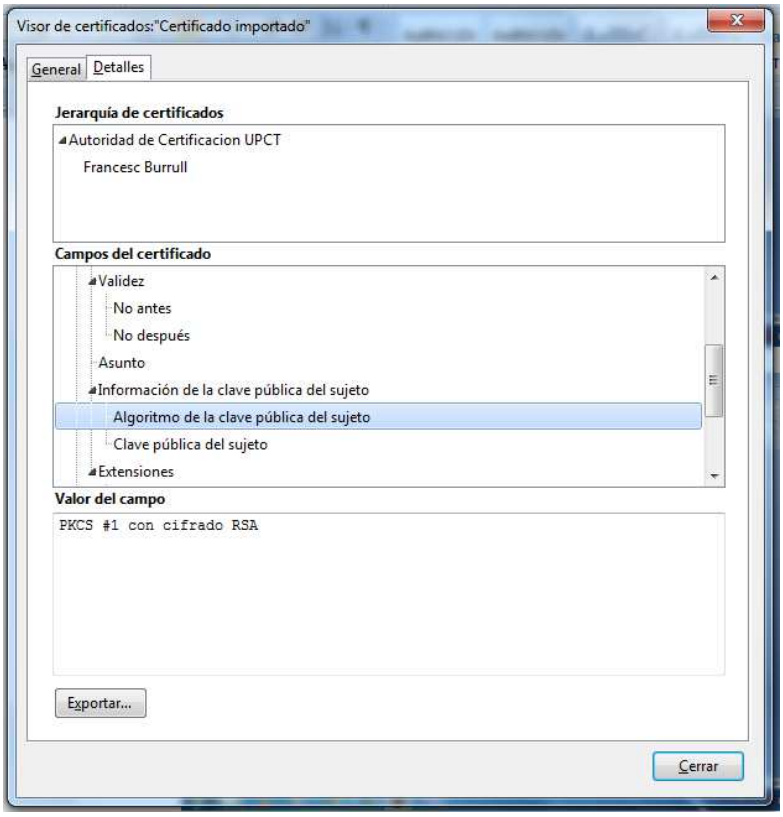

Figura 4.25. Características del certificado cliente instalado (4). Algoritmo de clave pública.

- Clave pública del sujeto puesto que puede ser conocida por cualquiera.

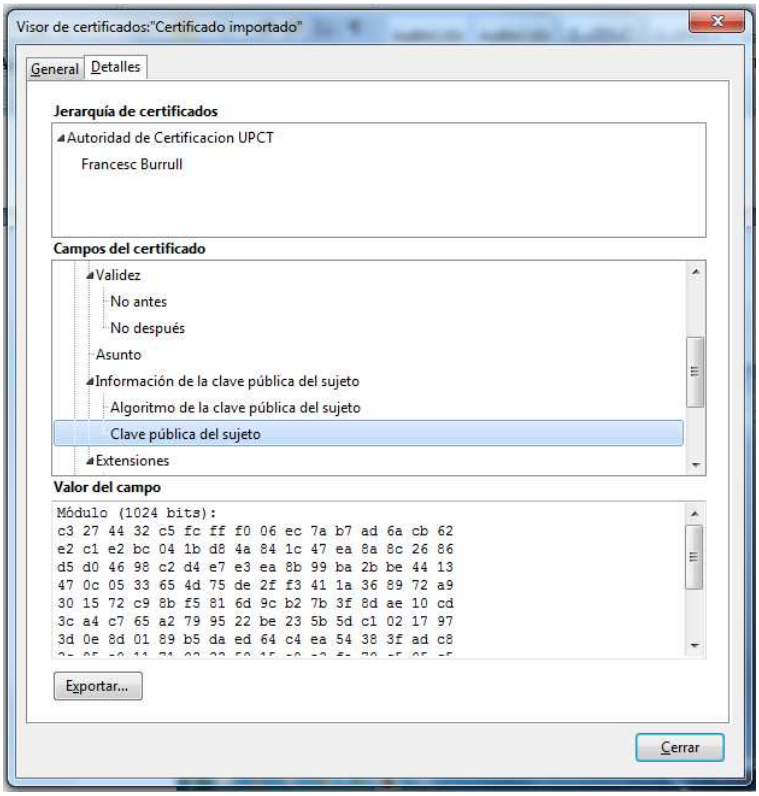

Figura 4.26. Características del certificado cliente instalado (5). Clave pública del sujeto.

- En el campo extensiones se muestran algunas características que se han añadido pero que no son imprescindibles, como por ejemplo el identificador de clave de asunto de certificado (huella digital obtenida de la clave pública del certificado), el identificador de la clave de la **Autoridad Certificadora** (donde se muestran los datos de esta) o las restricciones básicas del certificado donde se mostrarían éstas si el certificado emitido las tuviera.

- Valor de la firma del certificado y tamaño de la misma.

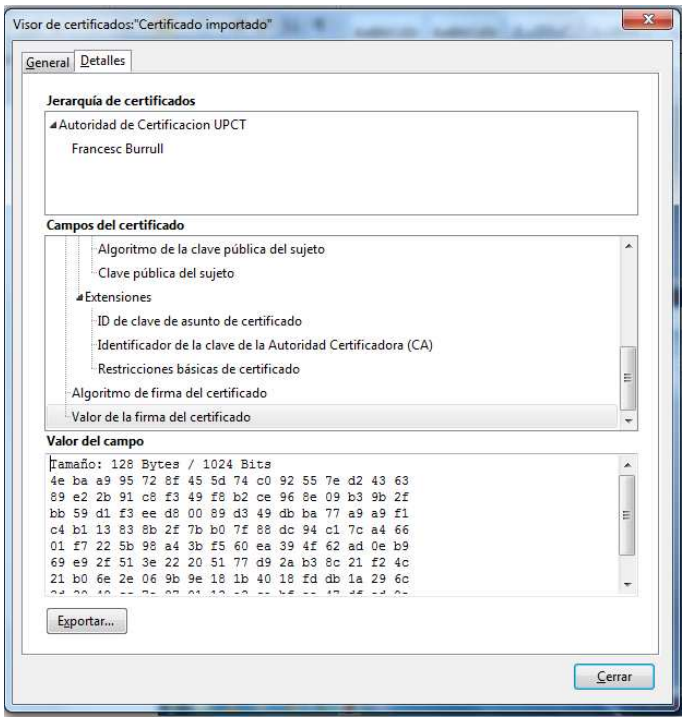

Figura 4.27. Características del certificado cliente instalado (6). Valor de la firma del certificado.

3.2. Por otro lado si el usuario selecciona el enlace ***'AC'*** (página ***'welcome.php'***) o bien ***'instalar esta AC como un servidor de Autoridad Certificadora de confianza'*** como vemos en la **Figura 4.12.,** aparecerá la siguiente imagen donde deberá seleccionar la opción correspondiente de manera que se indique que se confía en dicha **autoridad**.

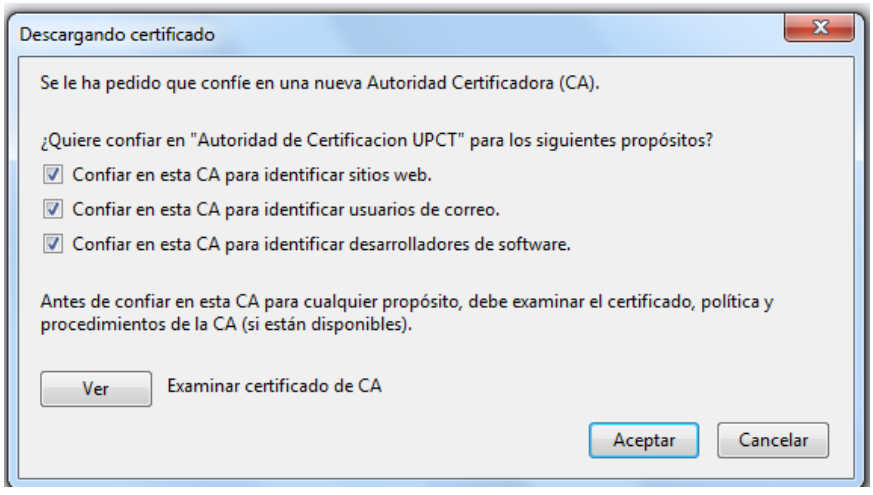

Figura 4.28. Certificado de la Autoridad Certificadora (1).

En este caso, como vemos en la imagen superior vamos a seleccionar todas las casillas, convirtiendo a esta Autoridad en una en la que el usuario confía para **identificar sitios web**, **identificar usuarios de correo** e **identificar desarrolladores de software**.
Si hacemos clic en el botón ***VER*** aparecerán las siguientes imágenes donde se pueden ver los detalles del **certificado de la Autoridad Certificadora** creado e instalado por el usuario.

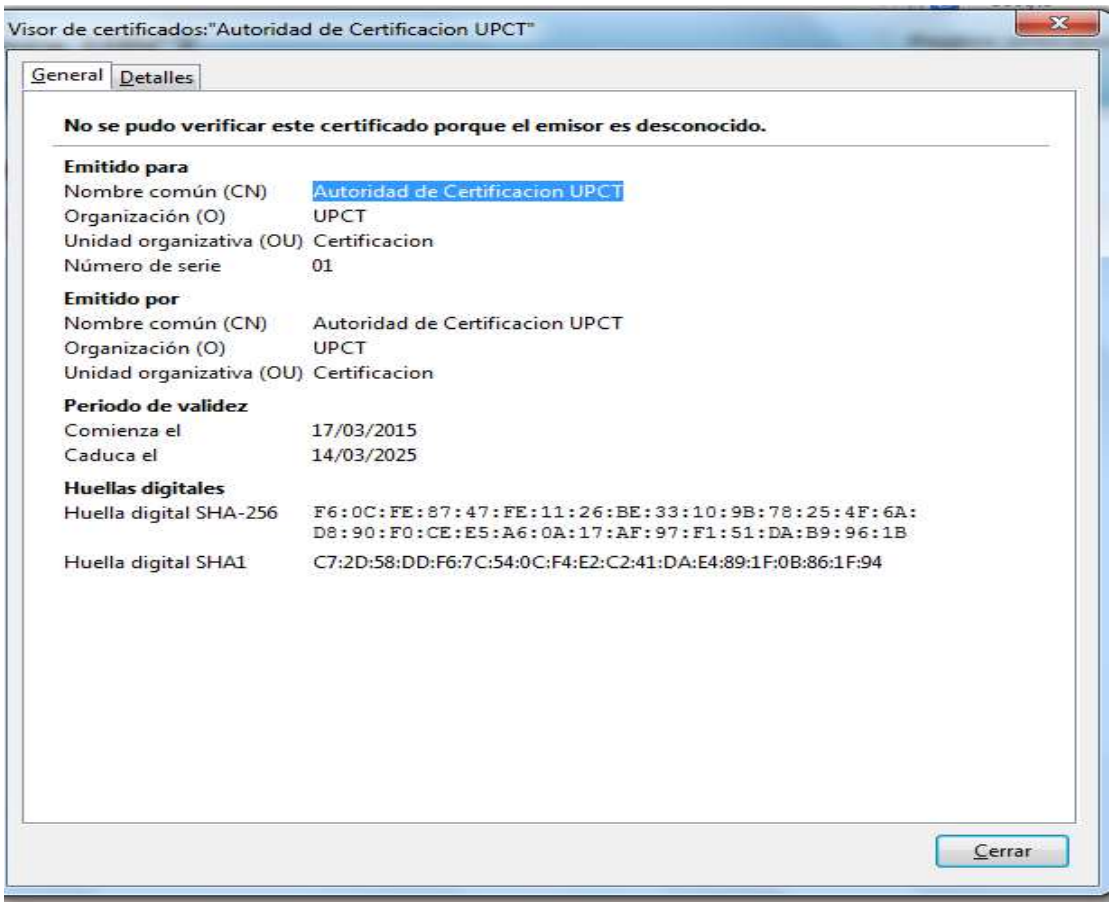

Figura 4.29. Certificado de la Autoridad Certificadora (2).

Como vemos en la imagen superior en la pestaña ***General*** se dan los detalles más generales de este certificado y en la pestaña ***Detalles*** veremos más detalladamente todos los parámetros de certificado.
Estas características que aquí se detallan son las mismas que las del **certificado cliente**, lo único que va a cambiar en este caso será el **destinatario** de este certificado (***'Emitido para'***), que como vemos es la propia **Autoridad Certificadora** puesto que como se ha explicado en la parte del código al crear este certificado se crea un **certificado auto firmado**, y el periodo de validez que en este caso será de 10 años puesto que no es algo que se deba renovar muy frecuentemente a no ser que sea revocado.
Las características del certificado se muestran de manera más detallada en la pestaña ***Detalles*** pero éstas son las mismas que las del **certificado cliente** que ha sido explicado anteriormente puesto que ambos certificados aunque se crean con extensiones diferentes (en el **anexo 2** se explica con detalle el formato y las extensiones de los certificados

digitales) los parámetros usados para crearlos son los mismos, como se ha explicado en la parte del código en el apartado referente al archivo de configuración ***'openssl.conf'***.

Una vez que la **Autoridad Certificadora** ha sido instalada podremos visualizarlo seleccionanado la opción de la barra de herramientas del navegador ***'Opciones'***, a continuación ***'Avanzado'*** y en la pestaña ***'Certificados'*** seleccionar el botón ***'Ver Certificados'***, después en la pestaña ***'Autoridades'*** podemos visualizar que ésta **Autoridad Certificadora** ya ha sido instalada en nuestro navegador y que por tanto nuestro certificado expedido por ella va a tener validez.

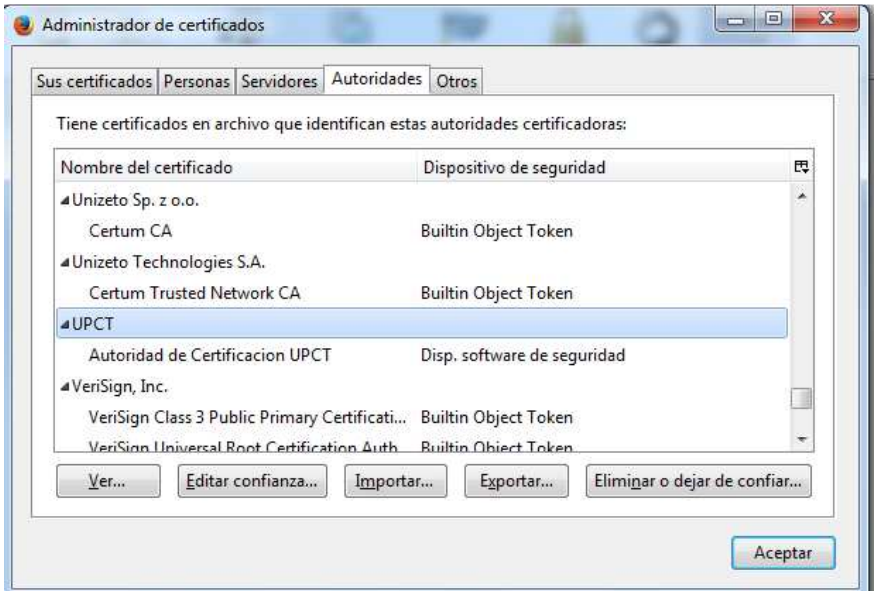

Figura 4.30. Autoridad Certificadora instalada.

Como vemos en la imagen anterior podremos, además de volver a visualizar el certificado, editar la confianza puesta en esta autoridad, importar o exportar alguno de estos certificados o eliminar o dejar de confiar en esta autoridad o en cualquier otra.

Una vez que ha finalizado todo el proceso de instalación, se recomienda realizar una copia de seguridad de las claves y del certificado en un medio alternativo (disquete, CD, etc.) y borrar la copia que se ha grabado en el ordenador en el paso de descarga.

De esta forma, en caso de borrado accidental o por mantenimiento de los datos del disco duro del ordenador podremos recuperar nuestras claves desde la copia de seguridad siempre que tengamos la contraseña de instalación del fichero que se envió en el proceso de emisión.

Es importante mantener la copia del fichero de claves y la contraseña de instalación del fichero de las claves custodiadas en sitios diferentes.

4. Una vez que ya se ha creado la **Autoridad Certificadora** y que el usuario ya dispone de su **certificado personal**, tanto el personal administrativo como el empleado podrán acceder a la pestaña **Administración** (página principal correspondiente al código ***'options.php'***) en la cual, como se ha explicado en el desarrollo del código, si el que accede es el personal administrativo podrá: **renovar el certificado de la Autoridad Certificadora,** o bien **solicitar que se vuelva a firmar el *CSR*** del certificado de la **Autoridad Certificadora**. En cambio si el que accede es el empleado podrá o bien de nuevo **instalar el certificado de la Autoridad**

Certificadora si aún no lo ha hecho (puesto que se proporciona otro enlace como vemos en la imagen) o bien **renovar su certificado personal.**

Figura 4.31. Pestaña Administración.

4.1. Si el usuario (ya sea personal administrativo o empleado) selecciona **renovar** un certificado (cualquiera de los dos) aparecerán las siguientes imágenes, en la que el usuario deberá introducir la **contraseña** empleada para crear el **certificado** de la **Autoridad Certificadora** si se desea renovar éste o bien la **contraseña** empleada para crear el **certificado personal** si es éste el que se desea renovar (código de los archivos ***'IntroducirPassw.php'*** y ***'IntroducirPasswClient.php'***). Una vez que el usuario introduce las contraseñas el software, de manera transparente (código de los archivos ***'renewCert.php'*** y ***'renewCertClient.php'***) **renueva** el certificado correspondiente y lo deja disponible para que el usuario lo instale (al igual que en el punto 3).

Figura 4.32. Renovar certificado AC. Introducir contraseña.

Figura 4.33. Renovar certificado AC. Instalar certificado.

Figura 4.34. Renovar certificado cliente. Introducir contraseña.

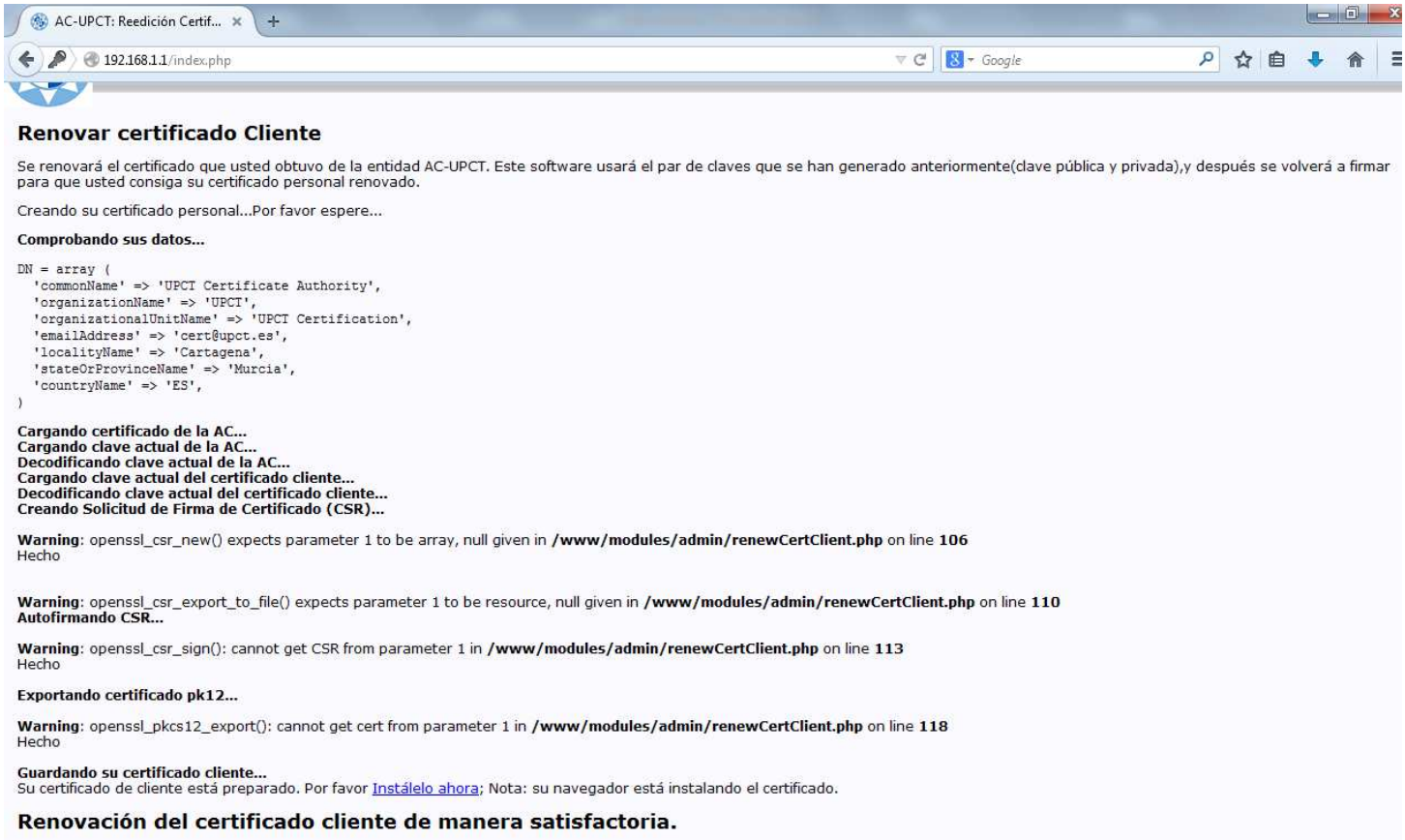

Figura 4.35. Renovar certificado cliente. Instalar certificado.

4.2. Además como se ha mencionado y como se ve en la **Figura 4.31.**, también se podrá refirmar el ***CSR*** del **certificado** de la **Autoridad Certificadora**, introduciendo la **contraseña** correctamente y el archivo de **Petición de Firma de Certificado** que se guarda una vez que se crea dicho **certificado**, de esta manera se **re-firma** el ***CSR*** pero con el **certificado** que ya se había creado, por lo que no se consigue un **certificado auto firmado**, si no un **certificado** firmado con el certificado auto firmado creado inicialmente (el archivo cuyo código crea este nuevo **certificado** es el ***'certSign.php'***). Este paso únicamente lo podrá realizar el personal administrativo puesto que es el único que conoce la **contraseña** que cifra las claves del certificado de la **Autoridad Certificadora**.

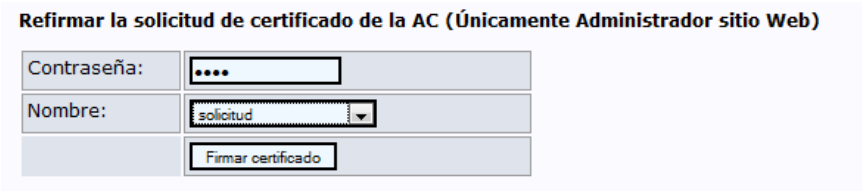

Figura 4.36. Refirmar certificado de la AC.

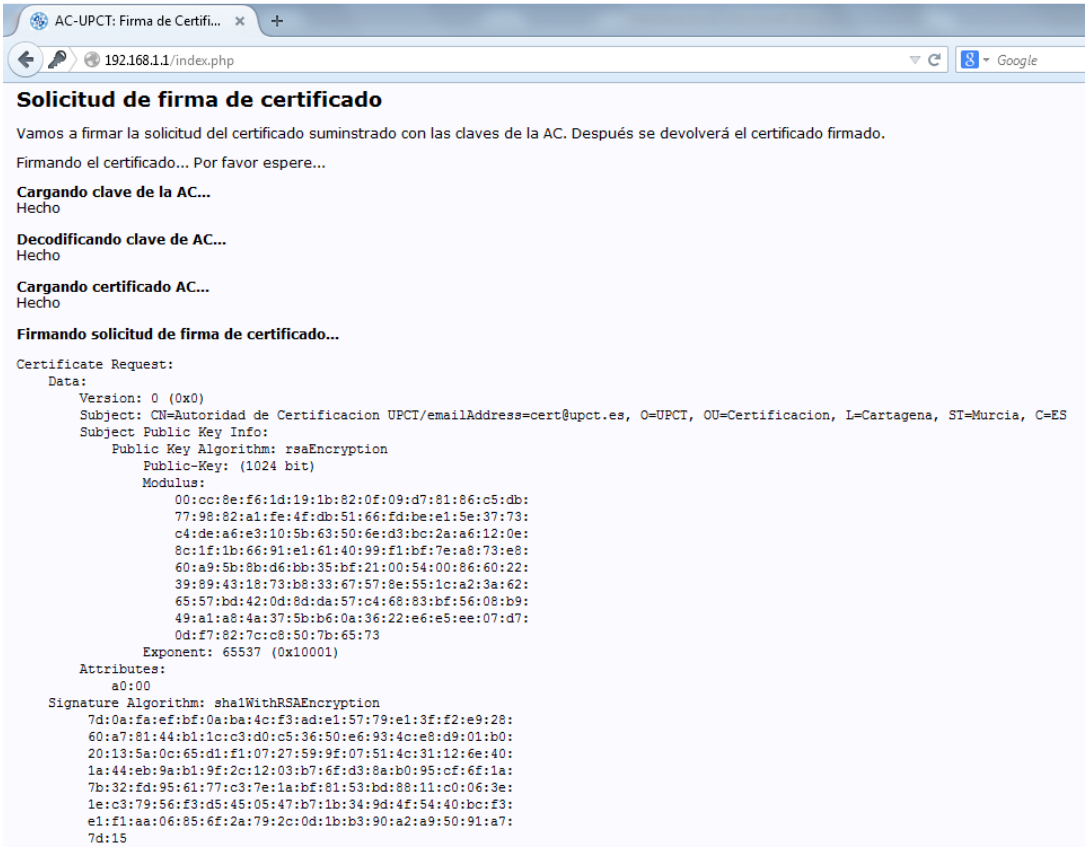

Figura 4.37. Refirmar certificado de la AC.

Como vemos en la imagen superior se muestran los datos de esta ***CSR*** que se ha vuelto a firmar.

5. Por último el usuario podrá acceder a la pestaña **About** (código del archivo ***'about.php'***) donde, como ya se ha explicado, se detalla la **información** acerca del **software** y además se permite descargar la **licencia** y donde se proporciona un enlace para descargar dicho **software**, además también se proporciona un enlace para enviar un correo electrónico al desarrollador si el usuario lo desea. Y a la pestaña **Help** (código del archivo ***'help.php'***), donde se proporciona una pequeña ayuda de los pasos que los usuarios habrán de seguir para crear la **Autoridad certificadora,** para **obtener** su **certificado personal** o las acciones que podrá realizar una vez que ya se disponga de ambos certificados.

Figura 4.38. Pestaña *About.*

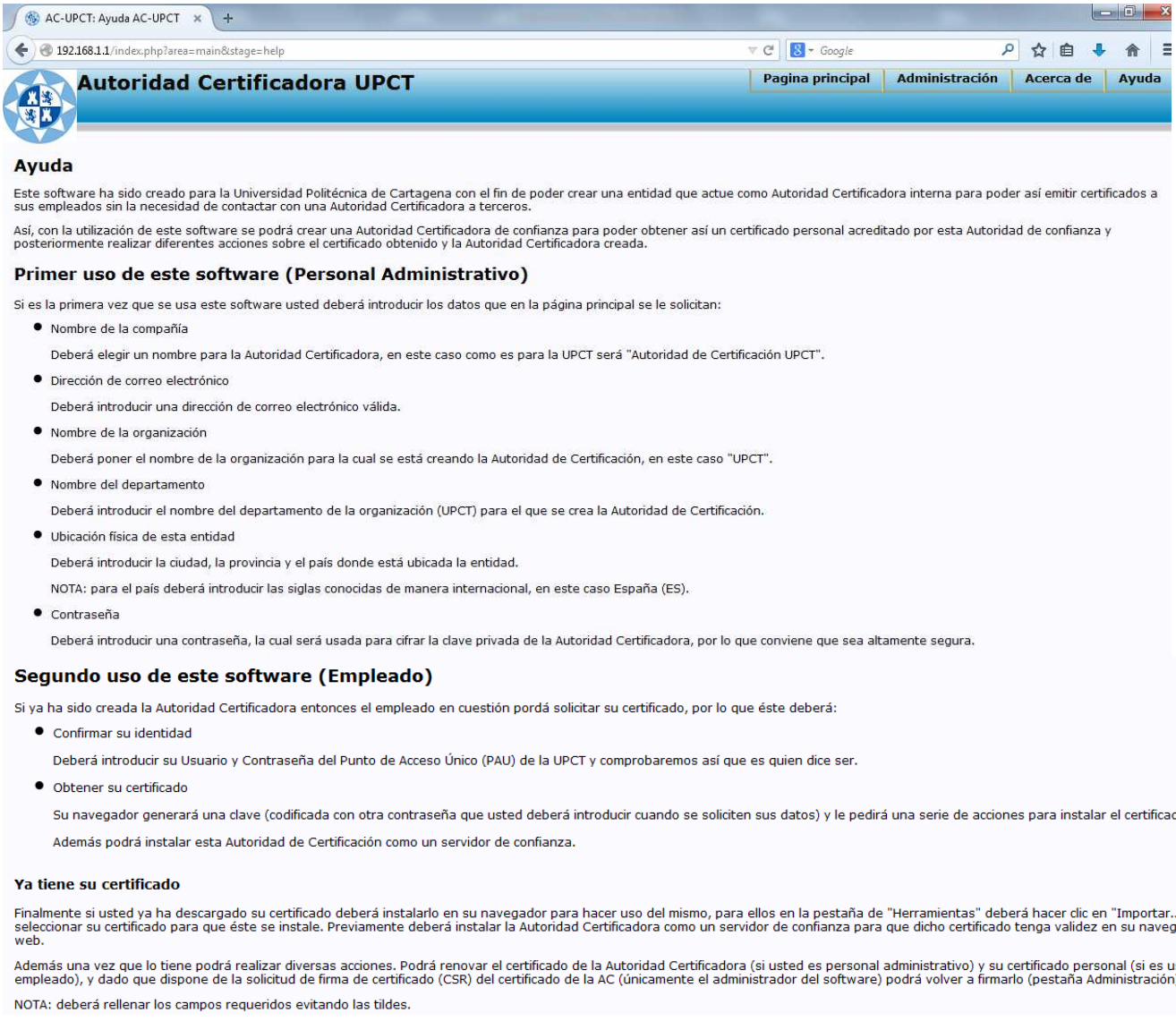

Figura 4.39. Pestaña *Help*.

CAPÍTULO 5.
Conclusiones y trabajos futuros.

A lo largo de los meses de desarrollo del proyecto se han creado y aprendido numerosas técnicas hardware y sobretodo de desarrollo de software, así como la forma de comunicarnos e interactuar con el ordenador y con el **router** empleado como herramienta de trabajo. En concreto hemos aprendido:

- ➔ Origen, funcionalidad y uso del sistema operativo **OpenWrt**.
- ➔ Los conocimientos necesarios para el desarrollo del proyecto referentes a la **seguridad en redes**, más concretamente lo referente a **certificados digitales** y **autoridades de certificación**.
- ➔ Cómo funciona y cómo usar correctamente el lenguaje de programación **PHP**.
- ➔ El funcionamiento y cómo ha sido creado el software utilizado como base para la creación de este proyecto (**php-ca**), además la instalación del mismo para trabajar sobre él.
- ➔ Funcionalidades ofrecidas por el **router** empleado como **hardware** para el desarrollo del proyecto, el cual alberga, como servidor, el software creado.
- ➔ Cómo elaborar una documentación completa.

Así cumpliendo cada uno de los objetivos secundarios fijados al inicio de la memoria, se ha conseguido llevar a buen término el objetivo principal el cual era crear un software que permitiera la creación de una **Autoridad Certificadora** para la **UPCT** y que ésta, por tanto, pudiera expedir **certificados personales** a sus empleados.

Como ya se ha comentado, en la actualidad, desde la aparición, y más aún desde la globalización de Internet, la **seguridad** en los sistemas de información se ha convertido en uno de los problemas más importantes al que debemos hacer frente. El aumento del uso de Internet ha dirigido la atención del mundo entero a un problema crucial: la **privacidad**. Hasta hace relativamente poco, no existía una protección real que garantizara que los mensajes que se enviaban o recibían no fueran interceptados, leídos o incluso alterados por algún desconocido, ya que nadie en realidad dirige o controla la red Internet.

Por ello la creación de este software se convierte en una manera de conseguir dicha **privacidad** y **seguridad** dado que se consigue de alguna manera verificar la **identidad** de una persona y establecer un cierto nivel de confianza. Además este software tiene como **ventaja** que es fácilmente **escalable** y por tanto, con las modificaciones adecuadas, podría llegar a tener una gran importancia en la **UPCT**.

Como **trabajos futuros** lo que aquí se propone es terminar de **completar** el proyecto aquí creado, dado que como se ha ido mencionando a lo largo de la memoria existen algunos puntos que no han sido debidamente tratados puesto que esto por el momento es una versión básica de lo que podría llegar a ser este **software**. Uno de estos puntos y que en un futuro sería el más importante a tratar es la creación de la **lista de certificados revocados** y por consiguiente que este software permitiera que los certificados pudieran ser **revocados**, bien porque ha finalizado el periodo de validez de dicho certificado o por cualquier otro motivo. Y el otro punto a tratar sería en lo referente a la **autenticación**, dado que actualmente esta parte se consigue de manera local pero en un futuro este software debería conectarse al **LDAP** de la **UPCT** y conseguir así comprobar que el usuario es empleado de esta entidad. Derivado de esto y dado que el usuario ya se habría identificado como empleado de la **UPCT**, éste tendría en

este software su propio **archivo local** en el que se fueran almacenando sus datos y sus **certificados personales** tanto **expedidos** como **revocados** y por tanto podría elegir qué acciones realizar sobre cada uno de ellos.

CAPÍTULO 6.
Bibliografía.

- Certificados y Firmas digitales.
  html.rincondelvago.com/certificados-y-firmas-digitales.html
  www.firma-digital.cr/como%20funciona/
  es.wikipedia.org/wiki/Certificado_digital

- Autoridades de Certificación.
  es.wikipedia.org/wiki/Autoridad_de_certificacion

- Seguridad en redes.
  www.dte.us.es/docencia/etsii/gii-ti/tecnologias-avanzadas-de-la-informacion/teoria/Tema1.pdf

- Apuntes asignatura Seguridad en redes (Ingeniería de Telecomunicaciones).

- Ataques pasivos y activos.
  feladazarodriguez.blogspot.com.es/2010/09/ataques-pasivos-y-ataques-activos.html

- Métodos de cifrado.
  www2.uah.es/libretics/concurso2014/files2014/Trabajos/CriptografiayMetodosdeCifrado.pdf

- Cifrado por bloques.
  es.wikipedia.org/wiki/Cifrado_por_bloques

- RSA.
  es.wikipedia.org/wiki/RSA

- Diffie-Hellman.
  es.wikipedia.org/wiki/Diffie-Hellman

- Curva elíptica.
  es.wikipedia.org/wiki/Criptografia_de_curva_eliptica

- Cifrado en flujo.
  es.wikipedia.org/wiki/Cifrador_de_flujo

- A5.
  es.wikipedia.org/wiki/A5/1

- Funciones HASH.
  www.genbetadev.com/seguridad-informatica/que-son-y-para-que-sirven-los-hash-funciones-de-resumen-y-firmas-digitales
  es.wikipedia.org/wiki/Secure_Hash_Algorithm

www.redeszone.net/2010/11/09/criptografia-algoritmos-de-autenticacion-hash/
www.globalsign.es/centro-informacion-ssl/transicion-a-sha-256.html

- Infraestructura de clave pública.
  es.wikipedia.org/wiki/Infraestructura_de_clave_publica

- Información OpenWrt.
  wiki.openwrt.org/es/doc/start
  wiki.openwrt.org/es/about/start
  www.adslzone.net/2014/10/03/nueva-version-de-openwrt-el-firmware-basado-en-linux-para-optimizar-nuestro-router/

- Como instalar OpenWrt.
  tombatossals.github.io/instalar-openwrt/
  overside.wordpress.com/2010/11/26/abrir-puerto-en-firewall-en-openwrt/

- OpenSSL.
  es.wikipedia.org/wiki/OpenSSL

- PHP.
  es.wikipedia.org/wiki/PHP
  php.net/docs.php

- Detalles del router TP-Link.TL-WR1043ND.
  www.tp-link.com/ar/products/details/?model=TL-WR1043ND

- Hoja de estilo.
  es.wikipedia.org/wiki/Hoja_de_estilo

- Librería Xenroll.dll.
  support.microsoft.com/kb/323172/es

- Formatos de certificados.
  www.sede.fnmt.gob.es/preguntas-frecuentes/acerca-de-los-certificados/-/asset_publisher/SA6CBDyHs9ZB/content/1553-diferencias-entre-pfx-p12-cer-y-crt-
  es.wikipedia.org/wiki/PKCS

- Formato de certificado X509.
  es.wikipedia.org/wiki/X.509

- Instalación de certificados.
  sede.elescorial.es/GDCarpetaCiudadano/certificadosMoz.pdf

Anexo 1. Instalación de OpenWrt y puesta en marcha del software.

En este anexo se van a explicar los sencillos pasos que se han seguido para instalar en nuestro router el sistema operativo **OpenWrt** y como posteriormente se instala el software basado en **php-ca** y se trabaja sobre el mismo para obtener el resultado final.

En primer lugar deberemos descargar el **firmware OpenWRT** adecuado para nuestro router, acceder a la administración web del firmware original del router, e instalar nuestro nuevo sistema operativo con el que podremos realizar operaciones de comunicación mucho más avanzadas que las que nos proporcionaba el firmware original y que para nosotros será imprescindible para desarrollar este proyecto.

Debido a que existen diferentes arquitecturas y chipsets dentro de toda la gama de routers soportados por OpenWRT, deberemos buscar la imagen del firmware adecuada entre todas las disponibles. En nuestro caso, como ya se ha mencionado, tenemos un **router TP-Link TL-W1043ND**, así que lo primero es mirar si está soportado, cosa que efectivamente confirmamos y encontramos en la página de los desarrolladores de este sistema operativo (bibliografía).

Así descargamos en la web de **OpenWrt** el firmware que mejor se adapta a nuestro router, que como se ha explicado en el **capítulo 3** corresponde a la versión **14.07** (de la cual se han explicado sus características en el **capítulo 3**).

Posteriormente deberemos sustituir el **firmware original** por el que hemos descargado, para ello:

- Conectamos el PC al router mediante un cable a cualquiera de los puertos LAN del router.
- El router nos servirá una IP por **DHCP**. Los dispositivos TP-LINK utilizan la subred 192.168.0.0/24, la IP del router será la 192.168.0.1, y la que nos asigne a nuestro PC será cualquier IP dentro de ese rango.
- Ya podemos acceder al panel de administración, vía web, a través de esta dirección: http://192.168.0.1, con el **usuario admin**, **contraseña admin**.
- Accedemos ahora al menú **System tools**, subapartado **Firmware Upgrade**.
- Del selector de archivos subimos el **firmware de OpenWRT** que hemos descargado antes, y apretamos el boton de **Upgrade Firmware**. Tardará unos 5 minutos en completar el proceso.
- Ya tenemos **OpenWRT** en nuestro router.

En este caso hemos instalado una **rama trunk** la cual no lleva instalada por defecto el gestor **LuCi** (interfaz Web) puesto que nos comunicaremos con el router a través de línea de comandos como se ha mencionado anteriormente.

Así deberemos configurar la contraseña del usuario **root** (la cual será **root**), en este caso lo haremos conectándonos vía **telnet**, para poder posteriormente conectarnos vía **ssh** al router y poder trabajar sobre el mismo.

Sabemos que los 4 puertos LAN están bridgeados con la Wireless LAN, y si conectamos nuestro PC a cualquiera de ellos obtendremos una IP por DHCP en el rango 192.168.1.0/24. En este caso la IP de nuestro router es la **192.168.1.1** y por tanto es la que usaremos para conectarnos

al router y para posteriormente visualizar el software creado. Aunque también nos podemos conectar a través del puerto WAN, siendo necesario para ello abrir los puertos de los servicios **WWW** y **SSH** en el Firewall, lo cual se hace añadiendo las siguientes líneas de código en el fichero de configuración de **/etc/config/firewall** (una vez que se ha conectado vía **ssh** a través de línea de comandos):

```
# Permitir SSH
config rule
option src wan
option proto tcp
option dest_port ssh
option target ACCEPT

# Permitir WEB
config rule
option src wan
option proto tcp
option dest_port 80
option target ACCEPT
```

Como ya se ha mencionado al principio de la memoria el sistema operativo del PC usado es **Windows 7** por lo que para poder establecer una conexión **ssh** con nuestro router ha sido necesario instalar la herramienta **Putty**, cuya imagen se muestra a continuación.

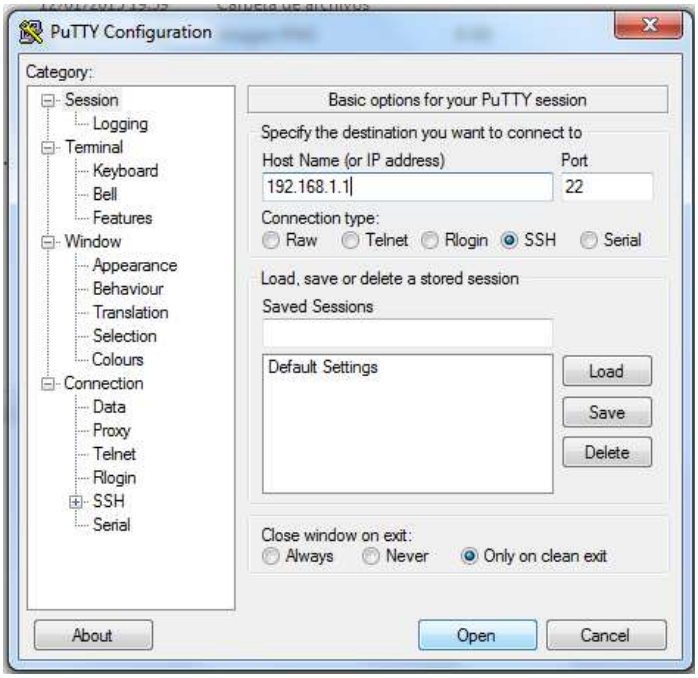

Figura A1.1. Conexión SSH a través del puerto 22.

Como vemos está seleccionada la casilla **SSH** y el puerto **22** siendo éste el que permite esta conexión, de manera que al pulsar el botón ***Open*** se nos abrirá una nueva conexión donde se deberá introducir usuario y contraseña, en este caso **root** y **root** como se ve en la siguiente imagen.

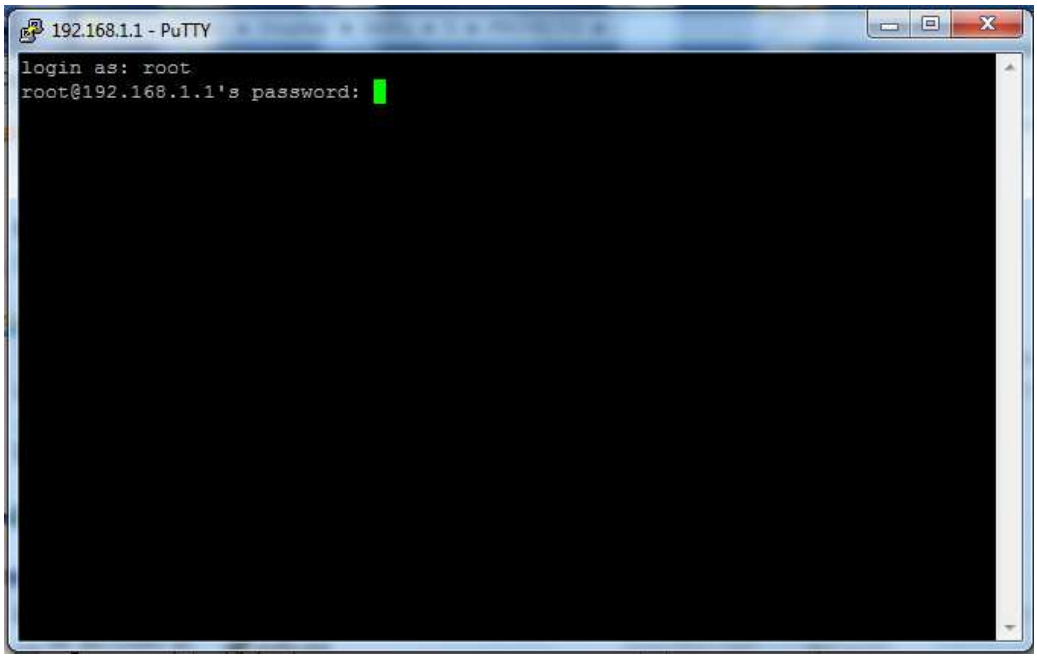

Figura A1.2. Introducir usuario y contraseña conexión SSH.

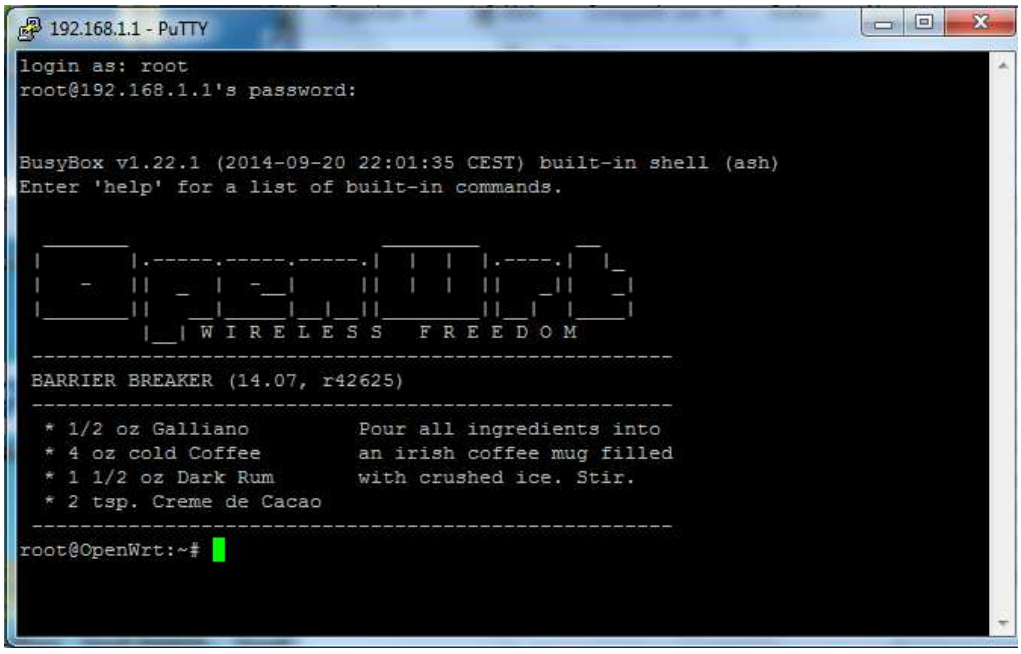

Figura A1.3. Conexión SSH con router OpenWrt.

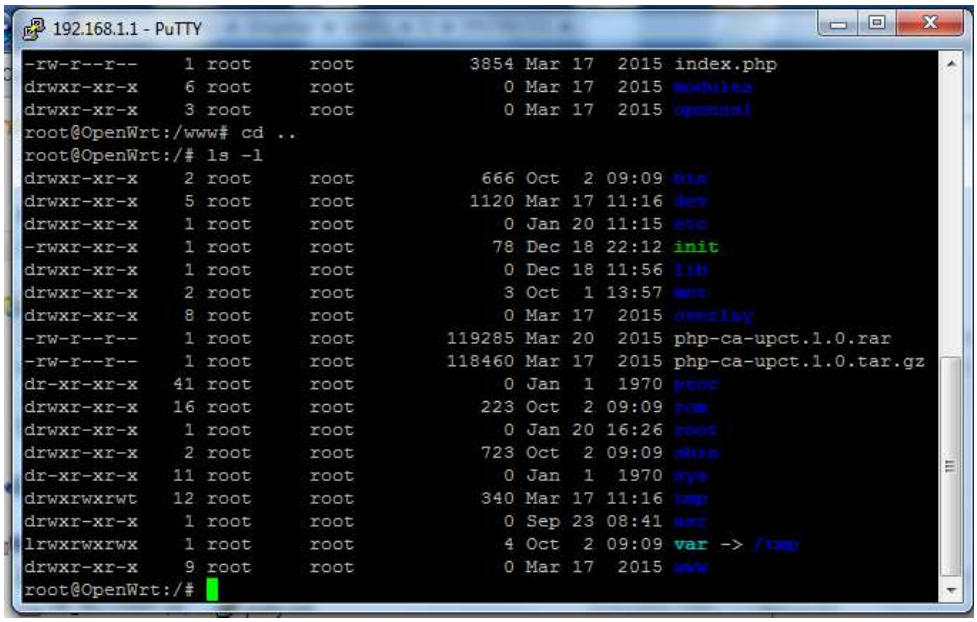

Figura A1. Gestor de archivos OpenWrt.

De esta manera ya podemos acceder a los ficheros del router y modificar lo que sea necesario. En primer lugar deberemos instalar los paquetes necesarios para el desarrollo del proyecto los cuales se han mencionado en el **capítulo 3** empleando para ello el gestor de paquetes **opkg**.

Una vez que lo tenemos todo listo para soportar el software **php-ca** deberemos copiar éste en el router para poder trabajar sobre el mismo, para ello se usaría el comando **scp** siendo éste el comando para copiar archivos de una máquina a otra del protocolo **ssh**. Sin embargo dado que el sistema operativo es **Windows 7** se dispone de un software llamado **WinSCP** que permite copiar archivos de una máquina a otra de manera sencilla.

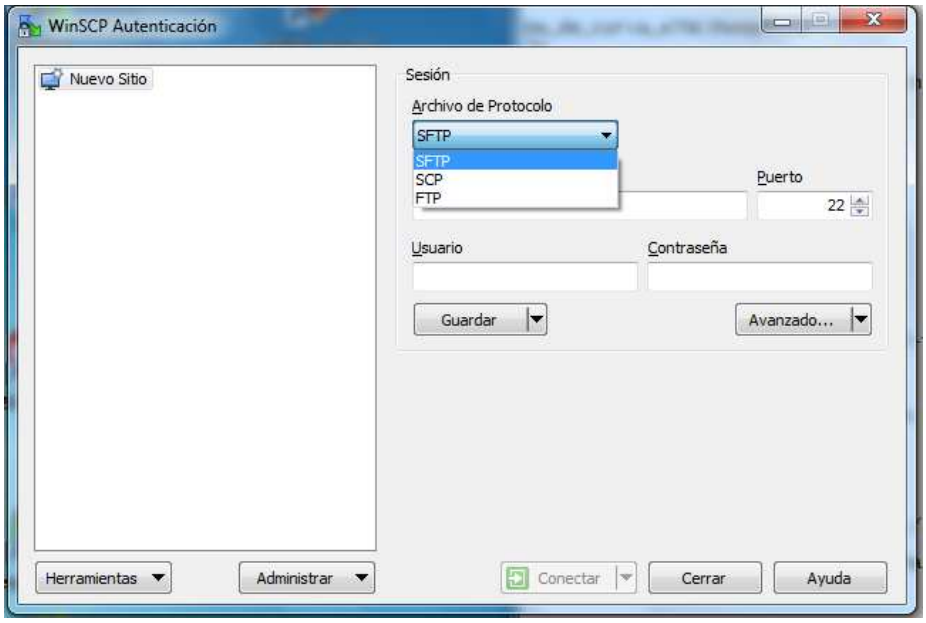

Figura A1.4. Software WinSCP.

Como vemos en la imagen superior se puede elegir el protocolo a usar para la copia de archivos pero nosotros emplearemos el protocolo **scp** puesto que es más seguro.

Así, como vemos en la **Figura A1.5.** introducimos la dirección IP, el puerto, el usuario y la contraseña al igual que se nos solicita para realizar la conexión **ssh** y pulsamos ***Conectar***.

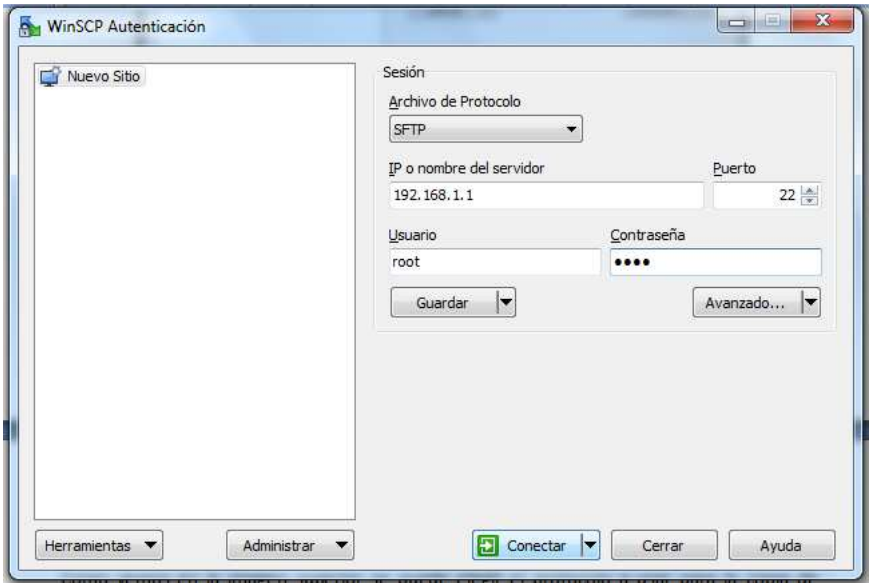

Figura A1.5. Software WinSCP. Datos de conexión.

Una vez que nos conectamos aparece la siguiente imagen, donde como vemos podemos subir o bajar archivos desde el dispositivo remoto (en este caso nuestro router). En la ventana de la izquierda se muestran los archivos del PC o de la máquina local (archivos a subir) y en la de la derecha los archivos del router o máquina remota (archivos a descargar).

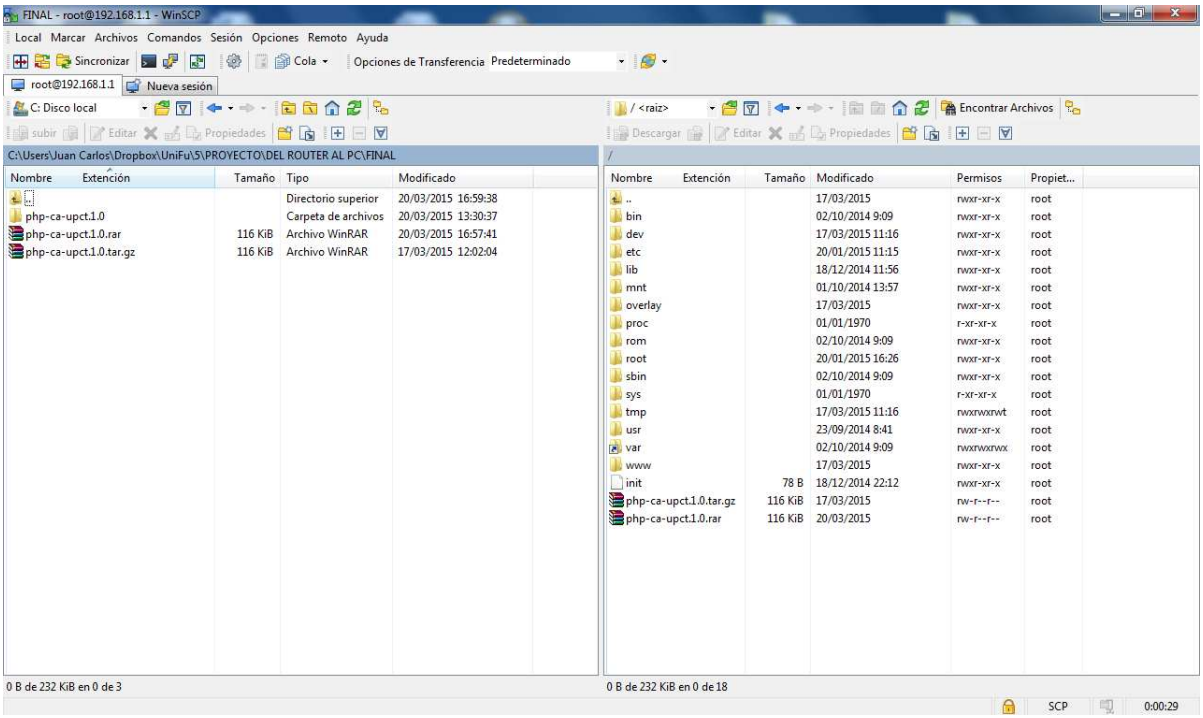

Figura A1.6. Software WinSCP. Visor de archives.

Finalmente subimos o copiamos el software creado sobre **php-ca** que posteriormente modificaremos en la carpeta **www** puesto que será de aquí donde se leerá (archivo principal ***Index.php***) para mostrar en el navegador web el software creado y del que el usuario final podrá hacer uso.

Anexo 2. Información adicional sobre certificados.

En este anexo se va a explicar de manera más extendida lo referente a los diferentes formatos, versiones y extensiones de los certificados creados, para poder entender así cómo y para qué son creados.

Como se ha mencionado en el **capítulo 2** el formato de los **certificados digitales** que se crean y que posteriormente instalan los usuarios en sus navegadores es el **X.509**, el cual es un estándar UIT-T para infraestructuras de claves públicas (en inglés, ***Public Key Infrastructure*** o PKI). Y como ya se ha dicho especifica, entre otras cosas, formatos estándar para certificados de claves públicas y un algoritmo de validación de la ruta de certificación.

Anexo 2.1. Historia del estándar X509.

Este estándar **X.509** publicado oficialmente en 1988 y comenzado conjuntamente con el estándar **X.500**, asume un **sistema jerárquico** estricto de **autoridades certificantes** (**ACs**) para emisión de certificados. Esto contrasta con modelos de **redes de confianza**, como PGP, donde cualquier nodo de la red (no solo las ACs) puede **firmar claves públicas**, y por tanto avalar la validez de certificados de claves de otras entidades. La **versión 3 de X.509** (la que actualmente se emplea, como se ha visto en el **capítulo 4**) incluye la flexibilidad de soporte de otras tecnologías como bridges y mallas. Puede usarse en una web de confianza ***peer to peer*** de tipo OpenPGP, pero desde 2004 raramente se usa así.

X.509 incluye también **estándares** para implementación de listas de **certificados en revocación** (**CRLs**), 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.

La forma aprobada por la IETF de **chequear la validez de un certificado** es el **Online Certificate Status Protocol** (**OCSP**).

Anexo 2.2. Certificados.

En el **sistema X.509**, una autoridad certificadora (AC) emite un certificado asociando una clave pública a un Nombre Distinguido particular en la tradición de X.500 o a un Nombre Alternativo tal como una dirección de correo electrónico o una entrada de DNS, como hemos podido comprobar a lo largo del desarrollo de la memoria.

Anexo 2.2.1. Estructura de un certificado digital.

La estructura de un certificado digital **X.509 v3** como hemos podido ver en el **capítulo 4** de manera detallada 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 (opcional)*

- *Algoritmo usado para firmar el certificado*
- *Firma digital del certificado*

Donde los **identificadores únicos de emisor y sujeto** fueron introducidos en la **versión 2**, y las **extensiones** en la **versión 3**. Obsérvese que el **número de serie debe ser único** para cada certificado emitido por una misma autoridad certificadora (tal como lo indica el RFC 2459).

El **estándar 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, en primer lugar, el que muestra el **algoritmo utilizado para crear la clave** (ej. RSA), y en segundo lugar, 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**.

Anexo 2.2.2. Extensiones de archivo de un certificado.

Como se ha mencionado a lo largo de la memoria para crear los certificados que aquí se generan se han empleado dos extensiones (una para el certificado de la **AC** y otra para el **certificado cliente**), pero existen diferentes extensiones para estos certificados, las cuales son las siguientes:

- ***.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-----". Puede contener certificados o claves privadas, encerrados entre las líneas BEGIN/END apropiadas. Extensión empleada para el certificado de la **AC**.
- ***.P7B*** - Ver .p7c.
- ***.P7C*** - Estructura ***PKCS#7* SignedData sin datos**, solo certificado(s) o CRL(s).
- ***.PFX*** - Ver .p12.
- ***.P12*** - ***PKCS#12***, puede contener certificado(s) (público) y claves privadas (protegido con clave. Extensión empleada para el **certificado cliente** dado que como vemos permitía exportar **claves privadas** lo cual era necesario a la hora de instalar el certificado en el navegador.

PKCS #7 es un **estándar** para **firmar o cifrar datos** (ellos lo llaman "sobreado"). Dado que el certificado es necesario para verificar datos firmados, es posible incluirlos en la estructura SignedData. Un archivo .P7C es simplemente una estructura SignedData, sin datos para firmar.

PKCS #12 evolucionó del estándar PFX (Personal inFormation eXchange) y se usa para **intercambiar objetos públicos y privados dentro de un archivo**.

Anexo 2.2.3. Proceso de validación de un certificado.

Como se observa en la **Figura 4.27.** del **capítulo 4** al final del certificado se encuentra la **firma de mismo**. Para estampar esta firma, la **autoridad certificadora** calcula un **hash SHA1** de la primera parte del certificado (la sección de Data: los datos del mismo más la clave pública) y **cifra ese hash con su clave privada**, que mantiene en secreto. Si el sitio web le devuelve a un cliente que se conecta el certificado anterior para probar su identidad, para validar este certificado, necesitamos el **certificado de la autoridad certificadora** (en este caso la de la **UPCT**). Se toma la **clave pública del certificado de la autoridad certificadora para decodificar la firma del primer certificado**, obteniéndose un **hash SHA1**. Este **hash SHA1** debe **coincidir** con el **hash SHA1** calculado sobre la primera parte del certificado. En ese caso el **proceso de validación termina con éxito**. Si no, la validación no tiene éxito, y no puede asegurarse que el certificado instalado está vinculado con esa **clave pública** y que por tanto tenga validez.

En el caso del certificado de la **Autoridad Certificadora** el certificado obtenido es, como hemos dicho un **certificado auto firmado**. Dado que, como vemos en la **Figura 4.29.** del **capítulo 4** el **CN del Emisor y el CN del Sujeto son iguales**. No hay forma de verificar este certificado, salvo que se comprueba contra sí mismo. En este caso, hemos alcanzado el **fin de la cadena de certificados**. ¿Cómo es entonces que este certificado se hace confiable? Simple: se configura manualmente como hemos hecho instalando esta **Autoridad** como una de **confianza**. La clave privada correspondiente a este certificado debe estar muy bien oculta.

Anexo 2.2.4. Tipos de certificados.

Existen distintos tipos de certificados, según el criterio que utilicemos de clasificación:

Verificación de datos

- **Certificados de Clase 1:** corresponde a los certificados más fáciles de obtener e involucran pocas verificaciones de los datos que figuran en él: sólo el nombre y la dirección de correo electrónico del titular.
- **Certificados de Clase 2:** en los que la Autoridad Certificadora comprueba además el Documento de identidad o permiso de conducir que incluya fotografía, el número de la Seguridad Social y la fecha de nacimiento.
- **Certificados de Clase 3:** en la que se añaden a las comprobaciones de la Clase 2 la verificación de crédito de la persona o empresa mediante un servicio del tipo Equifax, Datacredito.
- **Certificados de Clase 4:** que a todas las comprobaciones anteriores suma la verificación del cargo o la posición de una persona dentro de una organización.

Finalidad

- **Certificados SSL para cliente:** mediante el protocolo Secure Socket Layer, dirigido a una persona física.
- **Certificados SSL para servidor:** usados para identificar a un servidor ante un cliente en comunicaciones mediante SSL.
- **Certificados S/MIME:** usados para servicios de correo electrónico firmado y cifrado, que se expiden generalmente a una **persona física**.
- **Certificados para la firma de código:** usados para identificar al autor de ficheros o porciones de código en cualquier lenguaje de programación que se deba ejecutar en red (Java, JavaScript, CGI, etc).
- **Certificados para AC (Autoridades Certificadoras):** se usa por el software cliente para determinar si pueden confiar en un certificado cualquiera, accediendo al certificado de la AC y comprobando que ésta es de confianza.