Universitat Oberta de Catalunya

uoc.edu

Seguridad mediante Firma Digital y Certificados electrónicos

David de Dueñas Ledesma
Grado de Ingeniería Informática
Área de Seguridad Informática

Consultor: Helena Rifà Pous
Profesor: Jorge Miguel Moneo

31 de Mayo de 2022

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Seguridad mediante Firma Digital y Certificados electrónicos*
Nombre del autor:	*David de Dueñas Ledesma*
Nombre del consultor/a:	*Helena Rifà Pous*
Nombre del PRA:	*Jorge Miguel Moneo*
Fecha de entrega (mm/aaaa):	05/2022
Titulación:	*Grado de Ingeniería Informática*
Área del Trabajo Final:	*Seguridad informática*
Idioma del trabajo:	*Español*
Palabras clave	*Firma electrónica, Certificados digitales, Criptografía*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

Los trámites electrónicos, y en concreto el uso de certificados electrónicos y firmas digitales forman cada vez más parte del día a día, pero no todo el mundo es usuario de esta tecnología.

En este proyecto, en primer lugar, hemos planteado una encuesta para conocer si los ciudadanos tienen reticencia al uso de esta tecnología, la creen segura, la han usado y qué dificultades encuentran.

Hemos analizado los principales problemas descritos por los usuarios y estudiado que tipo de limitaciones se pueden encontrar, así como las obligaciones que hay en el uso de los trámites electrónicos.

Se ha estudiado el funcionamiento de los certificados digitales, cómo se emiten, qué información recogen, quién los emite y bajo qué normativas.

Se ha comprobado qué proceso es necesario seguir para la generación de una firma digital y cómo éstas pueden clasificarse en función de la información que contengan y su estructura, así como los algoritmos matemáticos en los que se apoyan. Unido a esto, hemos revisado la posibilidad de generar firmas longevas para posibilitar su validez en el futuro mediante el uso de sellos de tiempo.

Teniendo en cuenta la rápida evolución de las tecnologías, hemos revisado cómo la computación cuántica puede afectar a la base de la firma electrónica.

Por último, hemos planteado un caso de uso en el que las firmas electrónicas y certificados digitales pueden ser de utilidad para una actividad del día a día.

Abstract (in English, 250 words or less):

Online procedures, in particular, digital certificates and electronic signatures are used day to day, but, not everyone are using this tecnology yet.

In this project, firstly, we have made a survey in order to know the reluctance to use this technology, if they feel that it´s safe, they tried it and what problems they have found.

Also, we have studied the main problems of the users and know what limitations they could find, as well as the obligations of using online procedures.

We have studied the operation of digital certificates, how they are issued, what information they have, who issued them and what regulations applies to them.

We have studied the steps needed to generate a digital signature and how them can be classified by their information and structure, also, we have analiced what matematical algortims are used in digital signatures. Futhermore, we have analyzed the long term signatures generation using timestamps in order to achieve digital signatures that can be validated indefinitely

Considering the changing evolution of the technologies, we have studied how quantum computing can affect the key of digital signatures.

Finally, we have presented a case of use involving digital signatures and certificates for improve a daily activity

Índice

Lista de figuras

1. Introducción

1.1. Contexto y justificación del Trabajo

Una de las múltiples ventajas que nos ofrece la tecnología actual es el ahorro de desplazamientos y simplificación de las tareas por la posibilidad de realizar trámites de forma telemática. Esto se ha visto acrecentado por el impulso del teletrabajo y el confinamiento por la situación de la pandemia por la COVID. Como ejemplo de este aumento, si nos fijamos en los datos de uso según el OBSAE (Observatorio de Administración Electrónica) [1], vemos que en el año 2019 se usó la carpeta ciudadana 112.000.000 veces aproximadamente y se remitieron telemáticamente 1.696.816 registros electrónicos, mientras que en el año 2021 fueron 208.648.000 accesos aproximados a la carpeta ciudadana y se remitieron 4.850.700 registros electrónicos.

En este contexto, una de las opciones de poder realizar trámites de forma telemática del mismo modo que presencial es mediante el uso de certificados y firma electrónicos.

En España, hace 10 años aproximadamente, según el INE, se disponía de certificado para firma electrónica unos 11 millones de personas, sin embargo, el uso de estos para trámites con las Administraciones Públicas era escaso, un 4,7% haciendo uso del DNI electrónico y un 13% de otro tipo de certificados. [2]

A nivel de empresa, según informe del INE [3], las cifras varían drásticamente, ya que, más del 80% de las empresas con conexión a Internet hace uso de la firma digital.

En otros países, como Estados Unidos, Adobe calculó que el 27% de los estadounidenses uso la firma electrónica por primera vez el pasado año 2020. [4]

Por tanto, ¿Son los ciudadanos / usuarios realmente conscientes de esta posibilidad? ¿Se sienten seguros los usuarios con esta tecnología? ¿Hacen bien en sentirse seguros o inseguros con ella?

De forma teórica y práctica haremos un análisis de esta tecnología tras una evaluación previa.

1.2. Objetivos del Trabajo

Los principales objetivos de este trabajo son:

1. Identificar la recepción, conocimiento, uso y dudas de la firma electrónica y los certificados por parte de los ciudadanos.

2. Dar respuesta a las mayores inquietudes reportadas por los ciudadanos.

3. Ofrecer una explicación breve y sencilla que sea de utilidad para los ciudadanos del funcionamiento y pasos necesarios para el uso de la firma digital.

4. Analizar de forma teórica y práctica el nivel de seguridad que puede aportar el uso de la firma digital y los certificados digitales.

5. Plantear de forma teórica un caso cotidiano en el que poder aplicar la firma electrónica como opción de funcionamiento.

1.3. Enfoque y método seguido

1.3.1. Enfoque

Teniendo en cuenta los objetivos fijados anteriormente, enfocaremos el trabajo de la siguiente forma:

1. **Obtención de información inicial**: en primer lugar, definiremos las preguntas que deseamos realizar a los ciudadanos para conocer su punto de vista, para ello crearemos una encuesta usando los formularios de Google, cuyo enlace será remitido al mayor número de personas posibles con objeto de que la muestra sea lo más representativa posible.

2. **Análisis del funcionamiento**: basándonos en las respuestas obtenidas, se hará un estudio con los elementos más relevantes de la siguiente forma:

    a. Se investigará el funcionamiento teórico para obtener una página web estática (una única página) como si fueran unas FAQ con objeto de dar respuesta a las principales dudas de los usuarios, de forma que la misma pudiera ser a posteriori incorporada como una sección de una página web de una entidad (pública o privada) como información básica de firma electrónica.

    b. Se harán pruebas mediante Python que muestren el funcionamiento de la firma electrónica o partes de la misma para demostrar el porqué de su seguridad y cómo funciona.

3. **Creación de caso real**: se detallará una situación en la que el uso de la firma electrónica podría ser usada y se explicará de forma teórica qué elementos se usarían y de qué manera para poder obtener dicha funcionalidad. En este caso, al no realizarse una implementación, en principio las únicas herramientas a utilizar podrían ser las necesarias para introducir elementos gráficos para facilitar la explicación.

1.3.2. Metodología

En cuanto a la metodología a utilizar, se utilizará una metodología **ágil**, esta forma de trabajar nos hará actuar de forma reactiva en función de los resultados y necesidades. En este sentido, tomaremos la idea de "sprints" de la metodología *Scrum*, de forma que cada poco tiempo se finalice un apartado concreto del proyecto, dividiendo así el trabajo en muchas tareas pequeñas de forma que se pueda avanzar poco a poco. Asimismo:

- El análisis a realizar se hará de manera progresiva, en función de cada duda surgida, ya sea por el resultado de la encuesta o por el análisis previo, de forma que se irán resolviendo dudas paso a paso por cada necesidad surgida o tema concreto.

- Para el caso de la parte técnica o práctica se actuará de forma similar, en lugar de realizar una planificación previa de las pruebas o desarrollos necesarios, cuando encontremos una necesidad de hacer una prueba real o demostración con un software o bien desarrollar una pequeña parte de código para explicar, comprobar o confirmar una información lo haremos en ese momento de necesidad.

1.4. Planificación del Trabajo

Es importen indicar, que este punto se trata de una planificación inicial y durante la realización del proyecto los tiempos pueden diferir, así como detectarse nuevas tareas que no se ha considerado a posteriori, pero sirve como base para establecer el plan de trabajo.

El trabajo estará planificado en 3 etapas:

Etapa 0: En esta primera etapa se planteará el proyecto a realizar, explicando en qué consisten el mismo, cómo se realizará y qué pasos se seguirán, esta etapa se corresponde con la presente planificación y se estima finalizada a final de febrero, por eso, no se incorpora en la planificación de Gantt.

Etapa 1: Esta parte tomará el tiempo aproximado de la segunda PEC (02/03/2022 – 29/03/2022). Las tareas previstas son:

- **Definir encuesta (01/03/2022 a 02/03/2022)**: se establecerán qué preguntas se realizarán a los ciudadanos y se incorporarán en la encuesta para poder lanzarla y que sea respondida.
- **Realizar encuesta (03/03/2022 a 11/03/2022): en este** periodo se remitirá el enlace de las encuestas con objeto de que los ciudadanos puedan cumplimentar las mismas.
- **Comienzo respuestas teóricas (20/03/2022 -):** se estudiarán las dudas e inquietudes de la encuesta y se realizará un análisis explicativo de cómo funciona la firma electrónica y los certificados digitales, intentado además dar respuesta a las posibles dudas detectadas en las respuestas.
- **Comienzo respuestas técnicas (20/03/2022 -):** en función de las necesidades a explicar con respecto a las respuestas teóricas, se realizarán pequeños códigos en Python para apoyar la explicación – prueba.

- **Generar web-FAQ (20/03/2022 – 29/03/2022)**: Con las respuestas obtenidas se creará una sencilla web estática (1 sola página) que sirva como indicaciones – FAQ del funcionamiento y uso de la firma electrónica.

  - Hitos de esta etapa:
    - Encuesta finalizada.
    - Generar web-FAQ

Etapa 2: Esta parte tomará el tiempo aproximado de la tercera PEC (30/03/2022 – 26/04/2022). Las tareas previstas son:

- **Finalización respuestas teóricas (30/03/2022 – 27/04/2022):** Se continuará y terminará el análisis explicativo de cómo funciona la firma electrónica y los certificados digitales, intentado además dar respuesta a las posibles dudas detectadas en las respuestas.
- **Finalización respuestas técnicas (30/03/2022 - 27/04/2022):** Se seguirán realizando pequeños códigos en Python para apoyar la explicación – prueba.

  - Hitos de esta etapa:
    - Finalizar respuestas teóricas
    - Finalizar respuestas técnicas

Etapa 3: Esta parte tomará el tiempo aproximado de la cuarta PEC (27/04/2022 – 31/05/2022). Las tareas previstas son:

- **Crear caso de uso (27/04/2022 a 14/05/2022): en este** periodo se remitirá el enlace de las encuestas con objeto de que los ciudadanos puedan cumplimentar las mismas.
- **Conclusiones (15/05/2022 – 31/05/2022):** se estudiarán las dudas e inquietudes de la encuesta y se realizará un análisis explicativo de cómo funciona la firma electrónica y los certificados digitales, intentado además dar respuesta a las posibles dudas detectadas en las respuestas.

  - Hitos de esta etapa:
    - Crear caso de uso
    - Conclusiones
    - Finalizar memoria

Mostramos en un **diagrama de Gantt** la planificación estimada (la cual podría variar durante la ejecución).

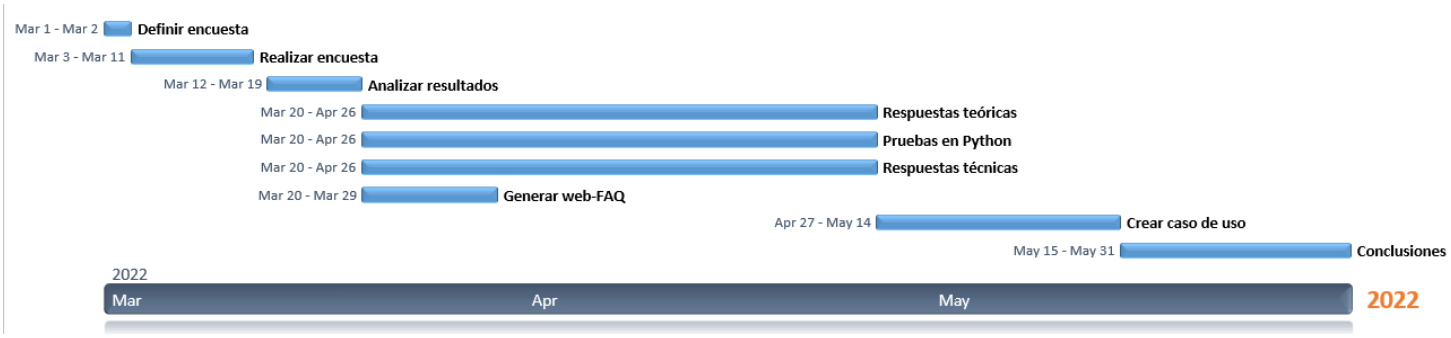


Ilustración 1 - Diagrama Gantt

1.5. Recursos necesarios

En el presente apartado vamos a listar los recursos que han sido utilizados para la elaboración del proyecto:

- **PyCharm 2021.3.2 Community Edition + Python 3.7**: Utilizado para elaborar pequeñas partes de código
- **ASN1 Editor**: Utilizado para la visualización de certificados
- **OpenSSL**: Utilizado para pruebas OCSP (Online Certificate Status Protocol)
- **Microsoft PowerPoint**: Utilizado para elaboración de gráficos
- **Microsoft Word**: Utilizado para la redacción de la memoria
- **Formularios Google**: Utilizado para la creación de la encuesta
- **Certificado software emitido por FNMT**: Utilizado para realizar diversas pruebas de funcionamiento
- **OfficeTimeline**: extensión instalada en PowerPoint para la realización del diagrama Gantt
- **Numpy**: Librería de Python que se ha utilizado para las operaciones matemáticas en cálculos de hash
- **Curl**: Software utilizado desde la línea de comandos para realizar la petición de sellado de tiempo

1.6. Análisis de riesgos

1.6.1. Identificación riesgos

Aunque al no tratarse de un proyecto de desarrollo o una implantación, y tratarse de un trabajo teórico, los riesgos serán pocos, vamos a listar posibles riesgos que se pueden detectar en el desarrollo del trabajo, separando los relativos a la encuesta y los relativos al proyecto en sí; a continuación, plantearemos las correspondientes acciones correctivas:

- Riesgos del instrumento de recogida de datos:

ID	Riesgo	Detalle	Probabilidad	Impacto
R001	Encuestas insuficientes	No se obtienen suficiente número de respuestas a la encuesta (100)	Media	Medio
R002	Respuestas no concluyentes	Las respuestas obtenidas no nos son de utilidad para obtener información	Medio	Alto

- Acciones correctivas (instrumento de recogida de datos):

ID	Riesgo	Acción
A001	R001	Se ampliará el tiempo de realización de encuestas
A002	R002	Se buscará un segmento de población concreto
A003	R002	Se modificarán las preguntas de la encuesta

- Riesgos de planificación:

ID	Riesgo	Detalle	Probabilidad	Impacto
R101	Problemas lenguaje Python	Python no aporta la funcionalidad requerida para hacer la operación	Baja	Alto
R102	Problemas de permisos	No se tiene acceso a servicios para realizar la prueba	Media	Medio
R103	Problemas de licencias	El software necesario no es gratuito y no se tiene licencia	Media	Media
R104	Falta de tiempo	Durante la ejecución se detecta que no es posible finalizar todos los apartados planificados	Media	Alta
R105	Falta de documentación	No se encuentra información suficiente para algún apartado	Baja	Media

- Acciones correctivas (planificación):

ID	Riesgo	Acción
A101	R101	Se buscaría una librería que pueda aportar la funcionalidad requerida.
A102	R101	Se utilizaría un lenguaje de programación alternativo que cubra esa posibilidad.
A103	R102	Se buscará un servicio similar con acceso libre
A104	R102	Se planteará el comportamiento teórico sin realizar la prueba
A105	R103	Se utilizará un software equivalente con licencia gratuita
A106	R103	Se planteará la opción de usar una licencia de prueba si el software lo permite
A107	R104	Se solicitarán vacaciones en el trabajo para poder dedicar jornadas completas al proyecto
A108	R104	Se reducirá alguno de los apartados
A109	R105	Se buscará información en idiomas distintos al inglés y español para ampliar las fuentes

1.6.2. Seguimiento riesgos

En el presente apartado vamos a indicar el seguimiento de los riesgos definidos y si han sido necesarias acciones correctivas:

R001 → Tras finalizar el plazo, el número de encuestas ha superado el definido (105), por lo que no han sido necesarias acciones correctivas.

R002 → Tras el análisis de los resultados, se han llegado a conclusiones con la población encuestada, por lo que no han sido necesarias acciones correctivas.
R101 → Nos hemos encontrado con problemas con la versión 3.7 de Python, ya que no permitía realizar la opción de inverso modular, lo cual ha sido solventado haciendo uso de la versión 3.10 (**A101**)
R102 → Se ha intentado remitir una petición de sello al endpoint "https://qtsa.cert.fnmt.es" de la FNMT, pero no hemos obtenido respuesta, por lo que ha sido necesario aplicar la acción **A103** y optar por otro endpoint (ACCV – http://tss.accv.es:8318/tsa)
R103 → La versión actual de ASN.1 Editor es de pago y no permite opción de prueba, por lo que ha sido necesaria aplicar la acción correctiva **A105**, aunque la alternativa no ha sido un software distinto si una versión previa del mismo software, la "2008.09.29 1.0.20", que sí es gratuita.
R104 → No se han encontrado problemas con el tiempo disponible.
R105 → No hemos tenido problemas por falta de documentación.

1.7. Breve sumario de productos obtenidos

Se espera la obtención de los siguientes productos:

- Información teórica de funcionamiento de firma electrónica orientada a ciudadanos
- Información teórica de carácter técnico del funcionamiento de firma electrónica plasmada en el trabajo – memoria.
- Pequeños códigos de programación de apoyo para el cálculo

1.8. Estado del arte

Actualmente, podemos encontrar diversas fuentes de información con respecto a los certificados digitales y la firma electrónica; a nivel de información para el usuario, podemos encontrar fuentes escritas, como "Firma Digital, Certificado Electrónico y Factura Electrónica", escrito por Ana García Alcázar orientado a explicar al usuario final estos temas o la propia web "firmaelectronica.gob.es" creada por parte del Gobierno de España con un objetivo similar, en este caso, para su consulta online.

Adicionalmente, la firma electrónica y transacciones telemáticas basan su legalidad (con respecto a España) en:

- **Ley 6/2020 de Firma electrónica** [5], que regula, entre otros, la caducidad y revocación de los certificados, la información a emitir en los mismos, las obligaciones de los PSC (Prestadores de Servicios de Comunicación).
- **Reglamento 910/2014 (eIDAS)** [6], que regula, entre otros, los niveles de seguridad en los sistemas de identificación, las responsabilidades y sanciones, los requisitos de seguridad de los PSC, los servicios de los prestadores cualificados, los requisitos para firmas avanzadas y cualificadas.

- **Política de Firma de la AGE** [7], que establece la información y estructura necesaria al firmar un documento en especial para trámites con la Administración Pública.

A nivel de seguridad no hemos encontrado documentación específica, no obstante, el funcionamiento de la firma electrónica y certificados digitales se basa en conceptos criptográficos, y al respecto podemos encontrar documentación cómo:

- "Técnicas Criptográficas de protección de Datos" de Amparo Fúster Sabater, Dolores de la Guía Martínez, Luis Hernández Encinas, Fausto Montoya Vitini y Jaime Muñoz Masqué (Editorial Ra-Ma)

Por último, debemos indicar que con respecto a los propios certificados y las firmas electrónicas:

- Los certificados electrónicos se basan en X.509 que se encuentra definido en el RFC 2380 [8]
- Las firmas electrónicas disponen de diferentes estándares en función de su formato, siendo los más comunes:
  - **XAdES**: Formato de firma electrónica que se basa en XML, podemos encontrar información al respecto en https://www.w3.org/TR/XAdES/
  - **PAdES**: Formato de firma electrónica que se basa en PDF, podemos encontrar información al respecto en diferentes normativas, siendo la primera de ellas la ETSI TS 102 778-1[9]
  - **CAdES:** Formato binario de firma electrónica que podemos encontrar en el ETSI TS 101 733 [10]

Teniendo en cuenta la encuesta planificada anteriormente, y la información actualmente disponible, como la citada en este apartado, así como posibles referencias que citaremos en los apartados correspondientes, intentaremos sintetizar y analizar el uso, funcionalidad y seguridad de la tecnología de firma electrónica y certificados digitales.

1.9. Breve descripción de los otros capítulos de la memoria

Encuesta: en esta sección se detallará la encuesta que realizaremos a los ciudadanos, qué preguntas haremos y qué herramienta usaremos. Asimismo, se incorporará un segundo apartado valorando los resultados obtenidos.

Análisis: este capítulo englobará el estudio teórico y técnico dividido en las siguientes secciones:

- **Análisis ético, limitaciones y obligaciones**: En esta sección se analizarán qué obligaciones existen con respecto a los trámites digitales y que limitaciones pueden encontrar sus usuarios.

- **Bases firma electrónica**: En esta sección se estudiará el funcionamiento de las bases de la firma electrónica, diferenciándose dos categorías: criptografía y *hashes*.

- **Firmas – Proceso de firma y clasificación**: En esta sección comprobaremos cómo funciona técnicamente el proceso de generación de una firma y cómo pueden clasificarse las firmas electrónicas en función de diferentes criterios.

- **Certificados – Seguridad en emisión / generación**: En esta sección se revisará el proceso de emisión de certificados y que problemas pueden detectarse al respecto.

- **Certificados – Seguridad en el uso**: En esta sección se revisará si puede existir algún problema de seguridad en el uso habitual de los usuarios de los certificados y si es posible evitarlo.

- **Certificados – Estructura e información**: En esta sección se estudiará como se cumplimentan los certificados: qué información recogen y cómo se limita y estructura.

- **Certificados – PSCs**: En esta sección es analizará cómo funcionan las entidades emisoras de certificados con respecto a su estructura, emisión y requisitos a nivel legal y de estándar.

- **Sellos de tiempo - Firmas longevas**: En esta sección se revisará cómo conseguir firmas que sean validables tras la caducidad o revocación de un certificado y cómo funcionan los sellos de tiempo; cómo se generan y cuáles son sus requisitos.

Futuro – computación cuántica: En esta sección revisaremos cómo puede afectar la computación cuántica.

Caso práctico: estudiaremos cómo puede aplicarse la tecnología de firma electrónica para un caso de nuestro día a día, ajeno a los propios trámites electrónicos.

2. Encuesta

2.1. Definición y alcance encuesta

Tal como hemos definido en nuestro planteamiento, como punto de partida queremos conocer el estado de uso y conocimiento de los ciudadanos sobre la firma electrónica y los certificados digitales, y conocer que reticencias o desconocimiento pueden tener al respecto.

Objetivo: la encuesta nos servirá para obtener de primera mano información sobre el nivel de conocimiento de la firma electrónica y certificados digitales, qué nivel de seguridad creen los ciudadanos que aporta y qué motivos pueden ocasionar que no usen esta tecnología.

Instrumento de recogida de datos: como medio para obtener la información elaboraremos una encuesta mediante formulario web. Al generarse un enlace que lleva a la misma, es fácilmente accesible, ya que puede compartirse por correo electrónico, aplicación de mensajería o red social, lo que ayuda a llegar al mayor número de personas. En concreto, usaremos los **formularios de Google**, ya que permite generar las encuestas de forma gráfica sin necesidad de realizar un desarrollo a medida.

Se ha realizado una prueba previa confirmando las siguientes ventajas de su uso:

- Gratuito
- Permite obtener un enlace o remitir la encuesta por mail
- No necesita que los destinatarios tengan una cuenta de Google
- Genera gráficos automáticos para las respuestas a las preguntas susceptibles de ello

De las opciones por defecto, desactivaremos el *check* “Mostrar enlace para enviar otra respuesta”, ya que no queremos instar a realizar más de una encuesta por persona.

Datos a recoger: Sólo existirá una limitación respecto a la población, la edad, optando sólo por personas mayores de edad (mayores de 18 años), ya que son más susceptibles de hacer uso de transacciones que impliquen la firma digital, y actualmente un menor de edad tendrá opciones limitadas para su uso.

La encuesta tendrá dos bloques, el demográfico y el apartado referente a la investigación. En el **Anexo 1 se listan las preguntas concretas**.

Procedimiento: Queremos que la encuesta sea realizada por el mayor número de personas posibles, por tanto, sólo haremos un total de 10 preguntas (sin contar el apartado demográfico), ya que deseamos no abrumar a los encuestados. La encuesta estará abierta durante 10 días, tras los cuales se limitará el acceso para poder procesar los datos sin que aumenten. Tras ello, se exportará la información en formato gráfico y analizarán los resultados (siguiente apartado, 2.2).

2.2. Resultados y análisis encuesta

En este apartado vamos a mostrar la interpretación de los resultados, pero no vamos a listar, como es de esperar, todas las respuestas, no obstante, se han exportado las mismas y se adjuntará como anexo en un fichero .csv.

Este análisis nos sirve para, por una parte, dar respuesta a nuestro primer objetivo del trabajo (Identificar la recepción, conocimiento, uso y dudas de la firma electrónica y los certificados por parte de los ciudadanos), y, por otra parte, como punto de partida para abordar el resto de los objetivos definidos anteriormente.

Con respecto a los datos demográficos, se han obtenido datos de un amplio espectro:

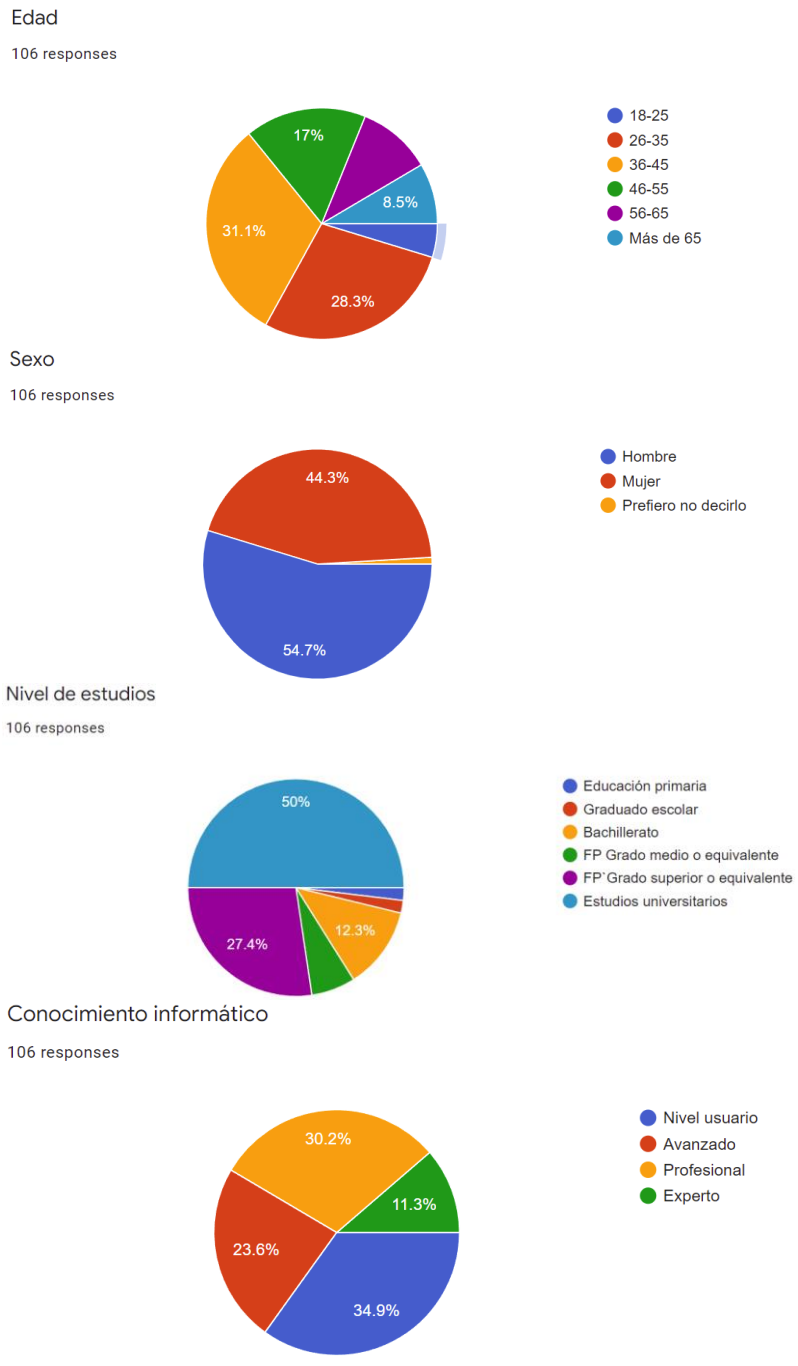


Ilustración 2 - Encuesta Resultados Demográficos

Por tanto, parece que tenemos una muestra que contempla diferentes tipos de usuario, ya sea por género, edad, estudios o nivel informático, lo cual parece es acertado para no dejar fuera algún grupo de personas que pueda tener un punto de visto que no quede reflejado.

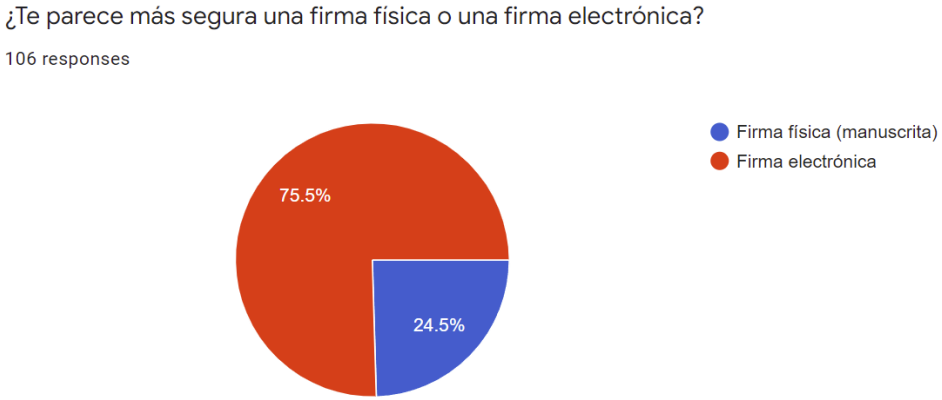


Ilustración 3 - Encuesta Firma física vs electrónica

Con respecto a la primera de las preguntas, podemos observar que hay un importante número de personas (25%) que consideran más segura la firma física (manuscrita) que la digital, por tanto, tenemos que considerar realizar esta comparativa posteriormente.

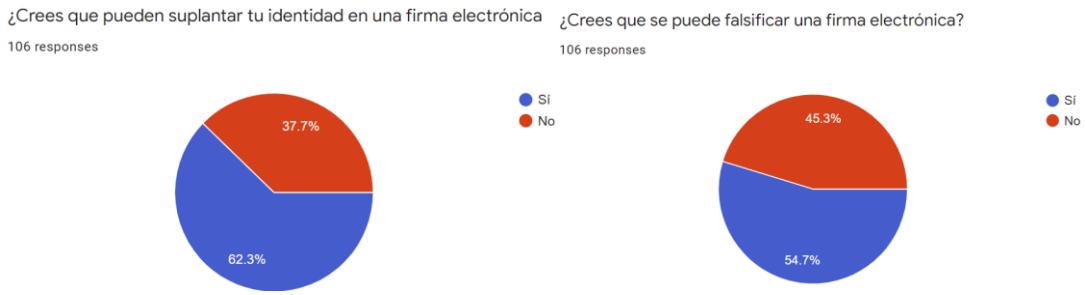


Ilustración 4 - Encuesta Suplantar y Falsificar

Si nos fijamos en las siguientes preguntas, más de la mitad de los encuestados consideran que es posible bien suplantar la identidad o bien falsificar una firma electrónica, por tanto, en un apartado posterior vamos a comprobar si es posible.

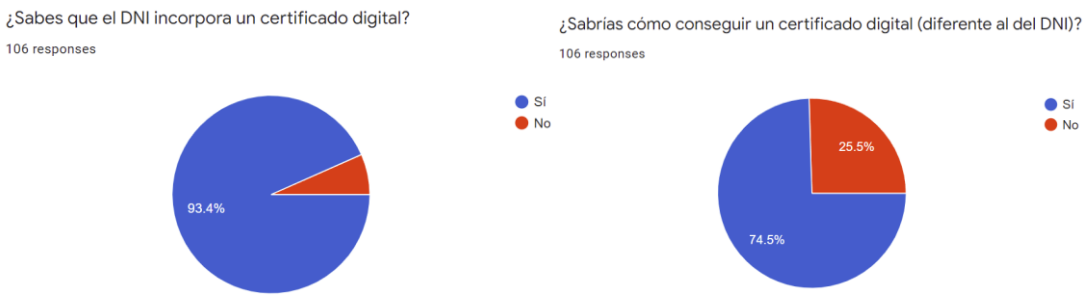


Ilustración 5 - Encuesta DNI

Si bien, la mayoría de las personas es conocedora de la existencia de un certificado en el DNIe, y, por tanto, parece que dicha información sí ha llegado a la mayor parte de la población, un porcentaje importante, aunque no mayoritario, no sabría cómo obtener otro certificado, por lo que tendremos que considerar explicar este trámite y comprobar qué pasos pueden ser necesarios.

De la pregunta abierta con respecto a los problemas de seguridad o confianza, se pueden destacar las siguientes respuestas:

- Varias personas muestran preocupación porque otras personas hagan uso de su certificado.
- Muestran preocupación por que sea atacada la web con la que se hace el trámite.
- Varias personas tienen miedo a que puedan ser *hackeados*.
- Preocupación por suplantación de identidad si se dispone de tu clave privada.

De la encuesta realizada sacamos las siguientes conclusiones:

- La mayoría de las personas confía en las firmas digitales y consideran que son seguras, pero todavía existen personas a las que no les inspiran confianza y creen que incluso las firmas físicas pueden ser más seguras.
- Las personas que han hecho uso del certificado lo consideran, salvo excepciones, un trámite sencillo, por lo que no parecen existir problemas en el funcionamiento en sí, sino en la falta de información y formación.
- Hay personas que creen que es posible falsificar o suplantar la identidad haciendo uso de la firma electrónica, en el análisis posterior revisaremos que opciones hay al respecto.

3. Análisis

Tras la introducción al trabajo y la encuesta previa, vamos a ir respondiendo a las cuestiones planteadas y analizando los diferentes aspectos de los certificados y la firma electrónica en diferentes apartados. Se elaborará una web básica (dos páginas estáticas a modo de FAQ, referenciada en los anexos), considerando lo estudiado en este apartado.

3.1. Análisis ético, limitaciones y obligaciones

Las nuevas tecnologías en muchas ocasiones nos aportan una accesibilidad y simplicidad que hacen nuestro día a día más fácil, pero no todos los usuarios son iguales.

Un ejemplo este tipo de problemática nos lo podemos encontrar en el funcionamiento con los bancos, la atención personalizada cada vez es menor y las entidades bancarias centran sus servicios en cajeros y especialmente en la banca online. Sin embargo, en especial las personas mayores se están viendo afectadas por ello, ya que, aunque no exista una obligación literal que obligue a tener una cuenta corriente, si no se hace uso de una entidad bancaria, los trámites del día a día son muy complicados y hay actividades que serían imposibles, por ejemplo, al no permitirse los pagos en efectivo por encima de los 1.000€ [11], toda acción que requiera un pago de este importe quedaría imposibilitada.

Teniendo en cuenta esto, podemos encontrar en una reciente encuesta (Febrero 2022) realizada por la revista 65ymas que más del 80% de los encuestados (personas mayores de 50 años) demanda atención personalizada a las entidades bancarias [12].

Por todo ello, tenemos que revisar si para el caso que nos ocupa (uso de certificados digitales y firma electrónica) existe algún tipo de limitación que impida o dificulte su uso y pensar cómo podemos mejorar estas posibles barreras.

Antes de ello, y en alusión al símil realizado con las entidades bancarias, vamos a revisar que nivel de obligación existe actualmente en el uso de esta tecnología, si es algo necesario, recomendable, obligado o meramente opcional.

Según el Ministerio del Interior [13], basándose en la Ley 39/2015 sí existe obligación de relacionarse con medios electrónicos, concretamente: las personas jurídicas, las entidades sin personalidad jurídica, personas que requieran colegiarse, representantes de los anteriores casos y empleados públicos (en algunas situaciones)

Adicionalmente, según podemos comprobar por parte de la Seguridad Social [14], teniendo en cuenta la Orden Ministerial ESS/214/2018 desde el pasado 1 de marzo del 2018 también tienen la obligación de realizar los trámites de forma electrónica los trabajadores autónomos, los trabajadores del Sistema Especial de Trabajadores Agrarios (SETA) y los del grupo I del Régimen Especial de Trabajadores del Mar.

Ya hemos, por tanto, encontrado varios grupos de ciudadanos para los que el uso de medios telemáticos en sus trámites es una **obligación**.

No obstante, vamos a revisar si se trata también de una opción recomendable, lo cual hace que para el resto de ciudadanos sea una ventaja aunque no una obligación, para ello, vamos a comparar algún trámite para tener una prueba real de ello:

Tramitación de Cambio de Domicilio:

En la dirección “https://cambiodomicilio.redsara.es” se permite al ciudadano comunicar el Cambio de Domicilio, podemos comprobar que es un trámite sencillo:

En primer lugar, se nos redirige para seleccionar método de acceso, entre los que podemos observar el Certificado Digital:

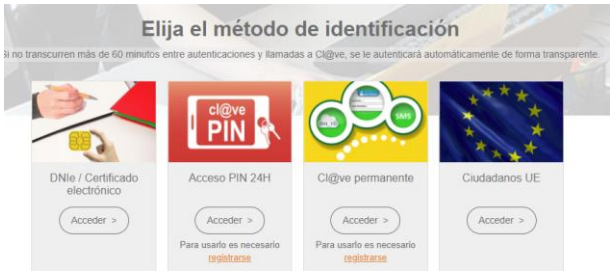


Ilustración 6 - CambioDomicilio Identificación

Tras ello, seleccionamos:

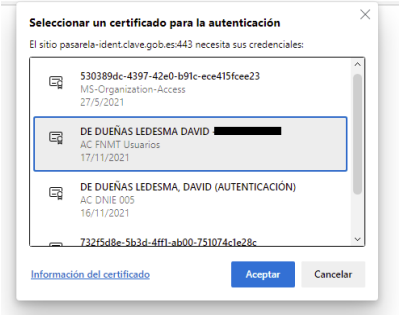


Ilustración 7 - CambioDomicilio Selección Certificado

Seleccionamos las entidades a comunicar:

Ilustración 8 - CambioDomicilio Selección Entidades

Y en el paso anterior carga la dirección que nos consta en el empadronamiento para trasladarlo a las entidades que seleccionamos en el paso previo:

Ilustración 9 - CambioDomicilio Tramitar

Como hemos podido observar, en 4 pasos que nos han llevado 5 minutos de nuestro tiempo podemos remitir la información hasta 6 Organismos públicos:

TGSS, Asturias (sólo residentes en dicha Comunidad), INSS, AEAT, DGT (Dirección General de Tráfico) y DGP (Dirección General de Policía).

En el caso de realizar el trámite de forma no telemática, deberíamos acudir a cada uno de los citados Organismos, con la consiguiente pérdida de tiempo y posible dinero (desplazamiento), por lo que el beneficio en este trámite es muy claro, además, como hemos visto, los pasos han sido pocos y sencillos.

Tramitación de Permiso Maternidad / Paternidad:

La Seguridad Social nos permite de forma telemática solicitar la prestación por nacimiento (maternidad / paternidad).

Para ello, accedemos al portal de Seguridad Social (https://sede-tu.seg-social.gob.es/):

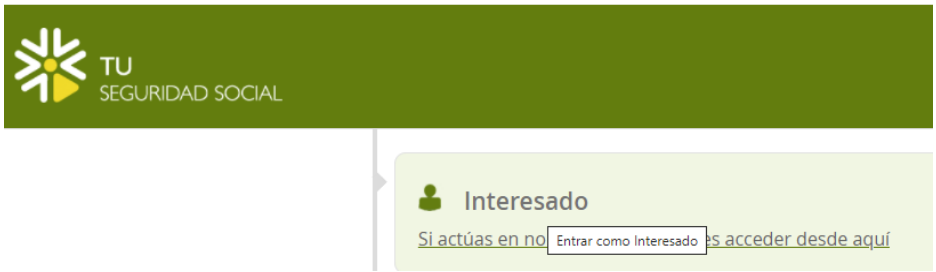


Ilustración 10 - SegSocial Acceso

Nos permite identificarnos con diferentes opciones, entre las que se encuentra DNIe – certificado:

Ilustración 11 - SegSocial Opciones

Tras ello, seleccionamos las opciones (no se muestran más detalles por privacidad) y tendremos el cálculo de nuestra prestación:

Ilustración 12 - SegSocial Prestación

Hemos podido realizar el trámite, tal como en el caso anterior, en unos sencillos pasos y sin necesidad de desplazarnos ni emplear un gran periodo de tiempo.

En el caso de tramitación presencial, el trámite sería más largo, ya que tal como se indica en la propia página https://revista.seg-social.es/-/c%C3%B3mo-pedir-la-prestaci%C3%B3n-por-nacimiento-y-cuidado-de-menor es necesario solicitar cita previa y posteriormente acudir en persona o bien remitir la información vía correo ordinario.

Hemos, por tanto, encontrado otro trámite, en el que si bien no es obligado su tramitación electrónica, el funcionamiento es más eficiente. Podemos incluso encontrar noticias [15] en las que se recomienda el trámite online, ya que los problemas y retrasos derivados del trámite en persona son mucho más elevados, de la propia noticia: "*Por ello, para aquellas solicitudes nuevas, se recomienda la presentación online, ya que el proceso de automatización de la gestión de los expedientes es mucho más ágil y el plazo de resolución es más rápido.*"

Con estas pruebas hemos podido, por tanto, llegar a las siguientes conclusiones para seguir nuestro análisis:

- Existen grupos de personas **obligados** por ley a realizar trámites vía digital.
- Hay trámites digitales que nos **benefician** ahorrando gestiones y tiempo.
- El funcionamiento ineficaz de algunos trámites personales **insta a usar** los servicios digitales en su lugar.

Teniendo en cuenta que gracias a este análisis previo hemos justificado que el uso de certificados digitales no es realmente opcional en muchos casos (sea por obligación, por los problemas o ineficiencias de no usarlo), vamos a revisar que limitaciones se podrían encontrar los usuarios por verse abocados a usar este tipo de tecnología:

- **Limitación económica**: una posible limitación a la que se puede enfrentar un usuario es que el coste de uso o adquisición de la herramienta para optar al servicio haga que el usuario desestime el mismo.

  En cuanto al coste de disponer de certificado digital, podemos contemplar dos opciones:

- Certificado DNI electrónico: Se incluye en la emisión del DNI [16], el certificado en sí es gratuito, con una caducidad de años, el precio del documento varía en función de la situación del ciudadano.

- Certificado FNMT: Se emite a cualquier ciudadano por parte de la Fábrica Nacional de Moneda y Timbre, como podemos comprobar en su web [17] es gratuito

Como herramienta para el uso del certificado digital, en el caso de FNMT no es necesaria, ya que es un certificado software, en el caso del DNIe se necesita un lector de tarjetas, cuyo precio puede rondar los 10 euros.

Teniendo en cuenta esta información, el uso de la firma digital no implica un desembolso económico que pueda ser limitante, por tanto, aunque puedan existir casos extremos en que no sea posible este desembolso, podemos descartar que el uso de esta tecnología se encuentre limitada por el poder adquisitivo del ciudadano.

- **Limitación de disponibilidad**: otra posible limitación es que los recursos necesarios sean limitados, lo cual no ocurre en este caso, ya que en el caso de la FNMT los certificados son en software y, por tanto, se pueden generar siempre que sean necesarios y en el caso del DNI electrónico no se ha conocido hasta la fecha ningún problema en la emisión de los mismos por limitaciones físicas.

  Por tanto, podemos descartar esta limitación, ya que es un recurso disponible para todo aquel que lo desee.

- **Limitación tecnológica**: como hemos visto, existen principalmente dos tipos de certificados, software y en tarjeta. Las pruebas realizadas previamente han sido realizadas con un certificado software, por lo que tenemos que añadir que sí se usa un certificado en tarjeta, como el DNIe, que además es el que por defecto pueden disponer fácilmente todos los ciudadanos:

  - Implica introducir el certificado en el lector en cada uso: aunque no sea una limitación si puede ser más incómodo.
  - Implica adquirir e instalar un lector de tarjetas: esto sí puede considerarse una limitación adicional, ya que obliga al ciudadano a realizar una gestión adicional, y aunque la configuración / instalación del lector se presume fácil, es posible que un usuario con pocos conocimientos informáticos pueda tener problemas.

  No obstante, a partir del DNIe 3.0 se permite el uso de NFC lo cual evitar el uso de un lector de tarjetas. Sin embargo, el problema es que por parte del emisor (DGP-FNMT) no existe una aplicación única que sirva reemplazar al lector, sino que existen o aplicaciones individuales para accesos concretos, o bien una que centraliza ciertos trámites con la administración [18]; por tanto, realmente en estos momentos el uso de NFC no es realmente un sustituto del lector y el hecho de que existan un gran número de aplicaciones para usar el NFC sin existir una única oficial puede hacerlo incluso más tedioso que la necesidad del lector.

Existe una versión más reciente del DNIe, la 4.0 [19], no obstante, a nivel del ciudadano en esta versión sólo puede encontrar cambios meramente estéticos, por lo que no va a variar su uso por el momento, aunque está previsto que esté DNIe 4.0 pueda gestionarse desde una aplicación móvil, en cuyo caso quizás sea una ventaja para el ciudadano y sea más accesible, pero esta funcionalidad no está aún disponible**.**

Por todo ello, que el certificado más fácil de disponer por parte de los ciudadanos sea en tarjeta puede suponer una limitación.

- **Limitación de conocimiento**: por último, podemos encontrar las limitaciones que pueden suponer la falta de información o conocimiento. En este sentido, tal como hemos podido comprobar en el apartado anterior relativo a la encuesta, existe un grupo de personas que desconoce cómo usar esta tecnología, sin embargo, si contrastamos con la opinión de las personas que sí lo han usado, vemos que más del 90% de usuarios lo ven sencillos y por las pruebas anteriormente indicadas así parece ser, por tanto, esto nos hace separar a las personas con estas limitaciones en dos grupos:

  - Personas que creen que se trata de una tecnología difícil, pero que una vez utilizada cambiarían su opinión.
  - Personas que por sus conocimientos les resultaría complicado, al igual que otro tipo de trámites electrónicos.

Podemos encontrar diversos estudios relacionados al respecto, por ejemplo, “Marek van de Waterning” [20] en su tesis sobre el impacto de tecnología y ordenadores en la población mayor encontró que existen barreras para este tipo de población como el miedo a lo desconocido, el desapego a algo que no les parece relevante y problemas con la accesibilidad que se podrían reducir siguiendo desarrollos y guías acordes a sus necesidades.

Si consultamos el uso de conocimiento informático por edades según vemos en el INE [21] podemos ver reflejada la siguiente información:

Utilizar al menos un conocimiento informático	
Total	
De 16 a 24 años	
2021	96,0
De 25 a 34 años	
2021	91,7
De 35 a 44 años	
2021	85,9
De 45 a 54 años	
2021	79,3
De 55 a 64 años	
2021	63,4
De 65 a 74 años	
2021	39,1

Ilustración 13 - Uso informático por edades INE

Lo cual nos enfatiza que el problema no se encuentra en sí sobre la tecnología en cuestión (firma digital / certificados electrónicos) si no sobre la falta de uso y conocimiento informático en general.

Pero estos problemas no se ciñen sólo a la edad, sino también a la implantación y acceso a las nuevas tecnologías en los medios rurales,

Teniendo en cuenta los resultados de la encuesta elaborada y el estudio realizado en este apartado, podemos llegar a una de las primeras conclusiones de este trabajo, confirmando que el uso de firmas electrónicas y certificados no está preparado para alcanzar a toda la población, encontrando especialmente problemas las personas mayores y/o con pocos conocimientos tecnológicos por lo que se proponen las siguientes acciones:

- Crear una guía de uso de la firma electrónica para ayudar a trasmitir el conocimiento.

- Las entidades públicas deberían fomentar el uso de nuevas tecnologías para aplicación directa en actividades que interesen o faciliten tanto a las personas mayores como a entornos rurales, de esta forma se puede atraer a estas personas a interesarse por las nuevas tecnologías, que pierdan el miedo y que cuando surjan tecnologías pasen a ser necesarias u obligatorias no supongan una barrera, como ejemplo podemos encontrar [22] [23]:
    - Drones para controlar pesticidas y fertilizantes
    - Robots para el control del ganado
    - Vehículos automatizados para granjas
    - Aplicaciones para que los ancianos que viven sólo contacten con la familia
    - Aplicaciones para hacer juegos y ejercicios que mejoren la memoria
    - Aplicaciones para fomentar el ejercicio en las personas mayores

- Crear espacios sean fijos o itinerantes dónde las personas con pocos conocimientos tecnológicos, independientemente de su causa, puedan ser ayudados a hacer los trámites digitales de forma que poco a poco vayan aprendiendo a ellos y también para que no les sea obligado el trámite presencial cuando existen otras alternativas.

- Conseguir el 100% de cobertura a Internet: aunque tal como indica el INE [24] la mayor parte de España tiene acceso a Internet (95%), es necesario que se cubra la totalidad del país, ya que aún quedan pequeñas poblaciones rurales que no disponen de esta opción, por tanto, en estos casos aunque tuviesen el conocimiento para usar los certificados digitales y la firma electrónica no podrían hacerlo, obligándoles a desplazarse, ya sea hasta un lugar con acceso a Internet o bien para hacerlo de manera presencial.

3.2. Bases firma electrónica

Como punto de partida, en este apartado vamos a introducir las bases usadas para la firma electrónica, hablando primero de la criptografía y posteriormente de los *hashes*.

3.2.1. Criptografía

La criptografía [43] es la base de las firmas digitales, pero ¿qué es la criptografía?, proveniente del griego (*criptos* = secreto/oculto), es la técnica utilizada para enmascarar una información de forma que sea confidencial.

Existen dos métodos básicos criptográficos:

- **Sustitución**: usar una correspondencia entre los caracteres utilizados en el texto y otro conjunto (que puede ser el mismo alfabeto o no) de forma que la clave es la que permite conocer la relación y, por tanto, conocer el texto original.

- **Transposición**: consiste en "barajar" los caracteres del mensaje de forma que se encuentren en otro orden, la clave en este caso nos serviría para poder ordenarlos correctamente y poder leer el mensaje origen.

Un ejemplo sencillo es el algoritmo César, que es un tipo de criptografía de sustitución, consistente en mover las letras del alfabeto X posiciones, por ejemplo:

Si tenemos el texto "Universidad" y usamos un algoritmo César de 2 posiciones, moveremos las letras dos espacios, así por ejemplo la A sería C y la D sería F, por tanto (hemos usado como ejemplo el alfabeto inglés, 24 caracteres):
Universidad (César 2) → Wpkxgtukfcf

Asimismo, dentro de la criptografía encontramos dos posibilidades:

- **Criptografía simétrica**: se usa la misma clave para cifrar y descifrar el mensaje.

- **Criptografía asimétrica**: existen dos claves, una pública y otra privada, que se pueden usar para cifrar o descifrar. Este tipo de criptografía es la usada en la firma digital

Este par de claves guarda una relación matemática (como veremos posteriormente en un ejemplo de proceso de firma). Es destacable que dicho par de claves suele tener dos usos:

Remitir **información cifrada** a un remitente: en este caso se usa la clave pública del destinatario para cifrar el mensaje, de esta forma, sólo se podrá descifrar con la clave privada que posee el destinatario, por tanto, nadie salvo él podrá leer el mensaje.

Realizar la **firma de un dato** (firma digital): en este caso se usa la clave privada para firmar la información, de forma que el propietario de la clave privada sea el único que pueda realizar el proceso, usándose su clave pública para validar la firma, pero siendo imposible generarla.

Las operaciones matemáticas que se llevan a cabo en cada caso son distintas, así, existen algoritmos que pueden hacer las dos cosas, como RSA, y otros que sólo pueden cifrar, como DSA.

Aunque el algoritmo difiera en la forma de hacer cada cálculo, el proceso en todos los casos es el mismo, el cual resumimos junto con la opción de cifrado:

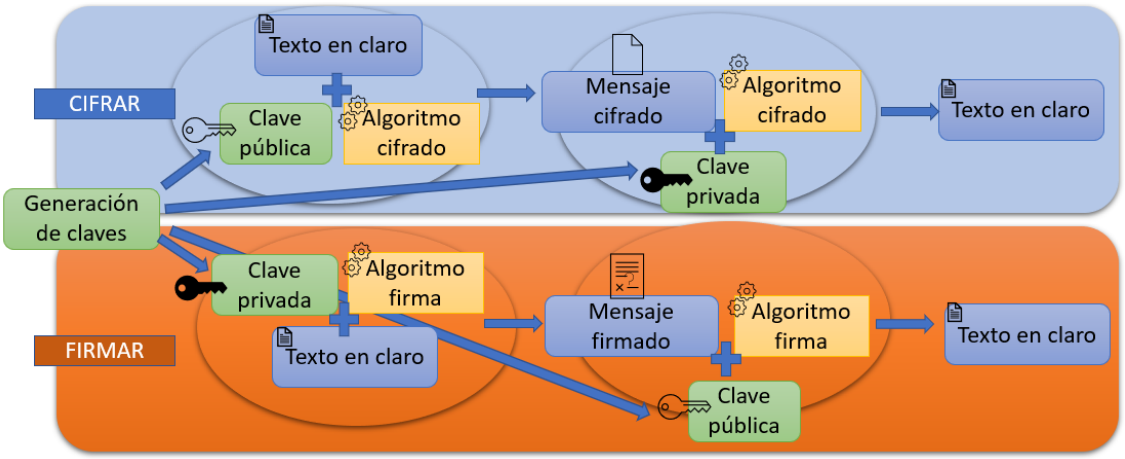


Ilustración 14 - Algoritmo firmar - cifrar

3.2.2. Hashes

Al hablar de firmas electrónicas, otro elemento que debemos revisar son las funciones *hash*. Las funciones *hash* son usadas, generalmente previo a la realización de la firma en sí, por dos motivos:

- Reducir el tamaño de la firma de modo que sea más fácil de procesar, ya que el proceso de generación de firma se realiza haciendo cálculos complejos matemáticos y siempre serán más ágiles con ficheros más pequeños.
- Conseguir que la propia firma tenga menor longitud, y así ser más fácil trabajar con ella. Por ejemplo, el contenido a incluir en una firma de un fichero de 1Mb, 5Mb o 10Mb quedaría reducido a 1Kb si le aplicamos un Hash de 256 bits.

El esquema de funcionamiento de un algoritmo *hash* sería el siguiente:

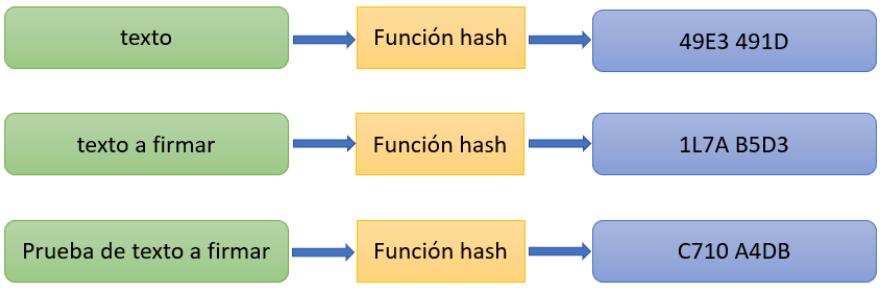


Ilustración 15 - Esquema funcionamiento hash

Basándonos en dicho esquema, una forma de simple de construir un algoritmo *hash* es usando la operación módulo, dicha operación consiste en calcular el resto de dividir dos términos, esto garantiza un número limitado máximo en el resultado, por ejemplo,

si usamos mod 555 sabemos que como mucho vamos a obtener 554, independientemente del número origen, si completamos el resultado para resultados inferiores a 3 cifras con 0 ya tenemos nuestra función:

H(x)=x (mod 555)

Sin embargo, tal como podemos ver en el libro “Técnicas criptográficas de protección de datos” [43], el funcionamiento resumido de este caso sería:

- Usuario A calcula el *hash* del mensaje que quiere firmar → $MH_1 = H(M_1)$
- Usuario A firma el *hash* → $Firma_1=Firma(MH_1)$
- Usuario B calcula la misma firma → $Firma_2= Firma(MH_1)$, con esto sabe que la firma es correcta.
- Usuario B calcula el *hash*, $MH_2=H(M_1)$, si , MH_1 y MH_2 coinciden puede confirmar que se ha firmado el dato que se esperaba.

Esta función que hemos creado puede que no sirva para esta premisa, ya que cómo podemos ver, es deseable poder asociar el fichero firmado con el *hash* para poder confirmar el dato origen con el usado para la firma y para ello es necesario que los resultados del hash sean siempre distintos; vamos a ver si esto se cumple en nuestro caso.

Hemos creado en Python una función que genera números aleatorios y otra que calcula el *hash* definido anteriormente, vamos, por tanto, a generar números y ejecutar nuestra función continuamente hasta que encontremos dos resultados iguales:

Prueba	Nº de intentos	Valor origen 1	Hash1	Valor origen 2	Hash 2
1	41	149936	86	844796	86
2	32	854478	333	245643	333
3	6	386085	360	251220	360
4	19	826376	536	166481	536
5	43	290055	345	150750	345
6	26	614786	401	809591	401

Ilustración 16 - Hash duplicados

Aunque la función nos sirva para definir el concepto y funcionamiento de un *hash*, confirmamos que una función hash tan sencilla o con sólo estas propiedades no vale para nuestro propósito, de hecho, si nos referimos al artículo publicado por Delton Rhodes [44] las 5 propiedades que deben tener las funciones *hash* cuando se usan para criptografía son:

- **Determinista**: esta propiedad hace referencia a que siempre que se indique como entrada el mismo valor, se obtendrá el mismo *hash* resultante, esta propiedad si la hemos conseguido y quizás es la más sencilla de obtener, basta con no incluir elementos aleatorios en el cálculo del *hash*.

- **Rápida (computacionalmente eficiente)**: teniendo en cuenta que las firmas u operación criptográfica para la que se use el *hash* debe hacerse en el momento que el usuario lo solicita, es necesario que el cálculo sea eficiente. En la propia web citada, podemos comprobar que algoritmos utilizados ampliamente como SHA-512 o SHA-256 tienen todas ellas tiempos inferiores a 1000 MiBps.

Vamos a hacer una prueba de velocidad de funciones *hash* mediante un programa en Python que nos permita conocer la diferencia de velocidad en función del algoritmo, para ello veremos qué algoritmo calcula más *hashes* durante el mismo tiempo (30 segundos) usando lógicamente todos ellos el mismo origen:

Posición	Algoritmo	Nº de hashes (5 segundos)	Nº de hashes (30 segundos)	Nº de hashes (1 minuto)
1	SHA-1	1.541.709	8.864.555	17.683.877
2	Md5	1.241.658	6.982.526	13.859.203
2	SHA512	1.108.801	6.589.293	13.042.795
3	Blacke2b	1.050.477	6.042.722	12.205.335
5	SHA256	892.018	5.040.272	9.807.418
6	Blacke2s	702.604	4.304.013	8.494.621

Ilustración 17 - Hash velocidad

Por lo que, a nivel de velocidad, el mejor algoritmo (al menos en su implementación en Python) es SHA-1, aunque posteriormente veremos que no es aconsejable su uso.

- **Unidireccionales**: No es posible conocer el dato origen usado para la generación del *hash*, salvo que se prueben todos los valores posibles hasta encontrarlo. Esto se consigue realizando operaciones matemáticas tales que el cálculo del *hash* sea rápido, pero no exista operación matemática inversa directa que nos devuelva el origen.

- **Resistente a colisiones**: Esta propiedad se refiere a que sea muy poco probable encontrar dos orígenes que obtengan el mismo hash. Esta propiedad la hemos visto incumplida en nuestra función de ejemplo.

Para obtener una aproximación debemos hacer referencia al problema del cumpleaños que calcula qué probabilidad existe de que dos personas al azar cumplan años el mismo día, esto es una paradoja, ya que el resultado es que hay un 50% de posibilidades cuando tenemos 23 personas, lo cual no parece intuitivo, ya que parece algo más complicado.

Teniendo en cuenta esto, [45] los ataques de este tipo suponen que un *hash* puede encontrar una colisión en $2^{n/2}$ pasos, una fórmula aproximada que nos sirve para ver cuando hay un 50% de posibilidades de encontrar una coincidencia, lo que en nuestro caso sería una colisión es la siguiente [46]:

$$n \approx \frac{1}{2} + \sqrt{\frac{1}{4} + 2 \times \ln(2) \times 365}$$

No obstante, en este caso se contemplan 365 posibilidades, que son los días del año, pero para nuestro caso debemos contar las opciones posibles en base a nuestro *hash*, así, por ejemplo, un *hash* de 8 bits, tendrá 2^8 opciones, es decir, 256.

Vamos, por tanto, a aplicar esta fórmula en Python para ver en función del número de bits las opciones de encontrar una colisión:

Nº de bits del hash	Nº de hashes posibles	Valores hasta 1ª colisión
8	256	19 (19,34)
16	65.536	301
32	4.294.967.296	77.163
64	$1{,}84467 \cdot 10^{19}$	5.056.937.541
128	$3{,}40282 \cdot 10^{38}$	$2{,}17193 \cdot 10^{19}$
256	$1{,}15792 \cdot 10^{77}$	$4{,}00651 \cdot 10^{38}$

Ilustración 18 - Hashes colisiones

Por tanto, podemos ver que sería recomendable usar un *hash* no inferior a 128 bits para que sea suficiente para evitar colisiones

- **Resistente a ataques de pre-imágenes**: Adicionalmente, a ser unidireccional (no existir operación matemática directamente inversa), también debe ser computacionalmente ineficaz para un atacante poder encontrar un mensaje origen que devuelve un hash en concreto.

Límites de las funciones *hash*: las funciones *hash* criptográficas tienen un máximo de tamaño que puede ser usado como entrada para el cálculo del hash. Este valor está determinado por el *padding*. El *padding* consiste en adaptar el mensaje origen para que tenga una longitud adecuada, ya que las funciones hash más comunes, de la familia SHA, realizando operaciones por bloques de bits y por tanto necesitan que sean múltiplo de un número. Por tanto, en función de esta implementación el límite varía, tal como podemos ver el Secure Hash Standard [47]:

- SHA-1 y SHA-256 usan un *padding* máximo de 64 bits, por lo que el tamaño del mensaje máximo será de 2^{64} bits, esto nos da un máximo aproximado de 2.090.000 Terabytes.
- SHA-512 usa un *padding* máximo de 128 bits, por lo que el tamaño del mensaje máximo será de 2^{128} bits, un gran número de Exabytes que ningún ordenador sería capaz de almacenar.

Hay funciones que en principio cumplían estos requisitos y eran utilizadas, pero han dejado de ser recomendadas por haberse considerado vulnerables, por ejemplo:

- **MD5**: En al año 2008 [48] se confirmó la teoría de que existe la posibilidad de sufrir un ataque por colisiones, que tal como hemos comentado anteriormente implica que dos orígenes distintos generen el mismo hash, lo cual puede servir para que un atacante consiga hacer creer que el dato usado para el origen ha sido otro distinto, al ser el cálculo resultante (hash) el mismo.
- **SHA-1**: Desde el año 2005 se han realizado diferentes análisis de ataques que han encontrado colisiones en partes del algoritmo, con cada vez menos esfuerzo computacional, hasta que en el año 2017 se consiguió demostrar una colisión entre dos ficheros PDF distintos [49] y, por tanto, por las mismas causas del caso anterior, dicho algoritmo debe dejar de ser usado por seguridad.

Con respecto a ejemplos concretos de funciones hash que cumplan estos criterios y que, por tanto, podrían ser usados en tareas criptográficas son:

- SHA-256: Tiene un *hash* resultante de 256 bits y hasta el momento (2022) no se conocen ataques de colisión que hayan sido exitosos.
- SHA-512: Tiene un *hash* resultante de 512 bits y hasta el momento (2022) no se conocen ataques de colisión que hayan sido exitosos.

Este apartado nos ha servido para conocer el funcionamiento de las funciones *hash*, conocer los beneficios de su uso para la firma digital, qué criterios de funcionamiento y seguridad deben cumplir para poder ser susceptibles de uso, así como sus limitaciones en cuanto a tamaño de origen.

3.3. Firmas – Proceso de firma y clasificación

Para analizar las firmas propiamente dichas, vamos a aportar un ejemplo para detallar como se generan las mismas y posteriormente explicar cómo podemos clasificar las mismas.

3.3.1. Proceso de firma

Ahora, basándonos en una firma XML (XAdES) cuyo estándar está indicado en [58] vamos a revisar cómo se realiza el proceso. Nos decantamos con este formato, ya que visualmente son más fáciles de tratar al no ser necesario un visor de código binario (como ocurre en firmas PAdES o CAdES), en el apartado posterior, detallamos los diferentes formatos de firma.

Su estructura es la siguiente:

Ilustración 19 - Estructura XAdES

Dentro de la firma (ds:Signature) podemos encontrar principalmente dos apartados:

- **SignedInfo**: Es el contenido sobre el que se realizará la firma
- **SignatureValue**: Es el valor calculado tras ejecutar el algoritmo de firma.

Ahora detallamos cada caso, empezando por **SignedInfo**:

En esta sección, se incluye:

- **CanonicalizationMethod**: es el estándar que se seguirá para canonicalizar, es decir, realizar las conversiones oportunas sobre el dato XML origen para que tenga un formado estándar (como puede ser, eliminar espacios en blanco

innecesarios). En este caso, se usa “http://www.w3.org/TR/2001/REC-xml-c14n-20010315”

- **SignatureMethod**: Algoritmo que se usará para generar el valor de firma (en este caso, SHA256)
- **Referencias**: estos son resúmenes de los elementos sobre los que se firman, previamente aplicándose la canonicliazación. En ellas se encuentra:
    - **DigestMethod**: algoritmo que se usará para el cálculo del hash
    - **DigestValue**: Hash en base64

Teniendo en cuenta esto, hemos creado una aplicación en Python que reproduce este comportamiento y nos sirve de utilidad para la comprobación de los DigestValue, su comportamiento es:

- Busca las referencias y extrae sus DigestValue
- Obtiene las URI de cada referencia
- Utiliza las URI para obtener los elementos sobre los que se ha calculado el DigestValue y se recalcula.
- Se comprueba el DigestValue extraído con el calculado, si coincide, esa parte de la firma es correcta.

En el anexo se referencia este programa.

Teniendo en cuenta lo anterior, el proceso de firma completo sería (se indican las etiquetas más importantes):

- Por cada referencia:
    - Se aplica la canonicalización (datoorigen1 se aplica REC-xml-c14n-20010315 → dato1Canonicalizado)
    - Se calcula el hash: dato1Canonicalizado se aplica SHA512 → Hash dato1Canonicalizado
    - Se calcula el base64: Hash dato1Canonicalizado se aplica base64 → b64 Hashdato1Canonicalizado
    - Se repite este cálculo por cada referencia y se incluye en el elemento SignedInfo en cada DigestValue, por lo que dicho elemento contiene: DigestValue1(Hashdato1Canonicalizado), DigestValue2(Hashdato2Canonicalizado)… DigestValuen(HashdatonCanonicalizado)
    - Se aplica la firma sobre el elemento SignedInfo: SignedInfo RSA-SHA256
    - Se convierte en base64: b64_RSA-SHA256_SignedInfo y se incorpora en el elemento SignatureValue

Ahora nos falta entrar en detalle sobre el algoritmo de firmado, en este caso, como vemos, se usa RSA, los pasos que sigue este algoritmo [42] son los siguientes:

- 1-. Generación de claves:
    - 1.1 Se seleccionan dos números primos (se recomienda que superen los 200 dígitos), que llamaremos *p* y *q*.
    - 1.2 Se calcula n multiplicando ambos números $n = p \cdot q$.
    - 1.3 Se selecciona un entero positivo (e) que sea menor que la función de Euler (φ) y sea coprimo

    - 1.4 Se utiliza el algoritmo de Euclides extendido para calcular el inverso de e, se cumple que $e \cdot d = 1 \pmod{\varphi(n)}$
    - 1.5 La clave pública es el conjunto (n,e)
    - 1.6 La clave privada es d
- 2-. Generación de firma
    - 2.1 Se calcula el hash del mensaje (m) a firmar h = hash(m)
    - 2.2 Se calcula $s=h^d \pmod{n}$.
- 3-. Verificación de firma
    - 3.1 Para verificar, se debe comprobar la siguiente ecuación: $c = m^e \pmod{n}$
    - 3.2 Se compara que coincidan ambos mensajes

Aunque existen funciones ya creadas en librerías que permiten ejecutar este algoritmo, vamos a hacer una aplicación en Python que sea capaz de aplicar este algoritmo paso a paso en base a la definición del algoritmo (en el Anexo se indica el programa en Python):

- 1-. Generación de claves:
    - 1.1 Se generan aleatoriamente los primos ***p* = 797** y ***q* = 587**
    - 1.2 ***n*** = 797·587 = **467839**
    - 1.3 ***φ(***467839***)*** = 466456
    - 1.4 ***e*** = coprimo aleatorio menor que 466456 = **440615**
    - 1.5 **d** = Inverso modular de e *φ(n)* → *d= inverso modular de* 440615 mod (466456)= ***398151***
    - 1.6 Clave pública = (n,e) = (**467839, 440615**)
    - 1.7 Clave pública = (n,d) = (**467839, 398151**)
- 2-. Generación de firma
    - 2.1 Se calcula el hash del mensaje (m) a firmar h = hash(m) (suponemos que el hash generado es 1234)
    - 2.2 $s=1234^{398151} \pmod{467839} = 404196$
- 3-. Verificación de firma
    - 3.1 $m= 404196^{440615} \pmod{467839} = 1234$
    - 3.2 Se compara que coincidan ambos mensajes

Como vemos, el algoritmo matemático que posibilita la firma mediante la parte privada permite la verificación usando la parte pública, pero la complejidad del mismo hace que no sea posible obtener la clave privada disponiendo de la parte pública. Asimismo, si se cambia cualquier valor del dato origen, el resultado sería distinto, esto hace que la firma sea segura y no pueda ser modificada.

3.3.2. Clasificación de firmas

En función de cómo la información de la firma se estructura, podemos encontrar principalmente tres formatos:

- **Firmas basadas en PDF**: Podemos encontrar su información en el estándar base definido en la ISO 32000-1:2009 [50]. Al incorporarse en el propio documento y el mismo ser visible por un lector de PDF es el más sencillo de visualizar para ciudadanos, ya que mediante los paneles de Adobe se puede acceder a la información de la firma. **Sólo es posible la firma de ficheros PDF** y no es posible firmar un *hash* del mismo, ya que la información se incorpora en el propio documento, por lo que nunca podremos hacer la firma más pequeña que el documento a firmar.

- **Firmas basadas en XML**: Podemos encontrar su información en el estándar base definido por W3C [51]. Este formato presenta la ventaja de su interpretación como XML y visualización del contenido por parte de las personas. **Permite firmar tanto XML como cualquier dato (binario).** A nivel de XML permite elegir nodos concretos o un XML concreto.

- **Firmas binarias**: Podemos encontrar su información en el estándar definido en la RFC 5652 [52]. En este caso, **también es posible firmar cualquier dato de origen**. Presenta el problema que la información se presenta en binario, por lo que sin un intérprete de dicha información no es posible visualizar la información contenida en la misma.

Todos estos formatos base, disponen de su versión avanzada para contener información adicional, concretamente:

- Firmas PDF → Formato PAdES → Definido en ETSI TS 102 778-1
- Firmas XML → Formato XAdES → Definido en https://www.w3.org/TR/XAdES/
- Firmas binarias (CMS) → Formato CAdES → Definido en RFC5126

Adicionalmente al propio formato indicado, podemos contemplar otra subcategoría:
- Firmas simples: en la firma sólo existe un firmante
- Firmas múltiples: en la firma existen varios firmantes.
    - **Firmas en serie (CounterSign)**; La firma se realiza sobre una firma ya realizada, esto tiene sentido cuando un trámite requiere un orden de firma, como puede ser una aprobación jerárquica. Tanto el formato XADES [53] que define esta opción con el elemento *CounterSignature* como el formato binario [54] que define el atributo *id-countersignature* permiten esta opción.
    - **Firmas en paralelo (coSign)**: Las firmas se añade sobre el mismo origen, por lo que no existe jerarquía entre las mismas, no se relacionan entre sí.

Otra importante forma de clasificar las firmas es respecto al nivel legislativo, lo cual podemos comprobar en el actual reglamento europeo 910/2014 [55] y el anterior, 1999/93/E [56], ya que en función de sus características regula que nivel:

- **SES (Simple electronic signature – Firma electrónica simple)**: En este caso sólo se cumple que unos datos en formato electrónico se vinculan a otros datos electrónicos para el uso de autenticación.

- **AES (Advanced electronic signature – Firma electrónica avanzada)**: se trata de una firma electrónica que cumple lo siguiente:
    - Permite identificar al firmante
    - Está vinculada de forma única al firmante
    - El firmante tiene capacidad de proteger su clave de forma que sólo él pueda realizar la firma
    - Los datos firmados están realizados de tal forma que cualquier cambio posterior de los mismos puede ser detectado (Integridad)

- **QES (Qualified electronic signature – Firma electrónica cualificada)**: este formato debe cumplir el requisito de las firmas avanzadas, y además ser generada por un certificado cualificado (ver sección “Funcionamiento PSCs - TSL”) y creada por un dispositivo seguro de creación (QSCD)

Para considerar un dispositivo seguro [57] debe contener altas garantías de proteger la clave con objeto de que sea usada sólo por su propietario. Por ejemplo, un certificado en tarjetas podría considerarse QSCD, ya que para acceder a la clave se requiere el acceso físico al mismo y una contraseña.

Es importante recalcar, que en el artículo 25 de la legislación referida, se indica que no se puede negar a efectos legales una firma cualificada y que es equivalente a una firma manuscrita. Asimismo, se indica que una firma que cumpla los requisitos para ser cualificada en un Estado Miembro debe ser reconocida por el resto de Estados.

Teniendo en cuenta esto, para cualquier trámite se debería optar por solicitar / realizar al menos el formato de firma avanzada.

Si consideramos la clasificación en todas sus posibilidades, podemos crear una tabla resumen para conocer la firma más acorde a nuestras necesidades:

Necesidad	XML	PDF	CMS	XAdES	PAdES	CAdES
Firmar hash del documento	✓	✗	✓	✓	✗	✓
Reconocimiento legal*	✗	✗	✗	✓	✓	✓
Firmar PDF	✓	✓	✓	✓	✓	✓
Firmar cualquier formato	✓	✗	✓	✓	✗	✓
Varios firmantes en paralelo	✓	✓	✓	✓	✓	✓
Varios firmantes en serie	✓	✗	✓	✓	✗	✓
Ver contenido facilmente	✓	✓	✗	✓	✓	✗

Ilustración 20 - Tabla formatos de firma

3.4. Certificados - Seguridad en emisión / generación de certificados

Uno de los primeros puntos en los que podemos analizar la seguridad de los certificados y firmas es en las emisiones de los propios certificados para su posterior uso.

Para ello, en primer lugar, debemos introducir la infraestructura de clave pública (PKI – Public Key Infraestructure), que es el conjunto de elementos que conforman el sistema que permite el correcto funcionamiento de las claves públicas y cuyos principales elementos son [25]:

- **Autoridades emisoras de certificados** (CA – Certificate Authority), que son las encargas de emitir (generar), revocar y permitir la validación de los certificados.
- **Autoridades de registro** (RA – registration authority), que son las encargadas de comprobar la identidad / información de quien desea obtener el certificado, generalmente requieren comprobación presencial, salvo excepciones en las que se disponga de una forma de identificarse reconocida, como puede ser disponer del DNIe para obtener el certificado de FNMT
- **Autoridad de validación** (VA – Validation authority): esta autoridad es la encargada de la gestión de validación del certificado.

- **Repositorios:** utilizados para almacenar la información referente a la PKI , principalmente los propios certificados y en especial la información de las listas de revocación (CRL: Certificate Revocation List)

El esquema general del funcionamiento de una PKI es el siguiente:

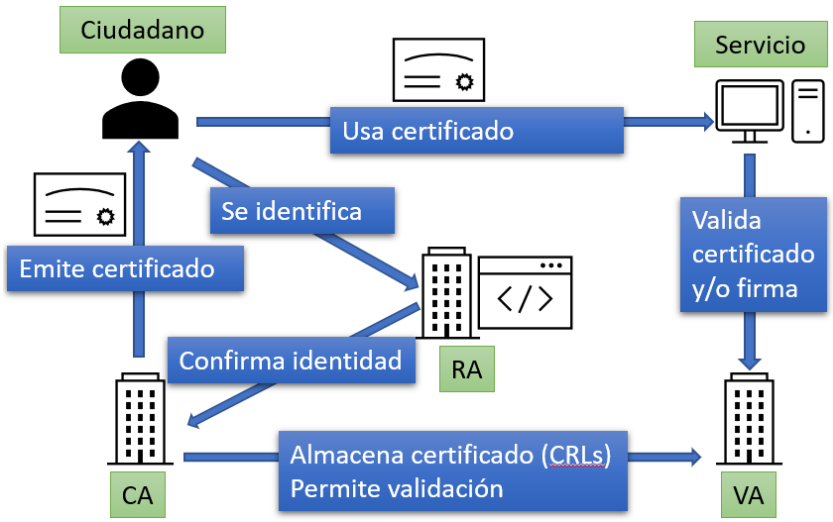


Ilustración 21 - Esquema funcionamiento PKI

Asimismo, es interesante mostrar cómo está construido un certificado. La información del mismo se encuentra en un formato estándar [26], X.509 en formato ASN.1 en una estructura de OID (Object Identifier). Por ese motivo, si intentamos abrir un certificado con un editor de textos no obtenemos información legible, en cambio, si usamos un editor específico, como ASN.1 Editor podemos observar la información, tal como mostramos a continuación:

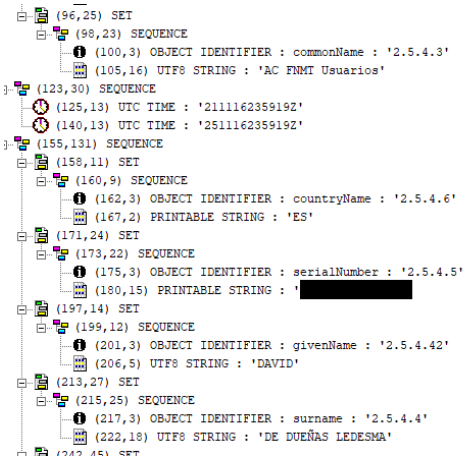


Ilustración 22 - Visión ASN.1 Certificado

Dos posibles vías de uso de certificado para la autenticación en una sede electrónica y, por tanto, prestar un servicio son:

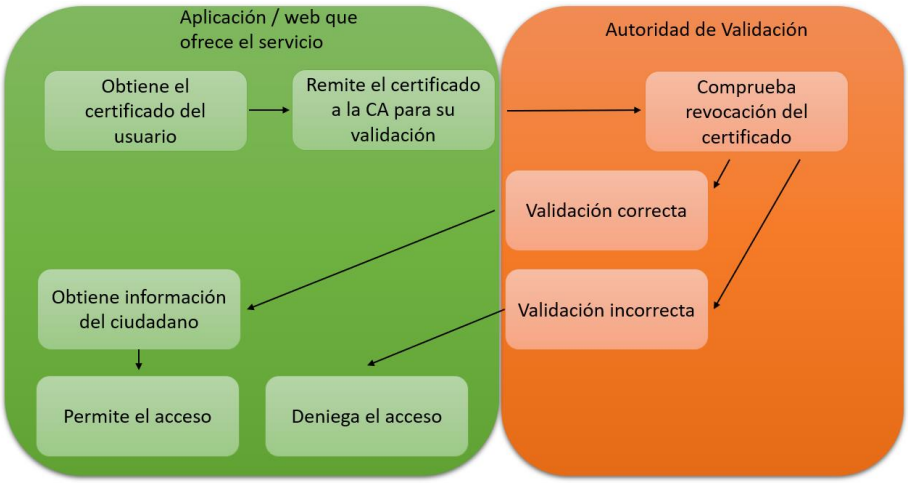


Ilustración 23 - Esquema uso certificado

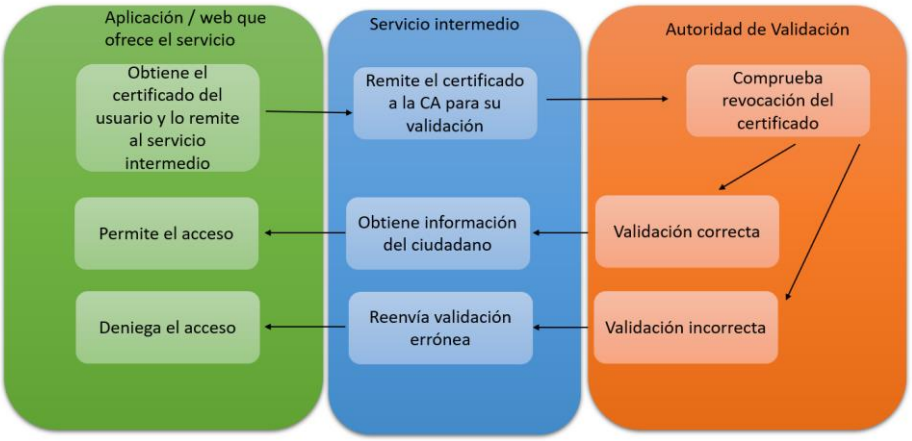


Ilustración 24 - Esquema uso certificado intermediado

El funcionamiento es similar en ambos casos, la aplicación o web que presta el servicio extrae el certificado del usuario, lo remite bien directamente al emisor o vía una autoridad intermedia y confirma la validez, si es correcta, extrae la información del propio certificado (o la extrae la autoridad intermedia), generalmente el NIF-NIE y en base a ella permite el acceso / operación.

Por tanto, si tal como vemos, la información se extrae del propio certificado, ¿qué impide que un usuario se genere un certificado con los datos de otra persona para suplantar su identidad? Vamos a comprobarlo:

En primer lugar, vamos a hacer uso de un certificado ya emitido, en este caso, por la FNMT y vamos a validarlo de las dos formas posibles, directa, e indirecta:

Para la validación directa, tenemos que consultar al emisor del mismo, vamos a comprobar en el propio certificado la información existente para su validación:

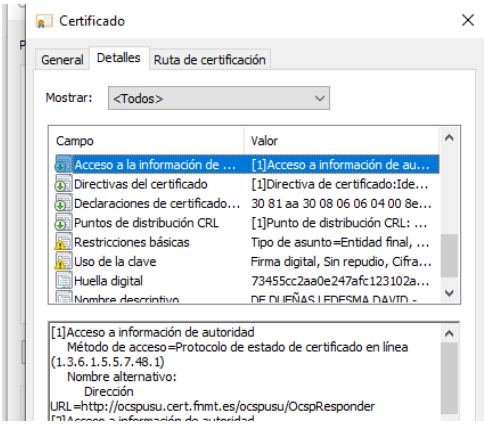


Ilustración 25 - Comprobación info certificado

Comprobamos que podemos comprobar la validez del certificado en el *endpoint* "http://ocspusu.cert.fnmt.es/ocspusu/OcspResponder"

Cabe indicar, que, al tratarse de un estándar, la información que aquí encontramos no es únicamente para los certificados de la FNMT sino para cualquier certificado, si consultamos el apartado del certificado en el que aparece dicha información (OID 1.3.6.1.5.5.7.48.1) en alguna página que registro la información de OIDs podemos ver [27]:

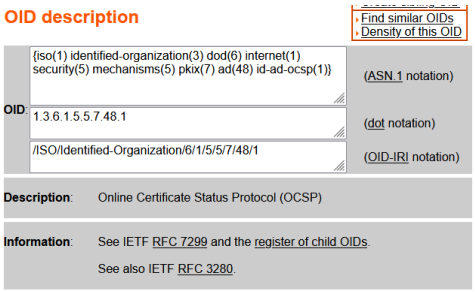


Ilustración 26 - OID Info Método validación

El estándar RFC 6960 [28] define cómo se realiza una petición OCSP, en nuestro caso, para poder generar una petición siguiendo dicho estándar vamos a usar OpenSSL, que permite dicha opción según se contempla en su manual [29]:

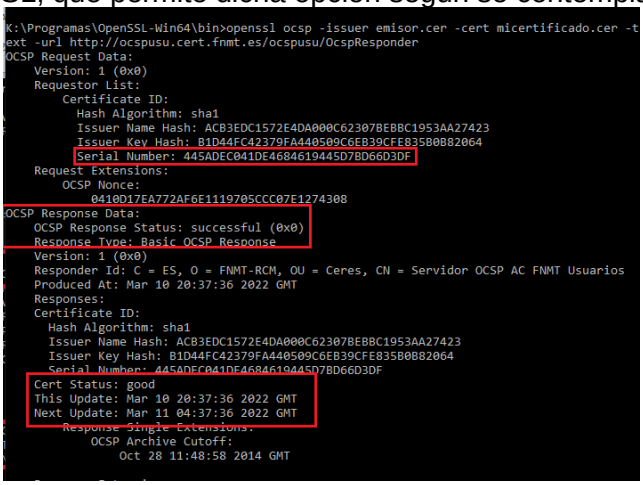


Ilustración 27 - Prueba OCSP

Como podemos ver, en la petición se remite la parte pública del certificado (micertificado.cer), la parte pública del emisor del certificado (emisor.cer) y la URL del OCSP.

En la petición, se muestra el número de serie del certificado, que si nos fijamos en el visor de Windows coincide:

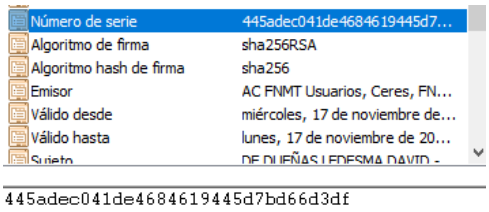


Ilustración 28 - Comparación nº de serie

Y la respuesta es “Good” lo cual quiere decir que el certificado es válido.

Ahora vamos a realizar la misma acción haciendo uso de un intermediario, en España, el Gobierno de España dispone de la web VALIDe [30] que permite validar los certificados haciendo uso de la Plataforma @firma también del estado, es decir, VALIDe obtiene el certificado, lo remite a @firma quien a su vez lo remite a la Autoridad de Validación correspondiente, vamos, por tanto, a hacer la misma prueba que hemos hecho de forma manual:

Ilustración 29 - VALIDe Prueba correcta

Ahora vamos a crear un certificado por nuestra parte para comprobar su funcionamiento, en el anexo se detallan los pasos seguidos. Tras su generación, vamos a repetir las pruebas hechas con el certificado emitido por la FNMT:

Validación vía OCSP:

```
cert_final_ddl_fnmt.cer: unknown
        This Update: Mar 20 20:50:20 2022 GMT
```

Ilustración 30 -Validación OCSP Unknown

Validación mediante VALIDe:

Ilustración 31 - Validación VALIDe no soportado

Tal como podemos ver, a pesar de haber generado un certificado con nuestros datos que sigue la estructura X.509 no es posible validar el mismo correctamente, en el primer caso el validador de FNMT no lo reconoce y en el segundo caso (VALIDe) detecta que el emisor del mismo no está soportado y no se intenta usar ningún método de validación. Por tanto, no es posible suplantar la identidad de una persona generando un certificado, ya que existe una seguridad a nivel de emisores de los mismos, en el apartado de PSC se explicará la parte correspondiente a las entidades emisoras.

3.5. Certificados - Seguridad en el uso

Tras la emisión del certificado, otro de los puntos a analizar es la seguridad en el uso de certificados por parte del usuario.

Vamos a diferenciar entre la seguridad que aplica al certificado según su formato y posteriormente detallar las consideraciones genéricas.

3.5.1. Certificados software

En el primer caso, software, podemos encontrar tres puntos problemáticos:

Exportación clave privada

Una vez que el certificado se nos emite, el mismo se encontrará en un fichero .p12, esta extensión representa un fichero PKCS12 que se corresponde con un almacén de certificados y que generalmente, y en el caso que nos ocupa, incluye la parte privada del certificado.

Para el uso del certificado podemos, bien importarlo en el navegador, que es el uso habitual, o bien mantener el fichero p12, no obstante, en la mayoría de los accesos no se nos permitirá seleccionar el certificado de un fichero, por lo que necesitaremos importarlo en el navegador.

A la hora de importarlo, nos encontraremos con esta opción:

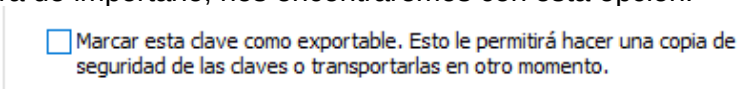


Ilustración 32 - Importación - exportable

Es muy importante que esta opción no se marque, salvo que estemos muy seguro de sus implicaciones, ya que, si alguien hiciera uso de nuestro ordenador y tuviéramos nuestro certificado instalado con dicha opción, podrían extraer nuestro certificado y hacer uso del mismo sin necesidad de conocer nuestra contraseña, desde el navegador, podríamos exportar la clave privada:

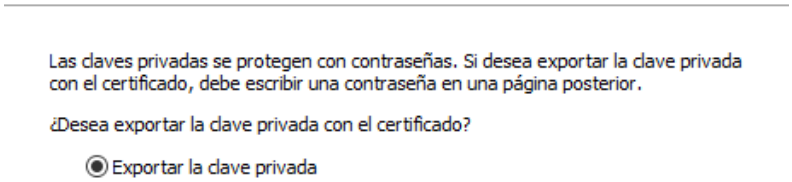


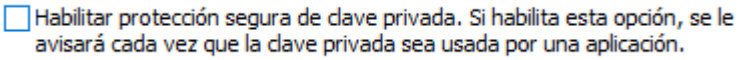
Ilustración 33 - Exportación clave privada

Si la importación se hace sin marcar la opción definida anteriormente, no se podrá exportar la parte privada, ya que dicha opción estará bloqueada:

Ilustración 34 - Exportación sin clave privada

Uso indebido

Con el paso previo evitamos que la clave privada pueda ser obtenida por otra persona, pero, si el certificado está instalado en el navegador, y alguien accede al mismo, podrían hacer uso del mismo, para evitar o minimizar este posible uso no autorizado, al importar nuestro certificado podemos marcar la opción:



Esto permite, que en el paso final de importación proteger el certificado con contraseña:

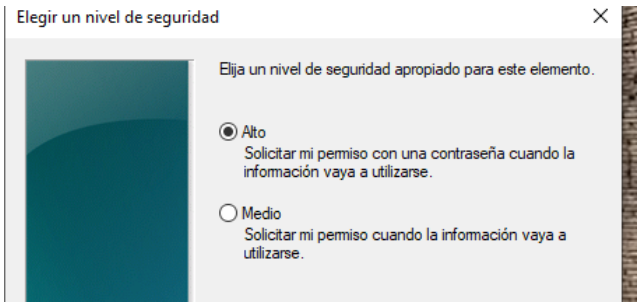


Ilustración 35 - Importación nivel seguridad

Así, cuando intentemos firmar con el mismo, se nos solicitará:

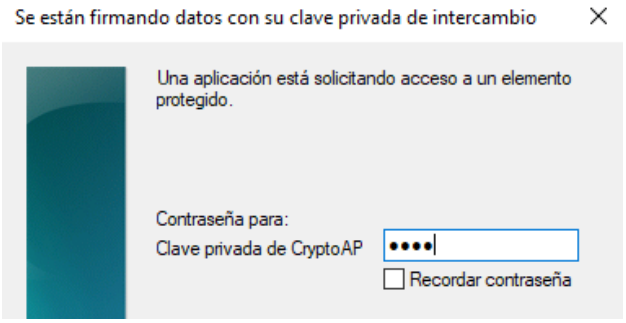


Ilustración 36 - Solicitud contraseña firmar

Es importante indicar, que este funcionamiento mostrado se ha realizado haciendo uso del almacén de certificados del sistema (Windows) del que hace uso Internet Explorer / Edge. Otro navegador, como Mozilla Firefox, no usa este almacén y tendremos que hacer la importación adicionalmente en este navegador si queremos hacer uso del certificado en este navegador, en este caso, la contraseña para el uso del certificado no se introduce en su importación, sino que se configura de forma global para el almacén:

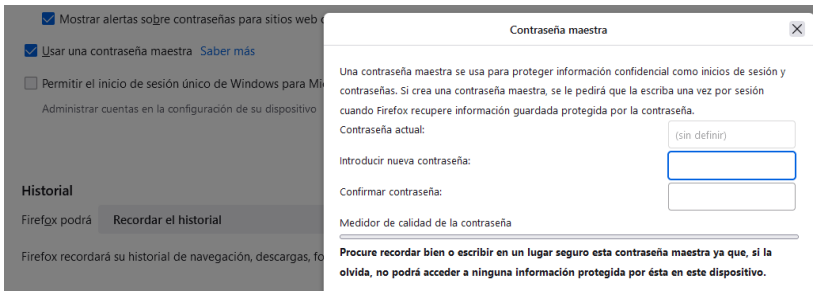


Ilustración 37 - Contraseña maestra Firefox

Ataque a la contraseña

Teniendo en cuenta los pasos previos, hemos podido proteger el uso del certificado software, pero ahora el elemento clave es la contraseña, vamos a analizar la seguridad de las contraseñas.

La seguridad de la contraseña viene determinada por la longitud de la misma y las posibles combinaciones: si denominamos *r* a la longitud y *n* a los posibles caracteres, las combinaciones posibles serán n^r

Vamos a comparar la diferencia de combinaciones, si usamos sólo números, es decir, 0-9, (10 caracteres) y si añadimos caracteres alfabéticos (sin distinguir mayúsculas y minúsculas y sin tener en cuenta la ñ), en cuyo caso tendríamos 26+10 = 36 posibilidades:

Longitud	Nº Caracteres	Combinaciones	Nº de Combinaciones
2	10	10^2	100
4	10	10^4	10.000
8	10	10^8	100.000.000
16	10	10^{16}	10.000.000.000.000.000
2	36	36^2	1.296
4	36	36^4	1.679.616
8	36	36^8	2.821.109.907.456
16	36	36^{16}	7.958.661.109.946.400.000.000.000

Ilustración 38 - Combinaciones contraseñas

Si se intentase averiguar la contraseña por fuerza bruta (probando contraseñas hasta averiguar la correcta) de forma manual, suponiendo que tardamos un segundo por cada intento y teniendo en cuenta este último ejemplo:

- Para longitud 2 → 1296(posibilidades)/60 segundos = 21,6 minutos
- Para longitud 4 → 1679616 (posibilidades)/60 segundos = 27.993,6 minutos → 466,56 horas → 19,4 días.
- Para longitud 8 → 2821109907456(posibilidades)/60 segundos = 47.018.498.457,6 minutos → 783.641.640,96 horas → 32.651.735,04 días → 89.456,8 años

Por tanto, a partir de dicha longitud ya sería inviable que atacaran nuestra contraseña de forma manual.

No obstante, los ordenadores pueden hacer estos ataques de forma más rápida, vamos a crear una aplicación en Python para demostrar un ataque con un ordenador.

Haciendo el cálculo directamente con 36 caracteres podemos ver como el tiempo se reduce considerablemente (se indica el tiempo medio, el total de las pruebas se indica en el Anexo 1)

Longitud	Nº Caracteres	Combinaciones	Tiempo medio
4	36	36^4	0,152 segundos
5	36	36^5	9,210 segundos
6	36	36^6	352,250 segundos
7	36	36^7	6672,652 segundos

Ilustración 39 - Tiempos obtener contraseña

A modo de ejemplo, si usáramos sólo caracteres numéricos, para este último caso probado (10^7), obtendríamos una media de 1,758 segundos.

Tras estas pruebas, se ha podido comprobar la importancia del uso de una contraseña con una longitud y complejidad elevada y cómo, aunque a nivel humano nos parezca una tarea complicada, lo rápido que un ordenador es capaz de realizar la misma tarea.

No obstante, un ordenador más potente y con un programa más elaborado podría mejorar notablemente estos tiempos, así, si observamos la tabla que nos ofrece *The Security Factor* [31]

Password Length	Numerical 0-9	Upper & Lower case a-Z	Numerical Upper & Lower case 0-9 a-Z	Numerical Upper & Lower case Special characters 0-9 a-Z %$
1	instantly	instantly	instantly	instantly
2	instantly	instantly	instantly	instantly
3	instantly	instantly	instantly	instantly
4	instantly	instantly	instantly	instantly
5	instantly	instantly	instantly	instantly
6	instantly	instantly	instantly	20 sec
7	instantly	2 sec	6 sec	49 min
8	instantly	1 min	6 min	5 days
9	instantly	1 hr	6 hr	2 years
10	instantly	3 days	15 days	330 years
11	instantly	138 days	3 years	50k years
12	2 sec	20 years	162 years	8m years
13	16 sec	1k years	10k years	1bn years
14	3 min	53k years	622k years	176bn years
15	26 min	3m years	39m years	27tn years
16	4 hr	143m years	2bn years	4qdn years
17	2 days	7bn years	148bn years	619qdn years
18	18 days	388bn years	9tn years	94qtn years
19	183 days	20tn years	570tn years	14sxn years
20	5 years	1qdn years	35qdn years	2sptn years

Ilustración 40 - Seguridad / fuerza bruta contraseñas

Podemos comprobar que para que una contraseña se considere suficiente segura debe tener al menos una longitud de 10 y contener números, mayúsculas, minúsculas y caracteres especiales.

Adicionalmente a esto, es útil mencionar la Ley de Moore [32] que indica que el número de transistores por cada chip se dobla cada año, por lo que los resultados de la tabla anterior diferirán y cada vez serán menos el tiempo estimado en obtener las contraseñas.

3.5.2. Certificados hardware - en tarjeta

El análisis indicado anteriormente aplica a certificados en software, pero no a los certificados emitidos en tarjeta, ya que los mismos no se instalan en el navegador.

Este tipo de certificados, se encuentran en el chip protegidos con contraseña y no es posible su exportación, por tanto, la única vulnerabilidad se encontraría en la propia contraseña, que ya hemos analizado previamente.

No obstante, es importante indicar que en este caso no sería posible realizar un ataque por fuerza bruta, ya que si revisamos las medidas de seguridad que incorporan este tipo de certificados, tal como podemos comprobar en la página web del DNIe [33] o en la de la FNMT [34] en ambos casos existe un número limitado de intentos para introducir la contraseña (PIN), generalmente 3, tras ello, la tarjeta se bloquea; en el caso del DNIe es necesario usar la propia huella dactilar para su desbloqueo y en el caso de la FNMT si se vuelve a introducir también de forma errónea el código de bloqueo es imposible volver a desbloquearlo, siendo necesario obtener otro certificado en tarjeta.

3.5.3. Niveles de seguridad

En el apartado anterior, hemos observado que no existe la misma seguridad entre certificados en tarjeta y software, este tipo de diferencias también está definido por parte de la legislación de forma que las aplicaciones, Organismos públicos o entidades puedan tenerlo en cuenta en sus operaciones.

Si revisamos el Reglamento UE 910/2014 [35] podemos comprobar que se hace referencia a tres niveles de seguridad: bajo, sustancial y alto.

Para considerarse de un nivel u otro se tienen en cuenta varios requisitos, entre los que destacamos:

- Si la persona se haya identificado de forma física para poder emitir el certificado
- El certificado ha sido emitido por un prestador que cumple las directrices marcadas
- Los medios técnicos de emisión y almacenamiento del certificado.

En los niveles bajo y sustancial se indica que se persigue reducir el riesgo de uso indebido o alteración de la identidad, mientras que en el alto se habla de evitarlo.

Un ejemplo de sistema que usa estos niveles de seguridad es Cl@ve, la Plataforma de identidad usada por gran parte de la Administración Pública en España, si nos fijamos en su documentación [36] podemos ver que considera en un nivel medio los certificados emitidos en Software y en un nivel alto los emitidos en tarjeta inclusive el DNIe.

¿Qué ventajas tiene esto? Que una aplicación puede limitar el tipo de certificado a utilizar si quiere hacer aún más seguros sus trámites, así, en función del trámite a realizar puede establecer distintos niveles de seguridad, siendo más laxo en algunos y más estricto en otros, en función de las acciones que se permitan y las implicaciones de la gestión.

3.6. Certificados – Estructura e información

En este apartado vamos a analizar la siguiente información sobre los certificados: como se estructura su información, qué información la componen y los campos más importantes.

3.6.1. Estructura

Tal como hemos mencionado anteriormente, la información que se puede encontrar en un certificado se encuentra en una estructura bajo el estándar X.509, y se usan OIDs para incorporar la diferente información del certificado.

Aunque ya lo mencionados en un caso previo, en este caso vamos a ver con detalle toda la cadena, tomando un ejemplo para ver que trata de una estructura jerárquica:

En un certificado personal de la FNMT vemos que se indica:

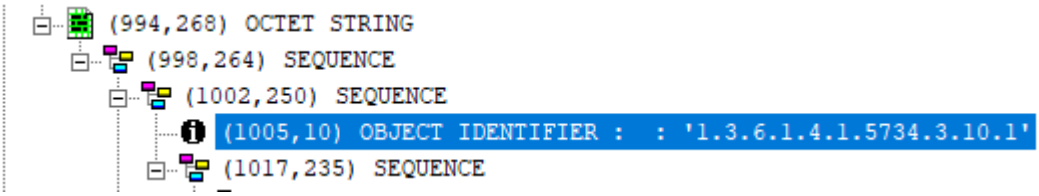


Ilustración 41 - ASN1 1.3.6.1.4.1.5734.3.10.1

Si revisamos la búsqueda paso a paso, partimos desde el primer OID (1.) hasta el último (1.3.6.1.4.1.5734.3.10.1) vemos que cada tramo de OID nos aporta información sobre el nivel de jerarquía, así podemos encontrar:

- 1. → Se corresponde con la Organización internacional de estandarización (ISO)
- 1.3. → Se trata de la identificación de esquemas acordes a ISO/IEC 6523-3
- 1.3.6. → Se corresponde al estándar OSI del DoD
- 1.3.6.1. → ID correspondiente a uso de Internet
- 1.3.6.1.4. → Esta jerarquía hace referencia a proyectos privados.
- 1.3.6.1.4.1. → Dentro de proyectos privados, se corresponde a empresas privadas
- 1.3.6.1.4.1.5734. →De entre las empresas privadas, este registro pertenece a la FNMT
- 1.3.6.1.4.1.5734.3. → Nos indica que se trata de las políticas usadas por la FNMT
- 1.3.6.1.4.1.5734.3.10. → Ahora podemos ver que de entre las políticas de la FNMT, se trata de aquellas para las personas físicas.
- 1.3.6.1.4.1.5734.3.10.1 → Por último, podemos comprobar que se trata de las políticas para un certificado cualificado de persona física.

Este tipo de información, lo podríamos comprobar para otros campos del certificado, y al ser estándar, las aplicaciones lo pueden utilizar para traducir la información, así, por ejemplo, en Windows vemos la siguiente información:

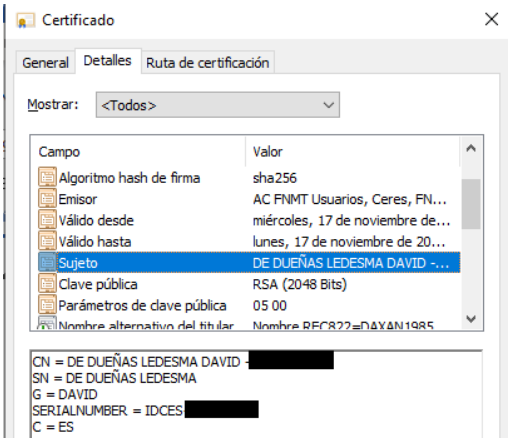


Ilustración 42 - Windows 1.3.6.1.4.1.5734.1.1

Y en un visor ASN1 vemos:

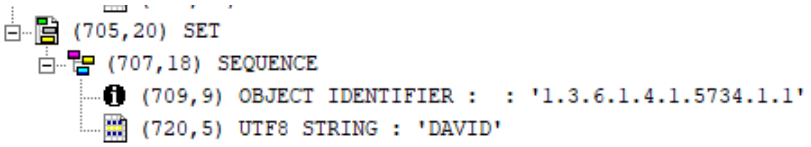


Ilustración 43 - ASN1 1.3.6.1.4.1.5734.1.1

Ha traducido G (según el estándar GivenName) ya que el OID "1.3.6.1.4.1.5734.1.1" indica dicha información:

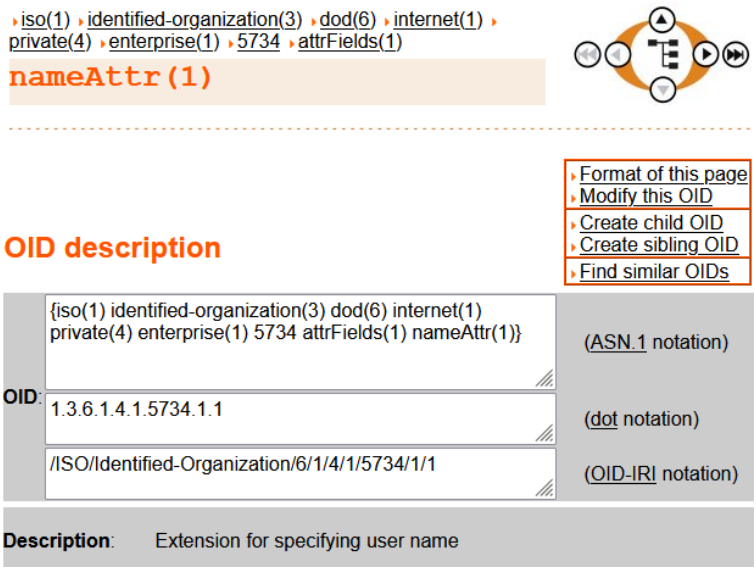


Ilustración 44 - OID 1.3.6.1.4.1.5734.1.1

3.6.2. Información

Adicionalmente a la estructura descrita en el apartado anterior, para establecer cómo se cumplimenta la información en el certificado se tienen en cuenta dos referencias:

- El estándar, en este caso, RFC 5280 en el que se incluye qué información viene recogida en cada campo, a modo de ejemplo:
  - Campo "Subject" (Sujeto): en este campo se incorpora la información pública de la entidad a la que se ha emitido certificado.

- Campo “Issuer” (Emisor): en este campo se incorpora la información de la entidad que emite y firma el certificado.

No sólo podemos encontrar qué información cumplimentar, y sí es opcional u obligatoria, sino cómo hacerlo, por ejemplo, nos indica que el campo *Issuer* debe estar preparado para recibir atributos como *country*, *organization* o *serial number* bajo el estándar X.520 [37], que, si nos referimos al mismo, observamos:

```
countryName ATTRIBUTE ::= {
  SUBTYPE OF              name
  WITH SYNTAX             CountryName
  SINGLE VALUE            TRUE
  LDAP-SYNTAX             countryString.&id
  LDAP-NAME               {"c"}
  ID                      id-at-countryName }

CountryName ::= PrintableString(SIZE (2)) -- ISO 3166 codes only
```

Ilustración 45 - X.520 Country

Si nos fijamos en el campo *Issuer* (Emisor) de nuestro certificado de FNMT vemos que se cumple que se indica la letra “C” y el tamaño de 2 (“ES”)

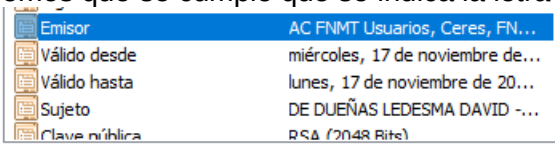


Ilustración 46 - Windows Country

Y si acudimos a ISO 3166 [38]:

Vemos que se usa la nomenclatura para España

English short name	French short name	Alpha-2 code
Singapore	Singapour	SG
Sint Maarten (Dutch part)	Saint-Martin (partie néerlandaise)	SX
Slovakia	Slovaquie (la)	SK
Slovenia	Slovénie (la)	SI
Solomon Islands	Salomon (les Îles)	SB
Somalia	Somalie (la)	SO
South Africa	Afrique du Sud (l')	ZA
South Georgia and the South Sandwich Islands	Géorgie du Sud-et-les Îles Sandwich du Sud (la)	GS
South Sudan	Soudan du Sud (le)	SS
Spain	Espagne (l')	ES

Ilustración 47 - ISO 3166 Country

Como ejemplo adicional, podemos ver el atributo Organización, que se nos indica que se trata de una cadena de texto que identifica al Organismo y que se refiere con la letra “O”:

6.4.1 Organization Name

The *Organization Name* attribute type specifies an organization. When used as a component of a directory name, it identifies an organization with which the named object is affiliated.

An attribute value for **OrganizationName** is a string chosen by the organization (e.g., O = "Scottish Telecommunications plc"). Any variants should be associated with the named Organization as separate and alternative attribute values.

```
organizationName ATTRIBUTE ::= {
  SUBTYPE OF                name
  WITH SYNTAX               UnboundedDirectoryString
  LDAP-SYNTAX               directoryString.&id
  LDAP-NAME                 {"o"}
  ID                        id-at-organizationName }
```

Ilustración 48 - X.520 OrganizationName

Vemos que en nuestro certificado así se cumple:

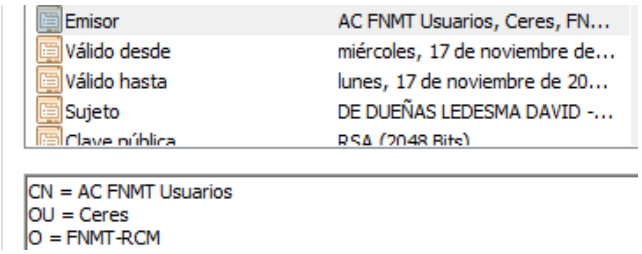


Ilustración 49 - Windows OrganizationName

- La política definida por la entidad emisora: la entidad que emite los certificados debe disponer de unos documentos denominados Declaración de Políticas de Certificación (DPC) que recoja las directrices seguidas por su parte para la emisión de cada tipo de certificado, así, para el ejemplo que estamos usando, FNMT, podemos comprobar que existe dicho documento [39] que entre otras cosas establece:
  - La cadena de certificación de los certificados
  - Las autoridades de registro (RA)
  - Quienes son los destinatarios de los certificados (en este caso personas físicas)
  - Cuál es el uso esperado de los certificados
  - Para qué no se deben usar los certificados
  - Cómo y dónde se publica la información de revocación (CRLs)
  - Estándares utilizados
  - Obligaciones de los suscriptores (por ejemplo, deben personarse para que se les pueda confirmar su identidad para generar el certificado)
  - Bajo qué criterios, quién y cómo es posible la revocación

También se hace referencia al perfil de certificados [40], que indica que campos contendrá el certificado, qué valor se espera y si el campo es obligatorio o no, a modo de ejemplo vamos a comprobar sí se cumple para nuestro certificado:

El algoritmo de firma es SHA256 y es obligatorio:

3.	Signature Algorithm	sha256WithRSAEncryption	Sí

Ilustración 50 - Perfil certificados - Algoritmo

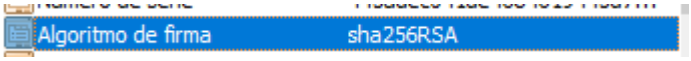


Ilustración 51 - Windows Algoritmo

La validez es de 4 años:

5.	Validity	4 años	Sí

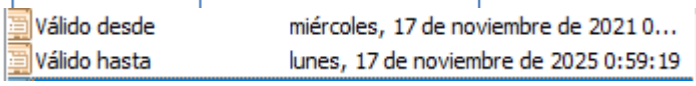


Ilustración 52 - Validez, Perfil y Windows

En el *Subject* (Sujeto) se informará del país (ES), número de serie (nif del titular), nombre, apellidos y *common name* (apellidos, nombre + nif):

6.	Subject	Identificación/descripción del titular de las claves certificadas	Sí
	6.1. Country	C=ES. (PrintableString, tamaño 2 (rfc5280))	Sí
	6.2. SerialNumber	NIF del titular. (PrintableString (rfc5280))	Sí
	6.3. Given Name	Nombre de pila, de acuerdo con documento de identidad. (UTF8String tamaño máximo 50 caracteres)	Sí
	6.4. Surname	Apellidos de acuerdo con documento de identificación. (UTF8String tamaño máximo 50 caracteres)	Sí
	6.5. Common Name	Apellidos, Nombre y NIF del titular. (UTF8String tamaño máximo 164 caracteres)	Sí

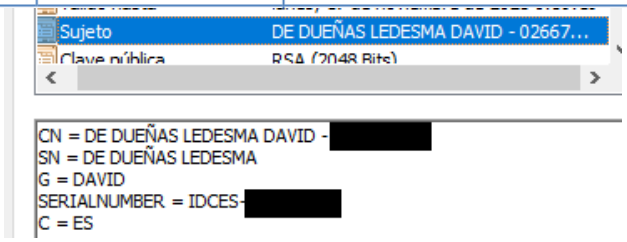


Ilustración 53 - Subject, Perfil y Windows

También se cumple la información.

Podemos, por tanto, comprobar que toda la información que se encuentra en un certificado y cómo incluirla está definido claramente, por lo que nos puede servir para confirmar que los mismos se emitan correctamente y para reclamar en caso de que no se hayan seguido estos parámetros.

3.7. Certificados - PSCs

3.7.1. Funcionamiento PSCs - TSL

Como hemos visto anteriormente, uno de los elementos de una PKI son los emisores de certificados a los que denominados PSC; estas entidades son las encargas de emitir los certificados y mantener la información de los mismos y de sus estados de revocación.

En primer lugar, hemos de indicar que los certificados se emiten de forma jerárquica:

- Existe un certificado raíz Autofirmado.
- Existe uno o varios certificados intermedios firmados por el raíz o anterior intermedio cuya orientación es clasificar los tipos de certificados
- Existen uno o varios tipos de certificados finales que son los que contienen los datos de la entidad o persona que hará uso del mismo.

Estas estructuras se conocen como cadenas o rutas de certificación:

Si tomamos como ejemplo el certificado que hemos usado en pruebas anteriores, su estructura jerárquica es:

- Certificado raíz: AC RAIZ FNMT-RCM
  - Certificado intermedio: AC FNMT Usuarios
    - Certificado final: Certificado de Persona Física (1.3.6.1.4.1.5734.3.10.1)

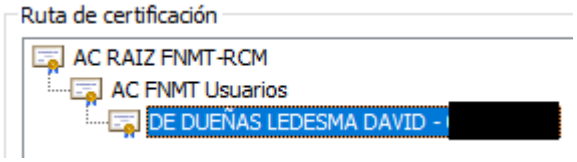


Ilustración 54 - Jerarquía 1.3.6.1.4.1.5734.3.10.1

Cada Prestador puede clasificar estos certificados como considere, en función del número de certificados, tipos de certificados a emitir, necesidad de crear nuevas ramas, etc.

Asimismo, los Prestadores suelen recoger su información (generalmente pública) sobre emisión de certificados en dos tipos de documentos [41], cuyos ejemplos hemos visto en otros apartados previos:

- DPC (Declaración de Prácticas de Certificación): Muestra la información de cómo emiten los certificados, para quien están destinados, qué uso se espera y que directrices legales, técnicas y pasos siguen para ello.
- Perfiles de certificados: Muestran una tabla con los campos que se esperan para el tipo de certificado en concreto, qué información se incluye, en qué formato y si los campos son obligatorios u opcionales.

Hemos visto en pruebas anteriores que un certificado emitido manualmente no era válido en algunas validaciones, y es que existe una forma de comprobar qué certificados son legalmente válidos.

En España, el Ministerio de Asuntos Económicos y Transformación Digital mantiene la lista de los servicios de prestadores que se consideran cualificados y se encuentran recogidos en una lista denominada TSL (Trusted Service status List) [42], que puede consultarse en formato XML y PDF, los Organismos que quieran permitir trámites con certificados digitales pueden basarse en esta información para filtrar qué certificados permiten para que los usuarios firmen y/o se identifiquen.

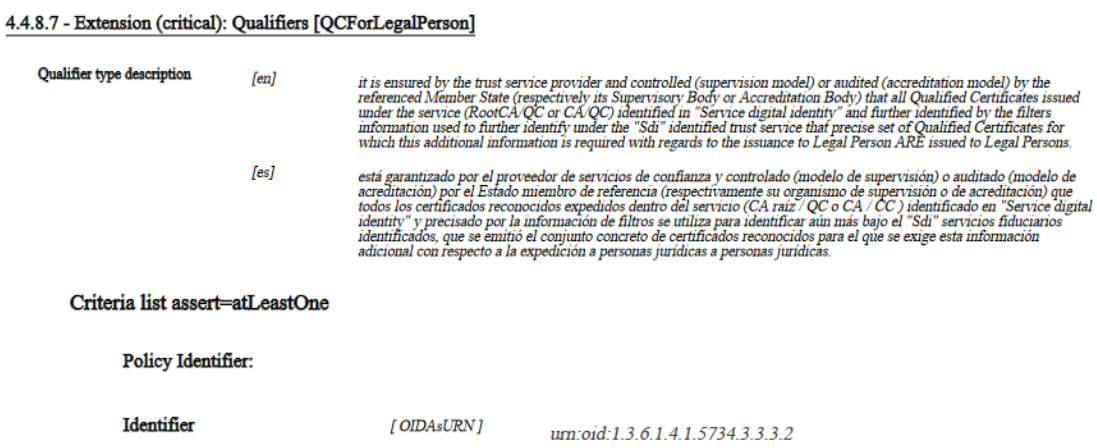
Es necesario, no obstante, indicar cómo identificar la información en dicha lista:

- La información se emite en base al emisor del certificado, debemos, por tanto, buscar si el emisor de nuestro certificado está en la lista, como “Service”, para el caso que hemos tomado como ejemplo → “*Service (granted): Qualified certificates for individuals issued by AC FNMT Usuario”*
- Es necesario comprobar que el servicio siga siendo válido, porque puede que por incumplimiento de la normativa vigente por modificaciones o por otro

motivo, aunque en su día estuviese reconocido, haya dejado de estarlo, en lugar de borrarlo de la lista aparecerá con estado *"withdrawn"*

- En la lista se indica para cada servicio, para que uso está destinado el certificado, ya que puede que un certificado esté reconocido como válido para que firme una persona física u otro para un sitio web y no deberían usarse indistintamente, así, podemos encontrar:
  - http://uri.etsi.org/TrstSvc/TrustedList/SvcInfoExt/ForeSignatures → Certificado emitido para la generacion de firmes
  - http://uri.etsi.org/TrstSvc/TrustedList/SvcInfoExt/ForeSeals → Certificado emitido para sellos Electrónicos
  - http://uri.etsi.org/TrstSvc/TrustedList/SvcInfoExt/ForWebSiteAuthentication → Certificado para autenticación de sitio web

- En ocasiones, la clasificación no aplica a todos los certificados emitidos por el certificado intermedio mostrado, en ese caso, se muestran excepciones como las que se indican:


4.4.8.7 - Extension (critical): Qualifiers [QCForLegalPerson]

Qualifier type description [en] it is ensured by the trust service provider and controlled (supervision model) or audited (accreditation model) by the referenced Member State (respectively its Supervisory Body or Accreditation Body) that all Qualified Certificates issued under the service (RootCA/QC or CA/QC) identified in "Service digital identity" and further identified by the filters information used to further identify under the "Sdi" identified trust service that precise set of Qualified Certificates for which this additional information is required with regards to the issuance to Legal Person ARE issued to Legal Persons.

[es] está garantizado por el proveedor de servicios de confianza y controlado (modelo de supervisión) o auditado (modelo de acreditación) por el Estado miembro de referencia (respectivamente su organismo de supervisión o de acreditación) que todos los certificados reconocidos expedidos dentro del servicio (CA raíz / QC o CA / CC) identificado en "Service digital identity" y precisado por la información de filtros se utiliza para identificar aún más bajo el "Sdi" servicios fiduciarios identificados, que se emitió el conjunto concreto de certificados reconocidos para el que se exige esta información adicional con respecto a la expedición a personas jurídicas a personas jurídicas.

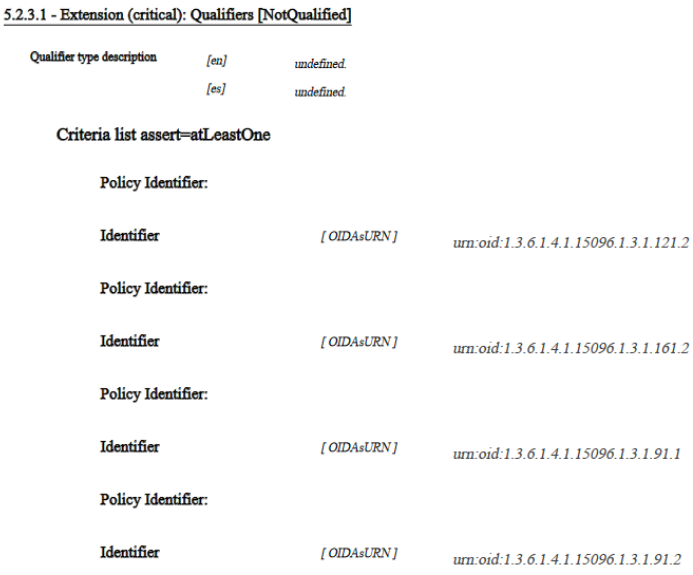
Criteria list assert=atLeastOne

Policy Identifier:

Identifier [OIDAsURN] urn:oid:1.3.6.1.4.1.5734.3.3.3.2

Ilustración 55 - TSL QCForLegalPerson

En este caso, la interpretación es que, de todos aquellos certificados que se indican en el servicio, el único que se considera cualificado para persona jurídica (QCForLegalPerson) es el que en su política incluya el OID "1.3.6.1.4.1.5734.3.3.3.2"


5.2.3.1 - Extension (critical): Qualifiers [NotQualified]

Qualifier type description [en] undefined.

[es] undefined.

Criteria list assert=atLeastOne

Policy Identifier:

Identifier [OIDAsURN] urn:oid:1.3.6.1.4.1.15096.1.3.1.121.2

Policy Identifier:

Identifier [OIDAsURN] urn:oid:1.3.6.1.4.1.15096.1.3.1.161.2

Policy Identifier:

Identifier [OIDAsURN] urn:oid:1.3.6.1.4.1.15096.1.3.1.91.1

Policy Identifier:

Identifier [OIDAsURN] urn:oid:1.3.6.1.4.1.15096.1.3.1.91.2

Ilustración 56 - TSL Excepciones

Este es el caso inverso, la interpretación es que, de todos aquellos certificados que se indican en el servicio se consideran cualificados, excepto aquellos que incluyan en su política alguno de los OIDs listados: “1.3.6.1.4.1.15096.1.3.1.121.2”, “1.3.6.1.4.1.15096.1.3.1.161.2”, “1.3.6.1.4.1.15096.1.3.1.91.1” o “1.3.6.1.4.1.15096.1.3.1.91.2”

Aunque esta información nos sirve para entender el funcionamiento de una TSL, recalcamos que existe un estándar (ETSI TS 119 61) en el que explica el funcionamiento de todos los campos de una TSL, cómo cumplimentarlos y su obligatoriedad o no.

Es importante indicar, que aunque hemos analizado esta información a nivel de realización de trámites online, los certificados de sitio web también son reconocidos por los propios navegadores, y disponen de unas entidades de confianza por defecto que consideran fiables (podríamos decir que es un equivalente a la TSL para los trámites, pero solo a nivel de SSL), por ejemplo, para Mozilla Firefox en https://ccadb-public.secure.force.com/mozilla/IncludedCACertificateReport podemos encontrar el listado

Como ejemplo, si accedemos a la web de la UOC hemos que el navegador lo muestra como de confianza

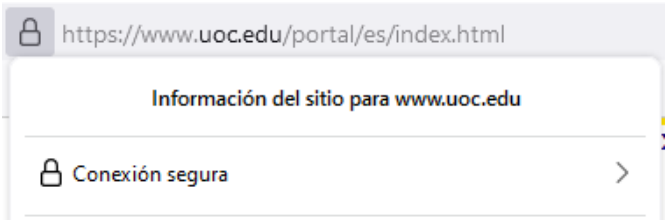


Ilustración 57 - Certificado SSL UOC

Si revisamos la cadena de certificación:

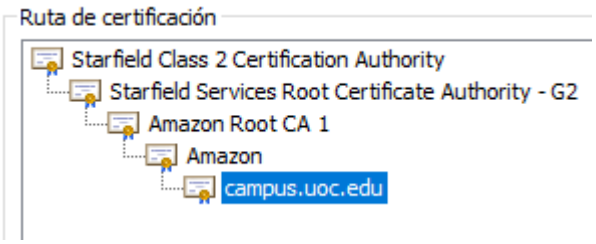


Ilustración 58 - Jerarquía certificado UOC

Y si comprobamos en el enlace anteriormente indicado:

Ilustración 59 - Confianza certificado UOC

En este apartado hemos aprendido a identificar que un certificado está emitido por una entidad de confianza, lo cual nos sirve para reforzar la seguridad de que, si la entidad que ofrece un servicio realiza las comprobaciones oportunas, no es posible suplantar la identidad salvo que estas entidades cometieran una infracción emitiendo un certificado indebido.

3.7.1. Listas de revocación

Uno de los aspectos básicos y más importantes de un Prestador es el mantenimiento de las listas de revocación. Estas listas CRL son un listado de los números de serie de los certificados que han sido revocados. Un certificado revocado es aquel que antes de su caducidad ha dejado de ser considerado válido por parte del PSC. Razones por las que esto puede ocurrir son:

- El propietario cree que existe un problema de seguridad (conocen su contraseña, han robado su equipo y es accesible el certificado, etc.)
- Se ha detectado algún error en la emisión (por ejemplo, no cumple algunos de los perfiles indicados en el punto anterior, y el campo no está cumplimentado como se espera).
- Cambios legislativos que afectan al uso o condiciones del certificado.

Generalmente, existen dos formas de comprobar la revocación de un certificado:

- **CRL**: Se trata de una lista que contiene el listado de todos los certificados revocados, indicando el número de serie del certificado y fecha de revocación. De forma general, los PSC no disponen de una CRL única si no que suelen disponer de una por cada certificado intermedio de su jerarquía, además, la CRL está firmada generalmente por dicho certificado.
  La CRL también dispone de una fecha de actualización, a partir de la cual debería emitirse una nueva versión, por lo que si la fecha actual es superior a la fecha de próxima actualización la CRL que estamos consultando no debería utilizarse.

- **OCSP**: En este protocolo, en lugar de descargarse la lista completa (CRL) se consulta expresamente por el estado del certificado en concreto, por lo que la velocidad de consulta es mayor y menos pesada, ya que no es necesario descargarse el listado de todos los certificados. La respuesta OCSP también está firmada.

A la hora de ofrecer un trámite en el que se utilice un certificado, es indispensable comprobar esta información, para evitar problemas de seguridad, ya que si un certificado está revocado, existe un motivo importante para ello.

3.8. Sellos de tiempo – Firmas longevas

En apartados anteriores hemos visto que una de las propiedades de los certificados es su caducidad, a partir de cierta fecha los mismos dejan de ser válidos. Esto querría decir que, a diferencia de una firma manuscrita, si usamos un certificado para realizar una firma digital, la misma tiene una caducidad ligada a nuestro certificado y una vez este expire la firma no será válida.

Por defecto, este es el comportamiento, y si intentamos validar una firma con un certificado que se encuentra caducado (o revocado) la firma será considerada inválida, para evitar esto tenemos que conseguir que la firma recoja información que confirme que en el momento de ser generada era válida y esta información se encuentre dentro de la propia firma, por lo que necesitaremos hacer uso de los sellos de tiempo.

El primer paso para ello es conseguir la hora de la firma, ya que la hora del sistema por defecto no es una hora fiable, al ser susceptible de ser modificada, por lo que el servicio, servidor o equipo que genere la información de la hora debe consultar o estar sincronizado con un servicio que garantiza que la hora es una fuente fiable.

En España, según el Real Decreto 1308/1992 [59] el Real Observatorio de la Armada es el responsable de mantener y ofrecer la hora legal. Tal cómo podemos comprobar en su propia página web, la información se obtiene entre una batería de patrones atómica de frecuencia y patrones de haz de cesio.

Entre los diferentes métodos que ofrece para la difusión de la hora, el ligado a Internet es el protocolo NTP, que puede ser consultado en hora.roa.es y minuto.roa.es.

En Windows podemos ver que por defecto ya tenemos un servidor de hora vía NTP:

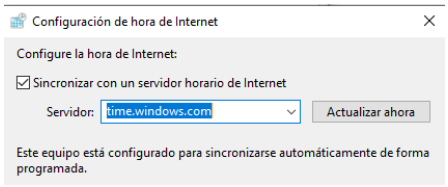


Ilustración 60 - Windows NTP

Podríamos modificar dicha URL por cualquiera de las dos citadas anteriormente.

En Linux, que es un Sistema Operativo más probable para ser usado por el servidor, el comando a utilizar sería: `ntpdate -u hora.roa.es`

Tras confirmar que nuestro equipo está sincronizado, abrimos un PDF, ya firmado, con Adobe, en el panel de firma seleccionamos “Agregar información de verificación…”, Si ahora revisamos la información de la firma:

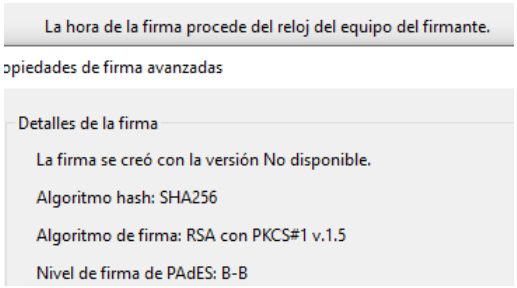


Ilustración 61 - Hora firma Adobe

Vemos que el formato es PAdES-B, y además se indica que la fuente de la hora es el equipo firmante. Según veremos a continuación, no sólo es necesario que la fuente de hora sea confiable para conseguir una firma longeva.

Si revisamos el documento de niveles de firma PAdES [60] vemos que se contemplan los siguientes niveles:

- **PAdES-BES** → Requisitos necesarios para la incorporación de los datos firmados e identificar al firmante en un fichero PDF.
- **PAdES-EPES** → Además de lo anterior, incorpora la posibilidad de añadir información de políticas de firma.
- **PAdES-LTV** → Además de lo anterior, añade información del sello de tiempo y requisitos adicionales que garantizará la validez de la firma longeva.

En el formato XAdES [58] vemos que existe un comportamiento similar:

- **XAdES** → Formato básico de firma XML que incluye la información del firmante, la firma y referencia firmada
- **XAdES-T** →Además de lo anterior, incluye un sello de tiempo
- **XAdES-C** → Además delo anterior, se añaden la información (referencias) de las evidencias de revocación
- **XAdES-X** → Además de lo anterior, se añade un sello de tiempo sobra las evidencias incorporadas a la firma -C
- **XAdES-XL** → Además de lo anterior, se incorpora en lugar de las referencias, la propia información de certificados y evidencias de revocación
- **XAdES-A** → Además de lo anterior, incluye información para posibilitar la validación periódica

Por tanto, si generalizamos, teniendo en cuenta el comportamiento para ambos formatos, para que una firma pueda ser validada de forma indefinida* es necesario:

- **Incluir en la propia firma la información de revocación** → Como hemos descrito en apartados anteriores, para comprobar la validez de una firma, entre otros aspectos, se necesita validar el certificado incluido y la misma y para ello se contactaría bien vía CRL o bien vía OCSP con las URL que ofrezca el emisor del certificado para su validación.

  No obstante, para que la firma sea validable en el futuro, queremos evitar este paso, ya que por una parte pueden dejar de mantenerse estas URL cuando termine su vigencia y, por otra parte, el certificado puede pasar a estado revocado o caducado, por lo que es necesario incluir en la firma la información existente en el momento de la generación de la misma o al menos antes de que ocurra lo descrito, ya que es la forma garantizar que en ese momento de tiempo el certificado firmante era válido.

- **Añadir uno o varios sellos de tiempo** → El sello de tiempo garantiza que lo firmado con ese sello ocurrió en ese momento, así, cuando se añade el primer sello se garantiza que la información del firmante fue incorporada en ese momento, pero es necesario añadir un segundo sello una vez incorporada la información de revocación, ya que este es el que garantiza que esa información de revocación se añadió en ese momento.

* La firma será válida de forma indefinida, pero siendo necesario que se genere un nuevo sello de tiempo cada vez que esté próximo a caducar.
Hemos definido qué requisitos debe cumplir una firma longeva, pero ¿qué requisitos se deben tener para emitir un sello de tiempo válido que podamos incorporar en nuestras firmas?

Si revisamos la legislación europea [6], la entidad que emita un Sello de Tiempo Cualificado debe tener en cuenta que el sello emitido debe:

- Asocia la información de los datos a la fecha / hora de forma que no pueda ser modificada
- Se basa en una fuente de tiempo fiable
- Está firmado con un certificado o sello de un servicio cualificado

Asimismo, a nivel técnico, la regulación RFC 3628 [61] indican los siguientes requisitos que debe tener el propio sello de tiempo:

- Debe incluir un identificador de la política de sello de tiempo.
- Cada sello de tiempo debe tener un identificador único
- El valor de la fecha / hora debe ser sincronizado con UTC
- Debe usarse una clave específica para este propósito
- Debe incorporar una referencia al dato origen, como puede ser el hash
- Debe incorporar información del país de la autoridad de sello de tiempo, un identificador de la autoridad y un identificador de la unidad que emite el sello.

Vamos a revisar cómo podemos comprobar que se cumpla esta información y por tanto el sello sea acorde a RFC 3628, para ello, vamos a generar un sello de tiempo:

Primero vamos a generar una petición de TSA, para ello, vamos a utilizar OpenSSL que mediante la opción “ts -query” permite crear una petición para posteriormente remitir a una autoridad de sello de tiempo. Como dato origen hemos usado el *hash* de un PDF, aunque también se puede utilizar la opción -digest para indicar el dato literal:

```
K:\Programas\OpenSSL-Win64\bin>openssl ts -query -data PruebaDDL.pdf -sha256 -out PruebaDDLHash
```

Ilustración 62 - Generación TSA Query

A continuación, vamos a remitir la petición a una plataforma de sellado de tiempo:

```
K:\Programas\curl\bin>curl -H "Content-Type:application/timestamp-query" --data-binary @PruebaDDLPetiHash
 http://tss.accv.es:8318/tsa -o RespACCV1.ts
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  1370  100  1304  100    66   8420    426 --:--:-- --:--:-- --:--:--  8896
```

Ilustración 63 - Petición TSA

Ya tenemos nuestro sello de tiempo, al ser un fichero binario, vamos a abrirlo con ASN1 Editor y revisamos la información que aparece en el mismo:

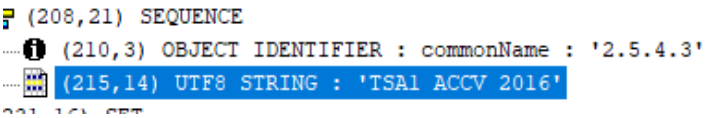


Ilustración 64 - ASN1 Emisor TSA

El emisor es "TSA1 ACCV 2016", si revisamos tanto la TSL de España [62] como el registro europeo de servicios cualificados [63] vemos que en ambos casos se reconoce como servicio cualificado de sello de tiempo.

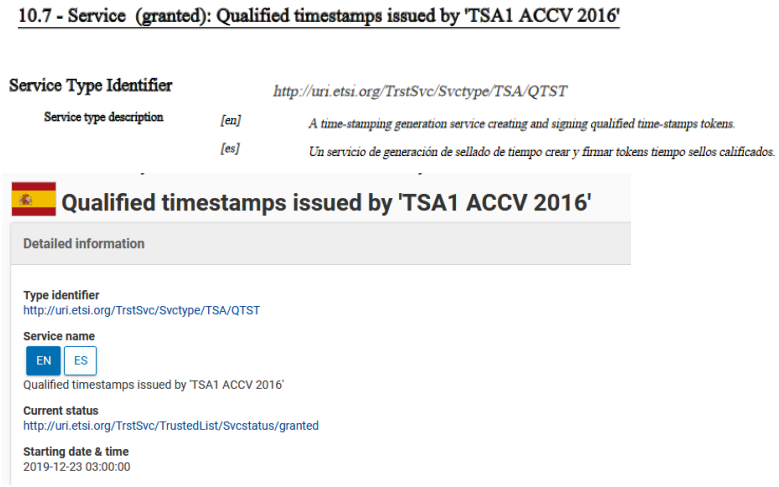


Ilustración 65 - Comprobación TSA Cualificada

Si revisamos la DPC del PSC (ACCV) [64] vemos que se indica […*La fuente de hora fiable que se codifica en los sellos de tiempo proviene de la red de servidores NTP de la ACCV, de fiabilidad probada y que enlaza, en última instancia, con el organismo encargado de forma oficial de mantener la fuente nacional de tiempos, el Real Instituto y Observatorio de la Armada (ROA), de San Fernando (Cádiz)* […], que se trata de la fuente nombrada al principio del presente apartado.

Asimismo, vemos que en el sello se indican los datos firmados y la fecha (ver apartado posterior).

Por tanto, hemos confirmado que se cumplen los 3 puntos mencionados por la legislación europea: Fuente tiempo fiable, cualificado y relaciona datos y fecha.
Ahora, vamos a confirmar si cumple la RFC 3628:

- Sí incluye un identificador de la política:

Ilustración 66 - ASN1 ID Política

- Hemos generado dos sellos, y vemos que el ID cambia, por lo que confirmamos que si tienen un identificador único:

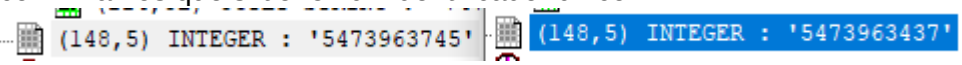


Ilustración 67 - ASN1 ID Sello

- El valor de la fecha / hora debe ser sincronizado con UTC

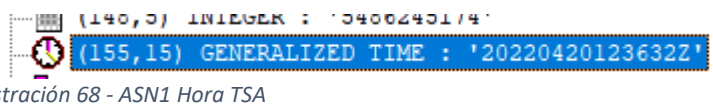


Ilustración 68 - ASN1 Hora TSA

- La política indicada anteriormente se indica en ACCV como orientada al sello de tiempo exclusivamente.

- Comprobamos que se referencia el dato firmado:

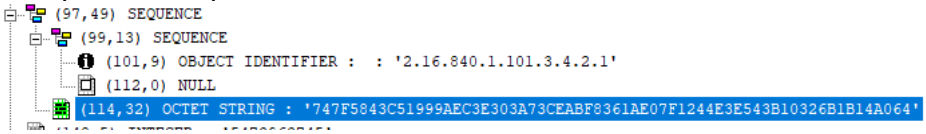


Ilustración 69 - ASN1 Dato firmado

- Se informa tanto del país de la autoridad, cómo de su nombre y unidad:

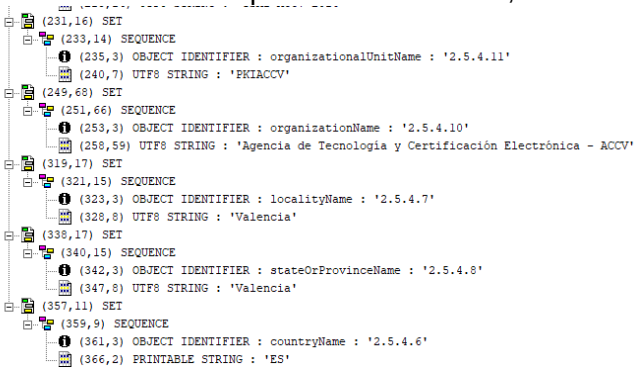


Ilustración 70 - ASN1 Info Autoridad Sello

Por lo que también cumple la RFC.

Adicionalmente al propio sello de tiempo, que acabamos de describir, es importante recordar que la firma debe contener la información de revocación en la propia firma, así, cuando se genere la firma, el proceso resumido (cómo hemos indicado anteriormente, cada formato de firma tiene un proceso distinto), debería ser:

- Se genera la firma con un certificado válido en ese momento
- Se añade un sello de tiempo para confirmar que en ese momento era válida
- Se realiza la validación para obtener la información del método de validación y se incorpora esta información en la propia firma
- Se añade un nuevo sello de tiempo que confirma que toda esta información se añadió cuando aún el certificado era válido

Concretando este punto, si nos referimos al formato XAdES y tomando el estándar referido anteriormente [58] la información que se añade (sin considerar los sellos de tiempo o información general de la firma) es:

- **Elemento "CompleteRevocationRefs"** → Incorpora las referencias de revocación que han sido usadas para validar el certificado firmante y el resto de los certificados del emisor, en general serán estructuras "CRLRefs" si la validación fue vía CRL o "OCSPRefs" si se hizo vía OCSP.
- **Elemento "CertificateValues"** → Incluye los certificados adicionales que han sido necesarios para validar el certificado firmante, como, por ejemplo, el certificado emisor
- **Elemento "RevocationValues"** → En lugar de las referencias en este elemento se incluye la información completa para comprobar la revocación, por ejemplo, la CRL completa que se ha usado para validar.

Tras todo ello, este apartado nos ha servido para conocer qué es una firma longeva, qué requisitos deben cumplirse y cómo podemos confirmar que así lo hagan.

4. Futuro – computación cuántica

Como hemos visto en apartados anteriores, la criptografía es la base del funcionamiento de las firmas electrónicas, sin embargo, existe una tecnología emergente y aún en sus primeros pasos (aunque su teoría ya empezó a formularse en los años 80 por parte de Paul Benioff [65]) como la criptografía cuántica que puede afectar en gran manera a la tecnología de firmas electrónicas.

Los computadores actuales basan su funcionamiento en bits, que permiten sólo dos estados, 0 y 1, sin embargo, los computadores cuánticos se basan en *qubits* (bits cuánticos) [66], que se basan en partículas subatómicas que aportan dos características importantes:

- **Superposición cuántica**: en lugar de poder representar 0 y 1, estas particulares pueden estar en múltiples estados al mismo tiempo, por lo que pueden representar combinaciones de 1 y 0 cada vez.

- **Enlazamiento cuántico**: esta propiedad en los *qubits* hace que cuando se añaden más *qubits* la capacidad de proceso aumenta de forma exponencial.

Una vez introducidos en esta tecnología, podemos ver que puede afectar en dos formas: cálculos sobre paradigmas actuales o nuevos algoritmos criptográficos.

4.1 Cálculos sobre paradigmas actuales

Hemos estudiado previamente la importancia de la potencia de computación, pero para realizar la comparación correspondiente en este caso debemos hablar de las clases de complejidad.

El tiempo de complejidad se denota con O [67] y determina cuanto tardaría un ordenador en ejecutar un algoritmo, teniendo en cuenta el número de operaciones básicas necesarias.

Una de las medidas más comunes es el tiempo logarítmico, T(n) = O((log n)x); y se considera una buena medida de complejidad, ya que según aumentamos *n* , es decir, la cantidad de datos, el tiempo no aumenta proporcionalmente. Para contrastar, podríamos considerar una complejidad exponencial, $O(n^x)$, en ese caso, cuantos más fuese la cantidad de datos, se incrementaría notablemente la dificultad, podemos comprobar de forma visual esta información con estos gráficos procedentes de *makeitreal* [67], vemos la complejidad exponencial (izquierda) contra la complejidad logarítmica (derecha):

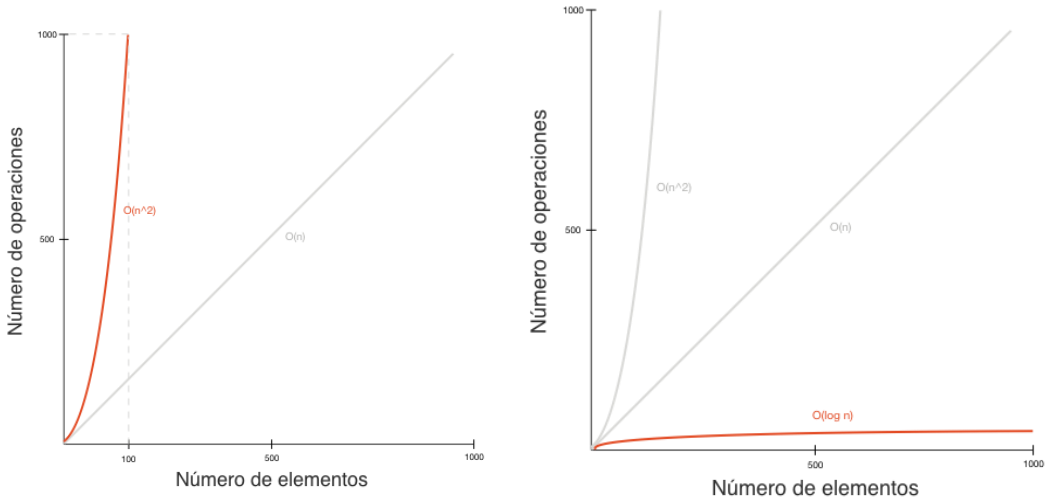


Ilustración 71 - Complejidad exponencial vs logarítmica

Si tomamos como base valores de diferentes tamaños, podemos ver la explicación de esas gráficas:

	O(n^2)	O(n^3)	O(log n)	O((log n)3)
n=2	4	8	0,301	0,903
n=8	64	512	0,903	2,709
n=16	256	4.096	1,204	3,612
n=32	1.024	32.768	1,505	4,515
n=256	65.536	16.777.216	2,408	7,224
n=512	262.144	134.217.728	2,709	8,127
n=1024	1.048.576	1.073.741.824	3,010	9,030
n= 8192	67.108.864	8.690.991.616	3,913	11,740

Siempre nos interesará, por tanto, un algoritmo que tenga una complejidad logarítmica frente a exponencial, ya que como vemos, si bien al principio las diferencias son leves, cuanto más aumentan los datos la complejidad se vuelve inasumible, mientras que en el caso logarítmico el dato no varía significativamente la complejidad.

Una vez indicado esto, vamos a analizar un caso concreto:

El algoritmo RSA se basa en el uso de números primos de gran tamaño, cuya multiplicación se usa cómo módulo para el cálculo de las claves (pública y privada), de forma que el cálculo sea fácil de generar y reversible fácilmente sólo si se conocen las claves.

La forma por tanto de atacar a este algoritmo es realizando la descomposición en factores de la clave, teniendo en cuenta que cómo hemos indicado antes se han usado dos números primos. Actualmente, dicho algoritmo es seguro, ya que no se ha encontrado ningún algoritmo capaz de realizar esta tarea en un tiempo razonable (logarítmico).

No obstante, basándose en el uso de la computación cuántica, Peter Shor ha ideado un algoritmo [68] que sí permitiría realizar la operación en un tiempo logarítmico.

Por el momento está pendiente de que existan ordenadores cuánticos con capacidad suficiente para aplicarse, pero en el año 2001 IBM [69] probó que el algoritmo era posible descomponiendo el número 15 en 3 y 5.

Actualmente, IBM ha creado un ordenador de 127 *qubits* [70], por lo que todavía están lejos de los necesarios para poder hacer uso de algoritmos como el de Shor, sobre el que se estima n·3 *qubits* necesarios por cada número de bit [71], así, si tuviéramos un número de 512 bits necesitaríamos 512·3 = 1536 *qubits*.

Si revisamos la siguiente comparativa [72], podemos ver el potencial de los ordenadores cuánticos:

Type of scaling	**Time to solve problem**				
Classical algorithm with exponential runtime	10 secs	2 mins	330 years	3300 years	Age of the universe
Quantum algorithm with polynomial runtime	1 min	2 mins	10 mins	11 mins	~24 mins

Ilustración 72 - Escalado classic vs quantum

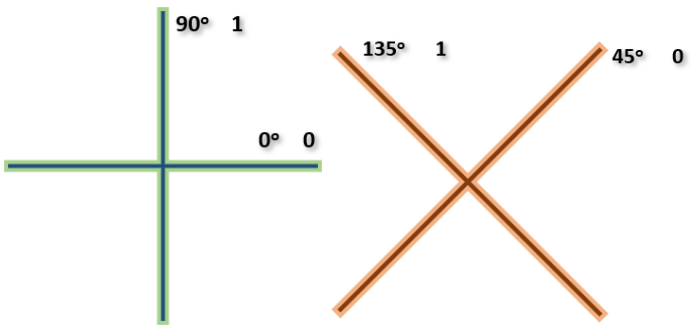
Por tanto, según avance la capacidad de *qubits* y se implementen nuevos algoritmos de ataque, los actuales algoritmos de firma / cifrado y las longitudes de clave usadas actualmente dejarán de considerarse seguras.

4.2 Nuevos protocolos

El funcionamiento de intercambio de claves presenta novedades con la criptografía cuántica, ya que los protocolos pueden varían teniendo en cuenta las propiedades de los *qubits*.

Un ejemplo lo encontramos en el protocolo BB84 [73], que sirve para distribución de claves usando computadores cuánticos, cuyo funcionamiento es el siguiente:

Partiendo del principio de la polarización de los protones, se definen dos bases:



Rectilíneas Diagonales

Como podemos ver, existen 2 pares de opciones:
Diagonales: 135º – 1 y 45º - 0
Rectilíneas: 90º – 1 y 0º - 0

- Usuario 1 genera una cadena de bits aleatoria
- Usuario 1 elije la base de forma aleatoria para cada bit:
  - Si elige 0, luego elegirá bien 45º o 0º
  - Si elige 1, luego elegirá bien 135º o 90º
- Usuario 2 mide cada protón usando una base aleatoria; si la base coincide con la que usó Usuario 1 el valor (1 o 0) coincidirá, si elige la base diferente el valor tiene un 50% de coincidir o ser distinto.
- Usuario 2 comunica por un canal (que puede ser inseguro) las bases que ha usado para medir los protones a Usuario 1.
- Usuario 1 indicará a Usuario 2 en cuáles ha acertado la base.
- Usuario 1 y Usuario 2 descartan los bits que se han medido con base errónea.
- Los bits restantes son la clave.

Hemos creado una simulación de este protocolo en Python, que se puede encontrar en el anexo, como ejemplo de ejecución:

Paso 1, Usuario 1 genera:
Bits [1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0]
Bases ['+', 'x', '+', '+', '+', 'x', 'x', '+', '+', 'x', 'x', 'x', '+', '+', 'x', 'x', '+', '+', '+', 'x', '+', 'x', 'x', '+', '+', 'x', 'x', 'x', '+', '+', '+', '+', '+', 'x', 'x', '+', '+', 'x', '+', '+']
Polarización: [90, 135, 90, 0, 0, 135, 135, 0, 0, 45, 135, 45, 0, 90, 135, 135, 90, 90, 0, 135, 90, 135, 45, 0, 0, 135, 45, 135, 90, 0, 90, 0, 90, 45, 135, 0, 0, 45, 0, 0]

Paso 2, Usuario 2 genera bases:
['+', 'x', '+', '+', '+', '+', 'x', 'x', '+', 'x', 'x', '+', 'x', 'x', '+', 'x', 'x', 'x', '+', '+', '+', '+', '+', '+', 'x', 'x', 'x', 'x', '+', '+', 'x', 'x', 'x', '+', '+', 'x', '+', '+', 'x', 'x']

Paso 3: Usuario 2 genera polarización:
[90, 135, 90, 0, 0, 0, 135, 45, 0, 45, 135, 0, 45, 135, 90, 135, 135, 45, 0, 0, 90, 90, 90, 0, 135, 135, 45, 135, 90, 0, 135, 45, 45, 0, 0, 45, 0, 0, 45, 135]

Paso 4: se comparan bases, eliminan bases erróneas y genera la clave:
[1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0]

Este es sólo un ejemplo de cómo aprovechar los ordenadores cuánticos, existen múltiples protocolos definidos y en el momento en que los mismos sean aplicables por la existencia de ordenadores con *qubits* suficientes también surgirán nuevos.

5. Caso práctico

En este apartado vamos a tomar una acción de la vida cotidiana que podría ser realizada mediante el uso de firmas electrónicas.

El caso que hemos pensado como ejemplo es la implementación de apertura y cierre de un automóvil. Es importante indicar, que se trata de un caso teórico, no vamos a detallar cómo se programaría, que tecnología se usaría para remitir y recibir la información y otros tipos de detalles.

Como hemos aprendido en capítulos anteriores, las firmas electrónicas nos aportan estas propiedades que pueden sernos de utilidad para este caso:

Integridad: no es posible modificar un dato firmado sin que se detecte en su validación. Esto nos es de utilidad para que nadie pueda modificar la orden que se da al coche.

Autenticación: sirve para confirmar que el firmante es quién dice ser. Esto nos sirve para que sólo el remitente / propietario / autorizado pueda realizar las acciones.

Para el funcionamiento del sistema se generarían dos pares de claves públicas / privadas, una asociada al vehículo y otra a la llave / mando.

En el coche existirían 3 tipos de usuarios:

- **Root / Fabricante**: permite abrir el coche, cerrar el coche y añadir y eliminar usuarios y administradores.
- **Administrador / Propietario**: permite abrir el coche, cerrar el coche y añadir y eliminar usuarios y administradores.
- **Usuario**: permite abrir y cerrar el coche,

En el coche la información se almacenaría usando las claves públicas del mando como IDs y asociándoles un rol:

ID - ClavePública	Rol
MFwwDQYJKoZI…	1
MIGeMA0GCSqG…	2
MIIBITANBgkq…	3
MIICIjANBgkqh…	3

Rol	Rol
1	Root
2	Admin
3	User

Este formato de almacenar la información permitiría añadir en un futuro nuevos roles sin afectar al registro de usuarios

La información que se almacenaría, por tanto, en cada lado sería:

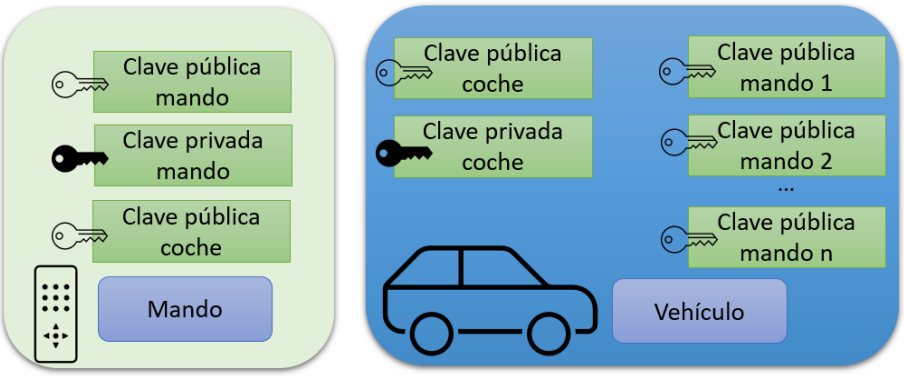


Ilustración 73 - Esquema almacenamiento claves

La llave / mando tendría un formato como el siguiente:

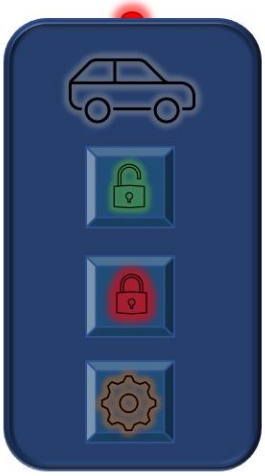


Ilustración 74 - Esquema llave - mando

Las acciones que se podrían hacer con el mando serían:

- Abrir coche
- Cerrar coche
- Acceso administrador
- Añadir usuario
- Eliminar usuario
- Solicitar acceso

Describimos ejemplos de funcionamiento:

Abrir Coche:

1- UsuarioMando1 pulsa el botón 1, lo cual genera y firma acción “AbrirCoche” con su clave privada.
2- Coche comprueba que la firma es válida y que la clave pública se encuentra registrada en su base de datos.
3- Coche firma respuesta “Abierto” o “No dado de alta” con su clave privada.
4- UsuarioLlave1 valida la firma de la respuesta, y pueden pasar las siguientes situaciones:
    - 5.1 No tiene la clave pública del coche registrada → Sirve para conocer que está intentando abrir un coche que no es el suyo o no ha sido suyo.

- 5.2 Si tiene la clave pública del coche registrada:
  - 5.2.1 Recibe una respuesta “No dado de alta” → sabe que ha sido dado de baja
  - 5.2.2 Recibe una respuesta “Abierto”
    - 5.2.2.1 El coche se abre:
    - 5.2.2.2 El coche no se abre: sabe que hay un problema en el sistema de apertura → debe comunicar con el taller o fabricante.

Esta acción la podría hacer un usuario con cualquier rol, no se indican los pasos para el cierre al ser equivalentes.

Alta de usuario

AdministradorLlave1 pulsa botón 3, lo que hace que se firme y remita la Acción “Administrar” con su clave privada.

Coche comprueba que la firma es válida y que la clave pública se encuentra registrada en su base de datos y que el rol sea 1 o 2.

Coche firma respuesta “Concedido” o “Denegado” con su clave privada y entra en modo configuración.

AdministradorLlave1 valida la firma de la respuesta, y pueden pasar 3 cosas:

- No tiene la clave pública del coche registrada → Sirve para conocer que está intentando abrir un coche que no es el suyo o no ha sido suyo.
- Si tiene la clave pública del coche registrada:
  - Recibe una respuesta “Denegado”, sabe que debe hablar con otro Administrador o con el concesionario para que pueda tener el permiso de administrador.
  - Recibe una respuesta “Concedido”
    - UsuarioLlave1 pulsa botón 3, lo que hace que se remita su clave pública al coche,
      - AdministradorLLave1 pulsa botón 1 para darle el rol de usuario
      - AdministradorLLave1 pulsa botón 2 para cancelar el alta.
      - AdministradorLLave1 pulsa botón 3 para darle el rol de administrador
        - En cualquiera de los 3 casos, se firma respuesta con: “Acción finalizada: alta rol 2”, “Acción finalizada: alta rol 3” o “Acción finalizada: alta cancelada”
    - Si no se recibe respuesta de un usuario que pulse botón Administrar en 10 segundos, se cancela la operación y se firma respuesta de “Cancelado por timeout”

Como muestra, mostramos el diagrama de flujo del primer caso:

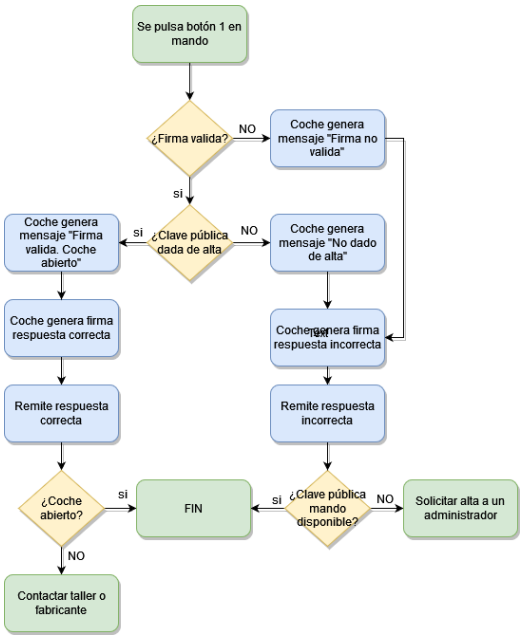


Ilustración 75 - Flujo botón 1

El esquema (resumido) general del funcionamiento por cada mensaje sería el siguiente:

- Mando1 usa la clave pública de Coche1 para cifrar la acción
- Mando1 usa su clave privada para realizar la firma.
- Coche1 usa clave pública de Mando1 para validar la firma
- Coche1 usa su clave privada para obtener el contenido cifrado.

Una forma de realizar estas acciones sería el uso de WS-Security [74], que permite la firma de mensajes autenticados, no obstante, como la implementación no forma parte del alcance de nuestro trabajo, es posible que otro estándar no orientado a web si no a otro canal de comunicación fuese más acorde, pero por su formato visual se muestra este como ejemplo:

Un ejemplo de petición (sin aplicar el cifrado) sería:

<?xml version="1.0" encoding="UTF-8"?>
- <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  - <soapenv:Header>
    - <wsse:Security soapenv:mustUnderstand="1" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-w
      secext-1.0.xsd">
      - <ds:Signature Id="SIG-5672" xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        - <ds:SignedInfo>
          - <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
              <ec:InclusiveNamespaces PrefixList="soapenv" xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#"/>
            </ds:CanonicalizationMethod>
            <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
          - <ds:Reference URI="#Id-14847766">
            - <ds:Transforms>
              - <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                  <ec:InclusiveNamespaces PrefixList="" xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#"/>
                </ds:Transform>
              </ds:Transforms>
              <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
              <ds:DigestValue>nCgl0VrbVlP2WMoCpRqedgAJsX8=</ds:DigestValue>
            </ds:Reference>
          </ds:SignedInfo>
          <ds:SignatureValue>LYj53K9A[...]Cwgsv1ma75YaIaERuQ731g==</ds:SignatureValue>
        - <ds:KeyInfo Id="KI-BA39385774350A03C1164924341781717015">
          - <wsse:SecurityTokenReference wsu:Id="STR-BA39385774350A03C1164924341781717016">
              <wsse:Reference URI="#X509-BA39385774350A03C1164924341781717014" ValueType="http://docs.oasi
            </wsse:SecurityTokenReference>
          </ds:KeyInfo>
        </ds:Signature>
        <wsse:BinarySecurityToken wsu:Id="X509-BA39385774350A03C1164924341781717014" ValueType="http://docs.oasi
          open.org/wss/2004/01/oasis-200401-wss-soap-message-
          security-1.0#Base64Binary">MIIHhzCCBm+gAwIBAgIQUue806ry19j/Ds4sUKYB7TANBgkqhkiG9w0BAQsFADCB
          [...]
          DA2oDSgMoYwaHR0cDovL2NybC5maXJtYXByb2Zlc2lvbmFsLmNvbS9jdWFsaWZpY2Fkb3MuY3JsMB0GA1UdDgQ
      </wsse:Security>
    </soapenv:Header>
  - <soapenv:Body xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id
    - <Peticion>
      - <Datos>
          <Accion>AbrirCoche</Accion>
        </Datos>
      </Peticion>
    </soapenv:Body>
</soapenv:Envelope>

Ilustración 76 - Ejemplo petición

Este sistema también podría aprovecharse para emitir comunicaciones seguras desde el vehículo, usando el cifrado, así usando la clave pública del destinatario podría emitir información cifrada con alguna utilidad. Un ejemplo de esto podría ser el realizar desde el fabricante un seguimiento del comportamiento del vehículo, ya sea para detectar problemas, obtener información para análisis de datos de todos sus coches o mejorar las revisiones periódicas.

El esquema de este caso sería el siguiente:

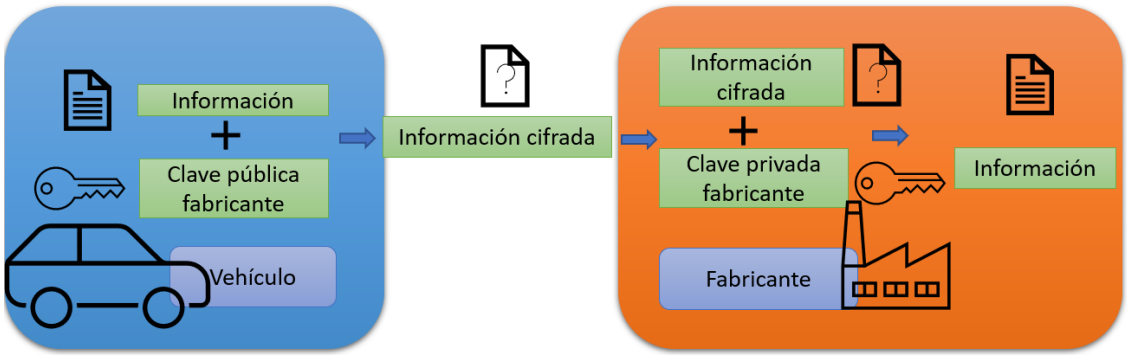


Ilustración 77 - Esquema caso cifrado

Esto permite que sólo el destinatario (fabricante) pueda leer esta información y ni otras empresas, ni los talleres, ni el propio usuario podría ni manipular ni visualizar la información.

Con respecto al caso general, es importante indicar que este sistema, al funcionar con permisos y tipos de usuario, podría tener un uso mucho más amplio en el futuro, así, en función de la llave del usuario podría obtener unos permisos o servicios del coche.

Si nos fijamos por ejemplo en Tesla [75] vemos que existen servicios de pago, estos servicios no necesitan que el coche sea adaptado, sino que se desbloquean al abonar su precio (suscripción), esta misma implementación podría realizarse con este sistema que hemos descrito, en ese caso, la llave podría ser adaptada para tener más opciones, ya sea con más botones o con una pantalla integrada.

También es interesante que este sistema se podría extrapolar al uso de teléfonos móviles, al permitir instalarse en los mismos certificados digitales, de forma que con una *app* podría, una vez esté dispuesto el sistema para llaves, usar el mismo como llave.

6. Conclusiones

En primer lugar, hemos podido concluir que **aún queda pendiente impulsar la implantación de la tecnología de firma digital y certificados electrónicos de forma general**, ya que existe aún ciudadanos bien reticentes a la misma, bien con falta de información o conocimientos para el uso de la misma. Se ha podido generar una web estática simple que puede servir para incorporar a modo de FAQ en páginas que ofrezcan servicios mediante el uso de certificados digitales.

Hemos podido observar que el uso de **la firma digital facilita los trámites en gran medida**, ofrecido un ahorro de tiempo y desplazamientos.

Asimismo, se ha podido confirmar que **las bases de la firma electrónica son sólidas**:

- Existe una legislación nacional y europea asociada que regula los requisitos que deben cumplir tanto los propios certificados como los emisores de los mismos.

- Las operaciones criptográficas que son utilizadas en tareas secundarias como los *hashes* o primarias como la firma se realizan con algoritmos matemáticos robustos que en la actualizad ofrecen suficiente seguridad para el uso de esta tecnología.

La tecnología de firma electrónica y certificados digitales **puede ser implementada para diferentes acciones de nuestro día a día**, hemos podido plantear que sería una opción posible para un sistema de apertura de un vehículo, incluso para control de diferentes funciones.

Hemos revisado que la **computación cuántica traerá importantes novedades** a las bases de la firma electrónica, con el uso de nuevos algoritmos, así como potencias superiores de cálculos que pueden hacer vulnerables algoritmos que ahora se consideran seguros.

Con respecto al desarrollo del propio proyecto, **ha sido posible abarcar las metas fijadas**, pudiendo analizar cada uno de los puntos planteados inicialmente haciendo uso de las pequeñas aplicaciones creadas para comprobar ciertos cálculos y funciones, se han encontrado ciertos problemas, tal como se indican en la sección de seguimientos de riesgos, pero todos ellos han sido solucionados sin afectar de forma relevante en el seguimiento del proyecto y plazos previstos.

En cuanto a **líneas de trabajo de futuro**, se podría hacer un análisis más a fondo centrándose en la computación cuántica, en especial cuando esta tecnología sea más accesible a la mayor parte de las personas, ya que actualmente únicamente planteado cómo afecta y uno de sus posibles algoritmos.

7. Glosario

- **AEAT**: Agencia Estatal de Administración Tributaria
- **ACCV**: Autoridad de Certificación de la Comunidad Valenciana
- **CMS**: Criptographic Message Syntax
- **CRL**: Certificate Revocation List (Lista de revocación de certificados)
- **DPC**: Declaración de Prácticas de Certificación
- **FNMT**: Fábrica Nacional de Moneda y Timbre
- **INE**: Instituto Nacional de Estadística
- **NTP**: Network Time Protocol
- **OCSP**: Online Certificate Status Protocol
- **OID**: Object Identifier (Identificador de Objeto)
- **PDF**: Portable Document Format
- **PKI**: Public Key Infraestructure (Infraestructura de Clave Pública)
- **PSC**: Prestador de Servicio de Certificación
- **RFC:** Request for Comments
- **RSA:** Rivest, Shamir y Adleman
- **TGSS**: Tesorería General de la Seguridad Social
- **TSA**: Time Stamp Authority (Autoridad de Sello de Tiempo)
- **TSL:** Trust Service status List (Lista de Servicios de Confianza)
- **XML:** eXtensible Markup Language (Lenguaje de Marcado Extensible)

8. Bibliografía

- [1] *Portal de Administración Electrónica - DatoObsae* [en línea] [consulta: 06/03/2022]. Disponible en: http://dataobsae.administracionelectronica.gob.es/
- [2] *INE - Encuesta sobre Equipamiento y Uso de Tecnologías de la Información y Comunicación en los hogares 2010* [en línea] [consulta: 23/02/2022]. Disponible en: https://www.ine.es/jaxi/Datos.htm?path=/t25/p450/a2010/l0/&file=08028.px
- [3] *INE - Encuesta sobre el uso de TIC y del comercio electrónico en las empresas Año 2020 – Primer trimestre de 202*1 [en línea] [consulta: 24/02/2022]. Disponible en: https://www.ine.es/prensa/tic_e_2020_2021.pdf
- [4*] Adobe - Adobe Digital Insights: Top 3 consumer trends shaping e-Signatures and how we will work in 2021* [en línea] [consulta: 25/02/2022]. Disponible en: https://blog.adobe.com/en/publish/2021/01/26/adobe-digital-insights-top-3-consumer-trends-shaping-e-signatures-and-how-we-will-work-in-2021
- [5] *Ley 6/2020, de 11 de noviembre – BOE* [en línea] [consulta: 09/03/2022]. Disponible en: https://www.boe.es/diario_boe/txt.php?id=BOE-A-2020-14046
- [6] *Reglamento UE 910/2014* [en línea] [consulta: 09/03/2022]. Disponible en https://www.boe.es/doue/2014/257/L00073-00114.pdf
- [7] *Política de firma de la AGE* [en línea] [consulta: 09/03/2022]. Disponible en: https://administracionelectronica.gob.es/ctt/politicafirma
- [8] *RFC 5280* [en línea] [consulta: 10/03/2022]. Disponible en: https://datatracker.ietf.org/doc/html/rfc5280
- [9] *ETSI TS 102 778-1 – PadES Overview* [en línea] [consulta: 11/03/2022]. Disponible en: https://www.etsi.org/deliver/etsi_ts/102700_102799/10277801/01.01.01_60/ts_10277801v010101p.pdf
- [10] *ETSI TS 101 733 - CMS Advanced Electronic Signatures (CAdES)* [en línea] [consulta: 11/03/2022]. Disponible en: https://www.etsi.org/deliver/etsi_ts/101700_101799/101733/02.01.01_60/ts_101733v020101p.pdf
- [11] *Ley 11/2021 de 8 de julio* [en línea] [consulta: 15/03/2022]. Disponible en: https://www.boe.es/buscar/doc.php?id=BOE-A-2021-11473
- [12] *Los mayores de 50 saben lo que quieren: el 80% reclama a su banco que vuelva la atención presencial - 65ymas* [en línea] [consulta: 11/03/2022]. Disponible en: https://www.65ymas.com/economia/banca/mayores-50-saben-quieren-80-reclama-su-banco-vuelva-atencion-presencial_36568_102.html
- [13] *Obligación de tramitación electrónica – Ministerio del interior* [en línea] [consulta: 07/03/2022]. Disponible en: http://www.interior.gob.es/web/servicios-al-ciudadano/registro-de-documentacion/obligacion-de-tramitacion-electronica
- [14] *Desde el pasado 1 de octubre los trabajadores autónomos están obligados a realizar sus trámites por vía electrónica, incluida la recepción y firma de notificacions - Seguridad Social* [en línea] [consulta: 07/03/2022]. Disponible en: https://www.seg-social.es/wps/portal/wss/internet/HerramientasWeb/51148#51149
- [15] *La Seguridad Social presenta retrasos de hasta seis meses en el pago de las prestaciones por maternidad y paternidad - BebesYmás* [en línea] [consulta: 07/03/2022]. Disponible en: https://www.bebesymas.com/noticias/seguridad-social-presenta-retrasos-seis-meses-pago-prestaciones-maternidad-paternidad

- [16] *Cuerpo Nacional de Policía – Certificados electrónicos* [en línea] [consulta: 07/03/2022]. Disponible en: https://www.dnielectronico.es/PortalDNIe/PRF1_Cons02.action?pag=REF_430&id_menu=30
- [17] *Sede – FNMT Certificado de Persona Física* [en línea] [consulta: 07/03/2022]. Disponible en: https://www.sede.fnmt.gob.es/preguntas-frecuentes/certificado-de-persona-fisica/-/asset_publisher/eIal9z2VE0Kb/content/1038-son-gratuitos-todos-estos-tramites-?inheritRedirect=false&redirect=https%3A%2F%2Fwww.sede.fnmt.gob.es%2Fpreguntas-frecuentes%2Fcertificado-de-persona-fisica%3Fp_p_id%3D101_INSTANCE_eIal9z2VE0Kb%26p_p_lifecycle%3D0%26p_p_state%3Dnormal%26p_p_mode%3Dview%26p_p_col_id%3Dcolumn-2%26p_p_col_count%3D1
- [18] *Acceso Administración con DNIe – PlayGoogle.com* [en línea] [consulta: 18/04/2022]. Disponible en: https://play.google.com/store/apps/details?id=com.dnieloginwidget
- [19] *Descripción DNIe 4.0* [en línea] [consulta: 07/03/2022]. Disponible en: https://www.dnielectronico.es/PortalDNie/PRF1_Cons02.action?pag=REF_110&id_menu=1
- [20] van de Watering, Marek, *The Impact of Computer Technology on the Elderly* [en línea] Essay – Vrije Universiteit, 2004 [consulta: 13/03/2022]. Disponible en: https://www.marekvandewatering.com/texts/HCI_Essay_Marek_van_de_Watering.pdf
- [21] *Porcentaje de adultos (16 a 74 años) que en los últimos 12 meses han realizado tareas relacionadas con conocimientos informáticos por sexo, grupos de edad y tipo de tarea - INE* [en línea] [consulta: 13/03/2022]. Disponible en: https://www.ine.es/jaxiT3/Tabla.htm?t=46292
- [22] *Los avances tecnológicos que revolucionan el mundo rural - Levante-EMV* [en línea] [consulta: 13/03/2022]. Disponible en: https://www.levante-emv.com/vida-y-estilo/tecnologia/2016/11/03/avances-tecnologicos-revolucionan-mundo-rural-12318523.html
- [23] *Las nuevas tecnologías para los ancianos y mayores: ¿Cómo podemos ayudarles? - Atenzia* [en línea] [consulta: en 13/03/2022]. https://teleasistencia.es/es/blog/ocio-en-la-tercera-edad/las-nuevas-tecnologias-para-los-ancianos-y-mayores-como-podemos-ayudarles
- [24] *6.6 Hogares que tienen acceso a Internet y hogares que tienen ordenador. Porcentaje de menores usuarios de TIC - INE* [en línea] [consulta: 13/03/2022]. Disponble en: https://www.ine.es/ss/Satellite?L=es_ES&c=INESeccion_C&cid=1259925529799&p=%5C&pagename=ProductosYServicios%2FPYSLayout&param1=PYSDetalle&param3=1259924822888
- [25] *Components of a PKI* [en línea] [consulta: 10/03/2022]. Disponible en: https://www.ncsc.gov.uk/collection/in-house-public-key-infrastructure/introduction-to-public-key-infrastructure/components-of-a-pki
- [26] *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile - DataTracker IETF* [en línea] [consulta: 10/03/2022]. Disponible en: https://datatracker.ietf.org/doc/html/rfc5280
- [27] *OID Repository - oid-info.com* [en línea] [consulta: 09/03/2022]. Disponible en: http://www.oid-info.com/cgi-bin/display?oid=1.3.6.1.5.5.7.48.1&action=display
- [28] *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP* [en línea] [Consultado en 10/03/2022]. Disponible en: https://datatracker.ietf.org/doc/html/rfc6960

- [29] *OpenSSL ocsp* [en línea] [consulta: 10/03/2022]. Disponible en: https://www.openssl.org/docs/man1.1.1/man1/ocsp.html
- [30] *Preguntas Frecuentes - VALIDe* [en línea] [consulta: 10/03/2022]. Disponible en: https://valide.redsara.es/valide/faqs.html#1
- [31] *Password cracking speed - The Security Factor* [en línea] [consulta: 11/03/2022]. Disponible en: https://www.thesecurityfactory.be/password-cracking-speed/
- [32] *Moore´s law - Britannica* [en línea] [consulta: 12/03/2022]. Disponible en: https://www.britannica.com/technology/Moores-law
- [33] *Seguridad - DNI y Pasaporte, Cuerpo Nacional de Policía* [en línea] [consulta: 12/03/2022]. Disponible en: https://www.dnielectronico.es/PortalDNIe/PRF1_Cons02.action?pag=REF_2100&id_menu=[6]
- [34] *Acerca de las Tarjetas Inteligentes - Sede electrónica FNMT* [en línea] [consulta: 12/03/2022]. Disponible en: https://www.sede.fnmt.gob.es/preguntas-frecuentes/acerca-de-las-tarjetas-inteligentes/-/asset_publisher/kL4tmWBYhjzW/content/1647-cuantos-intentos-de-pin-y-codigo-de-desbloqueo-admite-la-tarjeta-criptografica-?inheritRedirect=false
- [35] *Reglamento Nº 910/2014 - BOE* [en línea] [consulta: en 13/03/2022]. Disponible en: https://www.boe.es/doue/2014/257/L00073-00114.pdf
- [36] Cl@ve identificación – Descargas [en línea] [consulta en 13/03/2022]. Disponible en: https://administracionelectronica.gob.es/ctt/clave/descargas
- [37] *ITU-T X.520 - International Telecomunication Union* [en línea] [consulta: 14/03/2022]. Disponible en: https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-X.520-201210-S!!PDF-E&type=items
- [38] *ISO 3161 Country Codes* [en línea] [consulta: 14/03/2022]. Disponible en: https://www.iso.org/iso-3166-country-codes.html
- [39] *POLÍTICA Y PRÁCTICAS DE CERTIFICACIÓN PARTICULARES DE LOS CERTIFICADOS DE PERSONAS FÍSICAS DE LA "AC FNMT USUARIOS" - FNMT* [en línea] [consulta: 14/03/2022]. Disponible en: https://www.sede.fnmt.gob.es/documents/10445900/10536309/dpc_personasfisicas.pdf
- [40] *DEFINICIÓN DE PERFILES AC USUARIOS - FNMT* [en línea] [consulta: 14/03/2022]. Disponible en: https://www.sede.fnmt.gob.es/documents/10445900/10575386/perfiles_certificados_ac_usuarios.pdf
- [41] *AC FNMT Usuarios* [en línea] [consulta: 21/03/2022]. Disponible en: https://www.sede.fnmt.gob.es/dpcs/acusuarios
- [42] *Lista de confianza de prestadores cualificados de servicios electrónicos de confianza* [en línea] [consulta: 21/03/2022]. Disponible en: https://sedediatid.mineco.gob.es/Prestadores/Paginas/Inicio.aspx
- [43] SABATER FÚSTER, Sabater; DE LA GUÍA MARTÍNEZ, Dolores; HERNÁNDEZ ENCINAS, Luis; MONTOYA VITINI, Fausto; MUÑOZ MASQUÉ, Jaime. *Técnicas Criptográficas de protección de datos*. Madrid: Ra-Ma, 2004. ISBN 9788478975943.
- [44] Cryptographic Hash Functions Explained [en línea] [consulta: 05/04/2022]. Disponible en: https://komodoplatform.com/en/academy/cryptographic-hash-function/
- [45] *Collision Attack* [en línea] [consulta: 05/04/2022]. Disponible en: https://en.wikipedia.org/wiki/Collision_attack
- [46] *Birthday problem* [en línea] [consulta: 06/04/2022]. Disponible en: https://en.wikipedia.org/wiki/Birthday_problem

- [47] *FIPS PUB 180-4 Secure Hash Standard (SHS)* [en línea] [consulta: 06/04/2022]. Disponible en: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf
- [48] *MD5 vulnerable to collision atacks* [en línea] [consulta: 06/04/2022]. Disponible en: https://www.kb.cert.org/vuls/id/836068
- [49] *The first collision for full SHA-1* [en línea] [consulta: 06/04/2022]. Disponible en: https://link.springer.com/chapter/10.1007/978-3-319-63688-7_19
- [50] *ISO 32000-1:2008* [en línea] [consulta: 06/04/2022]. Disponible en: https://www.iso.org/standard/51502.html
- [51] *W3C - XML Signature Syntax and Processing Version 1.1* [en línea] [consulta: 06/04/2022]. Disponible en: https://www.w3.org/TR/xmldsig-core1/
- [52] *Cryptographic Message Syntax (CMS)* [en línea] [consulta: 07/04/2022]. Disponible en: https://datatracker.ietf.org/doc/html/rfc5652
- [53] *The counterSignature Element* [en línea] [consulta: 07/04/2022]. Disponible en: https://www.w3.org/TR/XAdES/#Syntax_for_XAdES_The_CounterSignature_element
- [54] *Cryptographic Message Syntax (CMS) - Countersignature* [en línea] [consulta: 07/04/2022]. Disponible en: https://datatracker.ietf.org/doc/html/rfc3852#section-11.4
- [55] *REGULATION (EU) No 910/2014 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL* [en línea] [consulta: 07/04/2022]. Disponible en: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=uriserv:OJ.L_.2014.257.01.0073.01.ENG
- [56] *Directive 1999/93/EC of the European Parliament and of the Council* [en línea] [consulta: 07/04/2022]. Disponible en: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=celex%3A31999L0093
- [57] *What is a qualified signature/seal creation device (QSCD)?* [en línea] [consulta: 08/04/2022]. Disponible en: https://ec.europa.eu/cefdigital/wiki/display/CEFDIGITAL/eSignature+FAQ#eSignatureFAQ-6
- [58] *XML Advanced Electronic Signature (XAdES)* [en línea] [consulta: 11/04/2022]. Disponible en: https://www.w3.org/TR/XAdES/
- [59] *Hora – Real Observatorio de la Armada* [en línea] [consulta: 13/04/2022]. Disponible en: https://armada.defensa.gob.es/ArmadaPortal/page/Portal/ArmadaEspannola/cienciaobservatorio/prefLang-es/06Hora--00Hora
- [60] *ETSI EN 319 142-2 V1.1.1 Armada* [en línea] [consulta: 13/04/2022]. Disponible en: https://www.etsi.org/deliver/etsi_en/319100_319199/31914202/01.01.01_60/en_31914202v010101p.pdf
- [61] *RFC 3628 – Policy Requirements for Time-Stamp Authorities* [en línea] [consulta: 13/04/2022]. Disponible en: https://datatracker.ietf.org/doc/html/rfc3628
- [62] *España – Trusted List* [en línea] [consulta: 13/04/2022]. Disponible en: https://sedediatid.mineco.gob.es/Prestadores/TSL/TSL.pdf
- [63] *EU Trust Services Dashboard* [en línea] [consulta: 13/04/2022]. Disponible en: https://esignature.ec.europa.eu/efda/tl-browser/#/screen/tl/ES/10/7
- [64] *ACCV - Política de Sellado de Tiempo de la Agencia de Tecnología y Certificación Electrónica* [en línea] [consulta: 13/04/2022]. Disponible en: https://www.accv.es/fileadmin/Archivos/Politicas_pdf/PoliticaSelladoTiempov2.0_pub.pdf
- [65] *How Benioff Paul´s Quantum Computing Theory Formed The Foundation Of Quantum Computers?* [en línea] [consulta: 29/05/2022]. Disponible en:

https://www.yourtechstory.com/2020/01/10/benioff-paul-quantum-computing-theory/
- [66] *Qué es un ordenador cuántico y su funcionamiento* [en línea] [consulta: 30/04/2022]. Disponible en: https://www.seas.es/blog/informatica/que-es-un-ordenador-cuantico-y-su-funcionamiento/
- [67] *Complejidad (Big-O)* [en línea] [consulta: 02/05/2022]. Disponible en: https://guias.makeitreal.camp/algoritmos/complejidad
- [68] *Shor´s factoring algorithm* [en línea] [consulta: 04/05/2022]. Disponible en: https://www.quantiki.org/wiki/shors-factoring-algorithm
- [69] *It's been 20 years since "15" was factored on quantum hardware* [en línea] [consulta: 04/05/2022]. Disponible en: https://research.ibm.com/blog/factor-15-shors-algorithm
- [70] *IBM creates largest ever superconducting quantum computer* [en línea] [consulta: 06/05/2022]. Disponible en: https://www.newscientist.com/article/2297583-ibm-creates-largest-ever-superconducting-quantum-computer/
- [71] *The mathematics behind quantum computing* [en línea] [consulta: 06/05/2022]. Disponible en: https://www.math.stonybrook.edu/~tony/whatsnew/may07/quantumI.html
- [72] *Five strategies to prepare for paradigm-shifting quantum technology* [en línea] [consulta: 07/05/2022]. Disponible en: https://www.ibm.com/thought-leadership/institute-business-value/report/quantumstrategy
- [73] *A Survey of the Prominent Quantum Key Distribution Protocol* [en línea] [consulta: 07/05/2022]. Disponible en: https://www.cse.wustl.edu/~jain/cse571-07/ftp/quantum/
- [74] *OASIS Web Services Security (WSS)* [en línea] [consulta: 09/05/2022]. Disponible en: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss
- [75] *Tesla – Soporte - Conectividad* [en línea] [consulta: 09/05/2022]. Disponible en: https://www.tesla.com/es_ES/support/connectivity

9. Anexos

Anexo 1: Preguntas encuestas

A continuación, listamos las preguntas utilizadas en la encuesta.

Con respecto a la demografía:

Pregunta	Tipo de pregunta	Opciones
Edad	Desplegable	• 18-25 • 26-35 • 36-45 • 46-55 • 56-65 • >65
Sexo	Opción a elegir	• Hombre • Mujer • Prefiero no contestar
Nivel de estudios	Desplegable	• Educación primaria • Graduado escolar • Bachillerato • FP Grado medio o equivalente • FP Grado superior o equivalente • Estudios universitarios

Mostramos el bloque de investigación:

Pregunta	Tipo de pregunta	Opciones
Conocimiento informático	Desplegable	• Nivel usuario • Avanzado • Profesional • Experto
¿Te parece más segura una firma física o una firma electrónica?	Opción a elegir	• Firma física (manuscrita) • Firma electrónica
¿Crees que pueden suplantar tu identidad en una firma electrónica	Opción a elegir (Sí/No)	• Sí • No
¿Crees que se puede falsificar una firma electrónica?	Opción a elegir (Sí/No)	• Sí • No
¿Tienes o has tenido certificado digital?	Opción a elegir (Sí/No)	• Sí • No
¿Sabes que el DNI incorpora un certificado digital?	Opción a elegir (Sí/No)	• Sí • No
¿Sabrías cómo conseguir un certificado digital (diferente al del DNI)?	Opción a elegir (Sí/No)	• Sí • No

¿Has usado en algún trámite la firma electrónica / certificado digital?	Opción a elegir (Sí/No)	• Sí • No
¿Te parece sencillo el uso de la firma electrónica?	Opción a elegir (Sí/No)	• Sí • No
¿Por qué no lo has usado?	Opción a elegir	• No dispongo de certificado digital • No sé cómo usarlo • No me parece seguro • Otro
En general, ¿Te parecen seguros los trámites online, como el uso de certificados digitales y firmas digitales?	Escala	• 1 (Nada seguro) a 5 (Muy seguro)
¿Qué problemas de seguridad / confianza te generan?	Texto libre	

Anexo 2: Creación de un certificado

1 – Creamos el certificado raíz

2 – Crearemos un CSR (Certificate Signing Request) con los datos de nuestro certificado
3 – Usaremos el certificado raíz para firmar nuestro certificado

Para ello nos ayudaremos de OpenSSL, y seguimos los siguientes pasos:

1.1 Creamos la clave privada (en nuestro caso, longitud 2048): openssl genrsa -out cert_raiz_ddl_fnmt.pem 2048

1.2 Generamos a partir de la clave el certificado auto-firmado raíz: openssl req -new -x509 -key cert_raiz_ddl_fnmt.pem -out raiz_ddl_fnmt.cer -days 5475 -config ddl.cnf

```
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AT]:ES
State or Province Name (full name) []:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:FNMT-RCM
Organizational Unit Name (eg, section) []:Ceres
Common Name (e.g. server FQDN or YOUR name) []:AC FNMT Usuarios
```

2.1 Creamos otra clave privada: openssl genrsa -out cert_final_ddl_fnmt.pem 2048

2.2 Generamos a partir de la clave anterior el CSR de generación de nuestro certificado personal: openssl req -new -key cert_final_ddl_fnmt.pem -out cert_final_ddl_fnmt.csr -config ddl.cnf

```
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AT]:ES
State or Province Name (full name) []:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:DE DUEÑAS LEDESMA DAVID - [redacted]
Email Address []:
Nombre []:DAVID
Apellidos []:DE DUEÑAS LEDESMA
ID []:[redacted]
```

3.1 Usamos el CSR como origen usando el certificado raíz para la firma: openssl x509 -req -in cert_final_ddl_fnmt.csr -CA raiz_ddl_fnmt.cer -CAkey cert_raiz_ddl_fnmt.pem -set_serial 12345 -days 1460 -out cert_final_ddl_fnmt.pem

4.1 Usamos tanto la clave pública como la privada para generar un fichero p12 y hacer nuestro certificado exportable e instalable: openssl pkcs12 -export -in cert_final_ddl_fnmt.pem -inkey cert_final_ddl_fnmt_clave.pem -out cert_final_ddl_fnmt.p12

Anexo 3: Cálculo fuerza bruta password

A continuación, se muestra el tiempo de averiguar contraseñas aleatorias en función de su longitud:

Longitud 4 → 36^4:

```
Prueba 1: Tiempo empleado: 0.045877695083618164 segundos
Prueba 2: Tiempo empleado: 0.21758818626403809 segundos
Prueba 3: Tiempo empleado: 0.17855072021484375 segundos
Prueba 4: Tiempo empleado: 0.41637754440307617 segundos
Prueba 5: Tiempo empleado: 0.07878971099853516 segundos
Prueba 6: Tiempo empleado: 0.038895368576049805 segundos
Prueba 7: Tiempo empleado: 0.1406242847442627 segundos
Prueba 8: Tiempo empleado: 0.1765270233154297 segundos
Prueba 9: Tiempo empleado: 0.23038291931152344 segundos
Prueba 10: Tiempo empleado: 0.03490567207336426 segundos
```

El tiempo medio es inferior a medio minuto.

Si aumentamos a 5, 36^5, el tiempo ya aumenta considerablemente:

```
Prueba 1: Tiempo empleado: 11.702519655227661 segundos
Prueba 2: Tiempo empleado: 4.672954797744751 segundos
Prueba 3: Tiempo empleado: 13.370466232299805 segundos
Prueba 4: Tiempo empleado: 12.095652103424072 segundos
Prueba 5: Tiempo empleado: 6.055535316467285 segundos
Prueba 6: Tiempo empleado: 6.670164346694946 segundos
Prueba 7: Tiempo empleado: 4.627625942230225 segundos
Prueba 8: Tiempo empleado: 2.281898021697998 segundos
Prueba 9: Tiempo empleado: 11.175145387649536 segundos
Prueba 10: Tiempo empleado: 7.494987964630127 segundos
```

Como vemos, el tiempo aumenta considerablemente, una media de 9,2 segundos.

Si ahora aumentamos la longitud a 6, 36^6:

```
Prueba 1: Tiempo empleado: 486.311972618103 segundos
Prueba 2: Tiempo empleado: 182.83970665931702 segundos
Prueba 3: Tiempo empleado: 441.78739285469055 segundos
Prueba 4: Tiempo empleado: 468.89511013031006 segundos
Prueba 5: Tiempo empleado: 459.075336933136 segundos
Prueba 6: Tiempo empleado: 159.90825533866882 segundos
Prueba 7: Tiempo empleado: 251.98185014724731 segundos
Prueba 8: Tiempo empleado: 410.36144518852234 segundos
Prueba 9: Tiempo empleado: 442.1203098297119 segundos
Prueba 10: Tiempo empleado: 264.2384157180786 segundos
```

Si ahora aumentamos la longitud a 7, 36^7:

```
Prueba 1: Tiempo empleado: 1093.3318796157837 segundos
Prueba 2: Tiempo empleado: 11134.887996912003 segundos
Prueba 3: Tiempo empleado: 1960.8448708057404 segundos
Prueba 4: Tiempo empleado: 5880.056178808212 segundos
Prueba 5: Tiempo empleado: 12015.24956703186 segundos
Prueba 6: Tiempo empleado: 11660.675790309906 segundos
Prueba 7: Tiempo empleado: 9116.321274757385 segundos
Prueba 8: Tiempo empleado: 2350.931504011154 segundos
Prueba 9: Tiempo empleado: 1185.2127101421356 segundos
Prueba 10: Tiempo empleado: 10329.014299631119 segundos
```

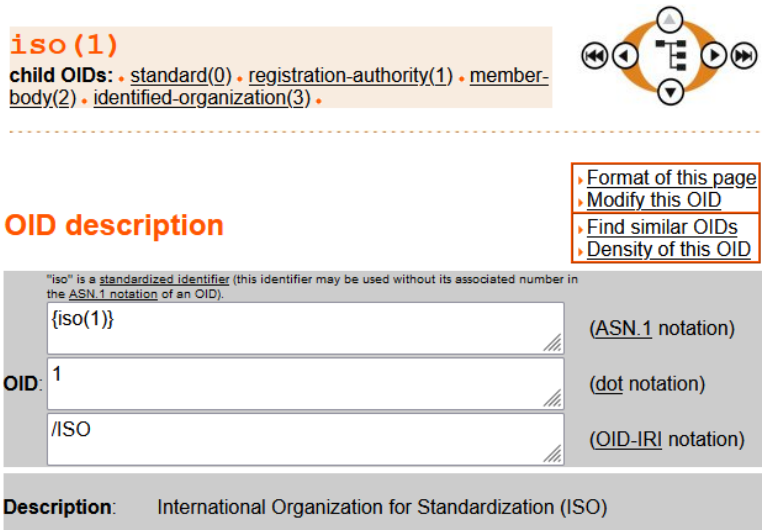
A modo de ejemplo, si usáramos sólo caracteres numéricos, para este último caso probado (10^7):

```
Prueba 1: Tiempo empleado: 1.6276192665100098 segundos
Prueba 2: Tiempo empleado: 1.4272119998931885 segundos
Prueba 3: Tiempo empleado: 1.6057333946228027 segundos
Prueba 4: Tiempo empleado: 1.4271833896636963 segundos
Prueba 5: Tiempo empleado: 1.8979520797729492 segundos
Prueba 6: Tiempo empleado: 1.8600525856018066 segundos
Prueba 7: Tiempo empleado: 2.313840866088867 segundos
Prueba 8: Tiempo empleado: 0.6981325149536133 segundos
Prueba 9: Tiempo empleado: 2.4424967765808105 segundos
Prueba 10: Tiempo empleado: 2.291900157928467 segundos
```

Anexo 4: Búsqueda paso a paso de un OID

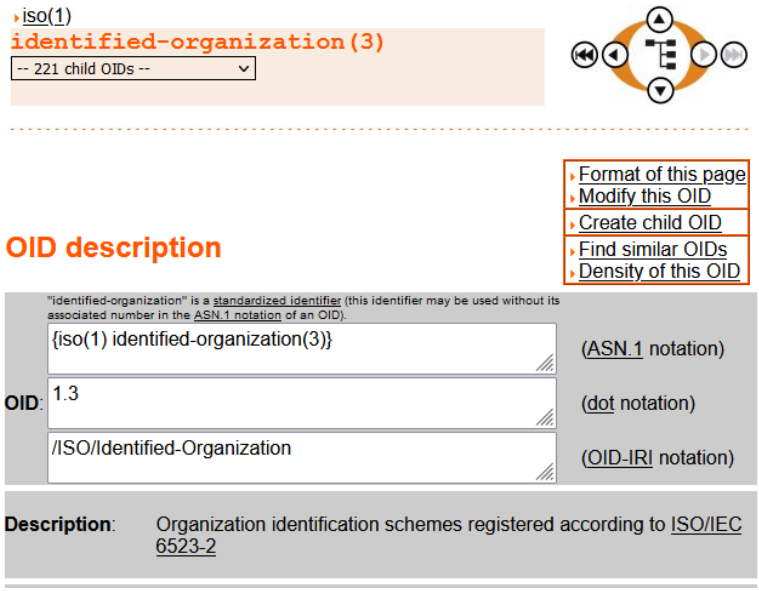
A continuación, listamos los pasos seguidos en la web “http://www.oid-info.com/” consultando un OID desde su nivel jerárquico inicial hasta el nivel más profundo:

OID 1.



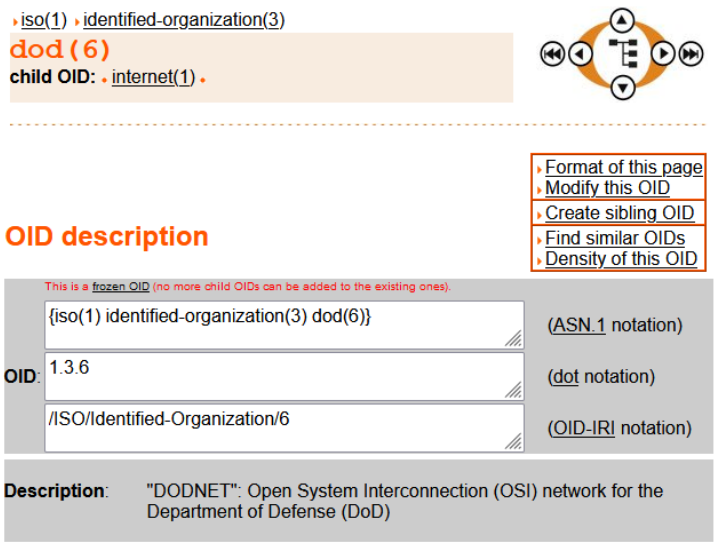
Se corresponde con la Organización internacional de estandarización (ISO) y finalizamos con

OID 1.3.



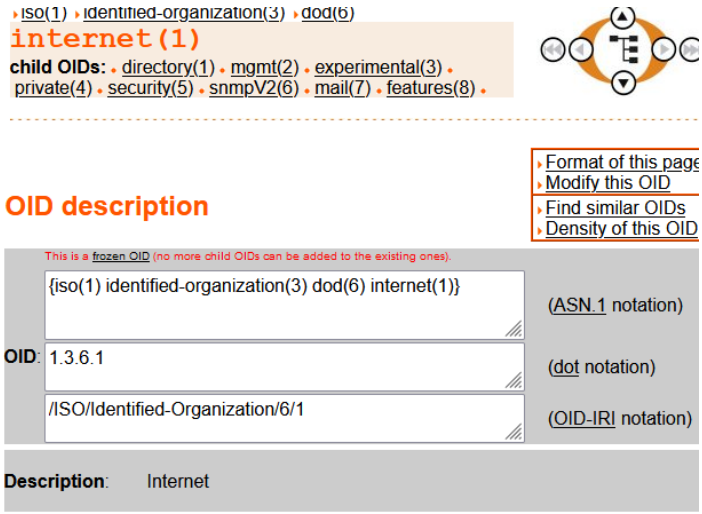
Hasta aquí vemos que se trata de la identificación de esquemas acordes a ISO/IEC 6523-3

OID 1.3.6.



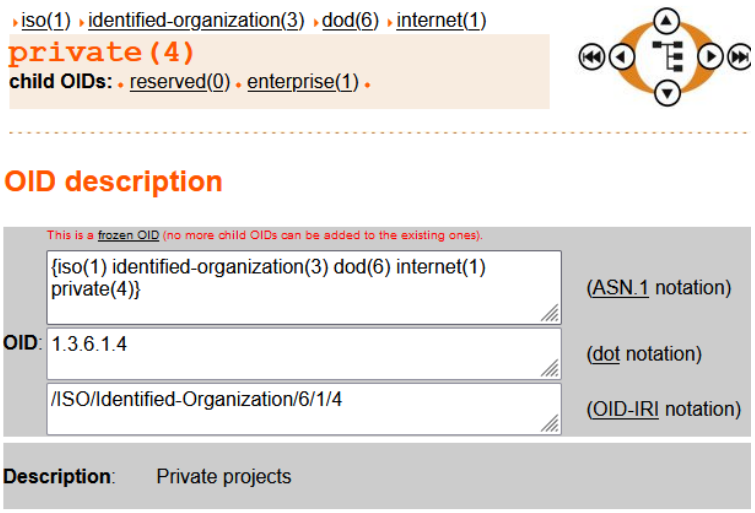
Este OID pertenece registrado al estándar OSI del DoD

OID 1.3.6.1.



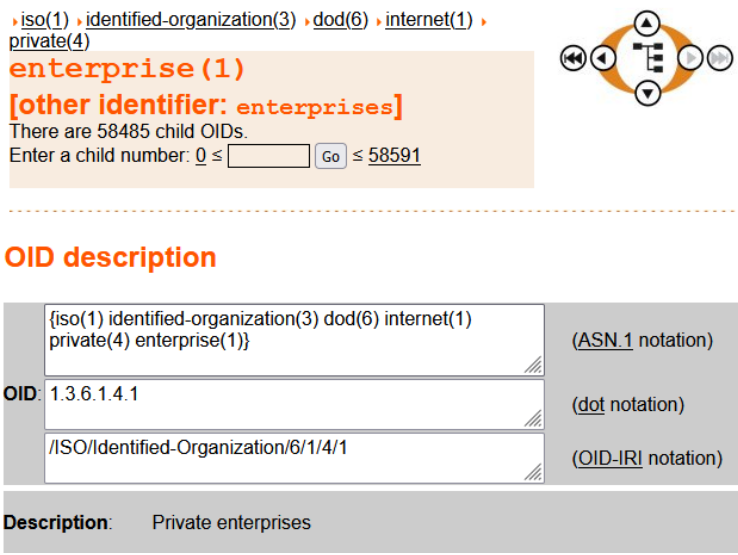
Esta parte pertenece registrado para el uso de Internet

OID 1.3.6.1.4.



Esto nos informa que pertenece a una organización privada.

OID 1.3.6.1.4.1



Ahora nos informa que dentro de los Organismos privados, pertenece a una empresa.

OID 1.3.6.1.4.1.5734

iso(1) identified-organization(3) dod(6) internet(1) private(4) enterprise(1)

5734

child OIDs: attrFields(1) fnmtCertificatePolicyID(3)

OID description

OID:	{iso(1) identified-organization(3) dod(6) internet(1) private(4) enterprise(1) 5734}	(ASN.1 notation)
	1.3.6.1.4.1.5734	(dot notation)
	/ISO/Identified-Organization/6/1/4/1/5734	(OID-IRI notation)

Description: Fábrica Nacional de Moneda y Timbre - Real Casa de la Moneda

Ahora vemos que la empresa privada que lo ha registrado es la FNMT.

OID 1.3.6.1.4.1.5734

iso(1) identified-organization(3) dod(6) internet(1) private(4) enterprise(1) 5734

fnmtCertificatePolicyID(3)

child OIDs: fnmtPublicAministrationCertPolicyID(3) europeanPublicAministrationCertPolicyID(4) fnmtQualifiedCertificatePolicyID(5) fnmtNaturalPersonCertPolicyID(10)

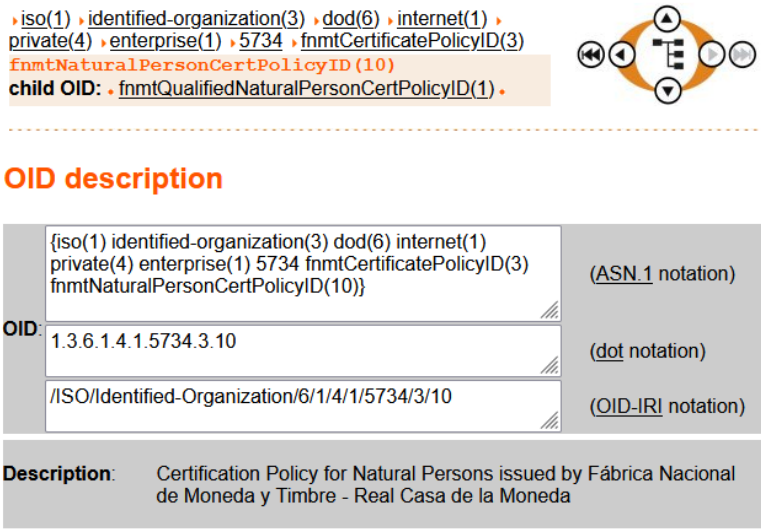
OID description

OID:	{iso(1) identified-organization(3) dod(6) internet(1) private(4) enterprise(1) 5734 fnmtCertificatePolicyID(3)}	(ASN.1 notation)
	1.3.6.1.4.1.5734.3	(dot notation)
	/ISO/Identified-Organization/6/1/4/1/5734/3	(OID-IRI notation)

Description: Policies used by Fábrica Nacional de Moneda y Timbre - Real Casa de la Moneda

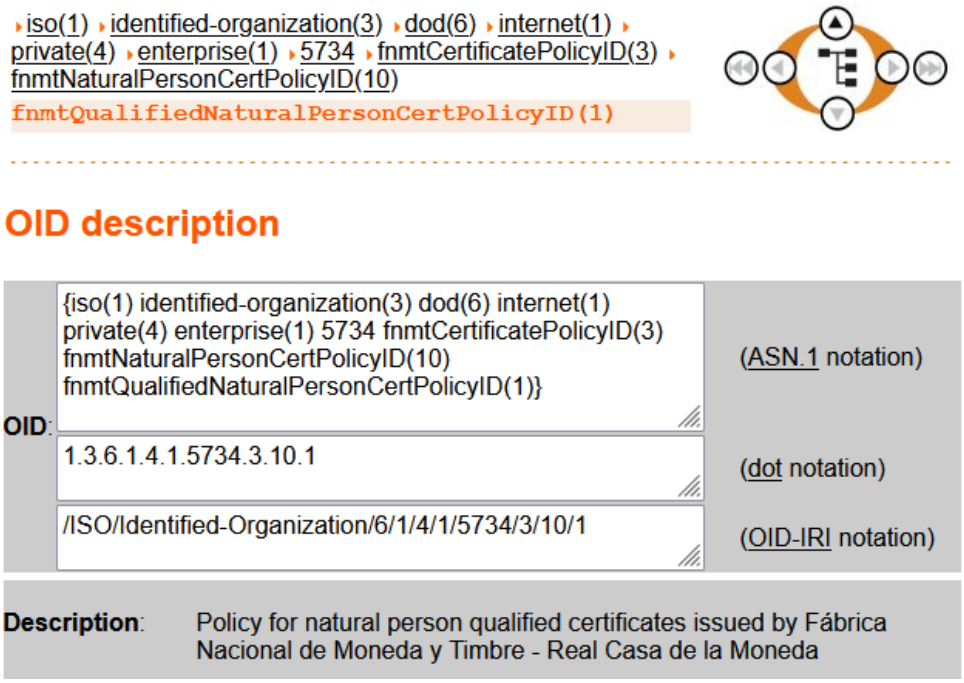
Nos indica que se trata de las políticas usadas por la FNMT.

OID 1.3.6.1.4.1.5734.3.10



Ahora podemos ver que de entre las políticas de la FNMT, se trata de aquellas para las personas físicas.

OID 1.3.6.1.4.1.5734.3.10.1



Por último, podemos comprobar que se trata de las políticas para un certificado cualificado de persona física.

Anexo 5: Detector de colisiones hash

Hemos creado en Python una aplicación que a partir de una función básica de hash definida por nosotros (hash = n mod 555) calcule cuantos intentos hacen falta para encontrar dos hashes iguales partiendo de números aleatorios distintos, lo cual se considera una colisión.

```
#TFG David de Dueñas Ledesma UOC 2022
import secrets #Librería que usaremos para generar un número aleatorio


dato = [] #Lista que contendrá cada dato origen
hash_dato = [] #Lista que contendrá el hash de cada dato origen
cont = 0 #Contador para detectar cuantos intentos han sido necesarios
para encontrar un hash duplicado (colisión)
dup1=0
dup2=0
hashdup=0

colision = False #Para iniciar el bucle partimos indicando que no hay
hashes iguales

while colision == False:
    cont+=1 #Por cada generaciónde hashes aumentamos el contador
    n = secrets.randbelow(999999) #Generamos un número aleatorio
    h = n % 555 #Aplicamos nuestra función hash
    if h in hash_dato: #Si el hash generado ya se encuentra entre los
generados
        dup1=dato[hash_dato.index(h)] #Obtenemos la posición dónde se
encuentra el hash ya generado para obtener el elemento sobre el que se
obtuvo
        dup2=n #Guardamos el nuevo número que ha generado el mismo
hash
        hashdup = h #Guardamos el hash duplicado en una variable
        colision = True #Indicamos que ya hemos detectado una colision
    else: #Si el hash generado no se encuentra entre los ya generados
        dato.append(n) #Por cada hash añadimos su dato origen en una
lista
        hash_dato.append(h) #Y el hash resultante en otra



print("Se ha encontrado un duplicado a los "+str(cont)+" intentos")
print("Número primero :"+str(dup1))
print("Número segundo :"+str(dup2))
print("Hash que ha generado la colisión :"+str(hashdup))
```

Se adjunta en el fichero “tfg_ddl_hash_detcol.py”

Anexo 6: Calculador velocidad algoritmos hash

Hemos creado una aplicación en Python que usando como dato origen una misma cadena de bytes de gran longitud calcula cuanto tiempo tarda cada algoritmo en realizar el hash.

```
#TFG David de Dueñas Ledesma UOC 2022
import hashlib
import time

cont_md5=0 #Iniciamos a 0 el contador de segundos para el tiempo que
tardará usando el algortimo MD5
cont_blacke2b=0 #Iniciamos a 0 el contador de segundos para el tiempo
que tardará usando el algortimo BLACKE2B
cont_black2s=0 #Iniciamos a 0 el contador  de segundos para el tiempo
que tardará usando el algortimo BLACKE2S
cont_sha1=0 #Iniciamos a 0 el contador de segundos para el tiempo que
tardará usando el algortimo SHA1
cont_sha256= 0#Iniciamos a 0 el contador de segundos para el tiempo
que tardará usando el algortimo SHA256
cont_sha512=0 #Iniciamos a 0 el contador de segundos para el tiempo
que tardará usando el algortimo SHA512

tiempo_actual = time.time() #Obtenemos la hora actual
while time.time() < tiempo_actual + 5: #Vamos generando hashes
continuamente hasta que pasen 5 segundos

hashlib.md5(b"PruebadeHash1234567890PruebadeHash1234567890PruebadeHash
1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234
567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash12345678
90PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890Pr
uebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890Prueba
deHash1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHa
sh1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash12
34567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash123456
7890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890
PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890Prue
badeHash1234567890PruebadeHash1234567890PruebadeHash1234567890Pruebade
Hash1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash
1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234
567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash12345678
90PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890Pr
uebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890Prueba
deHash1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHa
sh1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash12
34567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash123456
7890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890
PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890Prue
badeHash1234567890PruebadeHash1234567890PruebadeHash1234567890Pruebade
Hash1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash
1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234
567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash12345678
90PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890Pr
uebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890Prueba
deHash1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHa
sh1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash12
34567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash123456
7890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890
PruebadeHash1234567890").hexdigest()
        cont_md5+=1 #Sumamos 1 al contador de hahess generados
```

```
print("En MD5 se han generado "+str(cont_md5)+" hashes")

tiempo_actual = time.time()
while time.time() < tiempo_actual + 5:

hashlib.blake2b(b"PruebadeHash1234567890PruebadeHash1234567890Pruebade
Hash1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash
1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234
567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash12345678
90PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890Pr
uebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890Prueba
deHash1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHa
sh1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash12
34567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash123456
7890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890
PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890Prue
badeHash1234567890PruebadeHash1234567890PruebadeHash1234567890Pruebade
Hash1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash
1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234
567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash12345678
90PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890Pr
uebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890Prueba
deHash1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHa
sh1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash12
34567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash123456
7890PruebadeHash1234567890").hexdigest()
        cont_blacke2b+=1

print("En Blacke2b se han generado "+str(cont_blacke2b)+" hashes")

tiempo_actual = time.time()
while time.time() < tiempo_actual + 5:

hashlib.blake2s(b"PruebadeHash1234567890PruebadeHash1234567890Pruebade
Hash1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash
1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234
567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash12345678
90PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890Pr
uebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890Prueba
deHash1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHa
sh1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash12
34567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash123456
7890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890
PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890Prue
badeHash1234567890PruebadeHash1234567890PruebadeHash1234567890Pruebade
Hash1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash
1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234
567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash12345678
90PruebadeHash1234567890PruebadeHash1234567890Pr
```

```
uebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890Prueba
deHash1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHa
sh1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash12
34567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash123456
7890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890
PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890Prue
badeHash1234567890PruebadeHash1234567890PruebadeHash1234567890Pruebade
Hash1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash
1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234
567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash12345678
90PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890Pr
uebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890Prueba
deHash1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHa
sh1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash12
34567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash123456
7890PruebadeHash1234567890").hexdigest()
        cont_black2s+=1

print("En Blacke2s se han generado "+str(cont_black2s)+" hashes")

tiempo_actual = time.time()
while time.time() < tiempo_actual + 5:

hashlib.sha1(b"PruebadeHash1234567890PruebadeHash1234567890PruebadeHas
h1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash123
4567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567
890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890P
ruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890Prueb
adeHash1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeH
ash1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1
234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash12345
67890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash123456789
0PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890Pru
ebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890Pruebad
eHash1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHas
h1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash123
4567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567
890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890P
ruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890Prueb
adeHash1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeH
ash1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1
234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash12345
67890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash123456789
0PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890Pru
ebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890Pruebad
eHash1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHas
h1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash123
4567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567
890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890P
ruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890Prueb
adeHash1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeH
ash1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1
234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash12345
67890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash123456789
0PruebadeHash1234567890").hexdigest()
        cont_sha1+=1

print("En SHA1 se han generado "+str(cont_sha1)+" hashes")

tiempo_actual = time.time()
while time.time() < tiempo_actual + 5:
```

```
hashlib.sha256(b"PruebadeHash1234567890PruebadeHash1234567890PruebadeH
ash1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1
234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash12345
67890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash123456789
0PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890Pru
ebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890Pruebad
eHash1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHas
h1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash123
4567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567
890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890P
ruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890Prueb
adeHash1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeH
ash1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1
234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash12345
67890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash123456789
0PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890Pru
ebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890Pruebad
eHash1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHas
h1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash123
4567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567
890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890P
ruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890Prueb
adeHash1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeH
ash1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1
234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash12345
67890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash123456789
0PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890Pru
ebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890Pruebad
eHash1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHas
h1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash123
4567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567
890PruebadeHash1234567890").hexdigest()
        cont_sha256+=1

print("En SHA256 se han generado "+str(cont_sha256)+" hashes")

tiempo_actual = time.time()
while time.time() < tiempo_actual + 5:

hashlib.sha512(b"PruebadeHash1234567890PruebadeHash1234567890PruebadeH
ash1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1
234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash12345
67890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash123456789
0PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890Pru
ebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890Pruebad
eHash1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHas
h1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash123
4567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567
890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890P
ruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890Prueb
adeHash1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeH
ash1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1
234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash12345
67890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash123456789
0PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890Pru
ebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890Pruebad
eHash1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHas
h1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash123
4567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567
890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890P
```

```
ruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890Prueb
adeHash1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeH
ash1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1
234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash12345
67890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash123456789
0PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890Pru
ebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567890Pruebad
eHash1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHas
h1234567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash123
4567890PruebadeHash1234567890PruebadeHash1234567890PruebadeHash1234567
890PruebadeHash1234567890").hexdigest()
        cont_sha512+=1

print("En SHA512 se han generado "+str(cont_sha512)+" hashes")
```

Se adjunta en el fichero “tfg_ddl_hash_speed.py”

Anexo 7: Colisiones hash por longitud

Se ha creado una aplicación en Python que en función del nº de bits del hash calcula cuatas combinaciones distintas existan y cuando se estima la primera colisión.

```
#TFG David de Dueñas Ledesma UOC 2022
import math
import numpy

n_hash_bits = 32 #Variable que refleja el n° de bits del hash

pos_hash=pow(2,n_hash_bits) #Calculamos cuantos hash diferentes pueden
generarse con ese n° de bits

calc_prev=0.25+(2*(numpy.log(2))*pos_hash) #Cálculo auxiliar
prim_col=0.5+(math.sqrt(calc_prev)) #Calculamos cuando se producirá la
primera colisión

print("Existen "+str(pos_hash)+" hashes posibles con
"+str(n_hash_bits)+" bits")
print("La primera colusión se produciría tras "+str(prim_col)+"
valores")
```

Se adjunta en el fichero “tfg_ddl_hash_colisiones.py”

Anexo 8: Comprobador de Digest

Se ha generado en Python un programa que extrae las referencias de una firma XML y comprueba que su Digest sea correcto:

```
#TFG David de Dueñas Ledesma UOC 2022
import lxml.etree #Importamos libreria para el procesado del xml
import hashlib
import base64
import sys #Importamos la librería sys para poder parar la ejecución
del programa
from tkinter.filedialog import askopenfilename #Función para poder
seleccionar ficheros desde ventana

ruta_firma = askopenfilename() #Abrimos diálogo para obtener el
fichero de firma

tree = lxml.etree.ElementTree()
try:
    firma = tree.parse(ruta_firma) #Obtenemos la firma XML de la ruta
facilitada
except:
    sys.exit("El fichero no es un XML válido") #Si da error el
procesado, interpretamos que es un XML mal formado y paramos la
ejecución



refstotal = firma.xpath("//ds:Reference", namespaces={ #Obtenemos la
referencias incluidas en la firma
  'ds': 'http://www.w3.org/2000/09/xmldsig#',
  })


digests = [] #Lista que contendrá los digest de cada referencia
digmet = [] #Lista que contendrá los algoritmos usados para cada
digest
ref_uri = [] #Lista que contendrá las URI de cada referencia
refcont = 0 #Contador de referencias

for elem in refstotal: #Recorremos la referencias
    refcont+=1
    ref = str(lxml.etree.tostring(elem))
    inicio_dig = ref.find("<ds:DigestValue>")
    fin_dig = ref.find("</ds:DigestValue>")
    inicio_uri = ref.find('URI="#')
    uritemp = ref[inicio_uri:]
    fin_uri = uritemp.find(">")
    uri = '"'+uritemp[6:fin_uri]
    digest = ref[inicio_dig+16:fin_dig]
    digests.append(digest)
    ref_uri.append(uri)
    digmeth = elem.xpath("//ds:DigestMethod", namespaces={
  'ds': 'http://www.w3.org/2000/09/xmldsig#',
  })
    for dm in digmeth: #Recorremos para comprobar el algoritmo usado
        dmtext = str(lxml.etree.tostring(dm))
        algsha1 = dmtext.find("sha1")  #Buscamos si se usa el
algoritmo SHA1
        algsha256 = dmtext.find("sha256") #Buscamos si se usa el
```

```
algoritmo SHA256
        algsha512 = dmtext.find("sha512") #Buscamos si se usa el
algoritmo SHA512
        if algsha1 > -1:
            digmet.append("sha1")
            break
        elif algsha256> -1:
            digmet.append("sha256")
            break
        else:
            digmet.append("sha512")
            break



refs = [] #Lista que contendrá las referencias
refconta = 0
for ur in ref_uri: #Obtenemos el dato referenciado con la URI de cada
referencia
    xpathexp = "//*[@Id="+ref_uri[refconta]+"]"
    refs.append(firma.xpath(xpathexp))
    refconta+=1

calc_dig = []
cont = 0
for ref in refs:
    for at in ref:
        ref_canon = lxml.etree.tostring(at, method="c14n")
#Canonicalizamos
        if digmet[cont] == "sha1": #Calculamos hash con el algoritmo
que corresponda
            hash_ref_canon = hashlib.sha1(ref_canon).digest()
        if digmet[cont] == "sha256":
            hash_ref_canon = hashlib.sha256(ref_canon).digest()
        if digmet[cont] == "sha512":
            hash_ref_canon = hashlib.sha512(ref_canon).digest()
        b64_hash_ref_canon = base64.b64encode(hash_ref_canon)
#Codificamos en base64
        calc_dig.append(b64_hash_ref_canon) #Añadimos a la lista de
digests calculados
        cont += 1


valid = [] #Lista que contendrá la validación de cada referencia
print("Hay un total de "+str(refcont)+" referencias")
print()
for i in range(refcont):
    valid.append("Sin comprobar")

for i in range(refcont):
    a= digests[i]
    b = str(calc_dig[i])[2:-1]
    if a == b:
        valid[i] = "correcta"
    else:
        valid[i] = "incorrecta"


for i in range(refcont):
    print("La referencia "+str(i+1)+"("+str(ref_uri[i])+") es
"+valid[i]+". Su Digest es: "+str(calc_dig[i]))
```

Se adjunta en el fichero “tfg_ddl_digest.py”. y el fichero “FirmaEjemplo.xsig” para poder hacer una prueba de la aplicación.

Anexo 9: Generador / Validador RSA

Se ha creado en Python un programa que simula el comportamiento del algoritmo RSA, este programa realiza 3 funciones:

- Creación de las claves, pública y privada.
- Generación de una firma usando la clave privada
- Validación de la firma usando la clave pública

```
#TFG David de Dueñas Ledesma UOC 2022
import random
from math import gcd
from sympy.ntheory import randprime #Función que genera primos
aleatorios
from sympy import totient
import hashlib


def es_coprimo(x, y):
    return gcd(x, y) == 1

def gen_e(x): #Para selccionar e, buscamos un entero positivo que sea
comprimo con phi de n (eul_toi)
    prim_list = []
    for i in range(3, x):
        if (es_coprimo(i, x)):
            prim_list.append(i) #Creamos una lista de los candidatos
    return random.choice(prim_list) # Seleccionamos aleatoriamente uno
de los candidatos


#Generación de claves

p = randprime(100,1000) #Seleccionamos un primo aleatorio entre 100 y
1000
q = randprime(100,1000) #Seleccionamos un primo aleatorio entre 100 y
1000
#El rango usado se ha realizado por motivos de eficiencia al ser una
demostración
#para una generación real se recomienda que n al menos tenga 200
digitos por lo que p y q deberían ser mucho mayores

n = p*q #Calculamos n

eul_toi = totient(n)

e = gen_e(eul_toi) #Llamamos a la función para genrar e

d = pow(e, -1, eul_toi) #Generamos la clave privada, d

print("GENERACIÓN DE CLAVES")
print("--------------------")
print("El valor del primo p es "+str(p))
```

```
print("El valor del primo q es "+str(q))
print("El valor de n es "+str(n))
print("El valor de totiente es "+str(eul_toi))
print("El valor del exponente e es "+str(e))

print("La clave pública es n="+str(n)+", e="+str(e))
print("La clave privada es n="+str(n)+", d="+str(d))



#Creación de firma
print()
print("CREACIÓN DE FIRMA")
print("---------------------")
m = 12345 #Dato a firmar
h = hashlib.sha256(bytes(m)).hexdigest() #Calculamos el hash del dato
origen
h_dec = int(h, base=16) #Convertimos el hash en decimal
h_dec = h_dec%10000 #Acordamos el número
s = (h_dec**d)%n #Realizamos la firma sobre el hash usando la clave
privada
print("El mensaje origen m es "+str(m))
print("El hash de m es "+str(h_dec))
print("La firma del mensaje s es "+str(s))

#Validación de firma
print()
print("VALIDACIÓN DE FIRMA")
print("---------------------")
ms = s #Usamos la firma generada en el paso anterior para su
validación
h2=(ms**e)%n #Obtenemos el hash
print("El mensaje firmado s es "+str(ms))
print("El hash recuperado h2 es "+str(h2))
if h2 == h_dec: #Comparamos ambos hashes, si coinciden, la validación
es correcta
    print("El hash recuperado ("+str(h2)+") es igual al origen
("+str(h_dec)+"), la firma es válida")
else:
    print("El hash recuperado ("+str(h2)+") es distinto al origen
("+str(h_dec)+"), la firma es invalida")
```

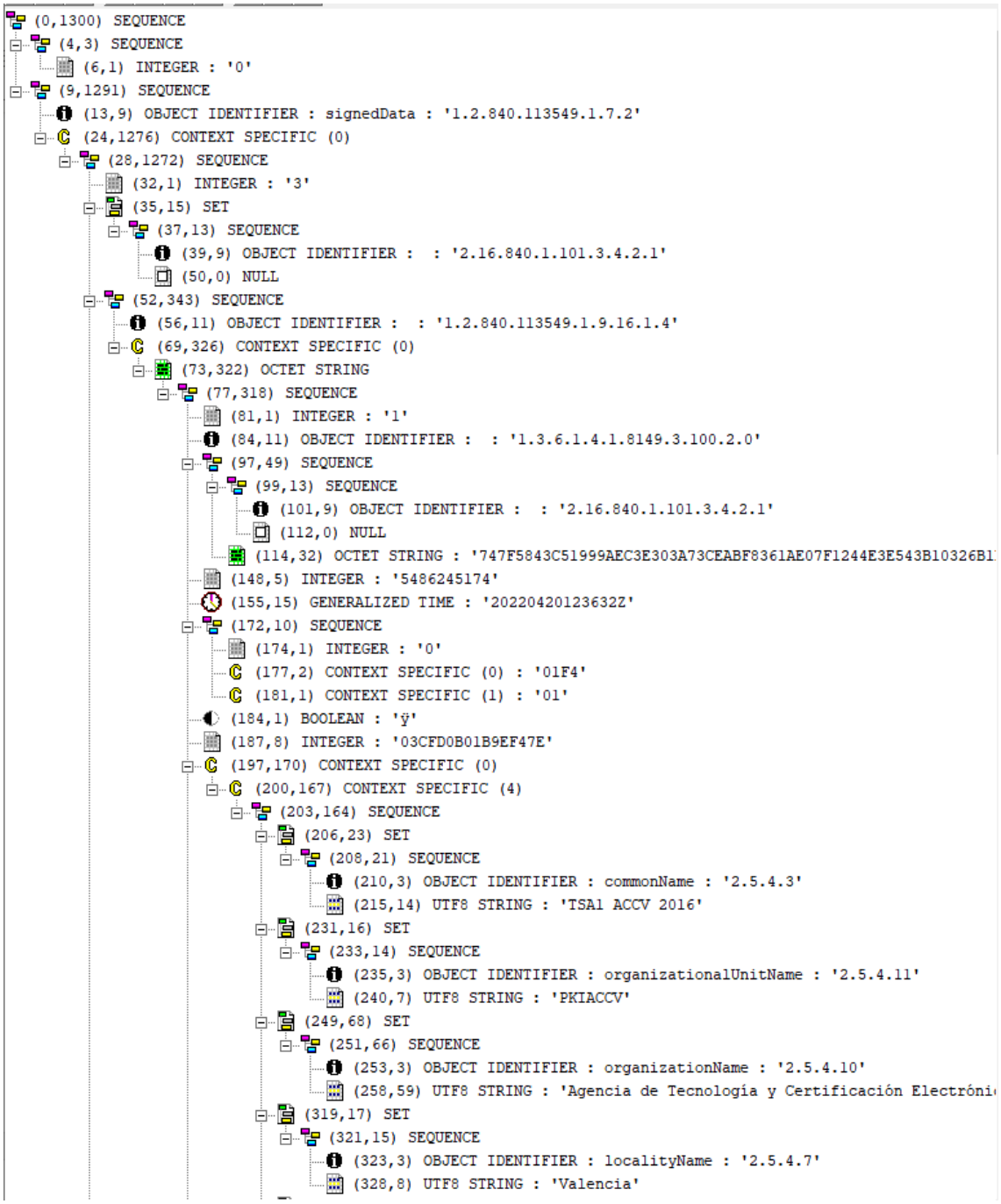
Es importante indicar las siguientes consideraciones:

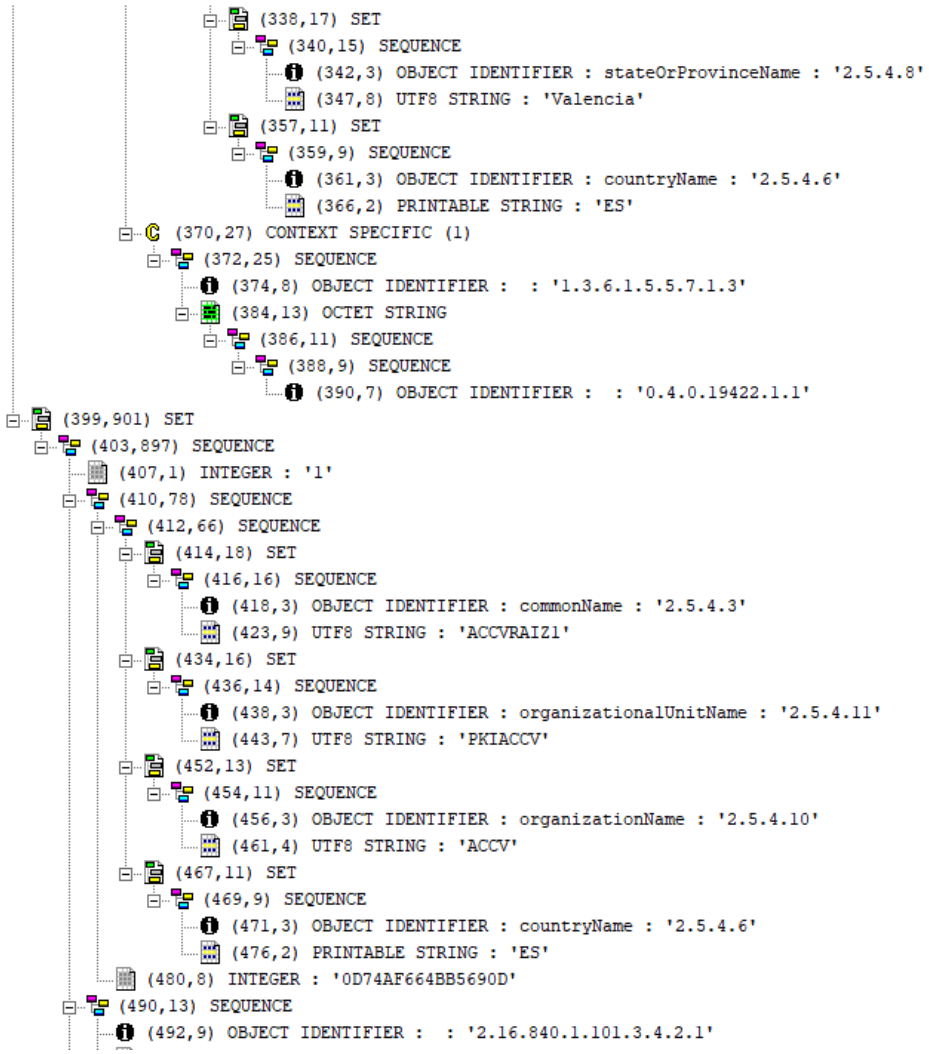
- Debido a la complejidad, se ha realizado el algoritmo para trabajar con datos de tamaño inferior y en decimal, el algoritmo real debería hacer las conversiones sobre el dato origen, que como podemos ver en la firma, no es sólo numérico, y por tanto no es decimal.
- Los primos usados como base se han limitado a un máximo de 1000, pero para una generación real se recomiendas números muy superiores.
- El hash utilizado en la conversión no es real, se ha acortado usando la operación módulo ya que en caso contrario el cálculo no podía realizarse al usar un *n* con poca longitud y no el esperado si se usasen primos de gran tamaño.

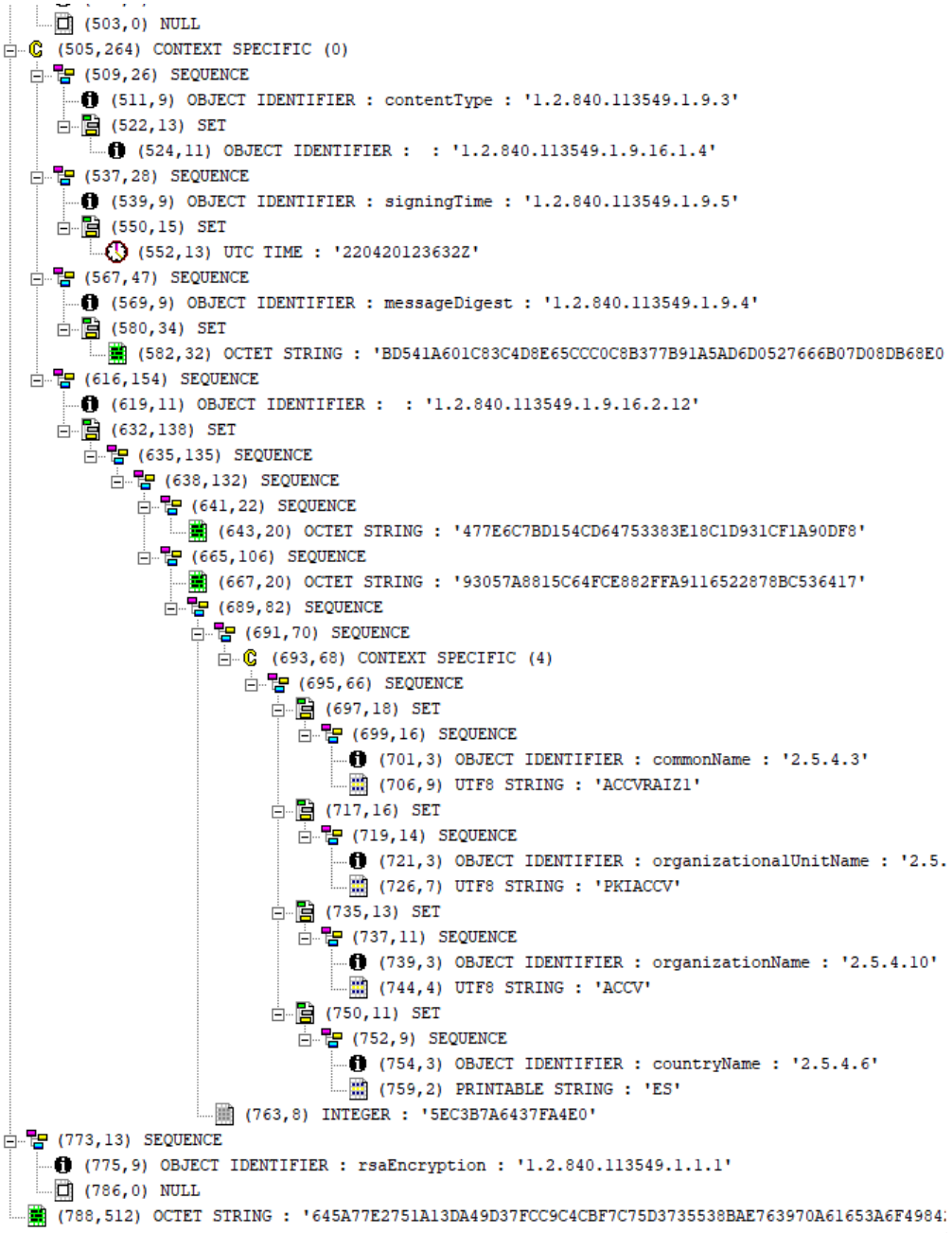
Se adjunta en el fichero “tfg_ddl_rsa.py”

Anexo 10: Respuesta TSA

A continuación, se muestra la respuesta completa TSA




(338,17) SET
(340,15) SEQUENCE
(342,3) OBJECT IDENTIFIER : stateOrProvinceName : '2.5.4.8'
(347,8) UTF8 STRING : 'Valencia'
(357,11) SET
(359,9) SEQUENCE
(361,3) OBJECT IDENTIFIER : countryName : '2.5.4.6'
(366,2) PRINTABLE STRING : 'ES'
(370,27) CONTEXT SPECIFIC (1)
(372,25) SEQUENCE
(374,8) OBJECT IDENTIFIER : : '1.3.6.1.5.5.7.1.3'
(384,13) OCTET STRING
(386,11) SEQUENCE
(388,9) SEQUENCE
(390,7) OBJECT IDENTIFIER : : '0.4.0.19422.1.1'
(399,901) SET
(403,897) SEQUENCE
(407,1) INTEGER : '1'
(410,78) SEQUENCE
(412,66) SEQUENCE
(414,18) SET
(416,16) SEQUENCE
(418,3) OBJECT IDENTIFIER : commonName : '2.5.4.3'
(423,9) UTF8 STRING : 'ACCVRAIZ1'
(434,16) SET
(436,14) SEQUENCE
(438,3) OBJECT IDENTIFIER : organizationalUnitName : '2.5.4.11'
(443,7) UTF8 STRING : 'PKIACCV'
(452,13) SET
(454,11) SEQUENCE
(456,3) OBJECT IDENTIFIER : organizationName : '2.5.4.10'
(461,4) UTF8 STRING : 'ACCV'
(467,11) SET
(469,9) SEQUENCE
(471,3) OBJECT IDENTIFIER : countryName : '2.5.4.6'
(476,2) PRINTABLE STRING : 'ES'
(480,8) INTEGER : '0D74AF664BB5690D'
(490,13) SEQUENCE
(492,9) OBJECT IDENTIFIER : : '2.16.840.1.101.3.4.2.1'


(503,0) NULL
(505,264) CONTEXT SPECIFIC (0)
(509,26) SEQUENCE
(511,9) OBJECT IDENTIFIER : contentType : '1.2.840.113549.1.9.3'
(522,13) SET
(524,11) OBJECT IDENTIFIER : : '1.2.840.113549.1.9.16.1.4'
(537,28) SEQUENCE
(539,9) OBJECT IDENTIFIER : signingTime : '1.2.840.113549.1.9.5'
(550,15) SET
(552,13) UTC TIME : '2204201236322'
(567,47) SEQUENCE
(569,9) OBJECT IDENTIFIER : messageDigest : '1.2.840.113549.1.9.4'
(580,34) SET
(582,32) OCTET STRING : 'BD541A601C83C4D8E65CCC0C8B377B91A5AD6D0527666B07D08DB68E0
(616,154) SEQUENCE
(619,11) OBJECT IDENTIFIER : : '1.2.840.113549.1.9.16.2.12'
(632,138) SET
(635,135) SEQUENCE
(638,132) SEQUENCE
(641,22) SEQUENCE
(643,20) OCTET STRING : '477E6C7BD154CD64753383E18C1D931CF1A90DF8'
(665,106) SEQUENCE
(667,20) OCTET STRING : '93057A8815C64FCE882FFA9116522878BC536417'
(689,82) SEQUENCE
(691,70) SEQUENCE
(693,68) CONTEXT SPECIFIC (4)
(695,66) SEQUENCE
(697,18) SET
(699,16) SEQUENCE
(701,3) OBJECT IDENTIFIER : commonName : '2.5.4.3'
(706,9) UTF8 STRING : 'ACCVRAIZ1'
(717,16) SET
(719,14) SEQUENCE
(721,3) OBJECT IDENTIFIER : organizationalUnitName : '2.5.
(726,7) UTF8 STRING : 'PKIACCV'
(735,13) SET
(737,11) SEQUENCE
(739,3) OBJECT IDENTIFIER : organizationName : '2.5.4.10'
(744,4) UTF8 STRING : 'ACCV'
(750,11) SET
(752,9) SEQUENCE
(754,3) OBJECT IDENTIFIER : countryName : '2.5.4.6'
(759,2) PRINTABLE STRING : 'ES'
(763,8) INTEGER : '5EC3B7A6437FA4E0'
(773,13) SEQUENCE
(775,9) OBJECT IDENTIFIER : rsaEncryption : '1.2.840.113549.1.1.1'
(786,0) NULL
(788,512) OCTET STRING : '645A77E2751A13DA49D37FCC9C4CBF7C75D3735538BAE763970A61653A6F4984

Anexo 11: Simulador Protocolo BB84

Se ha creado en Python un programa que simula el comportamiento del protocolo BB84, el cual realiza los pasos de este algoritmo hasta generar una clave compartida para dos usuarios:

```
#TFG David de Dueñas Ledesma UOC 2022
import random

l = 40 #Establecemos la longitud

usr1_bit = []
usr1_base = []
usr2_base = []
usr1_pol = []
usr2_pol = []
clave = []


for i in range(l): #Generamos bits, bases y polarizacion de usuario 1
    temp_bit = random.randint(0, 1)
    usr1_bit.append(temp_bit)
    temp_base = random.randint(0, 1)
    if temp_base == 0:
        usr1_base.append("+")
    else:
        usr1_base.append("x")
    if temp_bit == 1:
        if temp_base == 0:
            usr1_pol.append(90)
        else:
            usr1_pol.append(135)
    else:
        if temp_base == 0:
            usr1_pol.append(0)
        else:
            usr1_pol.append(45)

for i in range(l): #Generamos bases de usuario 2 y polarización en
función de usuario 1
    temp_base = random.randint(0, 1)
    if temp_base == 0:
        usr2_base.append("+")
    else:
        usr2_base.append("x")
    if usr2_base[i] == usr1_base[i]:
        usr2_pol.append(usr1_pol[i])
    else:
        if temp_base == 1:
            temp_pol = random.randint(0, 1)
            if temp_pol == 1:
                usr2_pol.append(45)
            else:
                usr2_pol.append(135)
        else:
            temp_pol = random.randint(0, 1)
            if temp_pol == 1:
                usr2_pol.append(0)
            else:
```

```
                usr2_pol.append(90)

#Comparamos bases
for i in range(l):
    if usr1_base[i] == usr2_base[i]:
        clave.append(usr1_bit[i])

print("Los bits generados por Usuario 1 son "+str(usr1_bit))
print("Las bases del Usuario 1 son "+str(usr1_base))
print("Las bases del Usuario 2 son "+str(usr2_base))
print("La polarización del Usuario 1 es :"+str(usr1_pol))
print("La polarización del Usuario 2 es :"+str(usr2_pol))
print("La clave es "+str(clave))
```

Se adjunta en el fichero “tfg_ddl_bb84.py”

Anexo 12: Ficheros

A continuación, se listan los ficheros adjuntos que complementan la memoria:

- Fichero “tfg_ddl_fuerza_bruta.py”: Fichero en formato Python que genera una contraseña aleatoria y calcula cuanto tiempo tardaría averiguarla.
- Fichero “tfg_ddl_web.zip”: Contiene la página web informativa creada para los ciudadanos.
- Fichero “tfg_ddl_digest.py”: Fichero en formato Python que comprueba los cálculos del Digest de una firma XML.
- Fichero “FirmaEjemplo.xsig”: Firma XML de ejemplo para poder probar la aplicación “tfg_ddl_digest.py”
- Fichero “tfg_ddl_hash_colisiones.py”: Fichero en formato Python que calcula cuando se producirá la colisión de hash en función de los bits.
- Fichero “tfg_ddl_hash_detcol.py”: Fichero en formato Python que calcula cuando se produce la primera colisión en un algoritmo concreto.
- Fichero “tfg_ddl_hash_speed.py”: Fichero en formato Python que comprueba la velocidad de diferentes algoritmos hash.
- Fichero “tfg_ddl_rsa.py”: Fichero en formato Python que simula el funcionamiento del algoritmo RSA (Generación claves, firma y validación)