Universidad Politécnica
de Madrid

**Escuela Técnica Superior de
Ingenieros Informáticos**

Grado en Ingeniería Informática

Trabajo Fin de Grado

Diseño de una Arquitectura de Codificación de Datos Patrimoniales Segurizada para un Escenario Post-Cuántico

Autor: Hugo de Juan Prieto
Tutor(a): Alberto Juan Sebastián Lombraña, Miguel Antonio Barbero Álvarez

Madrid, abril 2025

Tabla de contenidos

1 Introducción

Este trabajo se enfoca en la viabilidad del uso de criptografía para un escenario post-cuántico para la transmisión y almacenamiento seguro de datos patrimoniales.

1.1 Motivación

La información siempre ha sido un recurso fundamental y muy preciado. Sin embargo, con la creciente digitalización que han tenido todos los ámbitos de la vida en los últimos años, la información se ha situado como uno de los recursos más vitales y valiosos. Es por ello que tratarla y mantenerla almacenada de forma segura se ha convertido en una prioridad, no solo para evitar que se acceda a ella de forma no deseada, si no también para poder mantener su integridad y poder controlar quién accede a ella.

Los datos patrimoniales no son una excepción. Estos datos contienen información de vital importancia, que nos permite conservar la memoria cultural, histórica, etc. Por ello, a causa de la gran importancia que estos datos tienen, se ha generado un auge en la preservación de los datos patrimoniales mediante una digitalización masiva de los mismos.

La protección de la información, tanto en términos de confidencialidad como de integridad o autenticación, se consigue mediante el uso de métodos criptográficos, como los cifradores o las firmas digitales. Uno de estos métodos más conocidos y usados es RSA. Sin embargo, con la llegada de la computación cuántica, algoritmos como el de Shor podrían resolver el problema matemático que lo sustenta, comprometiendo la base de muchos sistemas de seguridad actuales.

Ante este escenario, denominado post-cuántico, es necesario desarrollar nuevas arquitecturas con nuevas formas de proteger la información, especialmente tratándose de datos patrimoniales, cuya preservación debe ser garantizada a largo plazo.

1.2 Alcance

En este trabajo de fin de grado se diseñará una arquitectura que permita la transmisión segura mediante el uso de criptografía resistente a la cuántica. Esta arquitectura estará orientada hacia el uso con datos de naturaleza patrimonial, que serán obtenidos de bases de datos patrimoniales públicas.

Además se realizará una evaluación de la misma en entornos controlados, y conforme a los formatos normalizados por la unión europea, siendo entornos como los policiales e institucionales de especial interés dada la naturaleza patrimonial de los datos.

Este trabajo se centra exclusivamente en el diseño e implementación de una arquitectura de transmisión segura de datos patrimoniales en un entorno controlado mediante el uso de criptografía post-cuántica.

Quedan fuera del alcance del proyecto cosas como la integración con infraestructuras reales, la evaluación exhaustiva del rendimiento de la arquitectura y el uso de certificados digitales oficiales.

1.3 Objetivos

El objetivo de este trabajo es el diseño de una arquitectura que realice una transmisión segura de datos patrimoniales para un escenario post-cuántico. Para ello se combinará tanto el uso de algoritmos clásico de criptografía con algoritmos post-cuánticos para mantener los datos seguros frente a ataques de ordenadores cuánticos.

Para ello los siguientes objetivos de este trabajo son:

- Estudio y comprensión de las actuales herramientas de codificación de datos patrimoniales.
- Identificación de riesgos de seguridad al manejar datos patrimoniales.
- Estudio y comprensión de las diferentes herramientas y técnicas de criptografía post-cuántica y su integración en bibliotecas modernas.
- Estudio de la criptografía cuántica y su aplicabilidad en el ámbito de protección de datos patrimoniales.
- Realización de una arquitectura adecuada para el escenario propuesto.
- Proponer el análisis del rendimiento de una posible implementación adecuada para la arquitectura.

1.4 Estructura

La realización del trabajo se divide en 3 fases, las cuales podemos observar en la figura 1:

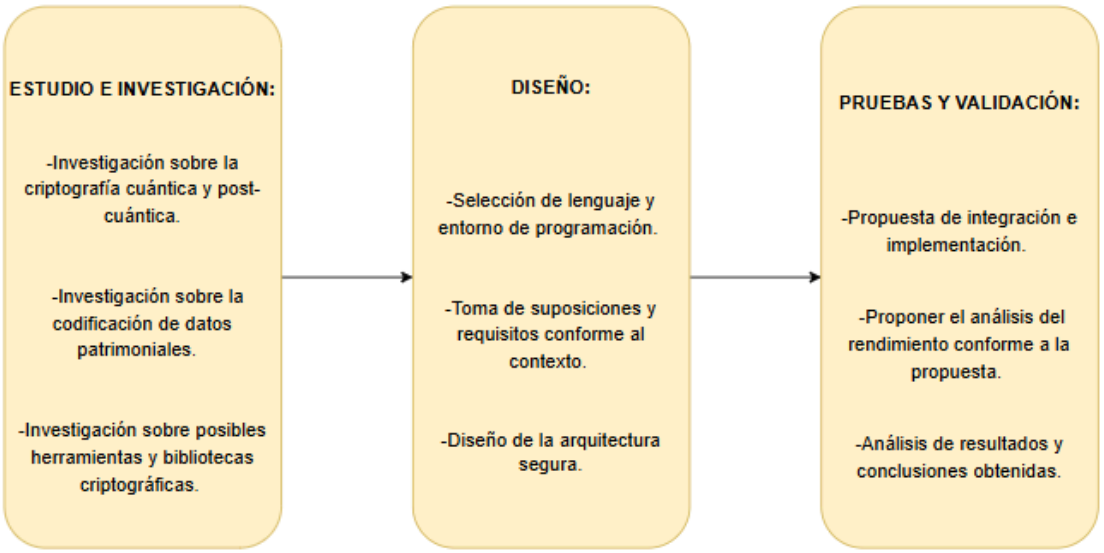

Figura 1: Fases del proyecto

Fase 1 (Estudio e investigación): en esta fase se realizan todas las tareas relacionadas con la investigación y estudio de los conceptos y herramientas que se consideran necesarias para el trabajo.

- Se investiga sobre la criptografía cuántica y la post-cuántica, entendiendo las diferencias entre ambas y como su aparición ha afectado a la criptografía tradicional.
- Se hace un análisis de la importancia de los datos patrimoniales y el creciente auge del interés por su preservación. También se investiga

sobre los diferentes riesgos y amenazas al transmitir y almacenar este tipo de datos.
- Se estudian las diferentes opciones de herramientas y bibliotecas criptográficas que pueden ser de utilidad para la realización del proyecto.

Fase 2 (Diseño): para el diseño de la arquitectura se opta por un enfoque por capas, donde cada nivel añade nuevas funcionalidades a la arquitectura.

Tras decidir el lenguaje de programación que se usaría, y hacer unas suposiciones y requisitos, se parte de una implementación básica con encriptación clásica, a la cual progresivamente se le incorporan funcionalidades hasta llegar a la arquitectura segura final.

Fase 3 (Pruebas y validación): una vez realizada la arquitectura, se usan entornos controlados para realizar diversas pruebas de funcionamiento comprobando que se realiza una transmisión segura e íntegra de la información.

2 Estado del arte

En este apartado se analizará el contexto teórico y tecnológico que fundamenta este trabajo. Se estudian tanto los avances en criptografía y computación cuántica como el tratamiento de los datos patrimoniales.

2.1 Trabajo relacionado con datos patrimoniales

La gestión digital del patrimonio ha evolucionado enormemente en los últimos años. Esta sección explora las iniciativas que han surgido para la conservación de estos datos, los distintos tipos de datos patrimoniales y como están tratados.

2.1.1 Patrimonio cultural digital

El término patrimonio cultural ha variado y evolucionado desde que empezó la digitalización de la sociedad, llegando a surgir un concepto nuevo denominado «patrimonio digital». Este concepto, según la UNESCO, abarca los recursos de conocimiento o expresión humana creados digitalmente o convertidos a digital a partir de un recurso analógico existente, que deben conservarse para generaciones futuras [1].

A nivel europeo, la digitalización del patrimonio ha sido reconocida como una prioridad: "*The momentum is now to preserve our cultural heritage and bring it to this digital decade*" [2]. Además, la Comisión Europea ha impulsado políticas e iniciativas como Europeana para facilitar la sostenibilidad y uso de este patrimonio digital.

De esta manera surge también el proyecto European Collaborative Cloud for Cultural Heritage (ECCCH), un proyecto de la Unión Europea cuyo objetivo es "ayudar a las instituciones de patrimonio cultural, organizaciones de investigación y otros profesionales de todos los tamaños y tipos a trabajar con sus objetos digitales de una manera más visible, interconectada, armonizada e informada, permitiéndoles afrontar con éxito los desafíos que la transición digital plantea al sector" [3].

2.1.2 Tipos de datos patrimoniales

Una de las características de los datos patrimoniales es la amplia variedad de tipo de datos que abarca. Esto si bien es algo positivo, pues muestra la riqueza cultural que existe, también supone un problema a la hora de clasificar y conservar este tipo de datos, ya que los conjuntos de datos de patrimonio cultural digital se caracterizan por características específicas [4].

Es por eso que uno de los aspectos más importante a día de hoy es ser capaz de documentar de una forma adecuada y eficiente los datos patrimoniales para su mejor entendimiento y uso en el futuro.

Con este objetivo en mente surgen iniciativas como la de Datasheets for Digital Cultural Heritage Datasets [4], la cual ha desarrollado una plantilla para su uso en el ámbito institucional y que incorpora información sobre la motivación y criterios de selección, la procedencia de los datos, etc.

Otros artículos como el redactado por el proyecto ENIGMA los divide en tangibles (museos, castillos, obras de arte, etc), intangibles (canciones, tradiciones, etc) y digitales [5]. Mientras que el Repositorio Digital Irlandés

(DRI) los clasifica centrándose en la infraestructura, describiendo los datos que gestionan como imágenes digitalizadas de obras de arte, documentos escaneados, modelos 3D, etc [6].

Como se observa, los datos patrimoniales son completamente heterogéneos, por ello su gestión requiere que su diversidad se tome en cuenta para garantizar su integridad y reutilización a largo plazo.

2.1.3 Tratamiento de datos patrimoniales en la actualidad

El tratamiento de datos patrimoniales en la actualidad ha sido un tema importante desde la digitalización masiva que ha habido en los últimos años.

Uno de los pilares fundamentales en este proceso es el uso de los metadatos, los cuales permiten describir, clasificar y contextualizar los recursos culturales. Según el enfoque propuesto por Gilliand [7], los metadatos son esenciales para describir el contenido de los datos así como para documentar su contexto y su estructura.

En Europa, ha habido un enriquecimiento de los metadatos patrimoniales gracias a plataformas como Europeana y su Europeana Data Model [8], el cual sigue un formato alineado con el siguiente:

```
<edm:ProvidedCHO rdf:about="http://example.org/cho/monalisa">
  <dc:title>La Mona Lisa</dc:title>
  <dc:creator>Leonardo da Vinci</dc:creator>
  <dc:date>1503-1506</dc:date>
  <dc:type>IMAGE</dc:type>
  <dc:language>it</dc:language>
  <dc:rights>Public Domain</dc:rights>
</edm:ProvidedCHO>
```

En conclusión, el tratamiento de los datos patrimoniales se encuentra en etapa de desarrollo que sigue avanzando gracias a la colaboración de los países.

2.2 Trabajo relacionado con criptografía y cuántica

La aparición de la computación cuántica ha supuesto un gran desafío para la protección de la información digital. Esta sección aborda desde la criptografía clásica hasta las nuevas técnicas post-cuánticas que se están desarrollando.

2.2.1 Criptografía clásica

La criptografía ha sido una herramienta fundamental e indispensable para mantener la información segura, cobrando más fuerza e importancia aún en entornos digitales. Existen dos grandes clasificaciones en la criptografía clásica: la asimétrica y la simétrica.

En la criptografía simétrica, un mismo secreto compartido se usa tanto para proteger como para acceder o validar la información. Algoritmos como el AES o los HMAC son usados con mucha frecuencia en la actualidad debido a la gran seguridad que proporcionan. En el caso de los cifrados, como el primer

ejemplo, el secreto compartido permite cifrar y descifrar la información mientras que en técnicas como los HMAC permite validar que la información permanece inalterada. La generalización de este último para dar integridad a certificados digitales de la identidad permite controlar quién accede a la información.

La criptografía asimétrica surge de la dificultad de distribuir el secreto compartido necesario para la criptografía simétrica, y se usa en la actualidad precisamente para hacer esa distribución. Algoritmos como RSA o Diffie Hellman son ejemplos de este tipo de criptografía. Ambos se fundamentan en problemas matemáticos los cuales se consideran poco resolubles por los ordenadores clásicos actuales. El algoritmo RSA se basa en la dificultad del problema de la factorización de números enteros grandes, mientras que el algoritmo Diffie-Hellman se basa en la dificultad del logaritmo discreto. Ambos algoritmos no son resolubles eficientemente pero la aparición de métodos matemáticos cuánticos como el algoritmo de Shor en 1994, el cual permite la factorización rápida de números enteros, demostró la existencia de algoritmos que son capaces de resolver estos problemas de forma eficiente [10].

2.2.2 Computación cuántica

La computación cuántica es un paradigma que hace uso de la mecánica cuántica, y que se basa en el uso de *qubits* en vez de bits ordinarios [11].

Si bien esto abre una enorme cantidad de nuevas posibilidades, el momento crucial surge en 1994, cuando Peter Shor publica un algoritmo el cual es capaz de "descomponer en factores un número N en tiempo $O((\log N)^3)$ y espacio $O(\log N)$" [12]. Factorizar grandes números nunca fue capaz de ser probado como difícil por lo que no se podía descartar la existencia de un algoritmo que fuera capaz de hacerlo eficazmente[10], y con la aparición del algoritmo de Shor esto empieza a ser posible y por tanto se compromete la seguridad de algoritmos como RSA.

Estos desarrollos y otros como la aparición del algoritmo de Grover, dejan clara la necesidad de desarrollar soluciones resistentes a la computación cuántica para seguir manteniendo la confidencialidad e integridad de la información.

2.2.3 Criptografía cuántica y post-cuántica

En este escenario «post-cuántico» en el cual la seguridad de los algoritmos clásicos ha sido amenazada por la aparición de la computación cuántica, surgen dos enfoques: la criptografía cuántica y la post-cuántica.

La criptografía cuántica se basa en el uso de los principios de la Mecánica Cuántica para garantizar la confidencialidad de la información [11]. El primer protocolo de distribución cuántica fue propuesto por Bennet y Brassard en 1984, conocido como BB84. Este protocolo hace uso de los qubits y el teorema de no clonación para asegurar que una comunicación no está siendo interceptada por un tercero [10]. Si bien la seguridad que proporcionan es incondicional, la criptografía cuántica no reemplaza completamente la criptografía clásica, sino que la complementa resolviendo problemas de distribución de claves [10]. Además requiere de infraestructuras avanzadas y sigue siendo una tecnología en desarrollo. Las claves distribuidas se pueden usar en las técnicas tradicionales como AES o los HMAC, degradando la

seguridad a la que proporcionan estas técnicas, como por ejemplo 256 bits para un AES256.

Por otro lado, la criptografía post-cuántica consiste en algoritmos resistentes a los ataques de ordenadores cuánticos. Estos pueden estar basados en código (CBC), hash (HBC) o retículos (LBC) entre otras cosas. El NIST (National Institute of Standards and Technology de los EE.UU.) ha tomado la iniciativa en el proceso de estandarización seleccionando algoritmos como Kyber o Dilithium como los elegidos para ir sustituyendo a los algoritmos clásicos como RSA [13]. En la actualidad no se sabe si estos algoritmos serán o no resistentes a los ordenadores cuánticos, pero son alternativas baratas y eficientes y se han aprobado como la técnica más adecuada para una transición hacia un mundo post-cuántico [14].

2.2.4 Herramientas y bibliotecas criptográficas

El desarrollo tecnológico y la creciente necesidad de mantener la información segura ante la aparición de nuevas amenazas como la computación cuántica, ha habido un crecimiento en el desarrollo de bibliotecas y herramientas que permiten afrontar este nuevo reto.

Una de las más destacadas es liboqs de Open Quantum Safe, la cual proporciona implementaciones en C de algoritmos de cifrado post-cuánticos y una versión de OpenSSL para poder realizar pruebas con protocolos como TLS [15].

Otra iniciativa es PQClean la cual recopila implementaciones seguras de los esquemas del proyecto post-cuántico del NIST. Estas implementaciones pretenden servir como base para otras bibliotecas más complejas como liboqs [16].

Estas herramientas son esenciales para que el mundo empiece una transición a un futuro escenario post-cuántico.

En cuanto a los servicios basados en la distribución cuántica de clave (QKD), están siendo integrados mediante interfaces normalizadas. Algún ejemplo de esto es la especificación ETSI GS QKD 014 desarrollada por el Instituto Europeo de Normas de Telecomunicaciones, o el SKIP (Standardized Key Interface Protocol) de Cisco.

3 Desarrollo

En este apartado se describe el proceso de diseño e implementación de la arquitectura segura para datos patrimoniales. Se detallan las decisiones de diseño tomadas durante el proceso, las herramientas empleadas y el funcionamiento de la arquitectura.

3.1 Descripción general de la arquitectura final

La arquitectura final desarrollada tiene como objetivo asegurar una comunicación segura de datos patrimoniales mediante el uso de criptografía post-cuántica, en concreto, mediante el encapsulamiento de claves ML-KEM y la firma digital post-cuántica con ML-DSA, estandarizados por el NIST.

En la figura 2 se presenta un esquema general del funcionamiento de esta arquitectura:

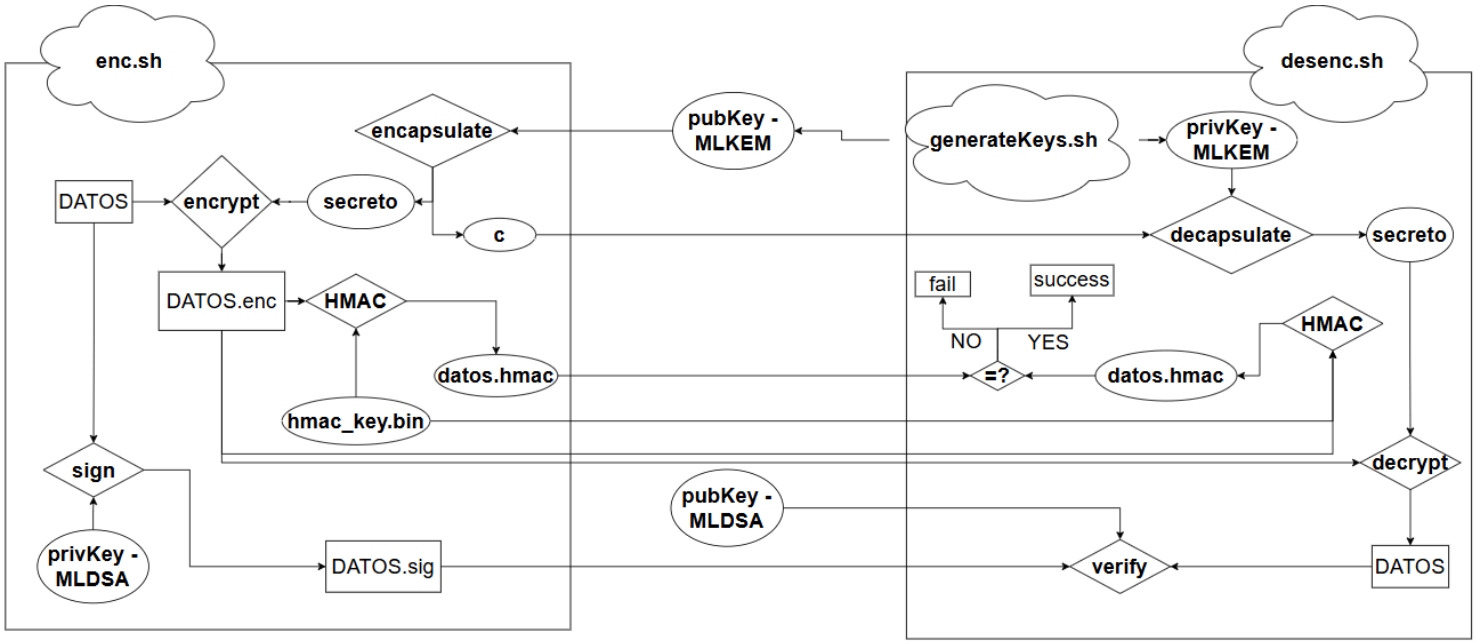

Figura 2: esquema de la arquitectura final. A la izquierda se ve la entidad emisora y a la derecha la entidad receptora de los datos. En la parte superior de cada entidad se ve el proceso de generación de claves ML-KEM, encapsulada en el emisor y desencapsulada en el receptor, obteniendose un secreto compartido. En la parte intermedia se ve el proceso de generación y comprobación del HMAC para la integridad de los datos. En la parte inferior se ve el proceso de firma y verificación de los datos mediante el algoritmo ML-DSA. Esta arquitectura se diseña agnóstica a los datos protegidos, pero en este trabajo se suponen datos patrimoniales.

La arquitectura se basa en el siguiente flujo de comunicación:

1) **Generación de claves:**
   La entidad B usará el script generateKeys.sh para generar un par de claves (pública y privada) utilizando el algoritmo ML-KEM. Una vez generadas enviará la clave pública a la entidad A.

A su vez se generarán también un par de claves ML-DSA en la entidad A que servirán para proporcionar una firma digital post-cuántica para autenticar los datos.

2) **Firma de datos:**
   La entidad A, antes de empezar con el encapsulamiento de claves y el cifrado de datos, usará la clave privada ML-DSA para firmar digitalmente los datos guardando el resultado en un archivo .sig.

3) **Encapsulamiento de claves y cifrado de datos:**
   Una vez que la entidad A ha recibido la clave pública realizará un encapsulamiento de clave. Este proceso generará una clave encapsulada *c* y un secreto compartido *K*.
   Este secreto *K* se usará como clave de un cifrado tradicional AES256 para cifrar los datos patrimoniales.

4) **Generación de HMAC:** una vez realizado el cifrado de datos, obtendremos, usando una clave de sesión aleatoria y los datos cifrados, un HMAC. Esto nos servirá para garantizar la integridad de los datos.

5) **Transmisión de datos:**
   Una vez se hayan cifrado los datos correctamente, la entidad A envía a B:
   - La clave encapsulada *c*.
   - Los datos cifrados.
   - El archivo de firma .sig.
   - La clave pública ML-DSA.
   - El HMAC generado y la clave de sesión utilizada.

6) **Verificación HMAC:**
   Una vez obtenidos todos los datos de la entidad A, la entidad B comprobará el HMAC. Para ello se volverá a generar un hmac usando los datos cifrados y la clave de sesión y se comparará con el generado en la entidad A. Si son iguales se confirma la integridad del mensaje.

7) **Desencapsulamiento y descifrado:**
   Verificada la integridad, la entidad B procederá a la desencapsulación de la clave *c* mediante el uso de la clave privada ML-KEM generada inicialmente. Esto dará como resultado el secreto *K*, consiguiendo así tener un secreto compartido entre ambas entidades.
   Una vez obtenido el secreto *K*, se pueden descifrar los datos recibidos, recuperando así la información patrimonial original.
   Gracias a este proceso se logra que solo la entidad B, poseedora de la clave privada, pueda desencapsular la clave y obtener el secreto compartido que se usa para el descifrado de los datos patrimoniales, garantizando así la confidencialidad incluso frente a amenazas cuánticas.

8) **Verificación firma digital:**
   Una vez la entidad B ha obtenido los datos patrimoniales descifrados, usará la clave pública ML-DSA y el archivo .sig para verificar la integridad y autenticación de los datos enviados.

Esta arquitectura permite establecer una comunicación segura entre dos entidades, garantizando confidencialidad, integridad y autenticación mediante la combinación de técnicas tradicionales y algoritmos post-cuánticos.

3.2 Herramientas utilizadas

Para llevar a cabo la implementación final, se han empleado múltiples herramientas y tecnologías. A continuación, se detallan los principales componentes utilizados:

1) **Máquinas Virtuales con Linux:**
   Todo el desarrollo se ha realizado sobre máquinas virtuales con el sistema operativo Linux, en concreto en Ubuntu 24.04.1.
   Esto ha permitido tener entornos controlados donde poder instalar versiones específicas de bibliotecas y herramientas, así como poder realizar pruebas sin afectar al *host*.

2) **Bash:**
   Para la realización de los scripts se decidió el uso de Bash, dada su integración nativa en el sistema y su simplicidad, que hacen que sea ideal para elaborar scripts que automatizan procesos como generar claves, o el cifrado y descifrado de datos.

3) **OpenSSL:**
   OpenSSL es una herramienta muy usada en el ámbito de la seguridad dada su amplia colección de funciones criptográficas, además de su fácil uso que te permite trabajar tanto desde línea de comandos como a través de bibliotecas.
   Se ha consolidado como una herramienta estándar de seguridad, además, gracias a que sigue en desarrollo, poco a poco se van incorporando nuevas tecnologías como la criptografía post-cuántica.

4) **Open Quantum Safe (OQS):**
   OQS proporciona bibliotecas de *software* de código abierto diseñadas para apoyar a la transición hacia la criptografía resistente a la computación cuántica [15].
   Una de sus dos líneas de trabajo consiste en la integración de prototipos en protocolos y aplicaciones, incluida la biblioteca OpenSSL.

5) **ML-KEM:**

En el FIPS 203 publicado por el NIST el 13 de agosto de 2024 [17] se define el estándar ML-KEM (Module-Lattice-Based Key-Encapsulation Mechanism).

Un KEM es un conjunto de algoritmos que se puede usar para establecer un secreto compartido entre dos entidades manteniendo la confidencialidad.

El ML-KEM sigue un esquema de KEM con 3 fases [18]:

- Se generan las claves pública y privada usando una matriz basada en retículos algebraicos y dos vectores pequeños secretos y de error.
- Se usa la clave pública para el encapsulamiento generando una clave encapsulada y un secreto compartido.
- Se usa la clave privada para la desencapsulación de la clave encapsulada obteniendo el secreto compartido.

  "La seguridad de los KEMs se cuantifica por su indistinguibilidad a ataques de texto cifrado elegido (IND-CCA)" [19], lo que significa que un atacante no puede distinguir entre la clave secreta real y una aleatoria [19]. En concreto, el ML-KEM se basa en el problema de MLWE, el cual se considera resistente frente a computadores cuánticos [18].

A esto se le suman otros elementos como la aleatoriedad generada por el uso de vectores de error y ruido que se generan durante la encapsulación.

6) **ML-DSA:**

El ML-DSA (Module Lattice Digital Signature Algorithm) es un algoritmo de firma digital post-cuántica basado en retículos modulares. Fue elegido y estandarizado por el NIST en el FIPS 204, publicado el 13 de agosto de 2024 [20].

El ML-DSA sigue un proceso con 3 fases [21]:

- Generación de claves pública y privada a partir de una semilla aleatoria de 32 *bytes* generada por un RBG.
- Se firma el mensaje usando la clave privada.
- El receptor puede verificar que una firma fue generada por la clave privada correspondiente, validando la integridad y autenticidad.

Al igual que ML-KEM, ML-DSA se basa en la dificultad computacional del problema Module-LWE, y está diseñado para ser fuertemente existencialmente infalsificable bajo ataque de mensajes elegidos (SUF-CMA), lo que garantiza que es computacionalmente inviable para un atacante falsificar firmas válidas, protegiendo la autenticidad e integridad de los mensajes firmados [21].

3.3 Formato de datos

a) **JSON:** durante la fase de desarrollo y pruebas de la arquitectura se utilizó el formato JSON debido a su formato simple, legible y a su amplia compatibilidad.
Este formato permitió validar el funcionamiento de la arquitectura diseñada y así poder realizar la transición al modelo EDM, descrito a continuación, para garantizar la compatibilidad con estándares patrimoniales.

b) **European Data Model (EDM):** para la correcta representación de datos patrimoniales, el estándar EDM es una de las principales propuestas a nivel europeo. Este fue desarrollado por el proyecto Europeana con el objetivo de integrar y conectar las descripciones proporcionadas por instituciones culturales como museos, bibliotecas, etc. de forma armonizada [22].
El EDM permite representar de forma estructurada objetos del patrimonio cultural y sus metadatos asociados (agentes, eventos, lugares).
El modelo define varias clases para representar distintas entidades relevantes:

CLASE	DEFINICIÓN	EJEMPLO
edm:Agent	Persona o institución relacionada con el objeto.	- Leonardo da Vinci - British Museum - W3C
edm:EuropeanaAggregation	Es el conjunto de recursos relacionados con un objeto de patrimonio cultural que representa colectivamente ese objeto de Europeana.	- Agregación de toda la información sobre la Gioconda (objeto cultural original, imagen digital, información de derechos, etc).
edm:EuropeanaObject	Cualquier objeto que sea resultado de las actividades de Europeana.	- Cualquier instancia de la clase EuropeanaAggregation - Una anotación creada por un usuario a través del portal de Europeana
edm:Event	Cambio de estados en sistemas culturales, sociales o físicos provocado por una serie de fenómenos físicos, culturales, tecnológicos o jurídicos coherentes.	- La 2ª Guerra Mundial - El acto de pintar la Mona Lisa
edm:InformationResource	Es un recurso cuyas características esenciales pueden	- Un texto de un libro - Un objeto digital

	ser transmitidas en un solo mensaje.	- Una partitura
edm:NonInformationResource	Todos los recursos que no sean recursos de información.	- Personas - Lugares
edm:PhysicalThing	Objetos de patrimonio cultural conocidos por Europeana como cosas físicas, así como todas las cosas físicas a las que Europeana hace referencia en las descripciones de objetos del patrimonio cultural.	- Mona Lisa - Piedra Rosetta - Venus de Praxiteles
edm:Place	Extensión en el espacio, en particular en la superficie de la tierra, en el sentido puro de la física: independiente de los fenómenos temporales y de la materia.	- Región de espacio ocupada por Roma en la actualidad - Región del espacio ocupada por la República de Crimea en 1945
edm:ProvidedCHO	Incluye los objetos de patrimonio cultural sobre los que Europeana recopila descripciones.	- Mona Lisa - Victoria alada de Samotracia
edm:TimeSpan	Extensiones temporales abstractas, en el sentido de la física Galileana, teniendo un inicio, un fin y una duración.	- 2021-12-31 - 1503 - 1506
edm:WebResource	Recursos de información que tienen al menos una representación web y al menos una URI.	- Un recurso web que contiene una descripción de la Mona Lisa

Tabla 1: Clases del lenguaje EDM

Además EDM, utiliza propiedades específicas para relacionar entre sí los distintos elementos. Algunas de las más relevantes incluyen:

PROPIEDAD	**DEFINICIÓN**	**EJEMPLO**
edm:aggregatedCHO	Asocia una agregación ORE con los objetos del patrimonio cultural de los que se trata.	- La agregación de la Mona Lisa
edm:dataProvider	El nombre o identificador de la organización que aporta datos indirectamente a un servicio de agregación como Europeana.	- El Instituto de Artes y Teatro de Praga
edm:provider	El nombre o identificador de la	- El proyecto Linked Heritage

	organización que aporta datos directamente a un servicio de agregación como Europeana.	
edm:hasView	Relaciona una agregación ORE sobre un objeto del patrimonio cultural con un recurso web que proporciona una vista de ese objeto.	- Una miniatura - Un resumen textual
edm:isShownAt	Una referencia URL inequívoca al objeto digital en el sitio web del proveedor en su contexto informativo completo.	- Una imagen de Henry Miller con otros datos.
edm:isShownBy	Una referencia URL inequívoca al objeto digital en el sitio web del proveedor en la mejor resolución disponible.	- Un link directo a una imágen
edm:language	Idioma asignado al recurso con referencia al Proveedor.	- El ISO 639 <eng> representa inglés
edm:rights	Información sobre el copyright y los derechos de uso y acceso de los objetos digitales de Europeana que representan el objeto de patrimonio cultural de origen descrito en los datos.	-https://creativecommons.org/publicdomain/mark/1.0
edm:type	El tipo de material Europeana del recurso.	- IMAGE - TEXT - SOUND

Tabla 2: Propiedades del lenguaje EDM

La arquitectura diseñada e implementada en este trabajo tiene como objetivo proteger datos con este formato. Sin embargo, por las herramientas usadas, no ha sido necesario hacer ningún tipo de adaptación.

3.4 Preparación del entorno

A continuación se describe el proceso seguido para preparar el entorno de desarrollo necesario para implementar y validar la arquitectura segura propuesta en este trabajo. Todas las instrucciones y configuraciones que se muestran a continuación están pensadas para ser ejecutadas sobre un sistema operativo Ubuntu 24.04.01 LTS, para garantizar la usabilidad de la solución.

El entorno requiere de tener instalado una versión de OpenSSL que pueda realizar las funciones de encapsulamiento y desencapsulamiento. Para ello se ha elegido la 3.5.0, que también tiene soporte para el proveedor OQSprovider.

Previamente a realizar cualquier acción, es necesario realizar un *sudo apt update* para preparar adecuadamente el sistema operativo. Esto asegura que el sistema cuente con las versiones más recientes de sus componentes,

evitando futuros errores y mejorando la compatibilidad con las herramientas que vamos a instalar.
Adicionalmente para evitar errores durante la preparación del entorno será necesario asegurarnos de que tenemos instaladas ciertas herramientas como git, gcc, make, cmake, etc. Para ello ejecutamos el comando: *sudo apt install -y build-essential git curl cmake ninja-build perl libssl-dev libtool autoconf automake*.
Una vez terminadas las preparaciones se procede a la instalación, la cual se realizará en un directorio aislado ~/pqc.
A continuación se descarga el repositorio git del oqsprovider [23].
Es importante usar el repositorio de git ya que proporcionará la versión más nueva. De este modo se evitan diversos errores que pueden producirse si se usan versiones antiguas.
Dado que no se va a utilizar la versión de OpenSSL que viene en el sistema de forma nativa (3.0.13), el OQSprovider proporciona un shell script "*fullbuild.sh*" que nos permite experimentar con una versión local de OpenSSL. Este paso es muy importante ya que actualizar la versión nativa puede acabar creando fallos en las dependencias y complica la instalación del OQSprovider.
Antes de realizar la ejecución del script (*./scripts/fullbuild.sh*) es necesario asignar, la versión de OpenSSL deseada a la variable de entorno OPENSSL_BRANCH, en este caso será openssl-3.5.0.
Terminada la ejecución del script comprobaremos que el archivo de biblioteca oqsprovider.so está en la ruta que le corresponde con el comando *ls -l .local/lib/ossl-modules*. Si esto no devuelve nada, significa que el archivo no está ahí y por tanto hay que copiarlo de la ruta donde esté (normalmente _build/lib/oqsprovider.so).
Para garantizar que el sistema utilice la versión de OpenSSL personalizada que se ha compilado, se tiene que modificar tres variables de entorno:

- "export PATH=~/pqc/oqs-provider/.local/bin:$PATH". Se añade al PATH el directorio donde se encuentra el binario de OpenSSL que se ha compilado localmente. Esto garantiza que cuando se use el comando openssl se ejecute la versión 3.5.0 y no la nativa del sistema.
- "export OPENSSL_MODULES= ~/pqc/oqs-provider/.local/lib/ossl-modules". Con esta línea se indica a OpenSSL dónde buscar sus módulos adicionales, como el oqsprovider.so.
- "export OPENSSL_CONF= ~/pqc/oqs-provider/.local/ssl/openssl.cnf". Esta línea le dice a OpenSSL qué archivo de configuración debe usar.

Para garantizar que el valor de estas variables de entorno se mantiene incluso cuando se reinicia o apaga la máquina, hay que añadir las tres líneas previamente dichas en el archivo ~/.profile. Tras añadirlas se ejecuta un *source .profile* para mantener los cambios del archivo.
Por último faltaría activar el OQSprovider, lo cuál se hace en el archivo openssl.cnf de la ruta definida en la variable de entorno OPENSSL_CONF. En este archivo hay que añadir "oqsprovider = oqs_sect" en la sección de provider_sect, descomentar el "activate = 1" de la de default_sect y debajo de

ella añadir una oqs_sect con el “activate = 1”. Todo esto viene explicado en el archivo USAGE.md del git oficial del OQSprovider [24].
Tras realizar todos los pasos hay que comprobar que todo se ha instalado de forma correcta. Esto lo haremos ejecutando un *openssl version* y *openssl list -providers*, siendo necesario obtener lo mostrado en la Imagen 1.

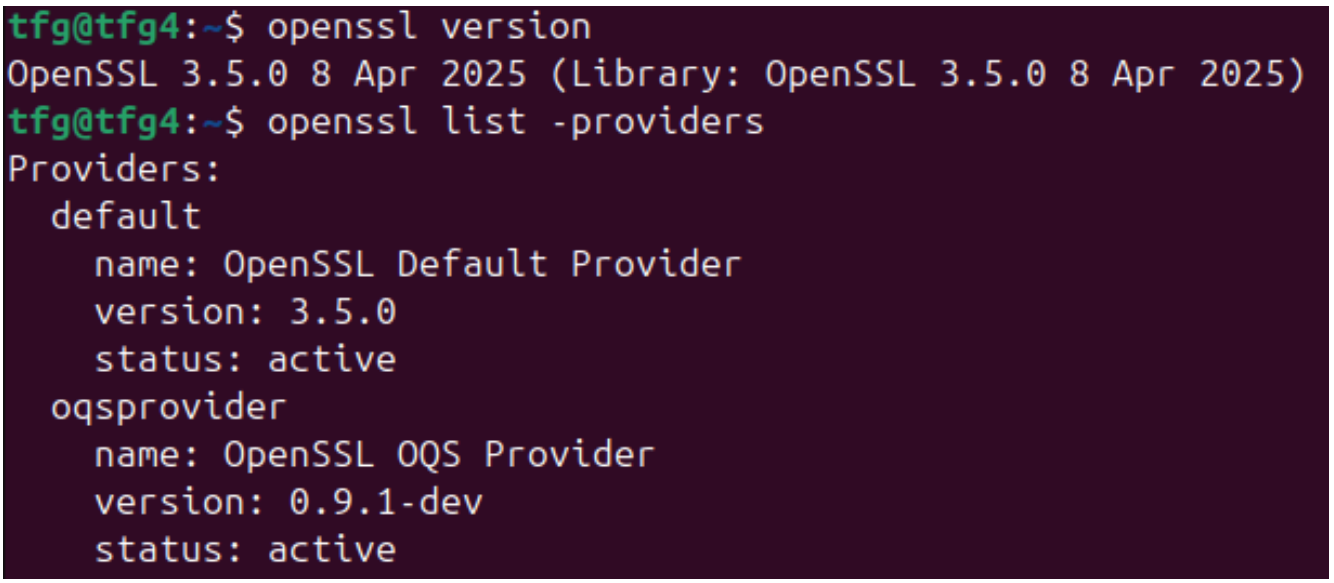

Imagen 1. Captura de la terminal con un resultado correcto de la instalación descrita.

Como se puede ver en la Imagen 1, la versión instalada de OpenSSL es la 3.5.0 y en la lista de proveedores están activados tanto el default como el oqsprovider. Con esto se puede confirmar que hemos finalizado la preparación del entorno.

3.5 Proceso de desarrollo

Para el desarrollo de la arquitectura propuesta se optó por un enfoque por capas, en el que partimos de una versión mínima y funcional y se le va añadiendo, progresivamente, nuevas capas de seguridad y funcionalidad.
Con este enfoque se consigue validar cada componente que se va añadiendo, lo que facilita la detección de errores, y favorece una evolución gradual del sistema.
Desde la primera fase, el desarrollo se realizó en una única máquina virtual y en un mismo directorio de trabajo. La configuración de una comunicación entre dos entidades separadas se dejó para el final del desarrollo, una vez comprobadas todos las funcionalidades de la arquitectura en el entorno local.

3.5.1 Enfoque inicial

En la primera fase del desarrollo, el objetivo fue establecer una base funcional sobre la que gradualmente ir construyendo la arquitectura final. Por ello, se optó por un esquema de cifrado clásico usando AES-256-CBC.
En este enfoque inicial se prescindió de criptografía post-cuántica con el fin de centrarse en validar un flujo básico de comunicación y cifrado simétrico. Se

genera manualmente una clave secreta compartida (*K*), la cuál se usaba tanto para cifrar como para descifrar los datos.
Este modelo sirvió como toma de contacto práctica con la herramienta OpenSSL, permitiendo familiarizarse con su sintaxis y funciones de cifrado.

3.5.2 Incorporación de HMAC

Antes de incorporar los algoritmos post-cuánticos, se añadió una nueva capa al sistema, con la intención de garantizar la integridad de los datos transmitidos. Concretamente se implementó un *hash-based message authentication code* (HMAC), utilizando el algoritmo SHA3-512. Con esta funcionalidad se asegura que los datos cifrados no fuesen modificados o manipulados durante la transmisión. La elección de una función resumen de longitud doble, 512 bits, garantiza que la seguridad global se mantiene al concatenarla con el cifrado de 256 bits, puesto que, por la paradoja del cumpleaños, la seguridad de las funciones hash es la mitad de su longitud.
El funcionamiento de esta etapa consiste en generar una clave de sesión aleatoria (KS) compartida, a partir de la cual se genera un código HMAC con el siguiente comando:
openssl dgst -sha3-512 -hmac hmac_key.bin -out datos.enc.hmac datos.enc
En el receptor, se realiza este comando de nuevo, generando nuevamente el HMAC y comparándolo con el recibido. Si ambos coinciden, se confirma que el contenido no ha sido alterado ni manipulado.

3.5.3 Encapsulamiento post-cuántico (ML-KEM)

Una vez validado el modelo básico con criptografía clásica, el siguiente paso fue la integración de un mecanismo post-cuántico de establecimiento de claves. Para este trabajo, se decidió el uso del esquema de encapsulamiento de claves estandarizado por el NIST en el FIPS 203 [17], el ML-KEM. Gracias a este mecanismo se permite que dos entidades compartan un secreto sin necesidad de acordar previamente una clave común, a través de un proceso seguro incluso frente a amenazas cuánticas.
El objetivo en esta fase era sustituir la clave compartida (*K*) generada de forma manual en el enfoque inicial por un proceso de encapsulamiento - desencapsulamiento mediante el uso de ML-KEM. Para lograrlo, se estudió la posibilidad de utilizar el proveedor OQSprovider junto con la versión nativa del sistema de OpenSSL. Sin embargo, esta versión no incluye el soporte completo para operaciones de encapsulamiento desde línea de comandos, ya que el comando *openssl pkeyutl* no dispone del flag *-encap*, necesario para encapsular claves ML-KEM.
Cabe señalar que OQSprovider sí proporciona scripts y ejemplos en C que permiten realizar las operaciones de encapsulamiento utilizando directamente las funciones de liboqs, pero dado que este proyecto se ha desarrollado completamente con scripts Bash, fue necesario que esta funcionalidad estuviera disponible a través de comandos estándar de OpenSSL. Es por esto que se optó por compilar la versión 3.5.0 de OpenSSL tal y como se describió en el capítulo 3.4.

Para el proceso de encapsulamiento primero se desarrolló un script "*generateKeys.sh"*, el cual se encarga de generar un par de claves ML-KEM768 con los comandos:
openssl genpkey -provider oqsprovider -algorithm mlkem768 -out privateKey.pem
openssl pkey -in privateKey.pem -pubout -out publicKey.pem
Tras tener el entorno listo con OQSprovider y OpenSSL 3.5.0, se procedió a realizar las primeras pruebas funcionales de encapsulamiento de claves. Esto se logra mediante el comando:
openssl pkeyutl -encap -pubin -inkey publicKey.pem -out encaps.bin -secret sharedkey.bin
Esto nos genera, a partir de la clave pública, una clave encapsulada "*encaps.bin*" y un secreto compartido. "*sharedkey.bin*", que usaremos para cifrar los datos de la misma manera que el enfoque inicial.
Como se ha comentado previamente, durante esta fase se detectaron y resolvieron problemas relacionados con la codificación de claves, errores de compatibilidad entre versiones, etc. Superados estos problemas, se consiguió tener un sistema capaz de realizar una distribución de clave segura resistente a ordenadores cuánticos, cumpliendo así uno de los objetivos principales del trabajo.

3.5.4 Firma digital post-cuántica (ML-DSA)

Habiendo completado la implementación del encapsulamiento de claves, se decidió añadir un mecanismo que garantizase tanto la autenticidad del emisor como la integridad del contenido, y que fuese resistente a amenazas cuánticas. Es por esto que se incorporó una firma digital basada en el algoritmo ML-DSA, estandarizado por el NIST en el FIPS 204 [20].
Este algoritmo permite a una entidad (A) firmar digitalmente un conjunto de datos y que otra entidad (B) pueda verificar que estos han sido generados por A y que no han sido modificados.
Mantener el HMAC una vez añadida la firma digital puede parecer redundante, puesto que la firma, además de aportar integridad, también aporta autenticación. Sin embargo, se decidió mantener su uso como capa adicional de protección. Emplear ambos mecanismos refuerza la arquitectura frente a diferentes tipos de ataques. Además de que la verificación de un HMAC es computacionalmente ligera por lo que no provocará lentitud a la hora de realizar las ejecuciones.
En cuanto a la implementación de la firma digital, al igual que con el encapsulamiento, se inicia con la generación del par de claves ML-DSA usando los comandos:
openssl genpkey -algorithm mldsa44 -out privDSA.pem
openssl pkey -in privDSA.pem -pubout -out pubDSA.pem
La diferencia respecto al encapsulamiento en el proceso de generación de claves, es que el encapsulamiento se realiza en la entidad receptora de los datos, mientras que en el caso de la firma se realiza en la entidad emisora.
Una vez generadas las claves se procede a firmar los datos:
openssl pkeyutl -sign -inkey privDSA.pem -in datos.json -out datos.json.sig

Este proceso genera un archivo .sig, el cual se envía a la entidad receptora que comprueba la firma:
openssl pkeyutl -verify -pubin -inkey pubDSA.pem -in datos_dec.json -sigfile datos.json.sig
Es importante recalcar que tanto la firma como la verificación se han de realizar sobre el contenido original, es decir, sobre contenido no cifrado. De este modo se garantiza la autenticidad del contenido original. Si firmamos los datos cifrados, no se garantiza que el mensaje original no haya sido modificado o alterado antes de cifrarse.
Es por ello que la firma se realiza antes del cifrado, y la verificación después del descifrado.

4 Pruebas y validación

En esta sección se presentan las pruebas funcionales y de rendimiento del sistema que validan la efectividad del diseño, incluyendo aspectos como la generación de claves, encapsulamiento de claves, firma digital y el cifrado.

4.1 Pruebas funcionales

Con este apartado vamos a realizar pruebas sobre cada una de las partes funcionales de la arquitectura para comprobar su correcto funcionamiento.

4.1.1 Generación de claves

Verificamos que el sistema genera correctamente un par de claves post-cuánticas utilizando ML-KEM768 como se muestra en la Imagen 2,

```
openssl genpkey -algorithm mlkem768 -out privateKey.pem
openssl pkey -in privateKey.pem -pubout -out publicKey.pem
```

Imagen 2

y también un par de claves utilizando ML-DSA44 a través de OpenSSL y oqsprovider como se muestra en la Imagen 3.

```
openssl genpkey -algorithm mldsa44 -out privDSA.pem
openssl pkey -in privDSA.pem -pubout -out pubDSA.pem
```

Imagen 3

El resultado son 2 pares de claves, un par KEM y un par DSA.
Se puede observar en la Imagen 4 que se han generado correctamente, tienen el formato correcto y que tienen los permisos correctos.

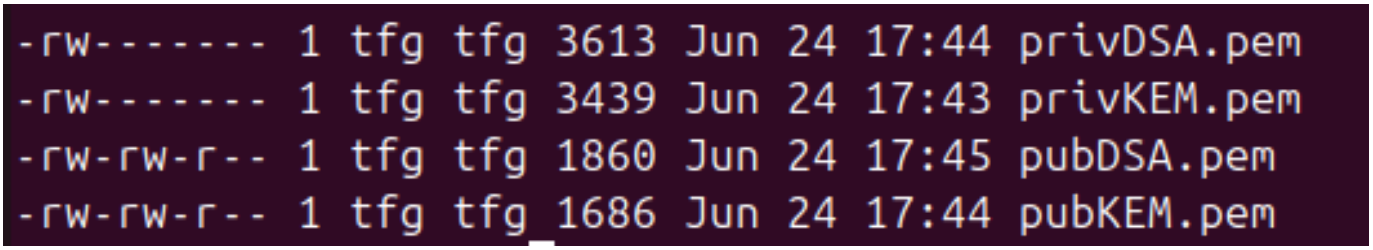

Imagen 4

Las claves tienen el formato PEM habitual en criptografía:
-----BEGIN PUBLIC KEY-----
MIIFMjALBglghkgBZQMEAxEDggUhAMXgXIJPvN9nYUn2TJMV405KBbgCIOgVZ
U7Z
LUUtbaqzGIk0fmWmiNeicDjUBJHdu0v97GQBUA3PeRxnr6OdHeZYn2ZVWpPiNO
fN
…
GA4bVfezTRxB1swmgQh2J5a0y0IpjJTeILyLctt+AeFeEPyxrxstOvZ+d8qzcZxE
f42ramdwkkVOm0uDq/f2xwuDE3QsyCpib5Mv4HlXV9nigIo/SBM=
-----END PUBLIC KEY-----

4.1.2 Encapsulado y desencapsulado de claves

Se verifica el proceso de encapsulamiento de una clave usando la clave pública ML-KEM:

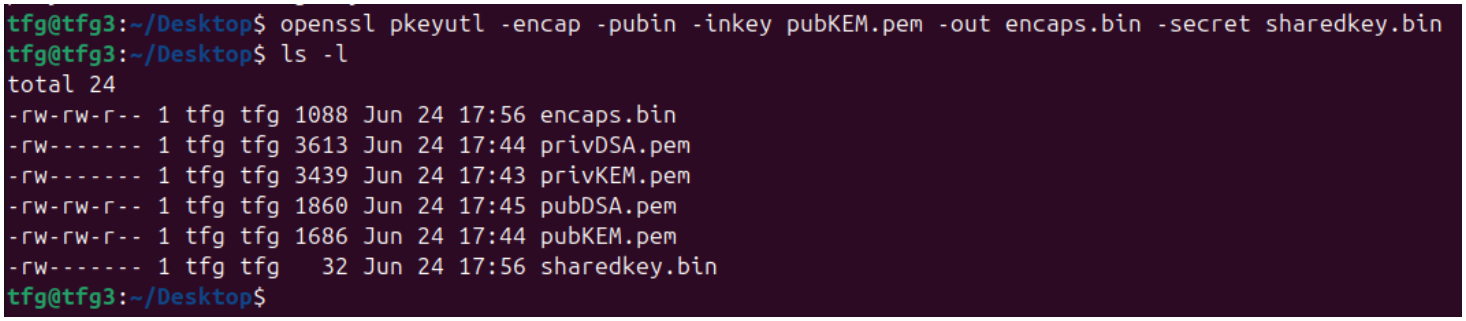

Imagen 5

En la Imagen 5 se ve como el comando de encapsulación genera el secreto compartido *K*, el cual se guarda en el binario *sharedkey.bin*, y la clave encapsulada en el archivo *encaps.bin*.
Ambos archivos tienen los tamaños especificados en el FIPS 203 [18], siendo estos, 32B para el secreto compartido y 1088B para la clave encapsulada.

Se verifica ahora el proceso de desencapsulamiento de la clave encapsulada en el archivo *encaps.bin* usando la clave privada ML-KEM.

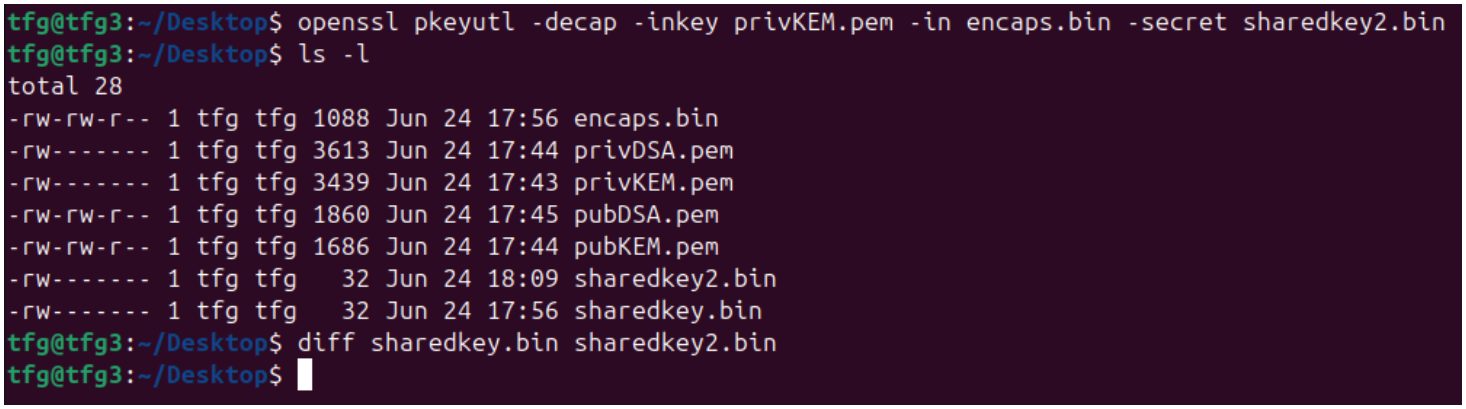

Imagen 6

En la Imagen 6 se puede ver como el comando de desencapsulación genera, usando la clave encapsulada *encaps.bin* y la clave privada ML-KEM, otro secreto compartido K de 32B en el archivo *sharedkey2.bin*.
Se hace un diff y comprobamos que ambos archivos son iguales por lo que se confirma que mediante este proceso se logra generar un secreto compartido seguro ante amenazas cuánticas.

4.1.3 Firma y verificación

Verificamos el proceso de firma de un archivo mediante el uso de una clave privada ML-DSA y un archivo json dummy *datos.json*:

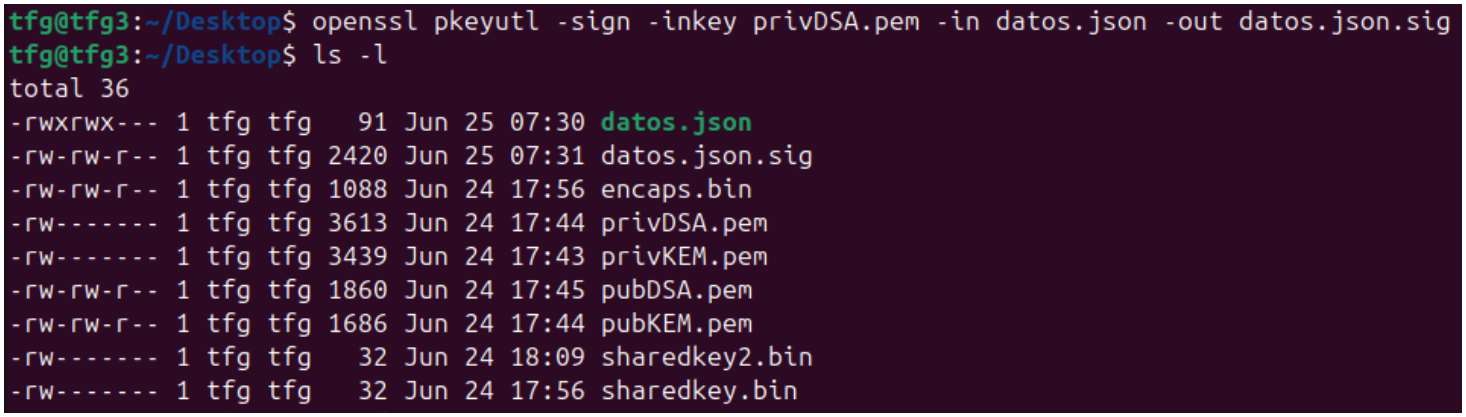

Imagen 7

Como se puede ver en la Imagen 7, se genera correctamente el archivo .sig, el cual contiene la firma digital. Además, también tiene el tamaño de 2420B, especificado en el FIPS 204 [21].

Se comprueba ahora el proceso de verificación de la firma digital mediante el uso de este archivo .sig y la clave pública ML-DSA.

```
tfg@tfg3:~/Desktop$ openssl pkeyutl -verify -pubin -inkey pubDSA.pem -in datos.json -sigfile datos.json.sig
Signature Verified Successfully
```

Imagen 8

En la Imagen 8 se observa como la verificación de la firma digital se realiza sin errores, lo que confirma que mediante este proceso se mantiene la integridad y la autenticidad de los datos, incluso frente a amenazas cuánticas.

4.1.4 Cifrado y descifrado de datos

Ahora, se verifica el proceso de cifrado y descifrado de datos del archivo *datos.json*, el cual contiene lo siguiente:

```
tfg@tfg3:~/Desktop$ cat datos.json
{
  "valor1": "hola",
  "valor2": "treasure",
  "valor3": "valor3",
  "valor4": "nuevo4"
}
```

Imagen 9

Para el proceso de cifrado y descifrado se va a utilizar el algoritmo aes-256.
Se puede observar en la Imagen 10 el contenido del archivo cifrado con éxito usando el secreto compartido generado en el proceso de encapsulamiento.

```
tfg@tfg3:~/Desktop$ openssl enc -aes-256-cbc -salt -pbkdf2 -in datos.json -out datos.enc -pass file:sharedkey.bin
tfg@tfg3:~/Desktop$ cat datos.enc
◆\K◆r◆pC◆◆◆◆TS◆◆◆#ዓ◆◆◆oK◆◆◆z◆0◆◆◆◆◆e◆q◆◆D◆F◆[◆#◆◆◆q◆4◆◆f◆◆
                                                    ◆L◆◆◆GGjmu◆◆◆^◆彼◆rO&;tfg@tfg3:~/Desktop$
```

Imagen 10

Se verifica el proceso de descifrado del archivo *datos.enc* usando el secreto compartido generado por el desencapsulamiento de clave y se comprueba que el contenido es igual al original:

```
tfg@tfg3:~/Desktop$ openssl enc -d -aes-256-cbc -salt -pbkdf2 -in datos.enc -out datos_dec.json -pass file:sharedkey2.bin
tfg@tfg3:~/Desktop$ cat datos_dec.json
{
  "valor1": "hola",
  "valor2": "treasure",
  "valor3": "valor3",
  "valor4": "nuevo4"
}
tfg@tfg3:~/Desktop$
```

Imagen 11

4.1.5 Integridad

Por último verificamos la funcionalidad del HMAC mediante el uso de unos scripts de prueba *hmac.sh y hmacVerify.sh.*

```
tfg@tfg3:~/Desktop$ cat hmac.sh
#!/bin/bash

# Generamos una clave HMAC
openssl rand 32 > hmac_key.bin

# generamos HMAC
openssl dgst -sha3-512 -hmac hmac_key.bin -out datos.enc.hmac datos.enc
tfg@tfg3:~/Desktop$ cat hmacVerify.sh
#!/bin/bash

hmac_desenc=$(openssl dgst -sha3-512 -hmac hmac_key.bin datos.enc | awk '{print $2}')
hmac_enc=$(awk '{print $2}' datos.enc.hmac)

if [ "$hmac_desenc" == "$hmac_enc" ]; then
    echo "HMAC verificado correctamente."
else
    echo "Error en la verificación del HMAC."
    exit 1
fi
```

Imagen 12

Se observa en la imagen 13 como el HMAC se verifica correctamente.

```
tfg@tfg3:~/Desktop$ ./hmac.sh
tfg@tfg3:~/Desktop$ ./hmacVerify.sh
HMAC verificado correctamente.
```

Imagen 13

4.2 Pruebas de rendimiento

Para evaluar el impacto que supone en el rendimiento sustituir un sistema clásico de establecimiento de claves por una versión post-cuántica, se ha realizado una comparación centrada en la fase del acuerdo de clave compartida.

Para el caso de un sistema de criptografía clásico se ha usado un Diffie-Hellman, el cual permite a dos entidades generar un secreto compartido de forma segura.

Todos los tiempos mostrados en la tabla son el promedio de diez ejecuciones consecutivas en un entorno local controlado, y se ha tomado el promedio de tiempo en milisegundos. Los resultados se muestran en la Tabla 3.

OPERACIÓN	CLÁSICA	POST-CUÁNTICA
Generación de claves	40 ms	20 ms
Derivación de la clave	20 ms	-
Encapsulamiento de clave	-	8 ms
Desencapsulamiento de clave	-	8 ms
Tiempo Total	60 ms	36 ms

Tabla 3: comparativa de tiempos de un proceso de intercambio de clave usando un Diffie-Hellman y la arquitectura desarrollada en este trabajo

Los resultados reflejan una leve diferencia de rendimiento, pero la solución post-cuántica es más rápida en este entorno concreto, lo cual refuerza que es viable, no solo en seguridad, sino también en rendimiento. Parte de esta diferencia se debe a la diferencia de tiempo en la generación de claves, ya que la solución post-cuántica solo tiene que generar un par de claves, mientras que la solución clásica tiene que generar dos pares de claves, provocando un aumento significativo del tiempo empleado en esta operación.
Cabe recalcar que no es el entorno de pruebas más controlado, pero sirve para asegurar que computacionalmente no es inviable el desarrollo de la arquitectura.
Además, para el caso de la solución clásica, se ha excluido del tiempo medido la generación inicial de parámetros DH, el cual es un proceso costoso que se realiza previamente, tardando unos 15 segundos.

4.3 Demostración práctica de la arquitectura

Para complementar las pruebas funcionales, se ha realizado una demostración práctica de la arquitectura completa simulando una comunicación entre dos entidades A y B que intercambiarán datos patrimoniales protegidos con criptografía post-cuántica.
Para realizar esta demostración se usarán dos máquinas virtuales configuradas tal y como se explicó en el apartado 3.4.
Para la comunicación y envío de datos se usará netcat por su simplicidad y disponibilidad en Linux. Esto nos permite centrar la prueba en la funcionalidad criptográfica de la arquitectura. No obstante, es importante señalar que netcat no proporciona características de seguridad por lo que su uso no sería adecuado en un sistema real.
En este caso, la entidad emisora cuenta con la IP 192.168.1.38 y la entidad receptora con la 192.168.1.37.

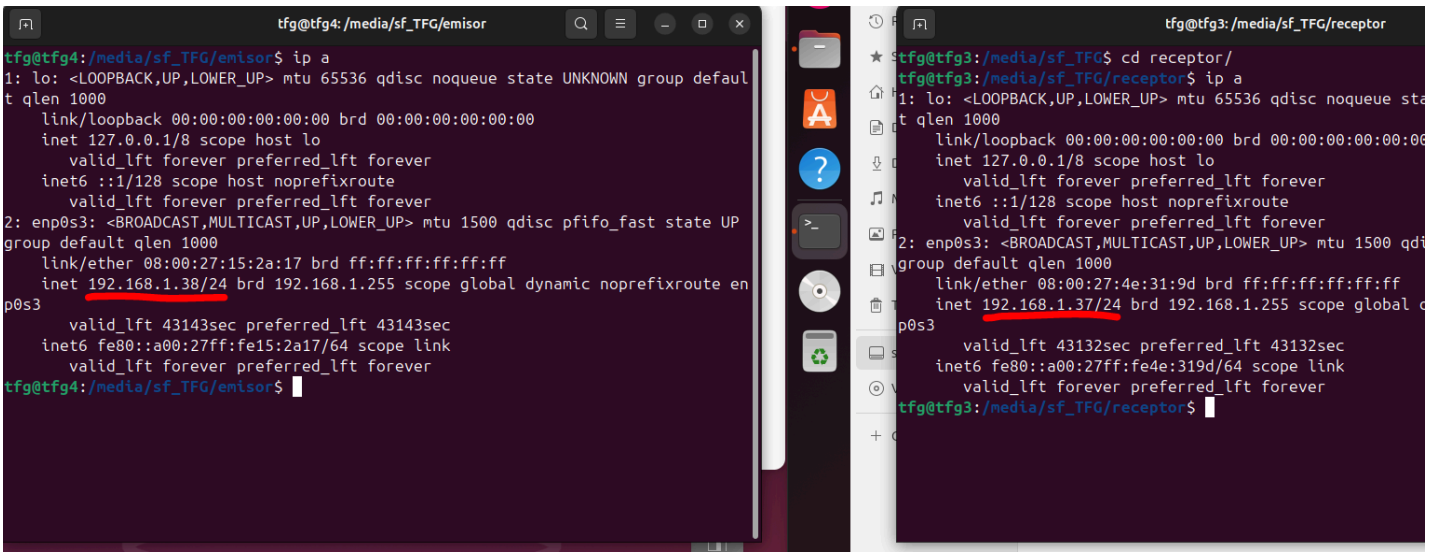

Imagen 14: se muestran las IPs de cada MV para realizar la conexión vía nc

Inicialmente el receptor generará las claves ML-KEM y enviará la pública al emisor.

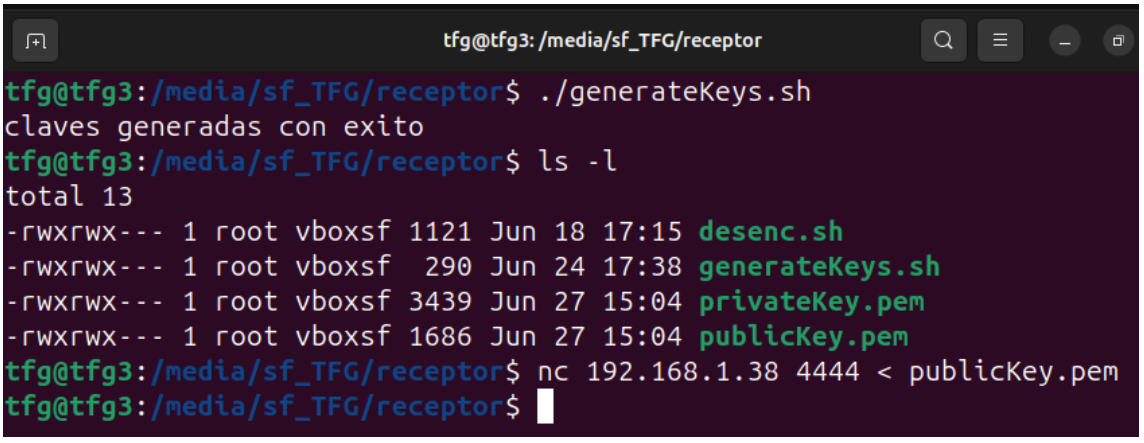

Imagen 15: el receptor genera el par de claves ML-KEM y envía la pública al emisor.

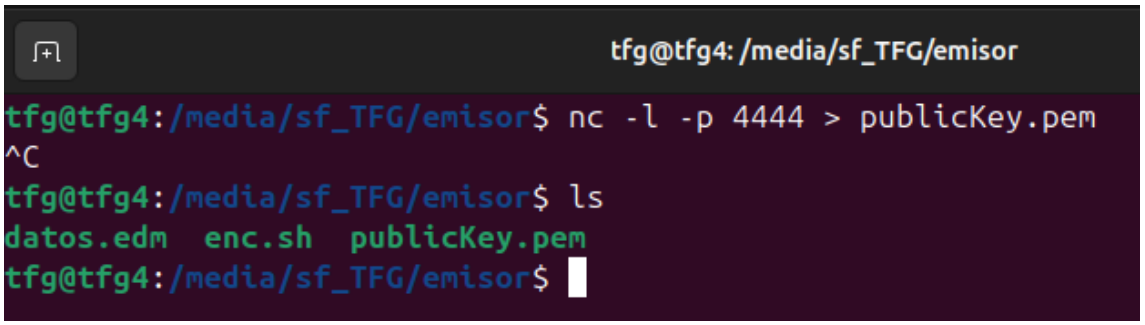

Imagen 16: el emisor recibe la clave pública ML-KEM

Una vez tiene la clave pública se va a ejecutar el script enc.sh que se va a encargar de generar las claves ML-DSA, encapsular la clave, firmar los datos, generar el HMAC y cifrar los datos. Una vez terminada su ejecución vamos a enviar al receptor todos los datos.

```
tfg@tfg4:/media/sf_TFG/emisor$ ./enc.sh
tfg@tfg4:/media/sf_TFG/emisor$ ls -l
total 34
-rwxrwx--- 1 root vboxsf 1908 Jun 27 14:41 datos.edm
-rwxrwx--- 1 root vboxsf 2420 Jun 27 15:12 datos.edm.sig
-rwxrwx--- 1 root vboxsf 1936 Jun 27 15:12 datos.enc
-rwxrwx--- 1 root vboxsf  155 Jun 27 15:12 datos.enc.hmac
-rwxrwx--- 1 root vboxsf 1088 Jun 27 15:12 encaps.bin
-rwxrwx--- 1 root vboxsf 1035 Jun 27 15:12 enc.sh
-rwxrwx--- 1 root vboxsf   32 Jun 27 15:12 hmac_key.bin
-rwxrwx--- 1 root vboxsf 3613 Jun 27 15:12 privDSA.pem
-rwxrwx--- 1 root vboxsf 1860 Jun 27 15:12 pubDSA.pem
-rwxrwx--- 1 root vboxsf 1686 Jun 27 15:09 publicKey.pem
-rwxrwx--- 1 root vboxsf   32 Jun 27 15:12 sharedkey.bin
tfg@tfg4:/media/sf_TFG/emisor$ nc 192.168.1.37 4444 < datos.enc
tfg@tfg4:/media/sf_TFG/emisor$ nc 192.168.1.37 4444 < datos.enc.hmac
tfg@tfg4:/media/sf_TFG/emisor$ nc 192.168.1.37 4444 < encaps.bin
tfg@tfg4:/media/sf_TFG/emisor$ nc 192.168.1.37 4444 < hmac_key.bin
tfg@tfg4:/media/sf_TFG/emisor$ nc 192.168.1.37 4444 < datos.edm.sig
tfg@tfg4:/media/sf_TFG/emisor$ nc 192.168.1.37 4444 < pubDSA.pem
```

Imagen 17: se realiza el proceso de encapsulación, firma y cifrado de datos para su posterior envío a la entidad receptora.

```
tfg@tfg3:/media/sf_TFG/receptor$ ls -l
total 30
-rwxrwx--- 1 root vboxsf 2420 Jun 27 15:16 datos.edm.sig
-rwxrwx--- 1 root vboxsf 1936 Jun 27 15:15 datos.enc
-rwxrwx--- 1 root vboxsf  155 Jun 27 15:15 datos.enc.hmac
-rwxrwx--- 1 root vboxsf 1121 Jun 18 17:15 desenc.sh
-rwxrwx--- 1 root vboxsf 1088 Jun 27 15:16 encaps.bin
-rwxrwx--- 1 root vboxsf  290 Jun 24 17:38 generateKeys.sh
-rwxrwx--- 1 root vboxsf   32 Jun 27 15:16 hmac_key.bin
-rwxrwx--- 1 root vboxsf 3439 Jun 27 15:04 privateKey.pem
-rwxrwx--- 1 root vboxsf 1860 Jun 27 15:17 pubDSA.pem
-rwxrwx--- 1 root vboxsf 1686 Jun 27 15:04 publicKey.pem
```

Imagen 18: la entidad receptora ha recibido todos los datos correctamente

Para finalizar, la entidad receptora ejecutará el script desenc.sh que se encargará de realizar la comprobación del HMAC, el desencapsulado de la clave, desencriptado de datos y por último la verificación de la firma.

```
tfg@tfg3:/media/sf_TFG/receptor$ ./desenc.sh
HMAC verificado correctamente.
Signature Verified Successfully
FIRMA VALIDA
tfg@tfg3:/media/sf_TFG/receptor$ cat datos_dec.edm
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:edm="http://www.europeana.eu/schemas/edm/"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:dcterms="http://purl.org/dc/terms/"
  xmlns:owl="http://www.w3.org/2002/07/owl#">

  <!-- Objeto patrimonial: La Gioconda -->
  <edm:ProvidedCHO rdf:about="http://museolouvre.fr/cho/gioconda">
    <dc:title>La Gioconda</dc:title>
    <dc:creator>Leonardo da Vinci</dc:creator>
    <dc:date>1503-1506</dc:date>
    <dc:language>it</dc:language>
    <dc:type>IMAGE</dc:type>
    <dc:rights>Public Domain</dc:rights>
    <dcterms:spatial rdf:resource="http://sws.geonames.org/2988507/"/>
  </edm:ProvidedCHO>
```

Imagen 19: ejecución del script desenc.sh y comprobación del resultado obtenido.

Como se puede observar, todas las operaciones se realizan correctamente y al comprobar el contenido del archivo desencriptado, este es correcto.
Con esto demostramos el correcto funcionamiento de la arquitectura diseñada y desarrollada durante este trabajo.

5 Conclusiones y futuras líneas de trabajo

Este trabajo ha abordado el diseño de una arquitectura segura para la transmisión de datos patrimoniales en un escenario post-cuántico y su posterior implementación. Se han logrado alcanzar los objetivos marcados en el plan de trabajo inicial demostrando que es posible integrar algoritmos criptográficos post-cuánticos en flujos de comunicación aplicables a un ámbito institucional.
El desarrollo ha requerido de un estudio tanto del contexto actual que tienen los datos patrimoniales como de los avances en criptografía con la aparición de las criptografías cuántica y post-cuántica. Gracias a esto se ha podido realizar el diseño de una arquitectura basada en estándares oficiales reconocidos como ML-KEM y ML-DSA, asegurando la confidencialidad, integridad y autenticidad de los datos frente a amenazas cuánticas.
Además, haber realizado la implementación mediante el uso de herramientas de código abierto como OpenSSL y OQSprovider facilita la replicación en futuros entornos reales.

Como futuras líneas de trabajo se puede plantear la integración en sistemas reales, realizar optimizaciones en el rendimiento, o la integración con infraestructuras cuánticas para explorar la posibilidad de combinar la arquitectura actual con sistemas de distribución cuántica de claves (QKD).
En definitiva, este trabajo demuestra que es posible comenzar hoy a construir sistemas resistentes frente a las amenazas cuánticas para lograr la preservación de nuestro patrimonio digital. La criptografía post-cuántica representa una oportunidad para protegernos de los riesgos que trae y que traerá en un futuro la computación cuántica. El patrimonio además de necesitar conservarse, debe poder transmitirse con la seguridad de que se mantendrá auténtico, íntegro y accesible para las generaciones futuras.

6 Análisis de Impacto

Este apartado evalúa el impacto de los resultados obtenidos desde distintas perspectivas.

6.1 Impacto Personal:

La realización de este trabajo ha tenido un gran impacto personal. Gracias a su realización se ha conseguido profundizar y aumentar los conocimientos en campos muy relevantes a día de hoy como la criptografía post-cuántica o la seguridad digital, siendo estos campos de especial interés personal. Además, se han reforzado distintas competencias como la capacidad de análisis o la integración de nuevas tecnologías que servirán para el desarrollo profesional.

6.2 Impacto Empresarial

Este trabajo tiene un gran potencial en el ámbito empresarial e institucional. Entidades como museos o archivos pueden beneficiarse de soluciones como la realizada en el trabajo para asegurar la confidencialidad, integridad y autenticación de los datos, así como para cumplir posibles normativas que sean necesarias en un futuro.

6.3 Impacto Social

En cuanto al impacto social, este trabajo ayuda a reforzar la confianza de la digitalización del patrimonio cultural. Además, al abordar una problemática como es la computación cuántica, se promueve una visión preventiva por parte de la ciberseguridad.

6.4 Impacto Económico

La transición hacia la criptografía resistente a la cuántica es algo inevitable en el futuro, y contar con trabajos como el realizado como base permite reducir costes en el futuro.

6.5 Impacto medioambiental

Aunque el trabajo no aborda directamente cuestiones ambientales, sí contribuye indirectamente. Al centrarse en la digitalización del patrimonio, se reduce la necesidad de transporte físico o tener que imprimir documentos, lo que tiene un efecto ecológico positivo.

6.6 Impacto cultural

El trabajo tiene como uno de sus pilares la preservación y protección del patrimonio cultural, por lo que el impacto en este ámbito es especialmente relevante. En un contexto donde gran parte del patrimonio está siendo digitalizado, es fundamental que se mantenga íntegro, accesible y auténtico a lo largo del tiempo.

El enfoque adoptado para el desarrollo de este trabajo permite a entidades como museos, archivos, bibliotecas y otras instituciones culturales mantener a salvo su legado y garantizar su transmisión a generaciones futuras.

7 Bibliografía

1. UNESCO, "Concept of Digital Heritage," *UNESCO Digital Heritage*, 2023. [Online]. Available: https://webarchive.unesco.org/web/20230616073538/https://en.unesco.org/themes/information-preservation/digital-heritage/concept-digital-heritage. [Accessed: 15-4-2025].
2. European Commission, "Cultural Heritage," *Digital Strategy*, 2023. [Online]. Available: https://digital-strategy.ec.europa.eu/en/policies/cultural-heritage. [Accessed: 15-4-2025].
3. European Commission, "Cultural Heritage Cloud," *Research and Innovation*, 2023. [Online]. Available: https://research-and-innovation.ec.europa.eu/research-area/social-sciences-and-humanities/cultural-heritage-and-cultural-and-creative-industries-ccis/cultural-heritage-cloud_en. [Accessed: 16-4-2025].
4. H. Alkemade, S. Claeyssens, G. Colavizza, N. Freire, J. Lehmann, C. Neudecker, G. Osti and D. van Strien, "Datasheets for Digital Cultural Heritage Datasets", *Journal of Open Humanities Data*, vol. 9, p. 17, 2023 [Online]. Available: https://openhumanitiesdata.metajnl.com/articles/10.5334/johd.124. [Accessed: 16-4-2025].
5. ENIGMA, "PersLab: Sharing Cultural Heritage Data," *ENIGMA EU*, Feb. 2024. [Online]. Available: https://eu-enigma.eu/2024/02/04/perslab-sharing-cultural-heritage-data/. [Accessed: 16-4-2025].
6. Digital Repository of Ireland, "Digital Cultural Heritage," *DRI*, 2023. [Online]. Available: https://dri.ie/digital-cultural-heritage/. [Accessed: 16-4-2025].
7. Getty Research Institute, "Setting the Stage," *Introduction to Metadata*, 2023. [Online]. Available: https://www.getty.edu/publications/intrometadata/setting-the-stage/. [Accessed: 16-4-2025].
8. Europeana, "EDM Documentation," *Europeana Pro*, 2023. [Online]. Available: https://pro.europeana.eu/page/edm-documentation. [Accessed: 16-4-2025].
9. Wikipedia,"Criptografía asimétrica," *Wikipedia*, 2023. [Online]. Available: https://es.wikipedia.org/wiki/Criptograf%C3%ADa_asim%C3%A9trica. [Accessed: Day-Month-Year].
10. N. Gisin, G. Ribordy, W. Tittel and H. Zbinden, "Quantum Cryptography," *arXiv*, 2024. [Online]. Available: https://arxiv.org/pdf/quant-ph/0101098. [Accessed: 17-4-2025].
11. Wikipedia,"Computación cuántica," *Wikipedia*, 2025. [Online]. Available: https://es.wikipedia.org/wiki/Computaci%C3%B3n_cu%C3%A1ntica. [Accessed: 17-4-2025].
12. Wikipedia,"Algoritmo de Shor," *Wikipedia*, 2025. [Online]. Available: https://es.wikipedia.org/wiki/Algoritmo_de_Shor. [Accessed: 18-4-2025].
13. Wikipedia,"Criptografía postcuántica", *Wikipedia*, 2025. [Online]. Available: https://es.wikipedia.org/wiki/Criptografía_postcuántica. [Accessed: 18-4-2025]
14. INCIBE, "Documento sobre ciberseguridad cuántica," *INCIBE*, 2023. [Online]. Available:

https://www.incibe.es/sites/default/files/eventos/Infoday%2025%20enero/NCC-ES%20INCIBE%20DOSIER%20.pdf?. [Accessed: 18-5-2025].
15. Open Quantum Safe, "Open Quantum Safe Project," *OQS*, 2023. [Online]. Available: https://openquantumsafe.org/. [Accessed: 26-5-2025].
16. PQClean, "PQClean GitHub Repository," *GitHub*, 2023. [Online]. Available: https://github.com/PQClean/PQClean. [Accessed: 18-4-2025].
17. NIST, "FIPS 203: Module-Lattice-Based Key-Encapsulation Mechanism Standard," *NIST*, 2023. [Online]. Available: https://csrc.nist.gov/pubs/fips/203/final. [Accessed: 26-5-2025].
18. NIST, "FIPS 203 (Full Text)," *NIST*, 2023. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.203.pdf. [Accessed: 22-6-2025].
19. Wikipedia,"Key Encapsulation Mechanism," *Wikipedia*, 2023. [Online]. Available: https://en.wikipedia.org/wiki/Key_encapsulation_mechanism. [Accessed: 30-5-2025].
20. NIST, "FIPS 204: Module-Lattice-Based Digital Signature Standard," *NIST*, 2023. [Online]. Available: https://csrc.nist.gov/pubs/fips/204/final. [Accessed: 31-5-2025].
21. NIST, "FIPS 204 (Full Text)," *NIST*, 2023. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.204.pdf. [Accessed: 31-5-2025].
22. Europeana, "EDM Definition v5.2.8," *Europeana Pro*, 2017. [Online]. Available: https://pro.europeana.eu/files/Europeana_Professional/Share_your_data/Technical_requirements/EDM_Documentation//EDM_Definition_v5.2.8_102017.pdf. [Accessed: 21-6-2025].
23. Open Quantum Safe, "OQS Provider GitHub," *GitHub*, 2023. [Online]. Available: https://github.com/open-quantum-safe/oqs-provider.git. [Accessed: 21-6-2025].
24. Open Quantum Safe, "OQS Provider Usage Guide," *GitHub*, 2023. [Online]. Available: https://github.com/open-quantum-safe/oqs-provider/blob/main/USAGE.md. [Accessed: 21-6-2025].

Este documento esta firmado por

	**Firmante**	CN=tfgm.fi.upm.es, OU=CCFI, O=ETS Ingenieros Informaticos - UPM, C=ES
	Fecha/Hora	Wed Jul 02 21:07:28 CEST 2025
	Emisor del Certificado	EMAILADDRESS=camanager@etsiinf.upm.es, CN=CA ETS Ingenieros Informaticos, O=ETS Ingenieros Informaticos - UPM, C=ES
	Numero de Serie	561
	Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)