ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

IMPLEMENTACIÓN DE SERVICIOS DE CÓMPUTO Y DE SEGURIDAD

IMPLEMENTACIÓN DE FIRMAS ELECTRÓNICAS MEDIANTE PYTHON CON BIBLIOTECAS LIBRES

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE TECNÓLOGO SUPERIOR EN REDES Y TELECOMUNICACIONES

RICHARD JAVIER MENA RAMOS

DIRECTOR: FERNANDO VINICIO BECERRA CAMACHO

DMQ, agosto 2024

CERTIFICACIONES

Yo, **RICHARD JAVIER MENA RAMOS** declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

RICHARD JAVIER MENA RAMOS

richard.mena@epn.edu.ec

richardmena2075@gmail.com

Certifico que el presente trabajo de integración curricular fue desarrollado por **RICHARD JAVIER MENA RAMOS**, bajo mi supervisión.

Ing. FERNANDO BECERRA.

DIRECTOR

fernando.becerrac@epn.edu.ec

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

RICHARD JAVIER MENA RAMOS

DEDICATORIA

A Dios.

Por ser mi guía y fortaleza en cada paso de mi vida.

A mi padre Giovanny.

Por sus sacrificios y valores que me han enseñado el significado de esfuerzo, trabajo y determinación.

A mi madre Adriana.

Por ser mi inspiración y apoyo constante, sus palabras de aliento siempre me han motivado a seguir adelante.

A mis hermanos Andrés y Danilo.

Por su motivación y apoyo, sus consejos y ánimos constantes siempre me han impulsado a perseguir mis sueños.

A una persona especial.

Por ser mi amor eterno y mi Luz en los momentos más oscuros, su presencia en mi vida ha sido un regalo invaluable.

A todos ustedes, mi familia.

Por ser mi mayor fuente de amor y fortaleza, por creer en mí en momentos en los que ni siquiera yo mismo lo hacía, Este logro también les pertenece.

AGRADECIMIENTO

Valoro profundamente esta etapa de mi vida y sé que siempre la recordaré con gran añoranza. Quiero expresar mi más sincero agradecimiento a todas las personas que fueron parte de este viaje lleno de enseñanzas y aprendizajes.

A mis seres queridos,

Quienes fueron el pilar fundamental en este camino. A mi familia, por su cariño y apoyo incondicional en los momentos más difíciles, siempre alentándome a seguir adelante. Mis padres y hermanos son mi fuerza y todo lo logrado no sería posible sin ellos. A mis verdaderos amigos, quienes demostraron su amistad con palabras de aliento sincero.

A esa persona especial que desde el inicio creyó en mí, mostrándome su cariño y confianza, y recordándome constantemente mi potencial para lograr grandes cosas.

Al Ing. Fernando Becerra,

Quien me brindó su apoyo incondicional, conocimientos y paciencia a lo largo de este trabajo y de toda la carrera, al igual que el resto de los docentes de la EPN

Agradezco a todos por su constante preocupación y respaldo en todo momento. Por último, pero no menos importante, a mí mismo, pues al final del día soy el reflejo de mis logros, y el culminar mi carrera es uno de esos reflejos.

ÍNDICE DE CONTENIDO

RESUMEN

El presente informe detalla el desarrollo de un sistema para la implementación de firmas electrónicas utilizando Python y bibliotecas de código abierto. Este componente se enfoca en la creación de certificados digitales para facilitar la firma digital de documentos en aplicaciones como Adobe Acrobat. Se aprovechan las capacidades avanzadas de las bibliotecas disponibles en la comunidad de software OpenSSL para asegurar la autenticación y validación de documentos en el entorno digital. El sistema no solo cumple con estándares rigurosos de seguridad y legalidad, sino que también se adapta fácilmente a diversos entornos profesionales y académicos.

El informe comienza con una descripción detallada del sistema desarrollado para implementar firmas electrónicas. Se explica cómo el sistema utiliza Python y bibliotecas de código abierto, especialmente OpenSSL y la biblioteca Cryptography, para generar claves criptográficas, crear certificados digitales y gestionar su exportación y renovación. El objetivo principal es proporcionar una solución segura, eficiente y fácil de integrar para la firma digital de documentos.

En la sección de marco teórico se revisan los fundamentos de la criptografía asimétrica y las firmas electrónicas. Se profundiza en el proceso de generación de claves públicas y privadas, así como en el papel de las autoridades de certificación (CA) en la emisión y validación de certificados digitales. Se destacan normativas y estándares internacionales relevantes para garantizar la seguridad y la legalidad de las firmas electrónicas.

En la sección de resultados se presenta la aplicación práctica del sistema desarrollado. Se incluyen ejemplos de documentos PDF firmados digitalmente utilizando los certificados digitales generados. Se muestran capturas de pantalla que ilustran las firmas aplicadas y se discuten los resultados de las pruebas de validación realizadas en Adobe Acrobat, enfatizando la autenticidad y la seguridad de las firmas digitales implementadas.

Las conclusiones resumen los principales hallazgos del estudio y subrayan la importancia de implementar firmas electrónicas seguras y efectivas en el entorno digital actual. Se revisan los objetivos alcanzados y se destaca la contribución del sistema desarrollado al cumplimiento de estándares de seguridad y legalidad en la firma digital de documentos.

Finalmente, se ofrecen recomendaciones para aquellos interesados en utilizar o expandir el sistema desarrollado. Se sugiere explorar nuevas aplicaciones y contextos para la implementación de firmas electrónicas, así como continuar mejorando la integración y la seguridad del sistema en diferentes plataformas y entornos de uso profesional y académico.

Este informe proporciona una visión integral y detallada del desarrollo e implementación de firmas electrónicas con Python y bibliotecas de código abierto, demostrando su utilidad y aplicabilidad en la era digital moderna.

PALABRAS CLAVE: firma electrónica, *cryptography*, certificados digitales, Python.

ABSTRACT

This report details the development of a system for the implementation of electronic signatures using Python and open-source libraries. This component focuses on the creation of digital certificates to facilitate the digital signing of documents in applications such as Adobe Acrobat. The advanced capabilities of libraries available in the OpenSSL software community are leveraged to ensure document authentication and validation in the digital environment. The system not only meets rigorous security and legal standards, but also easily adapts to various professional and academic environments.

The report begins with a detailed description of the system developed to implement electronic signatures. It explains how the system uses Python and open-source libraries, especially OpenSSL and the Cryptography library, to generate cryptographic keys, create digital certificates, and manage their export and renewal. The main objective is to provide a secure, efficient and easy to integrate solution for digital document signing.

In the theoretical framework section, the foundations of asymmetric cryptography and electronic signatures are reviewed. It delves into the process of generating public and private keys, as well as the role of certification authorities (CA) in the issuance and validation of digital certificates. Relevant international regulations and standards are highlighted to guarantee the security and legality of electronic signatures.

In the results section, the practical application of the developed system is presented. Examples of digitally signed PDF documents using the generated digital certificates are included. Screenshots illustrating the applied signatures are shown and the results of validation tests performed in Adobe Acrobat are discussed, emphasizing the authenticity and security of the implemented digital signatures.

The conclusions summarize the main findings of the study and highlight the importance of implementing secure and effective electronic signatures in today's digital environment. The objectives achieved are reviewed and the contribution of the developed system to compliance with security and legality standards in the digital signing of documents is highlighted.

Finally, recommendations are offered for those interested in using or expanding the developed system. It is suggested to explore new applications and contexts for the implementation of electronic signatures, as well as to continue improving the integration and security of the system in different platforms and environments for professional and academic use.

This report provides a comprehensive and detailed view of the development and implementation of electronic signatures with Python and open-source libraries, demonstrating their usefulness and applicability in the modern digital era.

KEYWORDS: *electronic signature, cryptography, digital certificates, Python.*

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

Se ha desarrollado un sistema con la finalidad de implementar firmas electrónicas mediante la creación de certificados digitales utilizando el lenguaje de programación Python y diversas bibliotecas de código abierto. Este componente se centra en proporcionar una solución segura, eficiente y fácil de integrar para la firma digital de documentos en aplicaciones como Adobe Acrobat, aprovechando las capacidades avanzadas de las bibliotecas libres disponibles en la comunidad de software *OpenSSL*.

Las firmas electrónicas son una herramienta esencial en la era digital, permitiendo la autenticación y validación de documentos sin necesidad de papel. La implementación desarrollada busca no solo cumplir con los estándares de seguridad y legalidad, sino también ofrecer una solución accesible y adaptable a distintos entornos.

Para este proyecto, se eligió la biblioteca *Cryptography* debido a su flexibilidad y amplio soporte en la comunidad. La biblioteca mencionada proporciona una amplia gama de funcionalidades criptográficas que son cruciales para la implementación de firmas electrónicas seguras.

El sistema desarrollado se estructura en varios módulos clave:

Generación de Claves: La biblioteca *Cryptography* implementa un módulo para la generación de pares de claves públicas y privadas. Este módulo permite la creación de claves RSA uno de los algoritmos más comunes para firmas digitales.

Creación de Certificados: Este módulo se encarga de la creación de certificados digitales utilizando las claves generadas. Incluye la creación de una solicitud de certificado digital y la emisión de este.

Exportación y Gestión de Certificados: Los certificados creados son exportados en formatos compatibles con aplicaciones como Adobe Acrobat. Este módulo también incluye funcionalidades para la gestión y renovación de certificados.

Adobe Acrobat es una herramienta ampliamente utilizada y reconocida para la gestión y firma de documentos PDF. Utilizar Adobe Acrobat para firmar documentos digitales tiene varias ventajas:

Compatibilidad y Estándar: Adobe Acrobat es compatible con el estándar PDF y es utilizado globalmente, lo que asegura que los documentos firmados sean accesibles y verificables en cualquier parte del mundo.

Seguridad y Confiabilidad: Adobe Acrobat implementa medidas de seguridad avanzadas para garantizar que los documentos firmados no sean alterados. Utiliza criptografía robusta y verifica las firmas digitales contra autoridades de certificación reconocidas.

Facilidad de Uso: Adobe Acrobat posee una interfaz de usuario intuitiva, lo que facilita la firma de documentos incluso para usuarios no técnicos. Además, permite la validación y visualización de las firmas digitales de manera clara y comprensible.

Cumplimiento Legal: Las firmas digitales realizadas con Adobe Acrobat cumplen con los requisitos legales en muchas jurisdicciones, asegurando que los documentos firmados tengan validez legal en procesos judiciales y administrativos.

Integridad del Documento: Utiliza técnicas criptográficas para asegurarse que el documento no haya sido alterado después de ser firmado. Si se detecta algún cambio, la firma digital es marcada como inválida.

Este componente no solo cumple con los objetivos específicos planteados, sino que también ofrece una herramienta práctica para su aplicación en diversos contextos profesionales y académicos.

1.1 Objetivo general

Desarrollar e implementar una solución integral de servicios de cómputo y seguridad en redes.

1.2 Objetivos específicos

1. Analizar soluciones a cada servicio de cómputo.
2. Diseñar la solución para cada servicio de cómputo mediante herramientas.
3. Implementar las soluciones mediante herramientas para el despliegue de los servicios de cómputo.
4. Verificar el funcionamiento de cada servicio de cómputo.

1.3 Alcance

El presente proyecto abarca el desarrollo e implementación de un sistema para la creación de certificados digitales utilizando Python y bibliotecas de código abierto, con el objetivo de

proporcionar una solución accesible, segura y eficiente para la autenticación y firma de documentos electrónicos. El alcance del proyecto se detalla en los siguientes puntos:

Generación de Certificados Digitales:

- Implementación de un módulo para la generación de pares de claves públicas y privadas utilizando el algoritmo RSA, asegurando la creación de claves seguras y robustas.
- Desarrollo de un módulo para la creación de certificados digitales autofirmados, cumpliendo con los estándares X.509.

Exportación de Certificados:

- Exportación de certificados digitales en formato PKCS#12, garantizando la compatibilidad con aplicaciones de firma de documentos, como Adobe Acrobat.
- Implementación de funcionalidades para la protección de certificados mediante contraseñas, asegurando la confidencialidad y seguridad de las claves privadas.

Interfaz de Usuario:

- Desarrollo de una interfaz de usuario que se encuentre basada en línea de comandos para la entrada de datos necesarios para la generación de certificados, incluyendo nombre completo, país, provincia, ciudad, organización y correo electrónico.

Integración con Adobe Acrobat:

- Explicación detallada de cómo los certificados generados pueden ser utilizados para firmar digitalmente documentos PDF en Adobe Acrobat.
- Descripción de los beneficios de utilizar Adobe Acrobat para la firma de documentos, destacando la compatibilidad, seguridad, facilidad de uso y cumplimiento legal.

Pruebas y Validación:

- Validación integral del sistema, verificando la interacción adecuada entre los diferentes componentes y la generación de certificados válidos y seguros.

1.4 Marco Teórico

Firmas Electrónicas

En la actualidad, las firmas electrónicas se han convertido en una herramienta esencial a nivel mundial, reemplazando a las firmas autógrafas que se realizaban físicamente en papel. Esta herramienta digital ha permitido realizar la autenticación de documentos mediante medios informáticos de manera fácil, por lo cual su adopción alrededor del mundo ha permitido formalizar acuerdos o negocios sin importar la índole. [1]

La firma electrónica dentro del contexto de seguridad informática corresponde a un grupo de datos electrónicos los cuales se vinculan lógicamente a un archivo digital, permitiendo garantizar la veracidad de la firma, así como la integridad del documento. Las firmas electrónicas son esenciales para la seguridad informática, puesto que proporcionan una manera de confirmar la identidad del firmante (autenticidad), asegurar que los documentos no han sido alterados (integridad) y prevenir la negación de la autoría (no repudio). [2]

Técnicamente, una firma electrónica está conformada por distintos componentes como: claves privadas, certificados digitales, contraseñas de acceso, algoritmos y protocolos de seguridad, con el propósito de asegurar la protección de datos y también la información personal del usuario.

Certificados Digitales

Un certificado digital es un archivo el cual es utilizado por un software para lograr firmar digitalmente un documento, la autenticación de certificados digitales garantiza a las empresas que solo los usuarios de confianza puedan acceder a su red.

Los certificados digitales son emitidos por autoridades de certificación. [3]

Poseen información como:

- Nombre del usuario
- Correo electrónico del usuario
- Organización a la que pertenece el usuario
- Clave pública
- Información de la autoridad certificadora
- Fecha de expiración del certificado

Se emplean para asegurar la comunicación y autenticar identidades en el entorno digital.

Diferencias entre firma electrónica y certificado digital

La firma electrónica es utilizada para firmar documentos electrónicos, sirve para dar la aprobación del firmante a un documento de manera online, mientras que el certificado digital es utilizado para autenticar la identidad del propietario y es emitido por una autoridad certificadora. [4]

En conclusión, la firma electrónica es un mecanismo utilizado para firmar digitalmente, mientras que el certificado digital garantiza la autenticidad de la firma y la identidad del usuario que firma.

Ventajas de la firma electrónica

La firma electrónica es un recurso de gran valor que ofrece múltiples beneficios tanto a organizaciones como a personas naturales, la firma electrónica cuenta con las siguientes ventajas:

- Ahorro: Al eliminar la necesidad de firmas tradicionales (físicas), se puede firmar documentos de manera más rápida ahorrando recursos y tiempo.
- Eficiencia de gestión: Reducción y eliminación de tareas administrativas asociadas con la gestión de documentos, como la impresión y escaneo de documentos.
- Seguridad: Al utilizar algoritmos criptográficos es más difícil de falsificar en comparación con una firma manuscrita.
- Accesibilidad: Permite realizar firmas desde cualquier sitio con acceso a internet, facilitando la colaboración remota.
- Auditoría: La firma electrónica permite un seguimiento de quién firmo junto con la hora exacta, mejorando el control de los documentos firmados.
- Errores humanos: La firma electrónica minimiza errores tales como: firmas incompletas o documentos en mal estado, contribuyendo a la fiabilidad de los documentos. [5]

Certificados X.509

Es un estándar definido por la UIT-T, el cual es usado para certificados de clave publica y son emitidos por autoridades certificadoras, utilizados para brindar seguridad informática y garantizar la autenticidad y confidencialidad de la información transmitida mediante redes públicas como el “Internet”. [6]

Este tipo de certificados incluyen distintos campos, entre ellos se encuentra:

- Versión: Muestra la versión del certificado.

- Serie: Muestra el número único asignado por la CA (Autoridad de certificación).
- Algoritmo: El algoritmo utilizado para la creación del certificado.
- Emisor: El nombre de la CA que emite el certificado.
- Validez: El tiempo de validez del certificado.

La implementación de dicho certificado mediante Python permite a los programadores elaborar soluciones seguras y fiables para la firma electrónica.

Entidades Certificadoras

Las autoridades certificadoras se encargan de emitir los certificados digitales a personas naturales o jurídicas, demostrando su autenticidad en el entorno digital; dichas entidades son consideradas “guardianes digitales” dado que son los únicos organismos que pueden asegurar la confiabilidad de credenciales de un certificado. [7]

Archivos PKCS12

Los archivos PKCS o archivos de intercambio de información personal, son documentos que almacenan datos criptográficos en un formato binario, estos archivos contienen el certificado digital y sus claves privadas de manera cifrada, Su función principal es permitir la importación y exportación de certificados junto con sus claves privadas, facilitando así la transferencia entre sistemas o aplicaciones. [8]

Un archivo PKCS incluye las extensiones “.p12” o “.pfx” que se pueden utilizar e instalar en el almacén de certificados de Windows.

Trabajar con archivos PKCS12 es beneficioso por las siguientes razones: [9]

- Soporte: PKSC12 al ser un archivo ampliamente reconocido es soportado y permitido por diversos softwares de seguridad y redes.
- Manejo de certificados: Puede ser utilizado por bibliotecas libres como OpenSSL, que proporcionan una gama amplia de comandos para su manipulación.
- Facilidad de importación y exportación: Los archivos PKCS12 son fáciles de importar y exportar en diferentes sistemas operativos además son fáciles de convertir utilizando OpenSSL.
- Protección de integridad: Soportan distintos mecanismos para verificar la integridad de los datos contenidos, asegurando que no hayan sido alterados.

Python

Python, como lenguaje de programación de código abierto, se caracteriza por una comunidad vibrante y en constante crecimiento. Esta característica lo convierte en una

herramienta invaluable para el desarrollo de proyectos de investigación, particularmente en el ámbito digital, Python ofrece una extensa colección de bibliotecas y módulos especializados en diversas áreas, tales como el análisis de datos, el aprendizaje automático, el desarrollo web y la automatización. Estos recursos, junto con su completa documentación, facilitan la realización de tareas complejas de manera eficiente y precisa. [10]

La sintaxis de Python se destaca por su simplicidad y claridad, lo que la convierte en un lenguaje accesible tanto para programadores experimentados como para aquellos que se encuentran en sus primeras etapas de aprendizaje. Esta facilidad de uso permite que los investigadores se enfoquen en los aspectos conceptuales de su trabajo, sin perder tiempo en complejidades sintácticas. [11]

Python encuentra una amplia gama de aplicaciones en el mundo real, incluyendo: [12]

- Análisis y visualización de datos: Python proporciona un conjunto de herramientas robustas como NumPy, Pandas y Matplotlib, las cuales permiten a los investigadores analizar y visualizar conjuntos de datos complejos de manera eficiente. Estas herramientas facilitan la extracción de información significativa de los datos y su comunicación efectiva.
- Aprendizaje automático: Python se ha convertido en el lenguaje de programación predilecto para el desarrollo de algoritmos de aprendizaje automático. Bibliotecas como scikit-learn y TensorFlow simplifican la creación de modelos predictivos y la implementación de técnicas de aprendizaje profundo.
- Desarrollo de software: Python es un lenguaje versátil que se utiliza para el desarrollo de aplicaciones de escritorio, web y móviles. Frameworks como Django y Flask facilitan la creación de aplicaciones web robustas y escalables con relativa facilidad.
- Desarrollo web: Python es ampliamente utilizado en el desarrollo web back-end, gracias a su capacidad para manejar grandes volúmenes de datos y su integración con bases de datos. Frameworks como Django y Flask simplifican la creación de aplicaciones web robustas y escalables.
- Automatización: Python se posiciona como una herramienta poderosa para la automatización de tareas repetitivas, liberando tiempo valioso a los investigadores para que se enfoquen en actividades más creativas y productivas. Bibliotecas como PyAutoGUI y Selenium permiten automatizar tareas interactuando con interfaces gráficas y navegadores web.

- DevOps: Python se utiliza ampliamente en las prácticas de DevOps, gracias a su capacidad para automatizar tareas de implementación, configuración y administración de sistemas. Herramientas como Ansible y Fabric facilitan la gestión de infraestructuras de manera eficiente y escalable.

OpenSSL

Forma parte de las bibliotecas de software libre las cuales son desarrolladas previamente por diferentes programadores, su función principal es brindar una colección de código que pueda ser reutilizado para poder resolver problemas o necesidades para agilizar el proceso de codificación y construir el software de manera más rápida y eficiente. [13]

OpenSSL se caracteriza por brindar bibliotecas de criptografía y seguridad, permite a los usuarios realizar tareas con algoritmos criptográficos los cuales son fundamentales para la generación de claves privadas. [14]

Uno de sus usos más comunes es ofrecer certificados que puedan ser utilizados con aplicaciones de software, los certificados proporcionados por OpenSSL aseguran que las credenciales de la organización o persona sean genuinas y no fraudulentas. [15]

Trabajar con bibliotecas libres tiene varias ventajas: [16]

- Coste: OpenSSL al ser una biblioteca de código abierto, no tiene un costo asociado. Esto permite a los desarrolladores ahorrar dinero en licencias.
- Transparencia: Un código OpenSSL tiene la capacidad de ser examinado, modificado o utilizado sin ningún tipo de restricción.
- Comunidad activa: OpenSSL cuenta con una comunidad activa que detecta y corrige errores rápidamente.
- Personalización: Las bibliotecas de código abierto pueden ser modificadas según las necesidades específicas del desarrollador
- Actualizaciones: Existen actualizaciones periódicas y frecuentes para mejorar la seguridad y adoptar los estándares más recientes.

Criptografía

Es una herramienta que sirve para almacenar información o datos que solo pueden ser visibles por el destinatario de forma legible.

Se encarga de la protección de datos mediante el uso de códigos, no permite alteraciones y sirve para la autenticación de usuarios.

Existen diferentes algoritmos criptográficos que son utilizados para la creación y verificación de firmas digitales.

La criptografía se enfoca en 4 objetivos:

- Confidencialidad: la información solo puede ser legible para las personas a quienes está destinada.
- Integridad: La información no puede ser modificada en el tránsito que existe del remitente al destinatario.
- Sin rechazo: No puede ser negada ninguna acción o modificación realizada por el remitente o el destinatario.
- Autenticación: Tanto el remitente como el destinatario pueden puede confirmar la identidad uno del otro. [17]

Algoritmos Criptográficos

Estos algoritmos se encargan de cifrar y descifrar un mensaje, de tal manera que dicha información solo sea visible en texto claro por el emisor y receptor.

Se pueden clasificar en dos grupos:

- Criptografía simétrica: Más conocida como criptografía de clave secreta, es aquella que posee una sola clave la cual se encarga de cifrar y descifrar la información, por ende tanto el emisor como el receptor deben poseer la misma clave, este fue el primer método utilizado para el cifrado de información.
- Criptografía asimétrica: Más conocida como criptografía de clave publica, es aquella que posee dos claves, la clave pública debe ser conocida por el emisor y el receptor, mientras que la privada debe ser únicamente conocida por el usuario emisor. [18]

Criptografía RSA

Es el algoritmo de clave pública más conocido, dicho cifrado comparte la clave pública abiertamente y la clave privada la mantiene secreta, solo la clave privada puede descifrar la información. RSA se basa en la dificultad de factorizar números grandes en primos siendo así computacionalmente inviable descifrarla.

RSA se utiliza en las firmas digitales para garantizar la integridad de los datos o mensaje. [19]

Funciones HASH

Se encargan de transformar los datos de entrada en una serie de datos finitos de salida, este tipo de algoritmos buscan proteger las contraseñas de tal forma que no sean legibles en texto claro. [20]

Las funciones HASH cuentan con una serie de algoritmos SHA para brindar seguridad, entre los más conocidos se encuentran:

- SHA-256: Conformado por un tamaño de palabra de 32 bits
- SHA-384: Conformado por un tamaño de palabra de 64 bits
- SHA-512: Conformado por un tamaño de palabra de 64 bits

Estos algoritmos se caracterizan por brindar seguridad en aplicaciones criptográficas.

SHA-256 es el más utilizado para aplicaciones de firmas digitales, por su característica de verificación de integridad de archivos.

Biblioteca *Cryptography*

Es una biblioteca utilizada en Python que proporciona herramientas de cifrado, descifrado, generación de claves, etc.

Es la biblioteca moderna más usada en Python orientada en criptografía. [21]

La biblioteca *cryptography* es considerada una herramienta poderosa y ampliamente utilizada en Python, permite realizar diversas operaciones criptográficas, dicha biblioteca fue creada para ser fácil de usar y ofrecer un alto nivel de seguridad.

Cuenta con primitivas criptográficas donde es posible la utilización de cifrado simétrico mediante AES o Blowfish, cifrado asimétrico mediante RSA o DSA. [22]

2 METODOLOGÍA

El proyecto de titulación propuesto tiene como objetivo la implementación de soluciones de servicios de cómputo. Se enfocó en el uso de herramientas específicas que adoptan herramientas para facilitar y eficientizar la automatización de servicios de cómputo. Además, se llevó a cabo una serie de pruebas para asegurar la funcionalidad óptima de las soluciones.

3 RESULTADOS

En este apartado se detalla el funcionamiento del sistema de firma electrónica implementado, proporcionando una comprensión clara de los procesos involucrados en la creación de firmas digitales mediante Python.

El sistema se desarrolla utilizando el lenguaje de programación Python y bibliotecas de código abierto, específicamente OpenSSL y *cryptography*. Estas bibliotecas proporcionan las herramientas necesarias para la manipulación de claves criptográficas, la generación de firmas digitales y la verificación de la integridad de los documentos.

El código Python se ejecuta en el intérprete de línea de comandos de Windows (CMD). Tras la ejecución, se solicita al usuario la información que desea incluir en su firma electrónica. El sistema genera un archivo .p12 que contiene el certificado digital, el cual tiene una validez de un año a partir de su emisión.

Con el certificado digital generado (.p12), se pueden firmar digitalmente archivos PDF utilizando aplicaciones como Adobe Acrobat Reader. Para ello, es necesario importar el certificado digital en el administrador de certificados de usuarios de Windows.

La implementación y creación de firmas digitales en la aplicación es altamente efectiva y a su vez relevante en la actualidad, proporcionando una seguridad y autenticidad equivalentes a las de una firma manuscrita tradicional. Utilizando algoritmos criptográficos en Python, la aplicación garantiza que cada documento firmado digitalmente sea único e inalterable, ofreciendo protección contra fraudes o manipulaciones. El uso de firmas electrónicas facilita procesos administrativos y comerciales, permitiendo la verificación inmediata de la identidad y la integridad de los documentos en un entorno digital.

La Figura 3.1 hace referencia al esquema a seguir para la creación de un certificado digital (.p12) que permita firmar digitalmente.

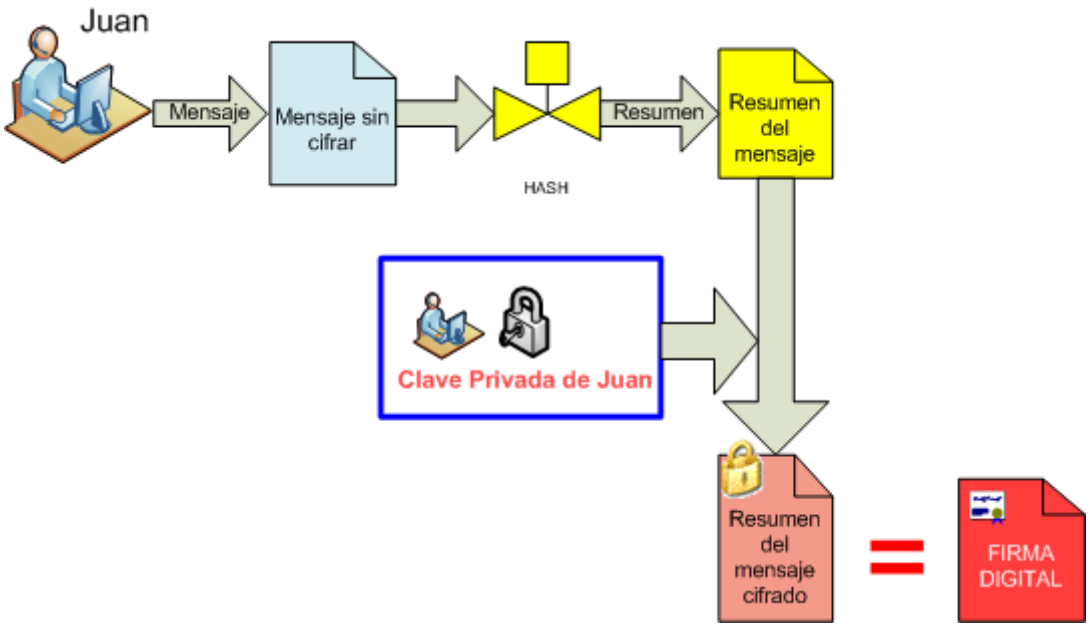

Figura 3.1 Cifrado y descifrado mediante el uso de algoritmos *[23]*.

3.1 Analizar soluciones a cada servicio de cómputo.

En esta sección, se describe los análisis e investigaciones realizadas sobre Python y soluciones de firmas electrónicas a través de bibliotecas libres. Mediante el análisis se logra proponer puntos necesarios a tomar en cuenta dentro de las soluciones a servicios de cómputo para asegurar la viabilidad y seguridad del sistema propuesto.

El objetivo es evaluar y comparar las diferentes opciones existentes en términos de rendimiento, facilidad de uso, compatibilidad con versiones de Python y el soporte activo de cada comunidad.

Identificación de servicios de cómputo

Es necesario realizar una identificación de los servicios de cómputo que se consideran relevantes para la implementación de firmas electrónicas. Existen puntos críticos a tomar en cuenta dentro de un sistema de firmas electrónicas, entre los cuales se encuentran:

1. Autenticación: Los protocolos a usar deben proporcionar seguridad para lograr verificar la identidad de los usuarios.
2. Claves: Los métodos a utilizar deben garantizar una base sólida para la creación de claves criptográficas para poder proteger los datos del usuario.
3. Certificado digital: Los certificados digitales deben tener la facilidad de integrarse a un sistema de firmas electrónicas.
4. Firma digital: La firma digital debe contener las características o datos del usuario proporcionados para la creación del certificado digital.

5. Verificación: El software a utilizar debe permitir visualizar la autenticidad de la firma electrónica.

Python

Python, como lenguaje de programación se destaca por sus diversos aspectos técnicos y su rendimiento, impulsados por su extensa comunidad y naturaleza de código abierto. La investigación realizada sobre Python resalta su facilidad de uso y su eficacia en el desarrollo de un sistema de firmas electrónicas. gracias a su compatibilidad con diferentes sistemas operativos y su capacidad para integrar bibliotecas criptográficas avanzadas, como *Cryptography*. Estas bibliotecas permiten la implementación de algoritmos de cifrado y firmas digitales de manera eficiente y segura.

Estas bibliotecas añaden nuevas funciones y características a Python, proporcionando herramientas y soluciones previamente diseñadas para una amplia variedad de tareas y problemas. El uso de bibliotecas de terceros es una práctica común en el desarrollo de software, ya que permite a los desarrolladores ahorrar tiempo y esfuerzo al reutilizar código comprobado y optimizado por otros expertos en codificación.

Python otorga la facilidad de implementar aplicaciones escalables, las cuales están diseñadas para permitir la incorporación de modificaciones, adición de nuevas funciones o mejoras a las ya existentes, sin necesidad de reconstruir la aplicación desde cero. Esto se logra mediante la capacidad de agregar nuevas características al código utilizado.

Además, su sintaxis clara y soporte para múltiples plataformas facilitan tanto su desarrollo como su crecimiento continuo en el enfoque de aplicaciones orientadas a firmas electrónicas, garantizando seguridad y confiabilidad en los procesos de autenticación y verificación de documentos electrónicos.

Bibliotecas Python

Entre las diversas bibliotecas de Python que trabajan con algoritmos de criptografía y que permiten implementar firmas electrónicas se destacan 3 bibliotecas:

- *PyCryptodome*: Ofrece una implementación extensa de criptografía, es adecuada para la generación y verificación de firmas digitales.
- *Cryptography*: Proporciona una interfaz amigable y es la más moderna en criptografía, cuenta con soporte para firmas digitales al igual que una documentación extensa.
- *M2Crypto y PyOpenSSL*: Destacan por su cualidad para generar y verificar firmas digitales pero se encuentran menos documentadas que las mencionadas

anteriormente y cuentan con una comunidad de soporte muy reducida en comparación con *cryptography*.

Cryptography

Cryptography es una biblioteca de Python diseñada para ofrecer herramientas seguras y sólidas para aplicar algoritmos criptográficos. Incluye funciones clave como generación de claves y cifrado tanto simétrico como asimétrico, fundamentales para implementar de manera eficaz algoritmos de firma digital como RSA, DSA y ECDSA."

Una de las fortalezas de *Cryptography* es su implementación eficiente y optimizada de algoritmos criptográficos. Utiliza bibliotecas OpenSSL, lo que garantiza que las operaciones criptográficas sean rápidas y confiables, sin comprometer la seguridad. Esto es crucial para aplicaciones que requieren alto rendimiento, como la verificación y generación de firmas electrónicas en tiempo real.

Cryptography utiliza certificados X.509 que cumplen con estándares definidos por organizaciones como ITU-T y ISO/IEC, especialmente diseñados para la gestión de infraestructuras de clave pública. Esta biblioteca ofrece funcionalidades especializadas para manejar eficazmente este tipo de certificados. Al adherirse a normativas internacionales reconocidas, *Cryptography* asegura la interoperabilidad y la fiabilidad en la autenticación y firma digital, facilitando su integración con otros sistemas y aplicaciones que también siguen estos estándares.

En resumen, *Cryptography* en Python ofrece un conjunto completo de herramientas para manejar certificados X.509, lo cual es crucial para asegurar la autenticidad, integridad y confidencialidad en aplicaciones que dependen de la infraestructura de clave pública.

Firmas Electrónicas

En esta sección se presenta la información obtenida para la implementación y evaluación de un sistema de firmas electrónicas el cual pueda ser desarrollado y ejecutado por Python y bibliotecas libres. La investigación realizada ha permitido identificar y analizar puntos claves tales como seguridad, eficiencia, facilidad de uso y cumplimiento de diversos estándares.

La seguridad es un aspecto crucial en un sistema de firmas electrónicas, un alto nivel de seguridad implica:

- Algoritmos criptográficos solidos: La utilización de algoritmos de cifrado reconocidos como: RSA o SHA-256 pueden ayudar a mejorar la seguridad, RSA es

el primer algoritmo criptográfico utilizado para cifrar y firmar digitalmente, este tipo de técnicas aseguran que las firmas electrónicas no puedan ser manipuladas o falsificadas sin ser detectadas como no legibles.

- Almacenamiento de claves: Es necesario realizar una generación y almacenamiento de claves, esto se logra mediante mecanismos de gestión de claves los cuales van a permitir que solo los usuarios autorizados puedan acceder a sus claves privadas para hacer uso de los certificados digitales y así poder firmar digitalmente.
- Integridad de datos: Las firmas electrónicas ayudan a garantizar la integridad de los datos contenidos así como tambien la integridad de los documentos firmados de tal manera que cualquier tipo de modificación en el documento invalidará la firma.

3.2 Diseñar la solución para cada servicio de cómputo mediante herramientas.

En esta sección se detalla la solución para la implementación de un sistema de firmas electrónicas mediante Python y el uso de bibliotecas libres donde se busca crear una aplicación con Python que permita la generación de certificados digitales, firma de documentos y verificación de las firmas de manera eficiente, segura y que no permita alteraciones.

Mediante el análisis realizado previamente donde se logra concluir que la biblioteca *Cryptography* es la mejor opción por diversos aspectos se logra obtener un panorama más claro sobre el tipo de aplicación que se desea realizar y los diferentes parámetros que debe contener la misma.

El diseño de la aplicación se basa en la selección de las herramientas adecuadas, dicho proceso conlleva a definir la arquitectura del sistema, y la planificación para posterior implementación de firmas electrónicas.

Para que un documento se pueda firmar digitalmente se requiere de la creación de una firma electrónica con los datos del usuario, los cuales pueden estar contenidos en datos biométricos o en un certificado digital, la Figura 3.2 representa a la creación de un documento con firma electrónica con los componentes mencionados previamente.

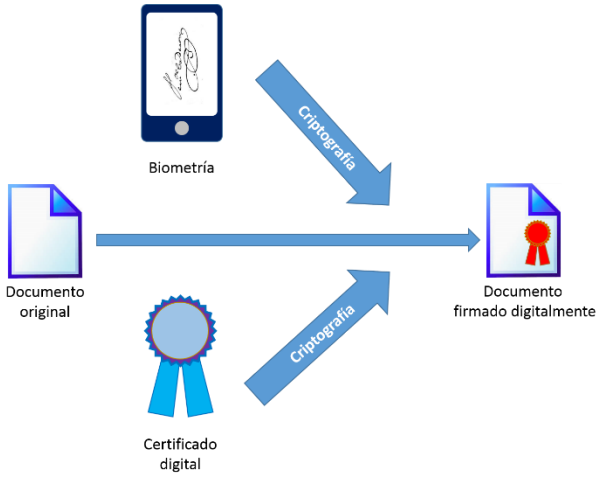

Figura 3.2 Creación de un documento firmado digitalmente *[24]*.

Diseño del sistema

Para poder realizar una aplicación o sistema de firmas electrónicas es necesario plantearse una arquitectura modular, es decir que permita la división de la aplicación en diferentes módulos, la finalidad de cada módulo es actuar como un subconjunto dentro de la aplicación y que cada uno tenga distintas funcionalidades que agreguen información a las firmas electrónicas y cumplan con una responsabilidad única.

Es decir, cada módulo debe tener características propias que aporten a la creación del sistema de firmas electrónicas y sean conectables entre sí.

La planeación de la arquitectura de un sistema permite la integración de módulos y componentes para que puedan coexistir, la Figura 3.3 hace referencia a que cada componente aporta a la creación de un sistema.

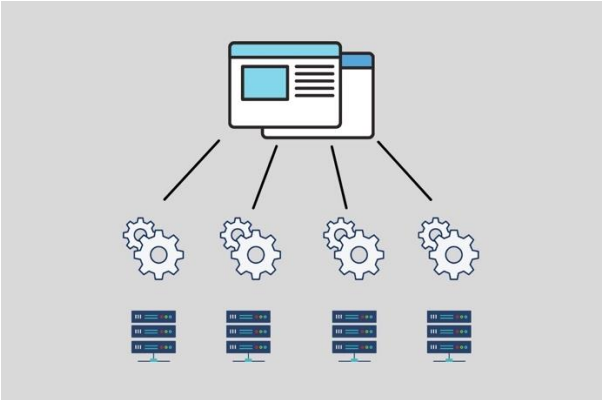

Figura 3.3 Unión de componentes para la creación de un sistema óptimo *[25]*.

Módulos y componentes

Los módulos en Python hacen referencia a los diferentes tipos de archivos que contienen distintas declaraciones, como funciones , clases y variables; para poder utilizar dichas declaraciones contenidas en un módulo se debe utilizar la palabra clave *import.*

Los componentes son las partes independientes de un sistema que cumplen cierta funcionalidad de manera específica y que pueden interactuar con otros componentes dentro del mismo sistema, los componentes pueden incluir funciones, clases y paquetes.

La biblioteca *Cryptography* proporciona varios módulos que pueden usarse para la implementación de firmas electrónicas, entre los más relevantes están:

1. ***hazmat.primitives.asymmetric.rsa:*** Este módulo proporciona la funcionalidad para la generación y manipulación de claves RSA, que son ampliamente utilizadas en criptografía de clave pública, incluidas las firmas digitales.
2. ***hazmat.primitives.asymmetric.dsa:*** Proporciona soporte para el algoritmo DSA el cual es otra opción para la creación de firmas digitales.
3. ***hazmat.primitives.asymmetric.ec:*** Este módulo ofrece soporte para criptografía de curva elíptica, que incluye algoritmos de firma como ECDSA.
4. ***hazmat.primitives.hashes:*** Utilizado para aplicar funciones hash criptográficas, que son parte esencial del proceso de firma digital.
5. ***hazmat.primitives.asymmetric.utils:*** Incluye utilidades para el procesamiento de datos relacionados con firmas, como la codificación.
6. ***hazmat.primitives.serialization:*** Proporciona funciones para la serialización y deserialización de claves criptográficas, permitiendo guardar y cargar claves públicas y privadas en diferentes formatos.

Creación de certificados autofirmados con atributos personalizados

El uso del estándar X.509 para la creación de certificados autofirmados se encarga de proporcionar un marco robusto para la implementación de una infraestructura de clave pública, que es esencial para garantizar la seguridad en las comunicaciones, a su vez los certificados X.509 permite la verificación de la identidad de las partes involucradas.

Los certificados autofirmados contienen información personalizada como el nombre, país, estado, ciudad, correo electrónico, etc. Estos atributos ayudan a identificar la identidad del propietario del certificado.

Interfaz de Usuario

El desarrollo de una interfaz de usuario basada en línea de comandos para la entrada de datos es sumamente necesario para la generación de certificados que puedan incluir nombre completo del usuario, país, provincia, ciudad, organización y correo electrónico.

Una interfaz de usuario bien diseñada facilita a los usuarios la creación y gestión de certificados digitales sin necesidad de conocimientos técnicos profundos. Esto permite a un mayor número de personas y organizaciones adoptar este tipo de tecnologías de firma digital.

Consideraciones de seguridad

Para poder realizar la implementación del sistema se debe tomar en cuenta previamente las medidas de seguridad para garantizar la integridad y confidencialidad de las firmas digitales y los certificados. El uso de RSA para generar un par de claves con un tamaño de clave de 2048 bits se considera seguro y se considera una práctica segura.

Utilizar funciones hash robustas como SHA-256 ayudan a garantizar que cualquier modificación en el documento pueda ser detectada. Firmar el certificado con la clave privada y un algoritmo de hash seguro como SHA-256 asegura la integridad del certificado y permite que terceros verifiquen su autenticidad.

Estas medidas de seguridad, que incluyen el uso de RSA, la personalización de atributos del certificado, la configuración de un período de validez, la asignación de números de serie únicos, la adición de extensiones críticas, la firma segura del certificado y la protección del archivo PKCS12 con una contraseña, son esenciales para garantizar la seguridad y la integridad de las firmas digitales y los certificados en tu implementación.

3.3 Implementar las soluciones mediante herramientas para el despliegue de los servicios de cómputo.

La implementación de la solución es una fase crucial para garantizar que las firmas electrónicas se realicen de manera segura y eficiente. Este apartado describe en detalle el proceso de implementación utilizando diversas herramientas y tecnologías.

Instalación de Python y OpenSSL

La presencia de ambos es esencial para implementar firmas electrónicas. Python y OpenSSL ofrecen las herramientas requeridas para desarrollar, desplegar y gestionar soluciones de firma digital de forma efectiva y conforme a estándares de seguridad globalmente reconocidos. La Figura 3.4 muestra la instalación de Python, es necesario

añadir Python al PATH del sistema para que sea más fácil ejecutar scripts desde cualquier ubicación sin especificar la ruta.

Figura 3.4 Instalación de Python.

La instalación de OpenSSL también debe ser en el PATH para que así cualquier comando pueda ser ejecutado desde cualquier ubicación sin especificar la ruta de ejecución. La Figura 3.5 indica la versión de la biblioteca OpenSSL instalada.

```
C:\Users\viva>openssl version
OpenSSL 3.3.0 9 Apr 2024 (Library: OpenSSL 3.3.0 9 Apr 2024)
```

Figura 3.5 Instalación de OpenSSL.

Instalación de bibliotecas y dependencias

La implementación se lleva a cabo en un entorno Windows, utilizando Python y la biblioteca *cryptography* para manejar las operaciones criptográficas. Para lograr aquello es necesario instalar la biblioteca cryptography en el entorno de trabajo de manera que pueda ser utilizado. La Figura 3.6 muestra la instalación de la biblioteca *cryptography* junto con todas sus dependencias.

```
C:\Users\viva>pip install cryptography
Requirement already satisfied: cryptography in c:\users\viva\appdata\local\pr
ograms\python\python310\lib\site-packages (42.0.7)
Requirement already satisfied: cffi>=1.12 in c:\users\viva\appdata\local\prog
rams\python\python310\lib\site-packages (from cryptography) (1.16.0)
Requirement already satisfied: pycparser in c:\users\viva\appdata\local\progr
ams\python\python310\lib\site-packages (from cffi>=1.12->cryptography) (2.22)


[notice] A new release of pip available: 22.2.2 -> 24.1.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

Figura 3.6 Instalación de la librería cryptography.

Script

El script desarrollado para la implementación del sistema de firmas electrónicas en Python utilizando bibliotecas libres contiene todos los elementos necesarios para la creación de certificados digitales y la gestión de firmas. El script completo se puede encontrar en la sección de ANEXO III: Códigos Fuente donde se detalla exhaustivamente su contenido. A continuación, se describe cada parte del script:

Bibliotecas importadas

El script incorpora las librerías de *cryptography* juntos con sus módulos respectivos para la creación y manejo de certificados digitales, la Figura 3.7 muestra las librerías usadas para la creación del sistema previsto junto con módulos para la generación de claves RSA, módulos de con funciones de serialización, etc.

```
import datetime
import os
from cryptography.hazmat import backends
from cryptography.hazmat.primitives import serialization
from cryptography import x509
from cryptography.hazmat.primitives.asymmetric import rsa
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.serialization import pkcs12
```

Figura 3.7 Bibliotecas cryptography y sus módulos implementados.

Gestión del contador

Se utiliza un archivo de texto “cont_serial.txt” para almacenar y gestionar el contador de series de certificados para que de esta manera cada certificado digital tengo un identificador único e irrepetible. Dicho archivo de texto se encarga de almacenar y gestionar el contador, de modo que se actualiza al último número de serie creado.

La Figura 3.8 muestra como está definida la función “serial()” la cual se encarga de leer el valor actual del contador serial desde un archivo de texto llamado “cont_serial.txt” Si el archivo existe, la función lee el contenido, lo convierte a un entero y lo retorna. Si el archivo no existe, la función retorna 0, indicando que el contador debe iniciarse desde cero. Además la función “guardar_serie(numeracion)” guarda el valor del contador serial en el archivo “cont_serial.txt” Se utiliza para actualizar el archivo con el nuevo valor del contador cada vez que se incrementa.

```
contador_path = "cont_serial.txt"

def serial():
    if os.path.exists(contador_path):
        with open(contador_path, "r") as file:
            return int(file.read().strip())
    else:
        return 0

def guardar_serie(numeracion):
    with open(contador_path, "w") as file:
        file.write(str(numeracion))

contador = serial()
```

Figura 3.8 Implementación de funciones para el contador serial.

Función de firma digital

La función principal del script es “**firma_digital()**”, que realiza las siguientes tareas:

Incremento de contador

Incrementa el contador global para asegurar la unicidad del número de serie del certificado, La Figura 3.9 hace referencia a dicho incremento en el contador.

```
contador += 1
```

Figura 3.9 Incremento del contador en 1 sucesivamente.

Datos de usuario

Solicita al usuario la información necesaria para generar su certificado digital:

La Figura 3.10 muestra el bloque de código que se encarga de recolectar la información proporcionada por el usuario, dicha información posteriormente es empleada para la creación del certificado digital personalizado.

```
nombreusuario = input("Ingrese su nombre completo: ")
pais = input("Ingrese su pais 'Ejemplo: EC, MX, US': ")
provincia = input ("Ingrese su estado o provincia: ")
ciudad = input("Ingrese su ciudad de residencia: ")
universidad_empresa = input("Ingrese el nombre de su universidad o empresa: ")
correo = input("Ingrese su correo electrónico: ")
contraseña = input("Ingrese su contraseña para proteger su certificado: ")
```

Figura 3.10 Implementación de formulario de datos para creación de certificados digitales.

Claves RSA

Crea una clave privada RSA y su clave pública correspondiente, La Figura 3.11 muestra la sección del script que genera un par de claves RSA.

```
private_key = rsa.generate_private_key(
    public_exponent=65537,
    key_size=2048,
    backend=backends.default_backend()
)
public_key = private_key.public_key()
```

Figura 3.11 Proceso de generación de clave RSA.

Certificado autofirmado

Construye un certificado autofirmado utilizando los datos recopilados por el usuario previamente junto con las claves RSA generadas, La Figura 3.12 muestra cada paso de dicho proceso, Este bloque de código es el que construye y firma un certificado digital autofirmado utilizando la biblioteca *cryptography*. El certificado incluye información personal del usuario, es válido por un año, y está firmado con una clave privada RSA.

```
builder = x509.CertificateBuilder()
builder = builder.subject_name(x509.Name([
   x509.NameAttribute(x509.NameOID.COMMON_NAME, nombreusuario),
   x509.NameAttribute(x509.NameOID.COUNTRY_NAME, pais),
   x509.NameAttribute(x509.NameOID.STATE_OR_PROVINCE_NAME, provincia),
   x509.NameAttribute(x509.NameOID.LOCALITY_NAME, ciudad),
   x509.NameAttribute(x509.NameOID.ORGANIZATION_NAME, universidad_empresa),
   x509.NameAttribute(x509.NameOID.EMAIL_ADDRESS, correo)
]))
builder = builder.issuer_name(x509.Name([
   x509.NameAttribute(x509.NameOID.COMMON_NAME, nombreusuario),
   x509.NameAttribute(x509.NameOID.COUNTRY_NAME, pais),
   x509.NameAttribute(x509.NameOID.STATE_OR_PROVINCE_NAME, provincia),
   x509.NameAttribute(x509.NameOID.LOCALITY_NAME, ciudad),
   x509.NameAttribute(x509.NameOID.ORGANIZATION_NAME, universidad_empresa),
   x509.NameAttribute(x509.NameOID.EMAIL_ADDRESS, correo)
]))
builder = builder.not_valid_before(datetime.datetime.utcnow())
builder = builder.not_valid_after(datetime.datetime.utcnow() + datetime.timedelta(days=365))
builder = builder.serial_number(contador)
builder = builder.public_key(public_key)
builder = builder.add_extension(
   x509.BasicConstraints(ca=True, path_length=None), critical=True,
)
certificate = builder.sign(
   private_key=private_key, algorithm=hashes.SHA256(),
   backend=backends.default_backend()
)
```

Figura 3.12 Bloque de código para la construcción y auto firma de certificados digitales.

Exportacion del certificado con su clave privada

Exporta la clave privada y el certificado en formato PKCS12 (.p12), es esencial para empaquetar la clave privada y el certificado digital en un mismo formato, La Figura 3.13 muestra la serialización de la clave privada y el certificado en un archivo (.p12).

```
p12 = pkcs12.serialize_key_and_certificates(
   nombreusuario.encode(), private_key, certificate, None, serialization.BestAvailableEncryption(contraseña.encode())
)
```

Figura 3.13 Creación del formato PKCS12.

Contador actualizado

La Figura 3.14 muestra como la parte del código “guardar_serie(contador)” llama a la función “guardar_serie(numeración)” para guardar el nuevo valor en el archivo de texto, para así seguir asegurando la unicidad de los números de serie de los certificados digitales.

```
guardar_serie(contador)
```

Figura 3.14 Parte del código que se encarga de guardar el nuevo valor de serie.

Retorno de archivo PKCS12

La función “return” retorna el archivo PKCS12 y el primer nombre del usuario, La Figura 3.15 devuelve dos valores: “.p12 y nombreusuario“ los cuales pueden ser usados posteriormente en el script.

```
return p12, nombreusuario.split()[0]
```

Figura 3.15 Retorno de valores para su uso posterior.

Generación y guardado de PKCS12

La función **“firma_digital()”** se ejecuta y el archivo PKCS12 generado se guarda con el primer nombre del usuario ingresado previamente en el script.

La Figura 3.16 muestra el uso de los valores retornados previamente y su respectiva ejecución, asegurando que cada archivo generado tenga un nombre distintivo.

```
firma_p12, primer_nombre = firma_digital()

certificadop12 = f"{primer_nombre}.p12"
with open(certificadop12, "wb") as f:
  f.write(firma_p12)
```

Figura 3.16 Ejecución y guardado del certificado.

Mensaje de confirmación

Finalmente, la aplicación imprime un mensaje de confirmación indicando que el certificado digital se ha creado y guardado exitosamente. La Figura 3.17 muestra el mensaje de confirmación para informar al usuario que el proceso se ha completado con éxito.

```
print(f"La firma digital y el certificado se han generado y guardado en '{certificadop12}'.")
```

Figura 3.17 Confirmación de creación exitosa del certificado.

3.4 Verificar el funcionamiento de cada servicio de cómputo.

En esta sección se describe la implementación y validación de certificados digitales a través de la ejecución de la aplicación desarrollada específicamente para este propósito. La aplicación fue diseñada para gestionar la creación de certificados digitales, asegurando la integridad y autenticidad de los documentos electrónicos firmados digitalmente.

Ejecución de la aplicación

Para poder generar y ajustar el certificado digital, es esencial ejecutar la aplicación desarrollada en Python de la siguiente manera:

La Figura 3.18 ilustra el método adecuado para ejecutar la aplicación desarrollada en Python desde el CMD de Windows. Es importante destacar que también se puede ejecutar simplemente haciendo doble clic sobre el archivo Python generado.

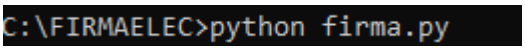

Figura 3.18 Ejecución del script de firma en Python.

Creación del certificado digital

Después de ejecutar la aplicación, crear el certificado digital es muy fácil: solo hay que introducir la información que la aplicación solicita al usuario. La Figura 3.19 ilustra cómo introducir correctamente la información y muestra el mensaje de confirmación de la aplicación que indica que el certificado se ha generado correctamente.

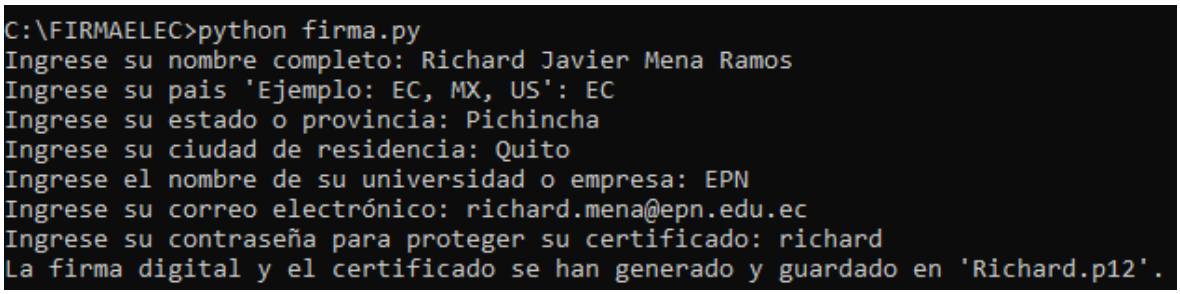

Figura 3.19 Generación exitosa del certificado en la aplicación.

Importación del certificado

Una vez generado el certificado en la aplicación, se debe importarlo. Para ello, es necesario ingresar la contraseña privada que se utilizó durante la creación del certificado. Una vez ingresada la contraseña, el asistente confirmará la importación del certificado, el cual se guarda automáticamente en el administrador de certificados de usuario.

La Figura 3.20 muestra cómo importar correctamente el certificado y seleccionar la ubicación donde se desea almacenarlo.

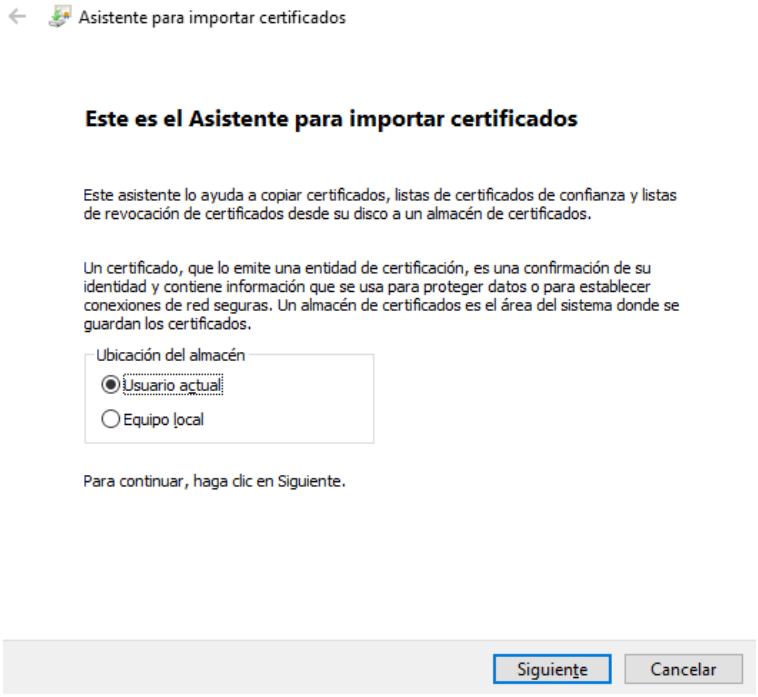

Figura 3.20 Ubicación de almacenamiento del certificado a importar.

La Figura 3.21 muestra la solicitud del asistente para ingresar la contraseña utilizada durante la creación del certificado en la aplicación.

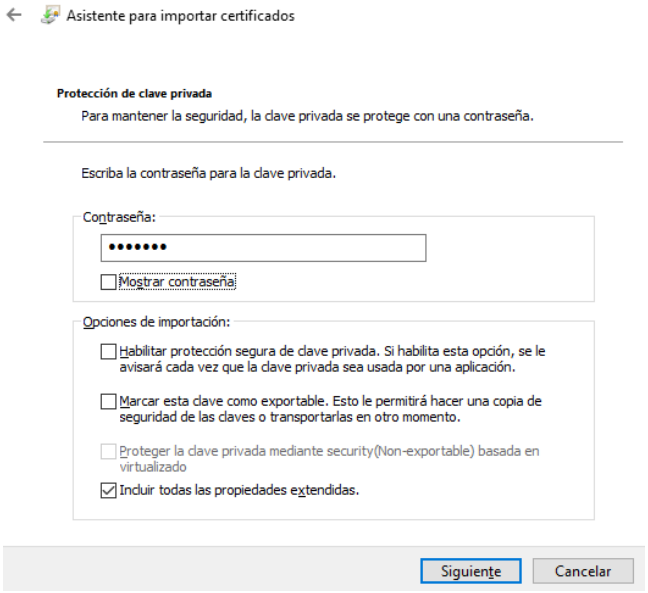

Figura 3.21 Solicitud de ingreso de contraseña para la importación.

La Figura 3.22 muestra el mensaje de confirmación de la correcta importación del certificado.

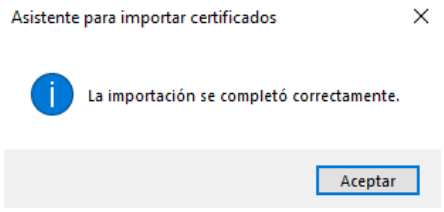

Figura 3.22 Importación completa del certificado.

Implementación de firmas electrónicas

Una vez configurados y preparados los certificados digitales generados mediante la aplicación desarrollada en Python, el siguiente paso es su integración y uso en Adobe Acrobat para la creación de firmas electrónicas. Este proceso asegura la autenticidad y la integridad de los documentos mediante la aplicación de firmas digitales verificables.

Los certificados digitales previamente generados e importados se deben configurar en Adobe Acrobat utilizando los certificados digitales preparados, se procede a aplicar firmas electrónicas a documentos PDF de prueba dentro de Adobe Acrobat. Este proceso se realiza seleccionando el certificado digital apropiado y aplicando la firma a través de las herramientas de gestión de firmas digitales del programa.

La Figura 3.23 presenta los certificados digitales generados por la aplicación en Python. Debe seleccionarse uno de ellos para poder firmar electrónicamente el documento PDF.

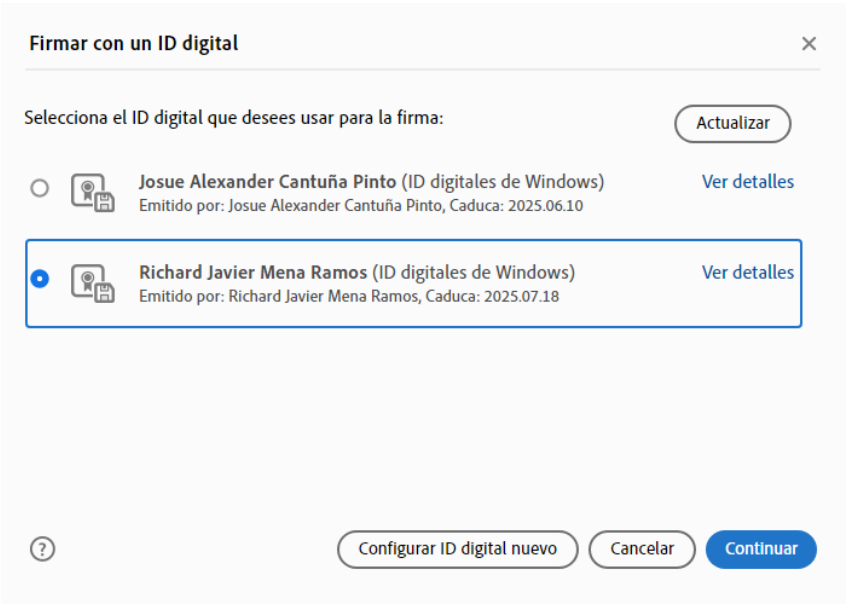

Figura 3.23 Selección del certificado digital para firma electrónica.

La Figura 3.24 presenta la aplicación de la firma electrónica en el documento utilizando el certificado digital seleccionado.

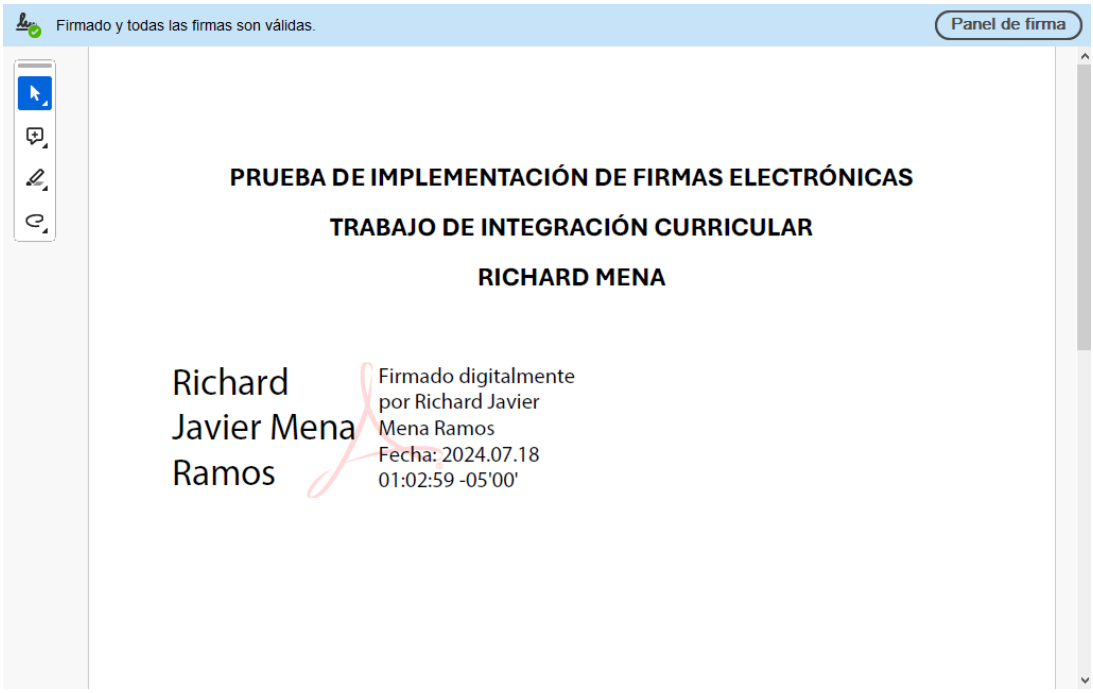

Figura 3.24 Documento Firmado Electrónicamente.

Validación de firmas electrónicas

Cada firma electrónica aplicada se somete a un riguroso proceso de validación dentro de Adobe Acrobat para garantizar su autenticidad y cumplimiento con los estándares de seguridad establecidos. Esto incluye la verificación de la integridad del documento y la autenticidad de la firma digital utilizada.

La Figura 3.25 muestra que la firma es válida y que el documento no ha sido modificado.

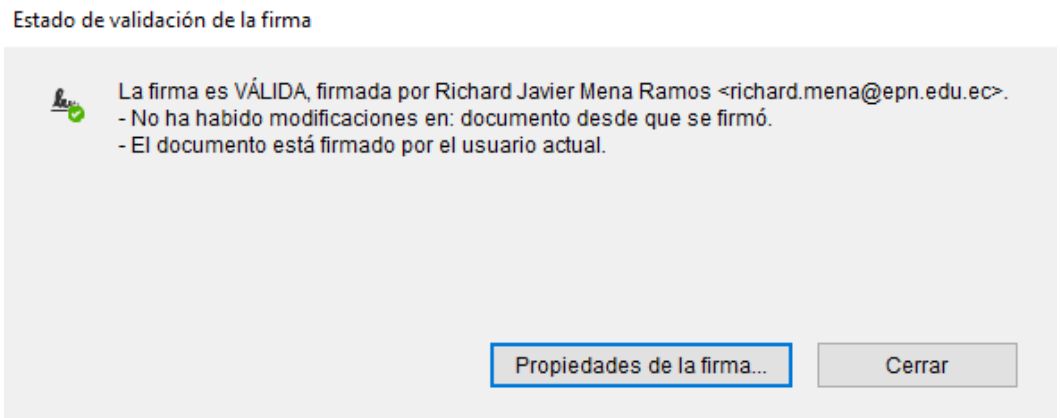

Figura 3.25 Estado de validación de la firma electrónica.

La Figura 3.26 presenta las propiedades de la firma junto con un resumen de su validez.

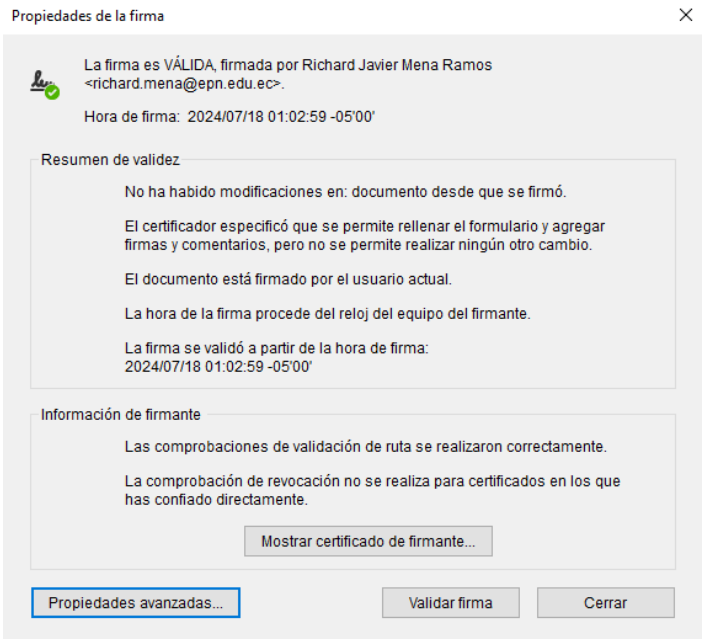

Figura 3.26 Propiedades de la firma.

La Figura 3.27 detalla las diversas propiedades avanzadas de la firma, como el algoritmo utilizado, el tiempo de validez y otros detalles relevantes.

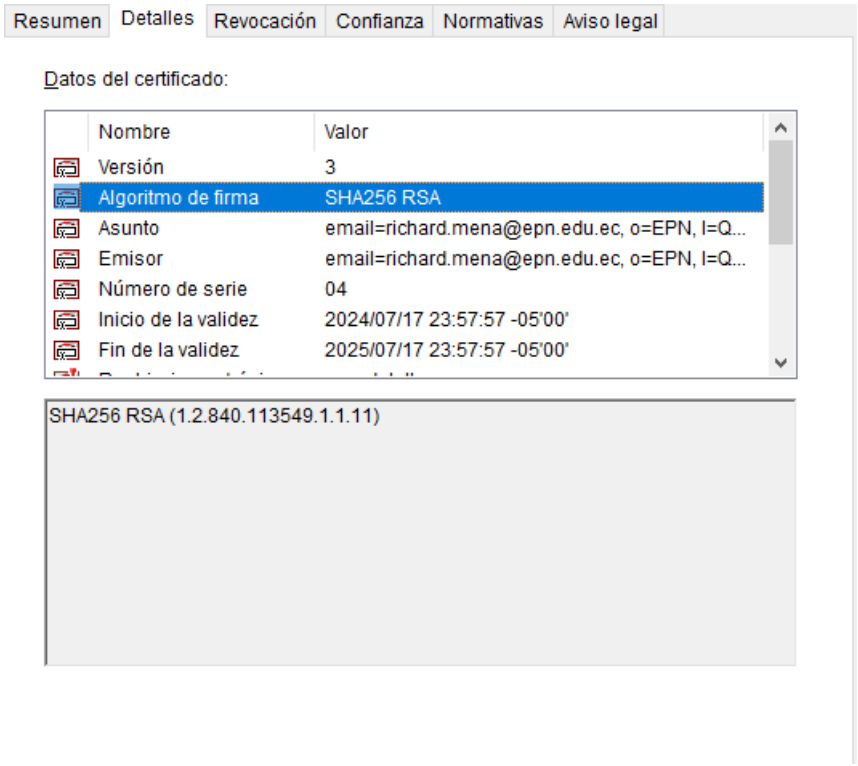

Figura 3.27 Propiedades avanzadas de la Firma Electrónica.

Este enfoque no solo asegura la correcta implementación de las firmas electrónicas, sino que también valida el proceso completo desde la generación inicial del certificado digital hasta su aplicación y verificación en un entorno práctico y reconocido como Adobe Acrobat.

Finalmente, la implementación del sistema de firmas electrónicas mediante Python y bibliotecas libres ha demostrado ser una solución efectiva y segura para la creación y gestión de certificados digitales. Los resultados obtenidos validan el cumplimiento de los objetivos específicos planteados, desde el análisis y diseño de soluciones hasta su implementación y verificación. La metodología empleada, junto con el uso de herramientas robustas como la biblioteca *cryptography*, ha permitido desarrollar un sistema flexible y escalable, capaz de integrarse fácilmente con aplicaciones comunes como Adobe Acrobat.

4 CONCLUSIONES

- Los resultados obtenidos subrayan la capacidad de Python para desarrollar soluciones de seguridad robustas y sugieren un camino prometedor para futuras investigaciones y desarrollos en el campo de las firmas electrónicas.
- La utilización de Python en el desarrollo del sistema de firma electrónica ha demostrado ser una elección acertada, aportando simplicidad, eficiencia, seguridad, flexibilidad y compatibilidad.
- La implementación de un sistema de firmas electrónicas utilizando Python y bibliotecas libres ha demostrado ser una solución viable y efectiva. Python, con su vasta colección de herramientas y su comunidad activa, proporcionó los recursos necesarios para desarrollar un sistema seguro, eficiente y fácil de mantener.
- La implementación de certificados digitales mediante Python y bibliotecas libres demuestra ser una solución viable para la autenticación y validación de documentos digitales. La elección de *Cryptography* como la biblioteca principal proporciona una base robusta y segura.
- La implementación de la aplicación para la creación de certificados digitales utilizando Python y bibliotecas de código abierto ha sido exitosa. El sistema desarrollado permite la generación de claves criptográficas, la creación y gestión de certificados digitales, y la integración de estos en aplicaciones como Adobe Acrobat para la firma y validación de documentos.
- El sistema es adaptable a diversos entornos profesionales y académicos, proporcionando una solución accesible y fácil de integrar para la firma digital de documentos. La interfaz intuitiva de Adobe Acrobat facilita el uso del sistema, incluso para usuarios con conocimientos técnicos limitados.
- Este proyecto contribuye al campo de la seguridad informática al ofrecer una herramienta práctica y segura para la firma digital de documentos. La metodología adoptada y las pruebas realizadas aseguran que el sistema cumple con altos estándares de seguridad, proporcionando una base sólida para futuras investigaciones y desarrollos en el ámbito de la criptografía y la seguridad digital.
- La automatización del proceso de creación de certificados digitales y la gestión del número de serie mediante un contador ha simplificado significativamente la administración de certificados. Esta metodología asegura que cada certificado sea único y evita conflictos o duplicaciones, mejorando la fiabilidad del sistema.

- La utilización de claves RSA y la implementación de algoritmos de hash seguros como SHA-256 garantizan la robustez de las firmas digitales creadas. Además, la protección de certificados mediante contraseñas y la exportación en formato PKCS12 aseguran que los datos sensibles estén resguardados contra accesos no autorizados.
- La solución desarrollada no solo es teóricamente sólida, sino también práctica y aplicable en contextos reales. La capacidad de generar certificados digitales personalizados y protegerlos adecuadamente hace que este sistema sea útil para una amplia variedad de aplicaciones, desde entornos académicos hasta empresariales.
- El proyecto ha cumplido con éxito los objetivos específicos planteados, desde el análisis de soluciones para servicios de cómputo, pasando por el diseño y la implementación de estas soluciones, hasta la verificación del correcto funcionamiento de cada componente. Esto valida la viabilidad técnica y funcional del sistema desarrollado.
- Aunque el sistema desarrollado ha cumplido con los objetivos específicos planteados, se identifican áreas para mejoras futuras. Esto incluye la exploración de nuevas bibliotecas y tecnologías emergentes, la mejora de la interfaz de usuario, y la ampliación de la compatibilidad con otras aplicaciones y plataformas.

5 RECOMENDACIONES

- Revisar y optimizar periódicamente el código utilizado para la generación de certificados y firmas digitales es fundamental. Es importante actualizar las bibliotecas a sus versiones más recientes para beneficiarse de mejoras en seguridad y rendimiento.
- Mantenerse actualizado con los avances en criptografía y tecnologías de firma digital. Evaluar periódicamente nuevas tecnologías y estándares que puedan ofrecer mejoras en seguridad, eficiencia y facilidad de uso, y considerar su implementación en futuras versiones del sistema.
- Aunque se utilizan claves RSA y algoritmos de hash seguros, se sugiere considerar la implementación de medidas de seguridad adicionales, como la autenticación multifactor (MFA) para el acceso a certificados y la utilización de Hardware Security Modules (HSM) para el almacenamiento de claves privadas.
- Es fundamental educar a los usuarios sobre la importancia de las firmas digitales y cómo utilizarlas correctamente. La capacitación debe incluir instrucciones sobre la creación de certificados, la protección de sus claves privadas y la verificación de firmas en documentos electrónicos.
- Si el sistema se va a utilizar ampliamente, el desarrollo una interfaz de usuario intuitiva y amigable ayudaría a facilitar la creación y gestión de certificados digitales. Una buena experiencia de usuario puede aumentar la adopción y el uso eficaz del sistema.

6 REFERENCIAS BIBLIOGRÁFICAS

[1] C. d. DocuSign, «Docusign,» 23 Enero 2023. [En línea]. Available: https://www.docusign.com/es-mx/blog/que-es-la-firma-electronica#:~:text=La%20firma%20electr%C3%B3nica%20es%20una,completar%20una%20solicitud%20de%20consentimiento.. [Último acceso: 26 Mayo 2024].

[2] C. d. Docusign, «Docusign,» 7 Noviembre 2022. [En línea]. Available: https://www.docusign.com/es-mx/blog/firma-digital-seguridad-informatica. [Último acceso: 28 Mayo 2024].

[3] Fortinet, «Fortinet,» [En línea]. Available: https://www.fortinet.com/lat/resources/cyberglossary/digital-certificates. [Último acceso: 27 Mayo 2024].

[4] Uanataca, «uanataca,» 16 Septiembre 2021. [En línea]. Available: https://web.uanataca.com/es/blog/firma-electronica/diferencias-con-certificado-digital. [Último acceso: 27 Mayo 2024].

[5] proofpoint, «proofpoint,» [En línea]. Available: https://www.proofpoint.com/es/threat-reference/digital-signature#:~:text=Algunos%20de%20los%20riesgos%20m%C3%A1s,para%20el%20robo%20de%20identidad.. [Último acceso: 27 Mayo 2024].

[6] E. d. s. d. SSL, «SSL.com,» 23 Septiembre 2019. [En línea]. Available: https://www.ssl.com/es/preguntas-frecuentes/%C2%BFQu%C3%A9-es-un-certificado-x-509%3F/. [Último acceso: 27 Mayo 2024].

[7] C. d. docusign, «docusign,» 12 Diciembre 2023. [En línea]. Available: https://www.docusign.com/es-mx/blog/autoridad-certificadora#:~:text=Una%20autoridad%20certificadora%2C%20o%20autoridad,autenticidad%20en%20el%20entorno%20digital.. [Último acceso: 27 Mayo 2024].

[8] E. d. s. d. SSL.com, «SSL.com,» 3 Marzo 2020. [En línea]. Available: https://www.ssl.com/es/c%C3%B3mo/exportar-clave-privada-de-certificados-del-archivo-pkcs12-con-openssl/. [Último acceso: 27 Mayo 2024].

[9] IBM, «IBM,» 16 Mayo 2023. [En línea]. Available: https://www.ibm.com/docs/es/rpa/21.0?topic=keys-generating-pkcs12-file. [Último acceso: 2 Junio 2024].

[10] Amazon, «AWS,» [En línea]. Available: https://aws.amazon.com/es/what-is/python/. [Último acceso: 26 Mayo 2024].

[11] E. Kosourova, «Datacamp,» Febrero 2024. [En línea]. Available: https://www.datacamp.com/es/blog/what-is-python-used-for. [Último acceso: 2 Junio 2024].

[12] E. C. Navone, «freeCodeCamp,» 22 Marzo 2022. [En línea]. Available: https://www.freecodecamp.org/espanol/news/para-que-se-usa-python-10-usos-del-lenguaje-de-programacion-python/. [Último acceso: 2 Junio 2024].

[13] OpenSSL, «OpenSSL,» [En línea]. Available: https://www.openssl.org/. [Último acceso: 2 Junio 2024].

[14] D. Gitlan, «SSL Dragon,» 4 Abril 2024. [En línea]. Available: https://www.ssldragon.com/es/blog/que-es-openssl/. [Último acceso: 26 Mayo 2024].

[15] EcuRED, «EcuRED,» [En línea]. Available: https://www.ecured.cu/OpenSSL. [Último acceso: 2 Junio 2024].

[16] «Universo Abierto,» 30 Diciembre 2015. [En línea]. Available: https://universoabierto.org/2015/12/30/los-beneficios-del-open-source-para-las-bibliotecas/. [Último acceso: 2 Junio 2024].

[17] J. S. Hurtado, «IEBS,» 8 Agosto 2022. [En línea]. Available: https://www.iebschool.com/blog/que-es-la-criptografia-y-para-que-sirve-finanzas/#:~:text=Es%20un%20m%C3%A9todo%20para%20almacenar,para%20la%20autenticaci%C3%B3n%20de%20usuarios.. [Último acceso: 27 Mayo 2024].

[18] A. Lopez, «redes zone,» 15 Mayo 2024. [En línea]. Available: https://www.redeszone.net/tutoriales/seguridad/criptografia-algoritmos-clave-simetrica-asimetrica/. [Último acceso: 27 Mayo 2024].

[19] «VERITAS,» [En línea]. Available: https://www.veritas.com/es/mx/information-center/rsa-encryption. [Último acceso: 17 Junio 2024].

[20] A. Lopez, «redes zone,» 16 Mayo 2024. [En línea]. Available: https://www.redeszone.net/tutoriales/seguridad/criptografia-algoritmos-hash/. [Último acceso: 27 Mayo 2024].

[21] F. Ruh, «Medium,» 25 Agosto 2023. [En línea]. Available: https://medium.com/@FridaRuh/encriptar-y-desencriptar-datos-en-pyhon-con-cryptography-5b186c669801. [Último acceso: 27 Mayo 2024].

[22] reaperhulk, «Pypi.org,» 04 Junio 2024. [En línea]. Available: https://pypi.org/project/cryptography/. [Último acceso: 15 Junio 2024].

[23] «Descom.es,» [En línea]. Available: https://www.descom.es/blog/correo-electronico/firma-digital/como-funciona-una-firma-digital.html. [Último acceso: 3 Junio 2024].

[24] «Telefonica Tech,» 26 Mayo 2015. [En línea]. Available: https://telefonicatech.com/blog/firma-digital-de-documentos-con. [Último acceso: 15 Junio 2024].

[25] «Solutions,» 12 Octubre 2021. [En línea]. Available: https://www.arrobasolutions.com/arquitectura-de-microservicios-que-es-y-cuales-son-sus-ventajas/. [Último acceso: 17 Junio 2024].

7 ANEXOS

El presente documento contiene distintos Anexos, que incorporan información relevante, pero que, por su extensión, no pueden ser incorporadas directamente en ninguna de las secciones anteriores.

La lista de los **Anexos** se muestran a continuación:

ANEXO I. Certificado de originalidad

ANEXO II. Enlaces

ANEXO III. Conjunto de datos extensos

ANEXO I: CERTIFICADO DE ORIGINALIDAD

CERTIFICADO DE ORIGINALIDAD

Quito, D.M. 30 de julio de 2024

De mi consideración:

Yo, FERNANDO VINICIO BECERRA CAMACHO, en calidad de Director del Trabajo de Integración Curricular titulado IMPLEMENTACIÓN DE SERVICIOS DE CÓMPUTO Y DE SEGURIDAD asociado a la IMPLEMENTACIÓN DE FIRMAS ELECTRÓNICAS MEDIANTE PYTHON CON BIBLIOTECAS LIBRES elaborado por la estudiante RICHARD JAVIER MENA RAMOS de la carrera en Tecnología Superior en Redes y Telecomunicaciones, certifico que he empleado la herramienta Turnitin para la revisión de originalidad del documento escrito completo, producto del Trabajo de Integración Curricular indicado.

El documento escrito tiene un índice de similitud del 12%.

Es todo cuanto puedo certificar en honor a la verdad, pudiendo el interesado hacer uso del presente documento para los trámites de titulación.

NOTA: Se adjunta el link del informe generado por la herramienta Turnitin.

Turnitin Final.pdf

Atentamente,

FERNANDO VINICIO BECERRA CAMACHO

Docente

Escuela de Formación de Tecnólogos

ANEXO II: ENLACES

A continuación, se puede acceder al video que muestra la implementación de la aplicación a través del siguiente enlace:

TESIS-Video de trabajo de integración curricular

Tambien, se puede ingresar mediante el siguiente código QR:

Anexo II.I Código QR de la implementación y pruebas de funcionamiento

ANEXO III: CÓDIGOS FUENTE

```
#BIBIOTECAS QUE UTILIZARE PARA LA CREACION DE LOS CERTIFICADOS DE LAS FIRMAS DIGITALES
import datetime
import os
from cryptography.hazmat import backends
from cryptography.hazmat.primitives import serialization
from cryptography import x509
from cryptography.hazmat.primitives.asymmetric import rsa
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.serialization import pkcs12
#Utilizaré un archivo txt que me ayude a guardar el incremento del contador
contador_path = "cont_serial.txt"
def serial():
    if os.path.exists(contador_path):
        with open(contador_path, "r") as file:
            return int(file.read().strip())
    else:
        return 0
def guardar_serie(numeracion):
    with open(contador_path, "w") as file:
        file.write(str(numeracion))
# Cargar el incremento de dicho contador
contador = serial()
def firma_digital():
    global contador
```

```
    # Realizar la incrementación del contador
    contador += 1
    # Datos que debe proporcionar el cliente para poder crear su certificado de su firma digital
    nombreusuario = input("Ingrese su nombre completo: ")
    pais = input("Ingrese su pais 'Ejemplo: EC, MX, US': ")
    provincia = input ("Ingrese su estado o provincia: ")
    ciudad = input("Ingrese su ciudad de residencia: ")
    universidad_empresa = input("Ingrese el nombre de su universidad o empresa: ")
    correo = input("Ingrese su correo electrónico: ")
    contraseña = input("Ingrese su contraseña para proteger su certificado: ")
    #Creación de las claves tipo RSA para poder generar la firma
    private_key = rsa.generate_private_key(
        public_exponent=65537,
        key_size=2048,
        backend=backends.default_backend()
    )
    public_key = private_key.public_key()
    # Creacion de un certificado autofirmado
    builder = x509.CertificateBuilder()
    builder = builder.subject_name(x509.Name([
```

```
        x509.NameAttribute(x509.NameOID.COMMON_NAME, nombreusuario),
        x509.NameAttribute(x509.NameOID.COUNTRY_NAME, pais),
        x509.NameAttribute(x509.NameOID.STATE_OR_PROVINCE_NAME, provincia),
        x509.NameAttribute(x509.NameOID.LOCALITY_NAME, ciudad),
        x509.NameAttribute(x509.NameOID.ORGANIZATION_NAME,
universidad_empresa),
        x509.NameAttribute(x509.NameOID.EMAIL_ADDRESS, correo)
    ]))
    builder = builder.issuer_name(x509.Name([
        x509.NameAttribute(x509.NameOID.COMMON_NAME, nombreusuario),
        x509.NameAttribute(x509.NameOID.COUNTRY_NAME, pais),
        x509.NameAttribute(x509.NameOID.STATE_OR_PROVINCE_NAME, provincia),
        x509.NameAttribute(x509.NameOID.LOCALITY_NAME, ciudad),
        x509.NameAttribute(x509.NameOID.ORGANIZATION_NAME,
universidad_empresa),
        x509.NameAttribute(x509.NameOID.EMAIL_ADDRESS, correo)
    ]))
    builder = builder.not_valid_before(datetime.datetime.utcnow())
    builder = builder.not_valid_after(datetime.datetime.utcnow() +
datetime.timedelta(days=365))
    builder = builder.serial_number(contador)
    builder = builder.public_key(public_key)
    builder = builder.add_extension(
        x509.BasicConstraints(ca=True, path_length=None), critical=True,
)
```

```
        certificate = builder.sign(
        private_key=private_key, algorithm=hashes.SHA256(),
        backend=backends.default_backend()
    )
    # Exportacion de la clave privada y su respectivo certificado a formato PKCS12 (.p12)
    p12 = pkcs12.serialize_key_and_certificates(
        nombreusuario.encode(), private_key, certificate, None,
serialization.BestAvailableEncryption(contraseña.encode())
    )
    # Guardar el valor del contador
    guardar_serie(contador)
    return p12, nombreusuario.split()[0]
# Ejemplo
firma_p12, primer_nombre = firma_digital()
# Guardar el archivo PKCS12 con el primer nombre del usuario
certificadop12 = f"{primer_nombre}.p12"
with open(certificadop12, "wb") as f:
    f.write(firma_p12)
#Mensaje del sistema indicando que se logró crear el certificado digital
print(f"La firma digital y el certificado se han generado y guardado en '{certificadop12}'.")
```