Universidad Cenfotec

Maestría en Ciberseguridad

Documento final de Proyecto de Investigación Aplicada 2

Diseño de un sistema para compartir información sensible con terceros mediante canales inseguros, con el fin de proteger dicha información de fugas accidentales.

Ortiz Sanabria Carlos José

Fecha: Julio, 2023

Declaratoria de derechos de autor

La presente investigación ha sido concebida y desarrollada por el autor único, Carlos José Ortiz Sanabria. Durante la elaboración de este trabajo, se ha empleado un conjunto diverso de fuentes bibliográficas, todas debidamente citadas y referenciadas en los diferentes capítulos de la investigación.

Se concede la autorización para utilizar parcial o totalmente esta investigación como referencia en futuros trabajos de carácter académico o científico, siempre y cuando realicen las referencias apropiadas.

Agradecimientos

Agradezco a todas aquellas personas me apoyaron a lo largo de este proceso en especial a mi querida esposa, por todos los fines de semana sacrificados y el oír cosas sin sentido, a mi familia por todo el apoyo directo e indirecto, y a todos los que ya no están en este plano por la huella que dejaron en este trabajo.

Aprobación del Tribunal

Universidad Cenfotec
Carrera de Postgrado
Maestría Profesional en Ciberseguridad

TRIBUNAL EXAMINADOR

Este proyecto fue aprobado por el Tribunal Examinador de la carrera: **Maestría Profesional en Ciberseguridad,** requisito para optar por el título de grado de **Maestría,** para el estudiante: **Ortiz Sanabria Carlos José.**

Digitally signed by MIGUEL PEREZ MONTERO (FIRMA)
Date: 2023.07.07 15:56:28 -06'00'

M.Sc. Miguel Pérez Montero
Tutor

JURGUEN AXEL KIRTON MENDEZ (FIRMA)
Firmado digitalmente por JURGUEN AXEL KIRTON MENDEZ (FIRMA)
Fecha: 2023.07.12 09:26:21 -06'00'

M.Sc. Jurguen Axel Kirton Méndez
Lector 1

IGNACIO TREJOS ZELAYA (FIRMA)
Firmado digitalmente por IGNACIO TREJOS ZELAYA (FIRMA)
Fecha: 2023.07.12 16:21:40 -06'00'

M.Sc. Ignacio Trejos Zelaya
Lector 2

San José, Costa Rica, 6 de julio de 2023

Firmada digitalmente, de conformidad con la Ley de Certificados, Firmas Digitales y Documentos Electrónicos N° 8454,

Tabla de Contenido

Tabla de Figuras

Resumen

En este documento se diseña e implementa un sistema para compartir información sensible dentro y fuera de una organización, a partir de dos grandes componentes, un cliente y un servidor. Los algoritmos que se utilizan para resguardar la información son los recomendados para su uso por instituciones como el Instituto Nacional de Estándares y Tecnología (NIST, por sus siglas en inglés) y por la Agencia de la Unión Europea para la Ciberseguridad (ENISA, por sus siglas en inglés). Además, se hacen pruebas comparativas para verificar cuáles serían los algoritmos y técnicas que no solo den mejor rendimiento, si no que utilicen de forma óptima los recursos. Uno de los grandes diferenciadores de este proyecto, con respecto a competidores como LastPass, Dashlane, 1Pass,Box o Dropbox, entre otros, es que la información sensible de la organización nunca será enviada a servicios de terceros, sino que se conservará dentro de las regulaciones y políticas de la entidad.

Palabras Clave: Encriptación, Canales inseguros, NIST, ENISA, Compartición, Estándares, Linux, Cloud Computing.

Capítulo 1. Introducción

1.1 Justificación

En un reporte de IBM sobre el costo de la filtración de datos (IBM Corporation, 2020) se reveló información interesante, con respecto a que el costo promedio de una fuga de datos es de 3.3 millones de dólares estadounidenses. Esta situación se presenta como uno de los mayores riesgos a los cuales las compañías actuales están expuestas, que además se incrementó debido a la pandemia por COVID 19, que inició a finales del 2019 y aún en el 2023 no se vislumbra todavía el final y advierte sobre situaciones similares que se puedan presentar en el futuro cercano. El riesgo de contagio forzó a muchas compañías a implementar metodologías de trabajo remoto, sin estar preparadas, ni tener adaptadas sus prácticas de trabajo a esta nueva realidad (Retina, 2020).

Un análisis realizado por Bárbara Hauer en el 2014 revela que de 1256 fugas de información, 214 fueron accidentales (Hauer, 2015); asimismo, afirma que la mayoría de estas fugas fueron internas, lo cual indica que la principal amenaza en las organizaciones usualmente no proviene de agentes externos, sino de los mismos empleados y cómo ellos comparten la información internamente o con terceras personas ajenas a la entidad. La finalidad de este proyecto es diseñar un sistema que funcione como repositorio central para información privilegiada y su distribución segura dentro y fuera de la empresa o institución.

1.2 Viabilidad

1.2.1 Punto de Vista Técnico.

Desde el punto de vista técnico la realización de este proyecto es completamente viable.

Se han investigado algoritmos ya probados y existentes relacionados con el almacenamiento y encriptación de información, lo cual garantiza que los algoritmos escogidos se pueden implementar con los recursos existentes. Además, estos métodos de cálculo son usados actualmente por la industria, por lo cual son estándares conocidos y bien documentados, lo que significa que existe suficiente material documental para apoyar esta investigación.

Sumado a esto, se utiliza un lenguaje de programación de alto nivel tal como Go, Python o Javascript, los cuales son familiares con otros lenguajes y de fácil adopción. Al ser los contenedores ultra portables (Poulton, 2019) se utilizarán para poder distribuir el resultado de forma más transparente y moderna.

1.2.2 Punto de Vista Operativo.

Para el proyecto se utilizan tecnologías Open Source, con estándares de facto para implementación y puesta a producción de la industria, como contenedores. El código fuente del proyecto usa la licencia BSD de tres cláusulas.(*The 3-Clause BSD License* | *Open Source Initiative*, s/f).

1.3.3 Punto de Vista Económico.

El producto final de este proyecto es un software libre, esto quiere decir que su código fuente estará disponible en repositorios de forma gratuita; por este motivo, su uso no implicará gastos de licenciamiento para usuarios finales. El autor de este trabajo sufragará los gastos de implementación y hospedaje en la nube pública Google Cloud y, de requerirlo la Universidad, la opción de ceder el código fuente a las bibliotecas de acceso público de la institución, para su uso y la aplicación de mejoras por parte de estudiantes y profesores. Para calcular la cantidad de horas totales que se invirtió en el proyecto se utilizó la fórmula $\mathrm{T_{horas}} = \mathrm{H} * 4^2$, que da como resultado un total de 480 horas, en un lapso de cuatro meses. Actualmente el salario promedio de un Ingeniero en Software, puesto actual del autor, es de 35.57 dólares americanos la hora (*Software Engineer Salary | PayScale*, s/f) por lo cual $T_{consultor} = T_{horas} * Salario_{hora}$ da un total de 17,073.06 dólares americanos.

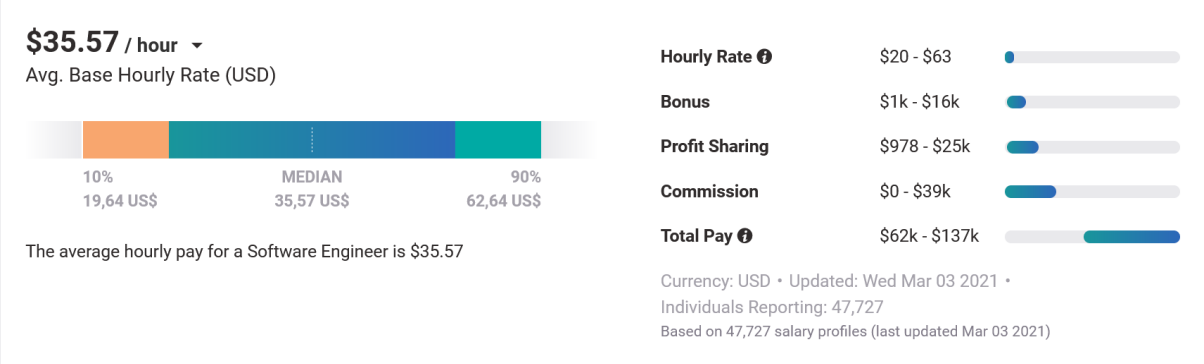

Figura 1: Salario Ingeniero de Software junio 2023 (Software Engineer Salary | PayScale, n.d.)

En las principales etapas de este proyecto se utilizaron los recursos computacionales del investigador, para el desarrollo de la solución. Conforme la implementación progresó esta fue desplegada en la nube pública Google Cloud y se utilizó la calculadora provista por el proveedor para calcular los costos mensuales de

la infraestructura mínima. El resultado de este costo es de 73.77 dólares americanos (*Calculadora de precios de Google Cloud Platform | Google Cloud*, s/f).

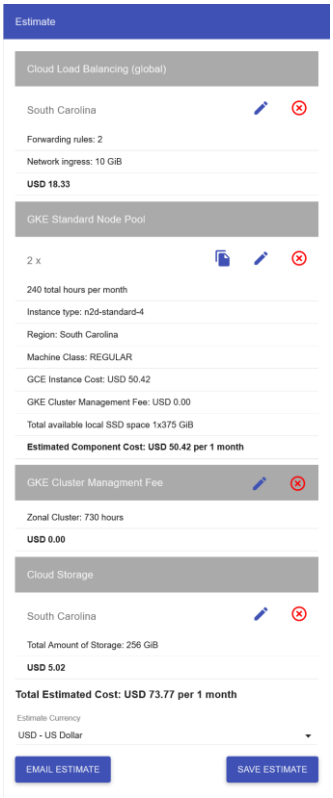

Figura 2:Estimación de costos de alojamiento por mes precios al 3 junio 2023 (Calculadora de Precios de Google Cloud Platform | Google Cloud, n.d.)

Se hace la salvedad que, debido a las características de la nube, estos costos podrían ser menores. Google Cloud ofrece hasta 300 dólares americanos de crédito para cuentas nuevas, por los primeros 90 días naturales o hasta agotar dicho financiamiento, lo que suceda primero (*Google Cloud Free Program*, s/f); ya que para el alojamiento en dicha nube solamente se utilizaron los últimos 60 días naturales, el

costo total de la infraestructura mínima es de 0 dólares americanos, usando la formula $T = (Costo_{mes} * Meses) - 300$.

Debido a que el proyecto estará bajo una licencia de código abierto, solo utilizará librerías con licencias compatibles y el uso de un ambiente de desarrollo de pago es opcional, por lo cual no se incluye en los costos.

Tabla 1 Resumen totales

Rubro	Total
Horas Consultor	$17,073.06
Alojamiento en la Nube	$0
Licencias	$0
Hardware	$0
	$17,073.06

Fuente: Elaboración Propia

1.4 Objetivos

1.4.1 Objetivo General

Diseñar un sistema para compartir información sensible con terceros, de forma segura, a través de canales inseguros, con el fin de proteger dicha información de fugas accidentales.

1.4.2 Objetivos Específicos.

- Identificar algoritmos simétricos y asimétricos de uso abierto, capaces de proveer un nivel de seguridad homologado por el NIST y el ENISA.
- Justificar, basado en los algoritmos seleccionados, el flujo de la información y cómo la información sensible será almacenada y compartida por el usuario destinatario.
- Desarrollar un servidor capaz de utilizar el algoritmo seleccionado para encriptar la información privilegiada, además de garantizar que dicha información está segura en reposo.

- Construir un cliente que pueda interactuar con el servidor mediante múltiples canales, además de utilizar autenticación de doble factor para determinar la identidad del receptor.

1.5 Alcances y Limitaciones

1.5.1 Alcances.

El alcance final del proyecto aquí expuesto es la implementación de un servidor capaz de encriptar y almacenar información sensible, tal como: documentos, texto plano, archivos comprimidos, imágenes y videos. El servidor también guardará registro del acceso y modificación de los datos almacenados, el cual incluirá la fecha y hora del evento, así como la identidad de los usuarios; todos los eventos de la bitácora serán firmados digitalmente para asegurar su integridad.

Adicionalmente, se entregará un cliente que interactuará con el servidor. Este cliente será el responsable de enviar los archivos al servidor para su almacenamiento, y se encargará de desplegar la información, ya sea mediante el navegador o como descarga, además de confirmar el destinatario mediante el uso de doble factor de autenticación.

En ambos casos, el cliente y el servidor no requieren el uso de servicios de terceros, a excepción del uso del doble factor de autenticación. La utilización de librerías o programas de soporte, como bases de datos, se justifica en este documento. Al usar tecnología de Contenedores se asegura un nivel muy alto de portabilidad, por lo cual cualquier servidor que sea capaz de ejecutar Contenedores compatibles con la especificación OCI (Open Container Initiative) será capaz de ejecutar el servidor y el cliente.

Ambos, cliente y servidor tendrán la capacidad de escalar de forma vertical y horizontal de forma nativa y sin necesidad de configuraciones adicionales, para eso, el servidor podrá leer y escribir un almacenamiento central, ya sea un disco en red o almacenamiento en la nube, tal como Amazon S3, Google Cloud Storage, Azure Blob Storage o similares. En el caso de discos locales o en red, la configuración será

delegada al Sistema Operativo y al ejecutor de Contenedores y el almacenamiento en la nube será integrado de forma nativa en el servidor, por lo cual solo se requiere que el ejecutor de Contenedores tenga los permisos necesarios para poder guardar, modificar y leer información del almacenamiento deseado.

1.5.2 Limitaciones.

Se excluyen del proyecto los siguientes elementos:

- Aplicaciones Móviles
- Sistema de control de acceso basado en roles
- Opciones de búsqueda de la información ya sea por nombre o metadatos
- Un visor de la bitácora de eventos.
- Manuales de Usuario o de Administrador
- Almacenamiento en la nube fuera de Google, AWS o Azure
- Integración con otro tipo de sistemas.

1.6 Marco de Referencia Organizacional y Socioeconómico

Estudios como los de IBM (IBM Corporation, 2020), Bárbara Hauer (Hauer, 2015) y el Diario el País (Retina, 2020) indican que las fugas accidentales de información son más frecuentes que el robo de información, ya sea por ataques cibernéticos, pérdida de credenciales o empleados malintencionados.

1.7 Estado de la Cuestión

A continuación, se muestra el proceso de revisión que se llevó a cabo usando como base la plantilla propuesta en la publicación Systematic Review in Software Engineering (Mian et al., s/f).

1.7.1 Planificación de la revisión

1.7.1.1 Formulación de la pregunta

¿Qué metodologías y prácticas existen para almacenar y compartir información sensible dentro y fuera de la organización con el propósito de evitar fugas accidentales?

1.7.1.1.1 Foco de la Pregunta

De acuerdo con la pregunta planteada en el punto anterior, y con apoyo en los objetivos específicos, se listan y filtran fuentes o soluciones existentes, con la finalidad de construir una solución diferenciada, y con una base tecnológica sólida.

1.7.1.1.2 Amplitud y calidad de la pregunta

1.7.1.1.2.1 Problema

La transmisión de información sensible es parte del diario quehacer de las organizaciones, la fuga accidental de datos es uno de los factores que inciden con más frecuencia en la pérdida de información y que puede llegar a costar millones de dólares. Este riesgo se acrecentó debido a la pandemia del COVID-19, dado que muchas empresas tuvieron que enviar a sus colaboradores a trabajar remotamente y compartir información de la organización por medio de redes o canales no seguros.

1.7.1.1.2.2 Pregunta

¿Cómo se guarda y comparte la información sensible dentro y fuera de las organizaciones? La pregunta anterior fue el resultado del proceso de revisión del foco y los objetivos general y específicos.

1.7.1.1.2.3 Palabras Clave

Debido a que el grueso de las publicaciones y estándares que se usan como base para la solución están en idioma inglés, las palabras clave fueron seleccionadas en ese idioma. A continuación, un listado de las palabras claves más relevantes sin ningún orden en particular: “encryption”, "encryption benchmark", ‟Homomorphic encryption”, “storage at rest”, “symmetic encryption”, “asymmetric encryption algorithms”, “NIST”, “ENISA”, “HOTP”, ‟Anti-Forensics”, “LUKS”, “Data Leakage”, “AES”, ‟elliptic curve”, “Blockchain”, “FIPS”, “key derivation”.

1.7.1.1.2.4 Intervención

De los documentos y estándares más relevantes se extrajo la información que era útil para la elaboración del proyecto.

1.7.1.1.2.5 Control

Se comienza la investigación sin ninguna base de información, pero existen parámetros aceptados por entes reguladores como el NIST o el ENISA; así como la información de fuentes como grandes compañías, o fundaciones, que no se descarta, pero se tiene en cuenta de acuerdo con su relevancia, a partir de los cuadrantes mágicos de Gartner. En el caso de la fundación de software, se toman en cuenta con la relevancia en la industria, basada en sus miembros fundadores y actuales.

1.7.1.1.2.6 Efectos

Con los resultados obtenidos en las búsquedas se espera entender las técnicas modernas y homologadas de encriptación, además de formas seguras de generación de llaves y almacenamiento de información en reposo.

1.7.1.1.2.7 Medida de Salida

Se asume que los documentos de fuentes como NIST, ENISA o Grupo de Trabajo de Ingeniería de Internet (IETF por sus siglas en inglés) son de entera confianza. Si la fuente del documento no pertenece a ninguna de las mencionadas anteriormente, se consulta su origen en sitos especializados. En caso del uso de documentación generada por empresas, el lugar que ocupa la organización en el cuadrante mágico es tomado como referencia por el autor.

1.7.1.1.2.8 Población

Al ser un proyecto nuevo, no existe población.

1.7.1.1.2.9 Aplicación

Este proyecto puede ayudar a todo tipo de organizaciones que trasieguen información sensible, ya sea de forma interna o con personas ajenas a la organización. Al ser un proyecto de código libre no existen restricciones monetarias para su uso más allá del costo de su alojamiento.

1.7.1.1.2.6 Diseño de experimental

Se clasificaron y analizaron los resultados de las búsquedas en la creación de un diseño experimental y se confirmó que se cuenta con suficiente información para

poder realizar el resto del proyecto. Varios de los campos investigados son muy amplios, por lo cual el diseño ayudó a limitar el espectro de la investigación.

1.7.2 Selección de fuentes

La gran mayoría de las fuentes seleccionadas son de muy alta fiabilidad como la proporcionadas por las instituciones NIST, ENISA y IETF, la cuales son referentes mundiales en múltiples campos: la IETF es el ente regulador de los estándares usados en internet, y por su parte, NIST y ENISA son las agencias reguladoras en temas de seguridad de Estados Unidos y la Unión Europa, respectivamente. Los documentos cuyas fuentes no eran las anteriormente mencionadas fueron sometidos a criterios de calidad y jerarquización de Scimago Journal & Country Rank.

En este sitio web público se calcula un índice denominado SJR, que mide las citaciones de un artículo por un periodo de al menos tres años, las revistas son categorizadas en subáreas, por ejemplo, Inteligencia Artificial, en lugar de Computación. El índice SJR tiene sus raíces en el modelo similar de Google Page Rank, como se menciona en la publicación A further step forward in measuring journals' scientific prestige: The SJR 2 indicator (Guerrero-Bote & Moya-Anegón, 2012), en el cual se explica detalladamente cómo se calcula dicho índice.

En el portal, también se hace referencia al índice-H o “H-index” por sus siglas en inglés, que fue planteado por el físico Jorge Hirsch en el 2005 (Hirsch, 2005). Al igual que el SJR, ambos califican el impacto científico de una publicación basado en la cantidad de citaciones o referencias en otras publicaciones. Para este documento se tomaron en cuenta ambos, pero si se destaca una preferencia del autor por dar más peso al índice SJR que al índice H.

Al mismo tiempo que Scimago Journal & Country Rank, se utilizó el sitio web QS World University Rankings para medir el peso de una universidad. Este sito califica más de 800 universidades a nivel mundial y fue usado para medir el impacto y calidad de referencias que no fueran publicadas en revistas del sector, por ejemplo, el caso de tesis o trabajos de investigación públicos. Debido a que no existen datos públicos

de como QS Quacquarelli Symonds genera el rango, este fue tomado como referencia, pero en caso de ser necesario, se consultaron otras fuentes para hacer una verificación cruzada.

Fuentes como IEEE Xplore, ACM Journals fueron exentas de su revisión ya que se consideran fidedignas y confiables, al ser también referentes de la industria y de la academia.

1.7.3 Ejecución de la revisión

Una vez realizada la selección, basado en los parámetros anteriormente mencionados, se escogen las siguientes fuentes para ser base de la investigación.

1.7.3.1 LUKS1 On-Disk Format Specification

En este documento (Fruhwirth, 2018) se especifica cómo funciona el “Linux Unifed Key Setup” el cual es el sistema de facto para la encriptación de discos duros en el sistema operativo de Linux, la relevancia de método se da ya que, al ser empleado en casi todas las verticales de la industria, se garantiza que ha sido probado y corregido contra vulnerabilidades conocidas. La técnica con la cual se guarda la información y las llaves puede ser perfectamente usada en ambientes de alto nivel, como guardar información en un sistema de archivos o base de datos.

1.7.3.2 PKCS #5: Password-Based Cryptography Specification & Key Derivation Function: The SCKDF Scheme

El RFC 2898 (Kaliski, 2000) & SCKDF (Chuah et al., 2013) defiende la especificación de funciones de derivación de llaves más utilizadas por la industria, con esta información se generan comparaciones evaluativas y cuadros, con el propósito de elegir la óptima para la solución buscada.

1.7.3.3 Algorithms, Key Sizes and Parameters Report *(European Network and Information Security Agency ., 2013)*

En este documento de la ENISA se detallan los algoritmos, tamaño de llaves y otros parámetros mínimos para que una solución que use criptografía sea considerada segura por esta agencia, por lo cual este documento es necesario para filtrar algoritmos que por su naturaleza no sean considerados seguros por esta agencia europea.

1.7.3.4 Guideline for Using Cryptographic Standards in the Federal Government *(Barker, 2016a)*

Al igual que en el documento de la sección 1.7.3.3, esta guía da las pautas a seguir para que una solución criptográfica sea considerada segura por las Agencias Federales de los Estados Unidos.

1.7.3.5 Publicly Verifiable Secret Sharing

Stadler (Stadler, 1996) propone una técnica para comprar secretos de tal manera que estos puedan ser verificables pero no descriptibles, esta técnica podría ser empleada en la solución que se plantea, por lo cual se considera fuente valiosa de información.

1.7.3.6 HOTP: An HMAC-Based One-Time Password Algorithm

En este RFC se detalla cómo se debería poner en funcionamiento esta técnica para generar contraseñas de solo un uso. Si bien, no es un objetivo de este proyecto generar una implementación, tener la especificación de esta ayuda a escoger el método para llevarla a cabo, y entender mejor su uso, esto con el fin de evitar problemas de seguridad por una implantación deficiente.

1.7.3.7 Anti-Forensics: Techniques, Detection and Countermeasures & TKS1 - An anti-forensic, two level, and iterated key setup schemes

La publicación de Garfinkel (Garfinkel, 2017) y (Fruhwirth, s/f, p. 1) detallan medidas y técnicas anti forenses. Se plantea el uso de las medidas aquí propuestas para aumentar la seguridad de la información cuando esta esté en reposo.

1.7.3.8 Cryptography and network security: principles and practice

Se usa Cryptography and network security: principles and practice (Stallings, 2017) como fuente de referencia para los temas relacionados con criptografía. La información se usa para decidir cuáles de los cifradores, llaves, o combinaciones pueden ser las óptimas para este proyecto.

Capítulo 2. Marco Teórico o Conceptual

Figura 3 Nube de palabras, Fuente Propia. Elaboración propia

En el presente capítulo se muestran los conceptos con mayor peso y relevancia en la investigación, en algunos de los casos se definen de forma general los algoritmos, pero se omite su parte técnica, queda a criterio del lector investigar el funcionamiento específico del algoritmo mencionado. Los conceptos definidos a continuación están ordenados por su relevancia dentro de la investigación.

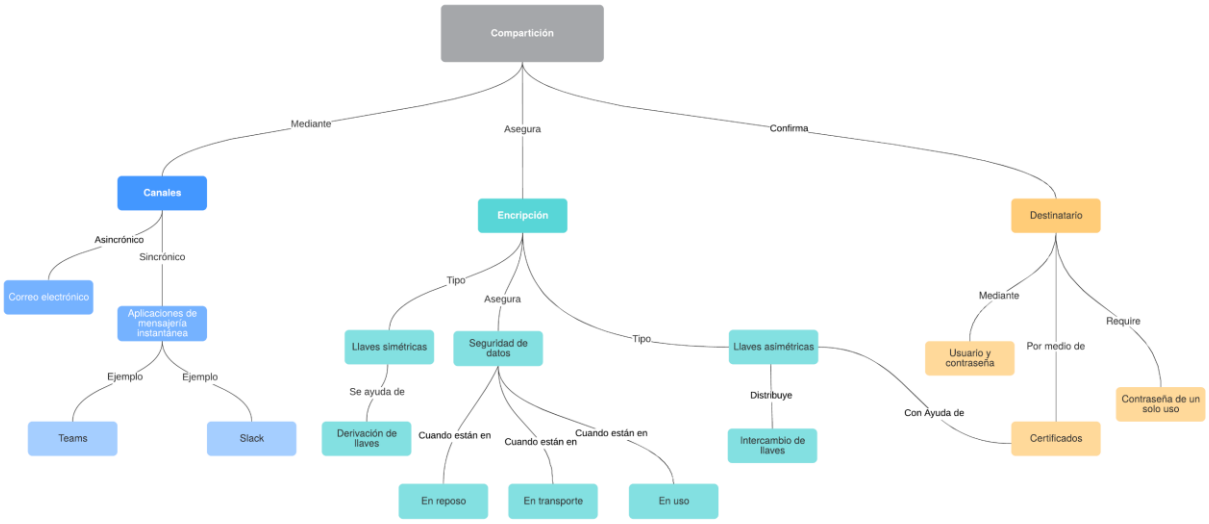

Figura 4 Mapa conceptual. Elaboración Propia

2.1 Conceptos sobre encriptación

La encriptación consiste en ofuscar un mensaje de tal forma que solo el transmisor y el receptor o receptores legítimos puedan comprender la información enviada. Si bien, la encriptación tiene cientos de años de historia, el uso masivo de Internet ha puesto en evidencia que esta es parte fundamental de la seguridad, ya que gracias a ella se puede garantizar seguridad, anonimidad y verificación de identidad e integridad.

2.1.1 Llaves simétricas

Digital Identity Guidelines(Grassi et al., 2017) define las llave simétricas como un instrumento criptográfico que se usa para encriptar y desencriptar el mensaje de la comunicación, y como su nombre lo sugiere, el receptor y transmisor del mensaje tienen que acordar esta llave antes de que la comunicación encriptada empiece.

Una de las principales debilidades de esta técnica es que si la llave cae en manos de terceros estos podrán desencriptar el mensaje; sin embargo, los algoritmos como Advanced Encryption Standard (AES) (*FIPS 197, Advanced Encryption Standard (AES)*, s/f), de llaves simétricas, son considerados seguros (Barker, 2016b) y de uso diario. En combinación con llaves asimétricas, esta técnica se usa en la comunicación segura en navegadores (Rescorla, 2018). Asimismo, los estándares de Linux (Fruhwirth, 2018), Windows (Dansimp, s/f-b), y mediante hardware (Trusted Computing Group, 2015), para asegurar la información en reposo, se basan en algoritmos de llaves simétricas.

Figura 5 Fuente (*U6.3 Esquema general de un sistema de cifrado simétrico*, s/f)

2.1.1.1 Derivación de llaves

La derivación de llaves o función de derivación de llaves es un componente básico en la criptografía. Las funciones de derivación de llaves se usan para derivar el material de claves de salida, partiendo de otra información secreta; esa información puede ser otra clave o, por ejemplo, una contraseña (Percival & Josefsson, 2016).

2.1.2 Llaves asimétricas

Como se mencionó, el mayor problema de las llaves simétricas es que el transmisor y el receptor tienen que conocer las mismas llaves, para eso tiene que existir algún centro de intercambio de llaves. Según Whitfield Diffie y Martin Hellman esto anula la idea de la criptografía, ya que todas las llaves tienen que estar en algún lugar centralizado (Stallings, 2017). Debido a esto, en 1979, Witfield Diffie y Martin Hellman publican el libro New Directions in Cryptography (Diffie & Hellman, 1976), en el cual proponen una nueva técnica de encriptación, mediante el uso de un Grupo multiplicativo de enteros módulo *n* para generar dos llaves: una pública y otra privada. En este caso, si un transmisor emite un mensaje privado usa la llave pública del receptor para encriptar dicho mensaje, solo el receptor, dueño de la llave privada que corresponde a esa pública, puede desencriptar y leer el mensaje; debido a esto, la transmisión de llaves públicas en canales inseguros no representa ningún riesgo, es más, existen repositorios con millones de llaves públicas, las cuales se usan para verificar firmas digitales de cualquier tipo de información como instaladores (Callas et al., 2007) y correos electrónicos (Schaad et al., 2019). Además, se usa como base para la firmas digitales, en el caso de Costa Rica se utiliza el algoritmo RSA con una llave de al menos 2048 bits, a la fecha de confección de este documento (Ministerio de Ciencia, Tecnología y Telecomunicaciones, 2013).

Figura 6 Fuente (*U6.6 Esquema general de un sistema de cifrado asimétrico*, s/f)

2.1.2.1 Intercambio de llaves

En el intercambio de llaves denominado Diffie-Hellman, las llaves de cifrado se pueden comunicar abiertamente ya que no representan ningún riesgo para la confidencialidad de los mensajes cifrados. Una parte intercambia las llaves con otra parte, ya que solo la llave de descifrado, en este caso corresponde a la llave privada, puede descifrar dicho mensaje. En ningún momento, durante el intercambio de claves Diffie-Hellman hay información sensible en riesgo de ser comprometida, a diferencia del intercambio de claves simétricas. Sin embargo, el protocolo de intercambio de claves Diffie-Hellman tiene un problema de autenticación, ya que no existe un método para probar la identidad real del destinatario, o del remitente.

La Infraestructura de llave pública o PKI (estándar X.509) resuelve este problema, ya que, en su implementación, cada usuario solicita a una autoridad certificadora (CA), en la que todas las partes confían, un certificado digital como una autenticación de identidad. En caso de que una CA fuera comprometida, el mismo protocolo contiene formas de revocar cualquier certificado (Cooper et al., 2008).

2.2 Conceptos sobre autenticación

En la autenticación se verifica que el usuario es quien dice ser, esto usualmente se hace mediante un factor de autenticación, una clave que solo el usuario sabe, también conocida como contraseña.

Existen otros tipos de autenticación como mediante “Mutual TLS”, la cual es un intercambio de certificados de mutua confianza (Oiwa et al., 2017). En este tipo de autenticación no es necesario tener un usuario o contraseña; y si bien este método es más seguro que el uso de contraseñas, como se menciona en la especificación de este, no es tan popular, ya que su empleo puede resultar más complejo para el usuario final. Un estudio comisionado por la empresa NordPass del 2020 revela cuáles son las contraseñas más comunes (*Top 200 Most Common Passwords of 2020 | NordPass*, s/f). Abonado a eso, el poder computacional y el uso de protocolos de derivación de llaves o “hash” que no se consideran seguros como el MD5 o SHA-1 (European Network and Information Security Agency, 2013) hacen que poder adivinar una contraseña mediante fuerza bruta sea más asequible, como se muestra en la figura 7.

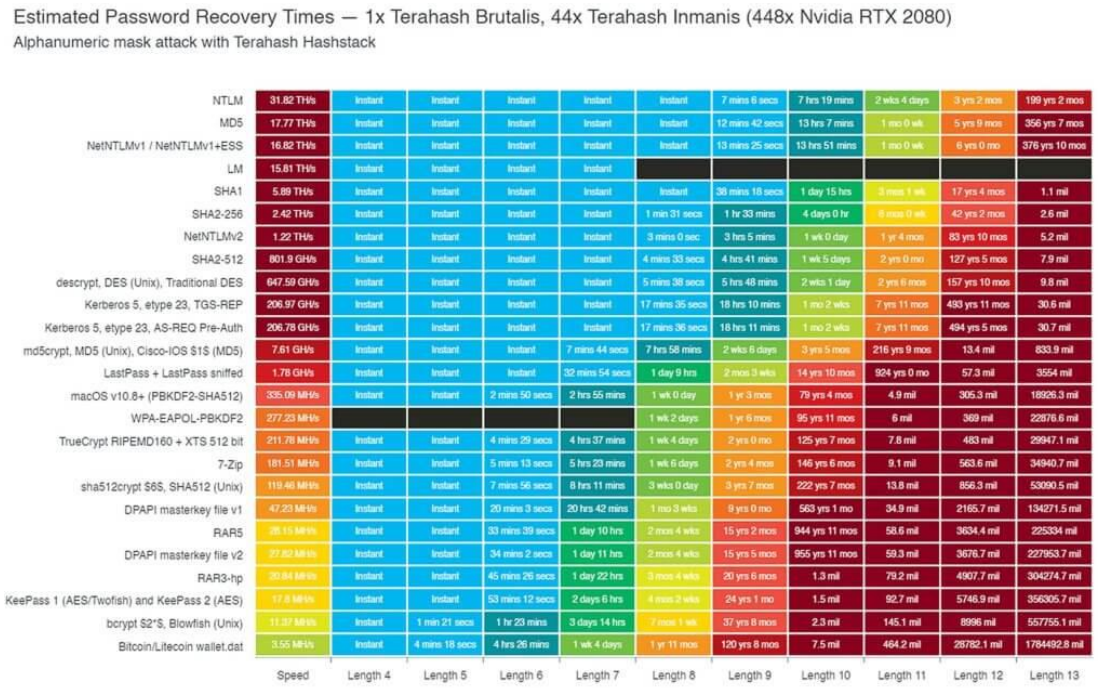

Figura 7 Tiempo necesario para adivinar contraseñas según el Hash usado; Fuente:(*IThemes Security Pro Feature Spotlight – Password Requirements*, 2021)

Como se observa en la figura 7, la práctica del uso de doble factor de autenticación está siendo introducida en su uso diario; el segundo factor de autenticación también se conoce como “algo que tengo”, ya sea en forma de una matriz de números, un generador de códigos “aleatorios”, tarjetas o llaves criptográficas.

Si bien, en la teoría, el envío de un correo o un mensaje de texto al celular se puede considerar un segundo factor de autenticación, existen casos en los cuales se han interceptado los mensajes de texto (KeyserSosa, 2018).

Capítulo 3. Marco Metodológico

3.1 Tipo de Investigación

Debido a que el objetivo de esta investigación no es generar un algoritmo nuevo de encriptación, sino todo lo contrario, usar algoritmos existentes, validados y respaldados tanto por la comunidad como por las agencias de estándares como la NIST y el ENISA, esta investigación es de tipo evaluativa. Lo anterior se respalda en lo afirmado por Tomás Escudero (Escudero, 2016) al señalar que este tipo de investigación añade diversidad y flexibilidad, las cuales son cualidades útiles en el desarrollo de software.

3.2 Alcance Investigativo

El tipo de investigación y desarrollo en el cual se desenvuelve este trabajo y el proyecto que se propone, se presta para establecer varios tipos de alcances, siendo los principales el Exploratorio y el Descriptivo.

3.2.1 Explicativo

Usando la definición esbozada en el libro Metodología de la Investigación (Hernández Sampieri et al., 2014) sobre el alcance explicativo, se puede afirmar que el proyecto aquí propuesto se adapta a un estudio de esta naturaleza, ya que explica de forma detallada cómo se realiza el aseguramiento de la información mientras está en reposo o tránsito, asimismo cómo y cuáles técnicas de autenticación se usarán para determinar que el receptor de la información sea el destinatario elegido.

3.2.2 Descriptivo

De igual forma, basado en el texto supra citado Metodología de la Investigación (Hernández Sampieri et al., 2014), este trabajo se apega a los alcances de un estudio descriptivo, ya que pretende comparar, analizar y medir cuáles de los algoritmos existentes de encriptación funcionarían mejor para el uso de la solución planteada.

3.3 Enfoque

Para la investigación desarrollada se empleará un enfoque alternativo. Esta decisión se sustenta, como referencia, en las ideas expuestas en los siguientes textos: “Investigación en Informática: El enfoque alternativo” (Naranjo-Zeledón et al., 2020) y “La dicotomía cuantitativo/cualitativo, falsos dilemas en la investigación social” (Chavarría-González, M., 2011). No obstante, y como este enfoque lo permite, se utilizan técnicas de investigación de los enfoques cuantitativos y cualitativos. Se realizan algunas pruebas de campo, como por ejemplo unas evaluaciones comparativas de los algoritmos, con el propósito de definir qué tan rápidos son, en comparación con otros.

3.4 Diseño

En “On theory development in design science research: anatomy of a research project” (Kuechler & Vaishnavi, 2008) se expone la ciencia del diseño como una herramienta útil en la propuesta de proyectos en el área de la informática; añadido a eso publicaciones como las de Contreras et al. (s/f) y Robles Sandoval (2019) demuestran la factibilidad de este diseño para investigación de índole similar. El modelo iterativo que detalla Robles Sandoval (2019) en su publicación se ajusta al modelo de prototipo sugerido por Floyd (1984) en el cual se le da énfasis a la exploración y la experimentación en cada una de las interacciones de forma similar a la ciencia del diseño.

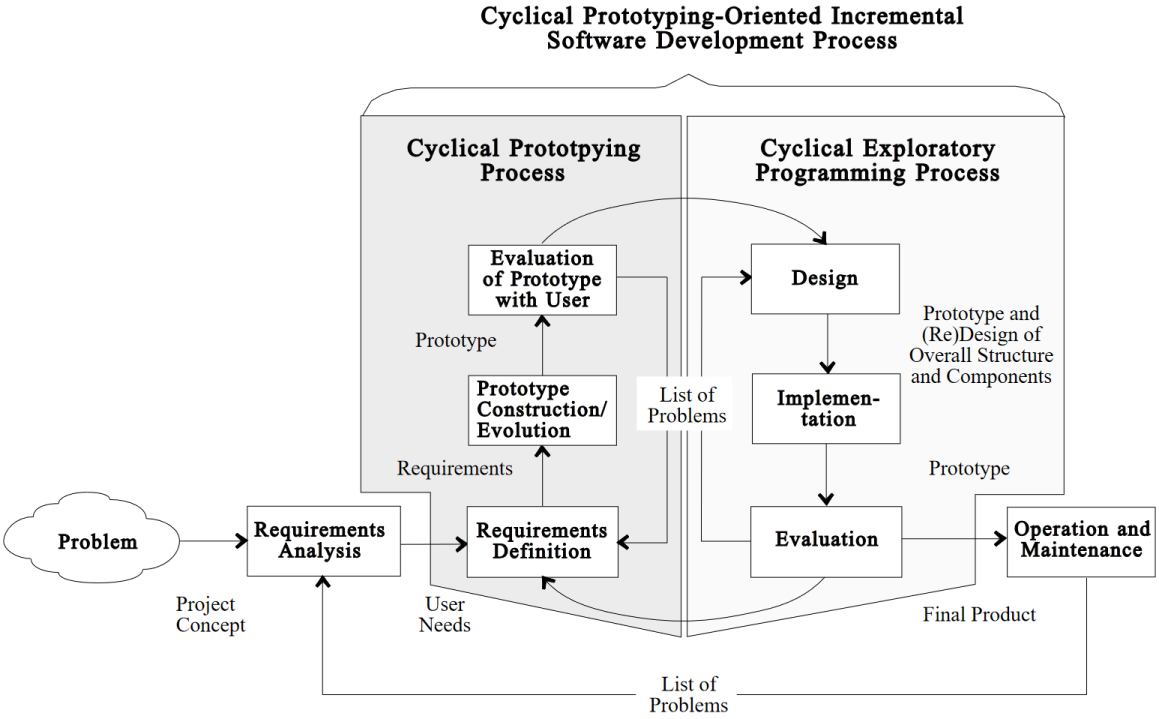

Figura 8 Diseño cíclico de prototipo; Tomado de (Pomberger et al., 1998)

Como se observa en la figura 8, el proceso cíclico de prototipos tiene similitudes con el modelo usado por Robles Sandoval (2019) como se muestra en la figura 9.

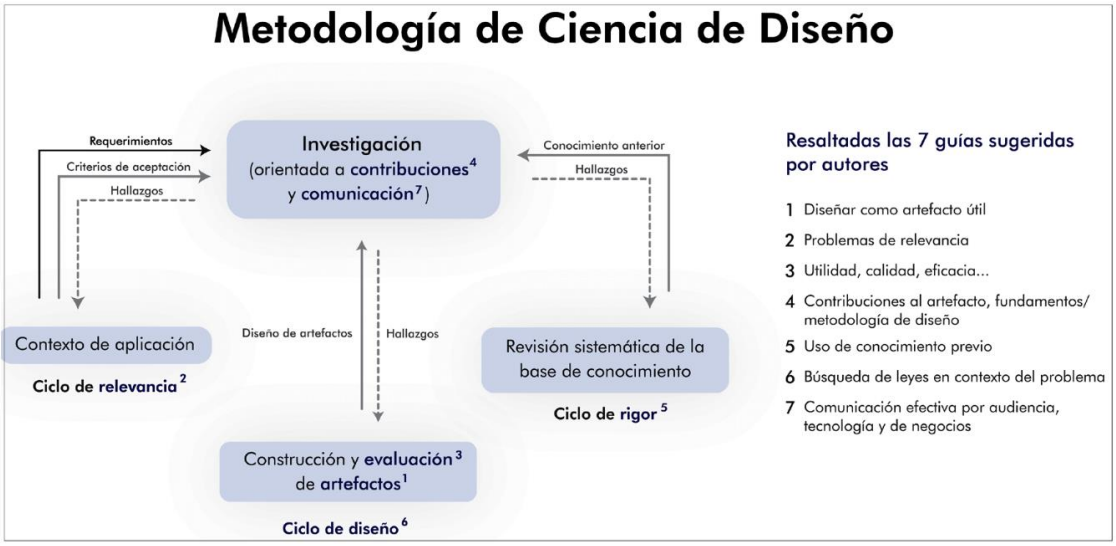

Figura 9 Ciclos de la ciencia del diseño; Tomado de (Robles-Sandoval, 2019)

3.5 Población y Muestreo

Debido al tipo de investigación no se requiere realizar ningún tipo de muestreo, ni definir una población.

3.6 Instrumentos de Recolección de Datos

En línea con los objetivos de esta investigación, se usan algoritmos ya existentes y avalados por la industria y sus referentes, como ENISA y NIST. Por lo tanto, se conoce de antemano una lista finita de estos algoritmos. De todos modos, se utilizan pruebas evaluativas y comparativas para evaluar cuál de estos tiene el mejor rendimiento para el caso de uso en el cual se emplearán, mediante la aplicación de pruebas existentes como las sugeridas en el Performance Analysis of Data Encryption Algorthms (s/f), (*/docs/man1.1.1/man1/openssl-speed.html*, s/f). De ser necesario, se codificará y documentará de forma apropiada cualquier otra prueba que se realice para medir la eficiencia de los algoritmos seleccionados. De igual forma, se evalúan de forma científica, medida e imparcial las herramientas o dependencias que serán utilizadas en el proyecto, apoyado en la metodología de ciencia del diseño y ciclo de prototipo descritos en el inciso 3.4.

3.7 Técnicas de Análisis de Información

Al tratarse de un desarrollo de software se disponen varias técnicas de análisis de datos tradicionales en la informática, en especial, al existir un flujo constante de datos, se emplean: casos de uso, flujo de datos y Diagramas de UML de tipo Diagramas de secuencias y de actividades. Se utiliza arquitectura para demostrar y argumentar el uso de los recursos proporcionados por Google Cloud.

Capítulo 4. Propuesta de Solución

4.1 Criptografía

4.1.1 Cifrado de bloques

En “Cryptography and network security: principles and practice” (Stallings, 2017) se define como un cifrado de bloque aquel cuya entrada es un solo bloque de un tamaño *n* y produce un bloque cifrado del mismo tamaño *n*; el tamaño de los bloques está predeterminado según el algoritmo de cifrado escogido, se encuentran bloques de 64,128 y 256 bits. Los cifrados de bloque se utilizan con llaves simétricas, el tamaño de esta llave también es determinado por el algoritmo a usar, los cifrados de bloques tienen una aplicación más amplia en comparación con los cifrados de flujo, en aplicaciones de aseguramiento de datos en reposo.

Al ser uno de los métodos de encriptación más comunes y utilizados en la informática, existen muchas implementaciones de cifrados de bloque, para efectos de este trabajo, se centrará en aquellos que se consideren actuales, esto quiere decir que, a la fecha de la redacción de este documento, no tengan vulnerabilidades conocidas. También, otro de los factores a sopesar, es que estos algoritmos tengan implementaciones sólidas, ya sea por proveedores de buena reputación, o implementaciones Open Source con una comunidad activa. En este punto también se valora si el proyecto es parte de software u organizaciones con buena reputación. A partir de los criterios anteriores, se seleccionaron los algoritmos que se tendrán en cuenta para el uso, por defecto, de la aplicación. El diseño, que se explica más adelante, deja la posibilidad de intercambiar la implementación por cualquier otra, pero, por el alcance del proyecto solo se implementa el algoritmo seleccionado, al final de esta sección. Los candidatos son: AES, Serpent, y Camellia. Se emplean varios rubros para generar una calificación y la propuesta que obtuvo la nota más alta, de 100 puntos posibles, fue la elegida para la implementación por defecto.

4.1.1.1 Rubros de calificación.

4.1.1.1.1 Aceptación por entes reguladores

Este sin duda alguna es el rubro con mayor peso a la hora de calificar, ya que se toman en consideración las recomendaciones de seguridad de los gobiernos, si bien se les da más prioridad a entes como ENISA y NIST. Los sugeridos por otros

entes paralelos, pero igualmente aceptados, serán tomados en cuenta; ahora bien, si existe una diferenciación entre los entes como ENISA y NIST, estos confieren más puntos que cualquier otra entidad. Los puntos otorgados en este aspecto son 45, como se mencionó es el elemento que más puntos otorga. Se determinó que esta cifra no superara los 50 puntos, ya que no se quiere descartar, a priori, algoritmos que pueden ser mejores en las otras categorías, que los seleccionados por estas agencias, debido a que puede existir la posibilidad de cadencias en la actuación de las recomendaciones y el uso de la industria.

4.1.1.1.2 Aceptación por la industria.

Esta es la segunda categoría con más peso, pues asigna 35 puntos. Aquí se evalúa el uso en la industria, y se examina si existen productos que empleen los algoritmos seleccionados, debido a la existencia de una gran variedad de soluciones y organizaciones. En este rubro se hace una distinción con respecto a quiénes utilizan el producto, por ejemplo, se dan más puntos a productos como el *Kernel* de Linux, que a un producto comercial de una compañía. El uso y aceptación en los principales proveedores de la nube también da un peso mayor. En caso de implementaciones Open Source se analiza la comunidad y actividad de repositorio de código, para determinar qué tan activa es.

4.1.1.1.3 Velocidad.

Este rubro otorga 20 puntos. Para probar la velocidad de los candidatos se utiliza de preferencia la implementación estándar del algoritmo, o sea, la que está incluida en el lenguaje de programación. En caso de que este no provea una implementación estándar, se utiliza una librería Open Source, la cual tenga un lanzamiento reciente y actividad en el repositorio reciente, entre otros. El racionamiento específico de la selección de la implantación es explicado en detalle en la sección de cada algoritmo. El set de pruebas es el mismo para todos los candidatos, el cual consiste en encriptar 2000 veces el libro Don Quijote de La Mancha en idioma español, en formato de texto plano utilizando la codificación “UTF-8” obtenido en el sitio https://gutenberg.org/ebooks/2000 y su firma MD5 es a4ebeeed524ddb934271d21a99de9802. Se emplea la misma llave para todos: 570dd9eaf172f4b4f1ddb15149b57e09b31b658277f3946b382fb7ba35179c6c, la cual

es resultado de algoritmo de hash basado en SHA3-256 de la cadena de caracteres “cenfotec”, por lo tanto, se utilizan bloques de 256 bits. Para estas pruebas se usa el modo de operación ECB[1].

El hardware para la prueba es virtualizado por Google Cloud, la instancia a utilizar es una c2-standard-4, la cual, según la documentación de Google, cuenta con 32 CPU virtuales y 32 GB de memoria RAM, y un disco duro de 375 GB usando una interfaz NVMe para obtener la mayor velocidad en lectura y escritura. Un CPU virtual se implementa como un "Hyper-Thread" en el Host físico, el CPU físico usado por Google, según la documentación, es un Intel Xeon, de la familia *Cascade Lake* de segunda generación, con velocidades hasta 3.8 GHz. La escogencia de este tipo de instancia se basa en que están optimizadas para los cálculos basados en CPU, y emplean una arquitectura que utiliza características como el acceso a memoria no uniforme (NUMA) para un rendimiento uniforme, confiablemente óptimo. Se genera una imagen de Debian-10, la cual es configurada una vez con todas las herramientas y utilidades que se ocupen para las pruebas de rendimiento. Se crean instancias nuevas, con esta imagen base, con el fin de evitar cualquier cambio a nivel del S.O. Los puntos se confieren de la siguiente forma: para el primer lugar se asignan once puntos, al segundo seis, y los restantes tres al último candidato.

4.1.1.1.4 Tabla final de puntajes

Tabla 2: Total de puntajes

Rubro	Puntuación
Aceptación por entes reguladores	45
Aceptación por industria	35
Rendimiento	20
Total	100

Fuente: Elaboración Propia

[1] ECB significa Electronic Code Book y se refiere a un modo de encadenamiento de bloques donde cada bloque se encripta y procesa independientemente de los demás.

4.1.1.2 AES

Sin duda alguna, Advanced Encryption Standard (AES) o Rijndael es el algoritmo de cifrado de bloque más utilizado en la actualidad. Este algoritmo fue el resultado del concurso Advanced Encryption Standard patrocinado por el Gobierno de Estados Unidos como reemplazo del DES, se publicó como estándar federal o FIPS, por sus siglas en inglés, a finales de noviembre del 2001, como FIPS-197 (FIPS 197, Advanced Encryption Standard (AES), n.d.).

4.1.1.2.1 Aceptación por entes reguladores

Autoridades como NIST y ENISA lo facultan como uno de los aceptados al usar bloques de 128, 192 y 256 para Rijndael y de 128 para AES. Además, es parte de la ISO/IEC 18033-3:2010.

4.1.1.2.2 Aceptación por la industria

AES es el estándar por defecto de la industria en la encriptación por bloques, ya que la mayoría de los lenguajes de programación de alto nivel tienen implementaciones estándares. Además, es el protocolo más empleado para la creación de certificados tipo SSL, así como el más utilizado por los proveedores de computación en la nube para la encriptación de la información en reposo, tanto en el nivel empresarial como para el hogar y la oficina, usado en la tecnología Bitlocker de Microsoft (Dansimp, s/f-a) y LUKS en Linux (Fruhwirth, 2018).

4.1.1.2.3 Rendimiento

A partir de las pruebas generadas para este proyecto y utilizar las herramientas incluidas en el kit de desarrollo de golang, se obtuvo un promedio de 56.91G operaciones por segundo.

4.1.1.2.4 Calificación final

Tabla 3: Calificación final AES

Rubro	Puntuación
Aceptación por entes reguladores	45
Aceptación por la industria	35
Rendimiento	3
Total	83

Fuente: Propia

4.1.1.3 Serpent

Fue uno de los finalistas del concurso en donde se eligió a Rijndael; su funcionamiento interior es muy similar a Rijndael, pero con una seguridad más alta (Nechvatal et al., 2001). Al final NIST escogió AES debido al poco uso de memoria y una mayor simplicidad en el diseño interior (Nechvatal et al., 2001).

4.1.1.3.1 Aceptación por entes reguladores

Serpent no está reconocido como un algoritmo de cifrado en NIST o ENISA, o cualquier otra agencia de gobiernos, aunque sí se menciona en documentos de ECRYPT(ECRYPT CSA, 2018). Tiene como objetivo fortalecer la excelencia europea en el área de la criptología para lograr una integración y estructuración duraderas de la comunidad criptográfica europea, al involucrar a la academia, la industria, las fuerzas del orden y las agencias de defensa.

4.1.1.3.2 Aceptación por la industria

No estar reconocido por los entes reguladores, no descalifica inmediatamente un algoritmo de cifrado para el uso de la industria, tal vez si para implementaciones para entes gubernamentales, que requieran adherirse a dichas normas. Es posible usar Serpent para encriptación de los datos en reposo en sistemas operativos Linux, empleando la tecnología LUKS (Arne Osvik, 2011/2002), además de poder utilizarse en conexiones SSH (Bellare et al., 2006) entre otros fines. Lamentablemente, al no ser un estándar, en muchas ocasiones hay que recurrir a librerías de terceros para poder aplicar Serpent en los lenguajes de programación.

4.1.1.3.3 Rendimiento

Tomando en cuenta las pruebas generadas para este proyecto y al utilizar las herramientas incluidas en el kit de desarrollo de golang, se obtuvo un promedio de 104.03G operaciones por segundo.

4.1.1.3.4 Calificación final

Tabla 4 Resultados Serpent

Rubro	Puntuación

Aceptación por entes reguladores	25
Aceptación por entes	25
Rendimiento	11
Total	61

Fuente: Propia

4.1.1.4 Camellia

Camellia fue desarrollado por Mitsubishi Electric and Nippon Telegraph and Telephone.(Aoki et al., s/f). Al igual que Rijndael y Serpent usa un conjunto de “S-Boxes” para la encriptación de los bloques.

4.1.1.4.1 Aceptación por entes reguladores

Camellia, es parte los algoritmos recomendados por ENISA (European Network and Information Security Agency, 2013) y es parte del estándar ISO/IEC 18033-3:2010 (ISO/IEC, 2010, p. 3).

4.1.1.4.2 Aceptación por la industria

Camellia es parte de la suite de cifrados posible de TLS desde el RFC 4132 a mediados del 2005. A diferencia de AES, Camellia está patentado, lo cual produce que su uso sea más restrictivo en software comercial u Open Source, y más complicado. De igual forma, proyectos como OpenSSL (Kanno & Kanda, 2011) incluyen este algoritmo.

4.1.1.4.3 Rendimiento

A partir de las pruebas generadas para este proyecto y al utilizar las herramientas incluidas en el kit de desarrollo de golang, se obtuvo un promedio de 87.63G operaciones por segundo.

4.1.1.4.4 Calificación final

Tabla 4: Resultados Camellia

Rubro	Puntuación

Aceptación por entes reguladores	45
Aceptación por industria	25
Rendimiento	6
Total	76

Fuente: Propia

4.1.1.5 Conclusiones

Basado en la metodología y pruebas que se describieron anteriormente, se elige AES como la implementación estándar para este proyecto. Lamentablemente la falta de respaldo por entidades regulatorias hace que Serpent no sea una opción viable puesto que podría limitar la implementación en organizaciones altamente reguladas. Camille es una muy buena opción, pero esta se descarta debido a su falta de adopción por parte de la industria, probablemente debido al factor de la patente que existe sobre este algoritmo (Nippon Telegraph and Telephone Corporation, 2001), o al hecho de que no sea aceptado por el NIST, pero esto es solamente especulativo. Rijndael tiene también a su favor que cuenta con el respaldo de múltiples organizaciones como ISO, NIST y ENISA, así como, tener una implementación estándar integrada en las librerías estándar del lenguaje de programación, lo que evita generar un criptoanálisis de las librerías de terceros para asegurarse que la calidad de estas implementaciones sea la adecuada.

Tabla 5: Resultados Finales

Rubros	AES	Camellia	Serpent
Aceptación por entes reguladores	45	45	25
Aceptación por industria	35	25	25
Rendimiento	3	6	11
Total	83	76	61

Fuente: Propia

4.1.2 Modo de operación[2]

El uso de AES por sí solo no asegura la calidad y seguridad de la información encriptada, por lo cual es común y recomendado que se utilice un modo de operación para añadir seguridad, confidencialidad y autenticidad a la información que se encripta, y existen múltiples modos de operación que son compatibles y aceptados, ya sea por NIST, ENISA o ISO. Para este proyecto, se consideran los modos que ofrecen confidencialidad y autenticación al mismo tiempo, por lo tanto, se descartan los modos ECB, CBC, OFB, CFB, CTR, XTS-AES que solo proveen confidencialidad y CMAC que solo provee autenticación. Dado lo anterior, se concentrará en los modos CCM, GCM, KW, KWP, and TKW que proveen ambas características requeridas.

El modo CCM es un modo de operación cuyo algoritmo está diseñado para proporcionar autenticación y confidencialidad. Solo se define para cifrados de bloque con una longitud de bloque de 128 bits (Dworkin, s/f). Este modo es el empleado en el estándar IEEE 802.11i (Bae et al., 2005) para su uso en IPsec (Frankel et al., 2003) TLS 1.2 (Dierks & Rescorla, 2008) y Bluetooth 4.0 (Bluetooth SIG, Inc, 2010). Presenta ciertos inconvenientes a la hora de implementarlo y aplicarlo, entre ellos que el “nonce”, por definición, solo puede ser utilizado una vez, por lo cual hay que tener cuidado con su empleo. El *nonce* es un número seudo aleatorio de solo un uso, que sirve, en especial, para dar aleatoriedad y puede depender de marcas de tiempo. Es útil, generalmente, para evitar ciertos tipos de ataques.

El GCM (Galois Counter Mode) es un modo de operación ampliamente adoptado por su desempeño, debido a que se pueden obtener altas velocidades utilizando hardware sencillo (Wang, 2006); su diseño también permite un uso más eficiente de procesamiento en paralelo, ya sea en software o mediante hardware. Similar a CCM, GCM es admitido como parte de los protocolos TLS 1.2 (Dierks & Rescorla, 2008) y 1.3 (Rescorla, 2018), múltiples estándares de la IEEE, como el 802.11ad (*IEEE Standard for Information technology--Telecommunications and information exchange between systems--Local and metropolitan area networks--Specific requirements-Part 11*, s/f).

---

[2] El modo de operación es la manera en que la encriptación de un bloque influye en los restantes bloques del mensaje o archivo a cifrar.

Se selecciona el modo GCM para este proyecto, ya que proporciona confidencialidad y autenticación de los mensajes a una muy buena velocidad y se descartan para la encriptación de los datos, los modos KW y KWP, ya que en ambos casos su uso es más específico para el aseguramiento de llaves criptográficas. Si bien, CCM cumple con las mismas características de seguridad y autenticación de la información, se escogió GCM por su velocidad y posibilidades de paralelización, así como productos líderes de la industria como el proyecto OpenSSL que lo habilitan en su kit de encriptación por defecto para el protocolo TLS 1.3, CCM está deshabilitado por defecto. Una de las diferencias más importantes a la hora de la autenticación en el modo CCM es que la etiqueta de autenticación se deriva del texto plano; y en el modo GCM, la etiqueta de autenticación se obtiene del texto cifrado.

4.1.3 Derivación de llave.

En la criptografía, lo más común es usar contraseñas en lugar de los bits en crudo que conforman las llaves, ya que son más fáciles de recordar y escribir; la conversión de la cadena de caracteres a bits que puedan ser usados por los algoritmos de encriptación como llaves es delegado al implementador de la encriptación. Una práctica común es usar funciones de “hash” para convertir estas contraseñas a llaves de un tamaño determinado, que pueden ser utilizados por los algoritmos. Esta práctica se considera como insegura, ya que son susceptibles a ataques de fuerza bruta, diccionario entre otros, algunos de estos ataques se pueden solucionar utilizando bits seudo aleatorios llamados “salt” los cuales se le agregan a la cadena de caracteres que se quiera usar como parámetro de entrada a la función "hash”.

La derivación de llaves es un proceso diseñado para que sea computacionalmente costoso adivinar las contraseñas mediante fuerza bruta, ya que usualmente, como parámetros de la función, se espera un “salt”, además de un tiempo de CPU y memoria RAM, lo cual produce un incremento del costo computacional, o la creación de circuitos integrados de aplicación específica, los cuales son generados con el único propósito de realizar una función, por ejemplo calcular el resultado de una función hash SHA256. Argon2 fue el ganador de una competencia realizada de forma similar al AES, llevada a cabo desde el 2013 hasta el 2015, con el fin de

estandarizar las funciones de derivación de llave. Argon2 fue diseñado por Lex Biryukov, Daniel Dinu, and Dmitry Khovratovich de la Universidad de Luxembourg. (Biryukov et al., 2016). Existen pequeñas variaciones de Argon2, diseñadas para mitigar posibles vectores de ataque; para el proyecto se utiliza Argon2di, la cual provee una buena relación entre seguridad contra ataques usando CIAS o ASIC en inglés, y *side-channel attack.* Argon2 se encuentra en un estado de borrador para un RFC, actualmente cuenta con 13 versiones, siendo la última versión de marzo del 2021.

4.2 Almacenamiento en Reposo.

Una vez que se ha definido cómo se van a asegurar los datos usando la criptografía aplicada, se determina cómo serán almacenados los documentos una vez encriptados. Tal y como se discutió en capítulos anteriores, la plataforma usada por defecto de Google Cloud, para el almacenamiento de los documentos encriptados es Cloud Storage. Se descarta el uso de discos duros internos en los servidores, ya que limita la facilidad de la escalabilidad, al tener que replicar los documentos entre los servidores y, por otra parte, también los administradores del proyecto tendrían que estar encargados de los respaldos de los datos. El uso de Cloud Storage transfiere los riesgos mencionados al proveedor de la nube, en este caso Google y utilizar este tipo de almacenamiento también permite poder escalar de forma vertical más fácil, ya que todos los servidores nuevos apuntarán al mismo repositorio, el cual, por su diseño en la práctica, no tiene un límite de tamaño y las velocidades de acceso son bastante rápidas, con la configuración adecuada.

Los documentos serán almacenados de tal forma que no sea posible identificarlos por nombre o extensión, para lograr dicho objetivo, se emplea la función SHA-3, usando como parámetro de entrada el archivo encriptado. Los archivos serán almacenados en un árbol de solo dos niveles, el primer nivel está constituido por los primeros tres caracteres de la firma SHA-3, el resto de los caracteres formarán el nombre del archivo.

4.3 Base de Datos

Se descartó el uso de bases de datos tradicionales y las no relacionales o NoSQL, en su lugar se optará por un motor de base de datos llave-valor. Al utilizar las

llaves como rutas, se puede crear un sistema de archivos virtual, por el cual se mapea el archivo en Cloud Storage; además se pueden guardar los metadatos de los archivos, como el nombre original, creador, fecha de subida y el tipo del archivo. Esta información será dispuesta para mostrar en la interfaz de usuario los datos de los documentos almacenados y reconstruir los archivos para su descarga o visualización. La comunicación entre *etcd* y el servidor de encriptación se realizará mediante HTTPS y la autenticación se llevará a cabo a través de certificados digitales, por lo cual se asegura que la información no pueda ser alterada en tránsito.

Existirán tres tipos de información guardada en *etcd*: metadatos de los documentos, configuración del sistema y finalmente, información de los usuarios.

4.3.1 Metadatos de los documentos

Se almacena la información de los archivos que sea estrictamente necesaria para desplegar o reconstruir el archivo para su descarga. El sistema en ningún momento modifica el archivo, por lo cual se espera que cualquier metadato propio del formato original del documento se conserve durante y después de su almacenamiento. Para los archivos se guarda la siguiente información:

4.3.1.1 Nombre

Nombre del archivo como fue enviado por el navegador al servidor. Se aceptarán todos los caracteres válidos excepto “/” y el byte “\null”. Si bien el sistema no valida el largo del nombre del archivo, es posible que si lo haga el navegador o el sistema operativo del usuario.

4.3.1.2 Creador

Usuario que subió por primera vez el archivo, se hace referencia a la ruta en donde se encuentra la información del usuario. Este usuario también será el dueño del documento, durante la permanencia de este en el sistema.

4.3.1.3 Tipo

Tipo de medio o MIME type según el estándar IANA. El servidor extraerá el tipo medio utilizando la extensión del archivo subido o mediante el identificador de la

cabecera del archivo (“magic number”). En ningún caso, se hará una validación exhaustiva del tipo de archivo, se confiará en el resultado de la librería seleccionada para realizar esta función.

4.3.1.4 Creación

Marca temporal de creación del archivo en el sistema. Se usará la zona horaria UTC. El cliente del servidor será el encargado de convertir la marca de tiempo, a la zona horaria del usuario.

4.3.1.5 Modificación

Marca temporal de la última modificación del archivo en el sistema. Se usará la zona horaria UTC. El cliente del servidor será el encargado de convertir la marca de tiempo, a la zona horaria del usuario.

4.3.1.6 SHA-3

Resultado de la firma SHA-3 aplicada a la última versión del archivo después de la encriptación. Este valor es la llave para encontrar el contenido del archivo en el sistema de almacenamiento, por defecto Cloud Storage. Este campo es actualizado cuando se suba una nueva versión o se modifique el archivo.

4.3.1.7 Historial

Lista de firmas SHA-3 y marcas temporales, este campo se utiliza para llevar y desplegar un historial de las versiones de los archivos, el sistema estará en la capacidad de usar esta información para recobrar versiones antiguas de los documentos.

4.3.1.8 Llave de encriptación

Se guarda la llave de encriptación que se generó seudo aleatoriamente. Se usa criptografía de llave pública/privada para asegurase que solo el usuario dueño del archivo pueda ver la llave usada en AES.

4.3.1.8 Acceso a Usuarios

En una lista se almacena la información de los usuarios y el resultado de la llave de encriptación utilizando las llaves públicas del usuario al cual se le concedió acceso al archivo. Este proceso está más detallado en futuras secciones.

4.3.2 Metadatos de los usuarios

La información de los usuarios, al igual que la de los archivos será la estrictamente necesaria.

4.3.2.1 Usuario

Nombre del usuario.

4.3.2.2 Correo electrónico

Correo electrónico del usuario, se usará como parte de la autenticación.

4.3.2.3 Contraseña

Contraseña del usuario, se guarda empleando Argon2di, se compara a la hora del inicio de sesión.

4.3.2.4 Llaves pública y privada

Se guarda la información de las llaves públicas y privadas que se generen, estas llaves usarán el formato Curva Elíptica de Diffie Hellman con una llave de 256 bits, con base en las recomendaciones de NIST (Brown, s/f); estas llaves serán utilizadas para encriptar y desencriptar las llaves de encriptación del archivo.

4.3.2.5 Llave MFA

Campo necesario para poder utilizar HOPT.

4.3.2.6 Estado

Estado actual del usuario.

4.4 Procesos.

En este apartado se definen las especificaciones de cómo funcionarán los principales procesos del sistema diseñado. Es importante aclarar que no todos los procesos serán descritos en esta sección, solo los que se conocieron como base para el funcionamiento, y que estén dentro del alcance del proyecto.

4.4.1 Creación de Usuario

Cuando el administrador crea un usuario, se generará una carpeta virtual dentro del folder “usuarios”, la nueva carpeta tendrá como nombre el correo electrónico del usuario. La información del usuario, como se describió en apartados anteriores será almacenada dentro de un archivo que se llamará “info”. Ya almacenada la información en dicho archivo, se generan un par de llaves criptográficas empleando el algoritmo de Curva Elíptica de Diffie Hellman o (ECDH), los parámetros para el uso y generación de llaves serán los mismos que se definen por NIST (Regenscheid, 2019). Las llaves se guardarán en la misma carpeta donde se almacenó la información del usuario. Se originará una contraseña aleatoria, por lo tanto, el usuario estará marcado para tener que cambiar la contraseña y registrarse al multifactor de autenticación la primera vez que ingrese. No será posible acceder o utilizar el sistema si este paso no es llevado a cabo.

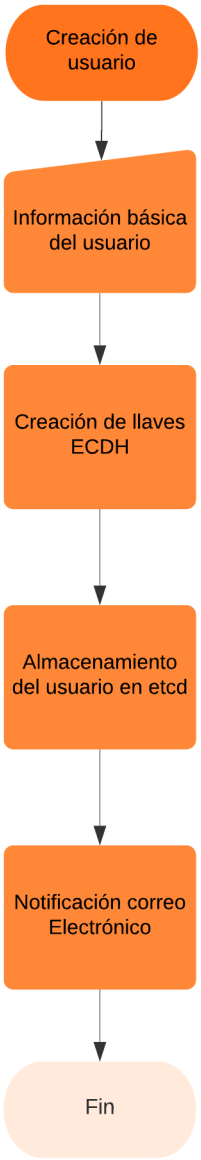

Figura 10 Diagrama de proceso: creación de Usuario

Fuente: confección propia

4.4.2 Almacenamiento inicial de archivos

La lógica y almacenamiento de los archivos tendrá como inspiración LUKS de Linux (REF) con algunas diferencias, la mayor de estas es que no se usarán técnicas anti forenses, ya que estas están implementadas con el fin de asegurar la eliminación de la información en discos.

El usuario, mediante la aplicación web, enviará un archivo al servidor; una vez en el servidor, primero se extraen los metadatos como el nombre original y el tipo, de

seguido se generará una llave de encriptación de forma pseudoaleatoria de 256 bits de largo, con esta llave se procede a encriptar el archivo usando el algoritmo AES, en el modo GCM; el resultado de este será pasado por el algoritmo de firma SHA-3. Se subirá el archivo encriptado al sistema de almacenamiento configurado, por defecto Cloud Storage, usando como folder raíz los primeros 3 caracteres de la firma SHA-3. El nombre del archivo con que se almacenará estará formado por el resto de los caracteres de la firma SHA-3. Cuando el proveedor de almacenamiento confirme la recepción del archivo, mediante el uso de la llave pública del usuario que envió el archivo, se encriptará la llave AES. El resultado será almacenado con el resto de los metadatos en *etcd*.

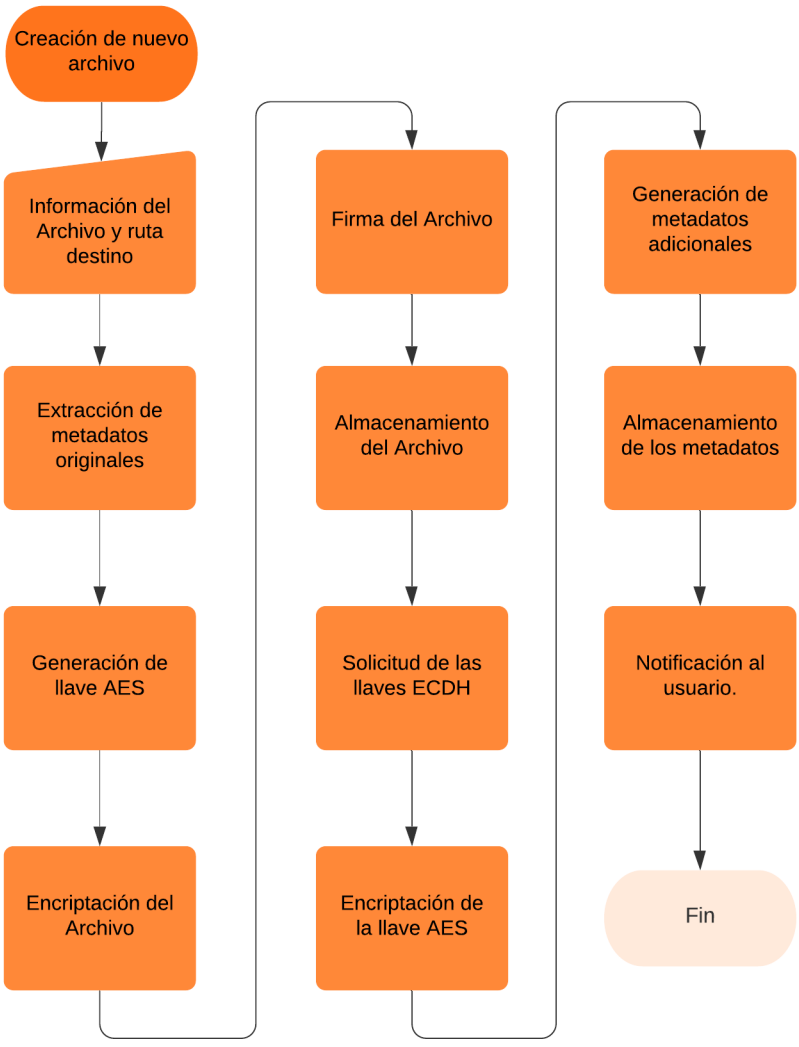

Figura 11 Diagrama de proceso: Creación de un archivo

Fuente: confección propia

4.4.3 Lectura de archivos almacenados

El usuario, mediante la aplicación web, solicita al servidor poder visualizar o descargar un archivo en una ruta que él mismo seleccionó. El servidor buscará los metadatos del archivo, una vez localizado el archivo de metadatos, el servidor verificará que el usuario tenga acceso, al comprobar que en los metadatos exista un registro de la encriptación de la llave AES, usando la llave pública del usuario. El servidor descarga a memoria el archivo desde el sistema de almacenamiento, empleando la llave privada del usuario que solicita el documento, desencriptará la llave AES, y procederá a recuperar la información original del archivo. Una vez recuperado el archivo, este será enviado al usuario mediante el canal seleccionado. Se utilizarán los metadatos almacenados en *etcd* para recuperar cualquier otro tipo de información.

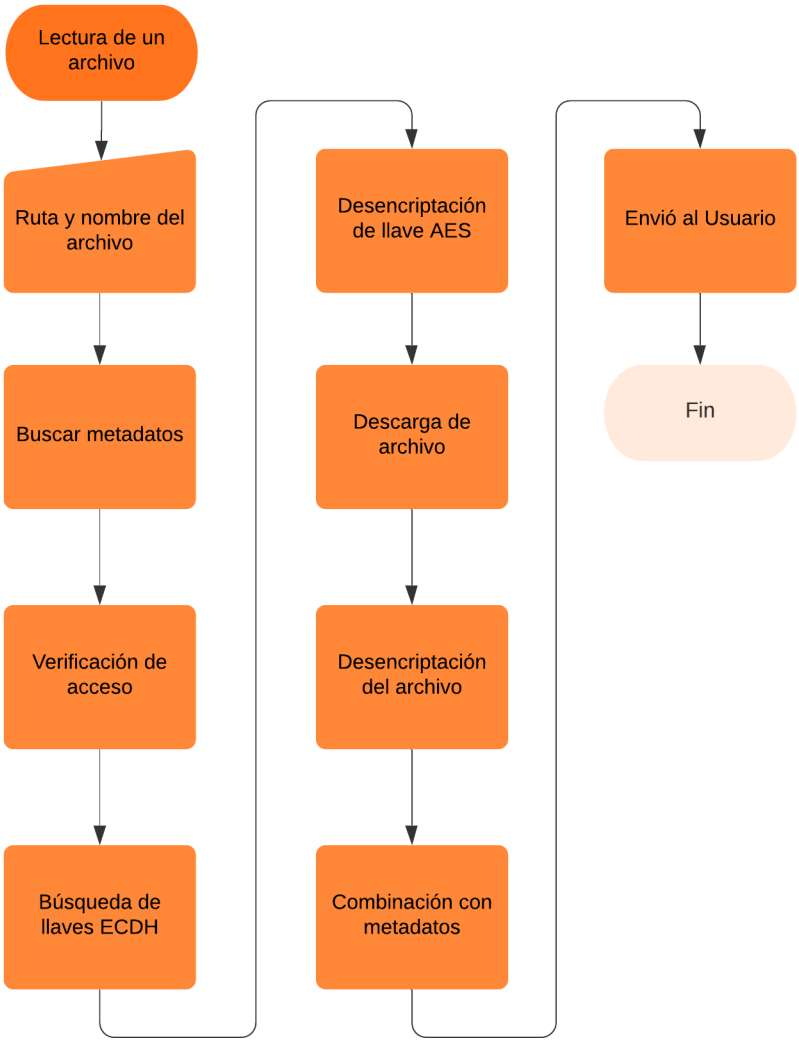

Figura 12 Diagrama de proceso: Lectura de un Archivo

Fuente: confección propia

4.4.4 Compartición de archivos.

Mediante la interfaz web, el usuario selecciona el archivo y el usuario con el cual quiere compartir la información, para este paso se pedirá que ingrese una contraseña de solo un uso, para verificar la autenticidad de la operación. Si el usuario está registrado en el sistema se buscará la información en *etcd*. Localizada la información del receptor, se buscan los metadatos del archivo seleccionado, usando la llave privada del emisor. Seguidamente, se desencripta la llave AES,

inmediatamente se vuelve a encriptar mediante el uso de la llave pública del receptor y el resultado de esta operación será almacenado en los metadatos del archivo. Se enviará un correo electrónico al receptor para informarle sobre el éxito de la operación. En caso de que el usuario no exista dentro del sistema, este genera un usuario nuevo y sigue el proceso normal. El dueño del documento podrá revocar el acceso al archivo en cualquier momento, para hacerlo el sistema solo borra el registro de los metadatos del documento. No será posible para un usuario compartir un documento de un tercero, esto quiere decir que no se le permitirá compartir un documento del cual no sea el dueño.

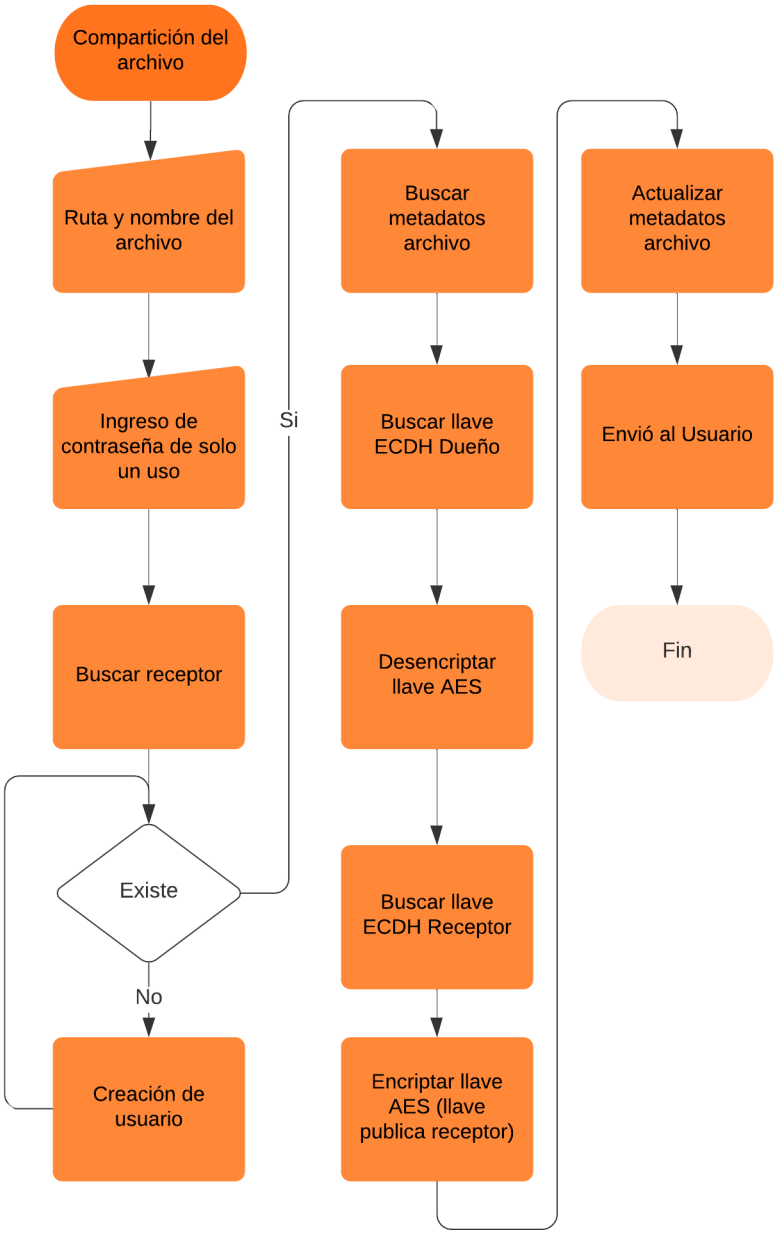

Figura 13 Diagrama de proceso: Compartición de un Archivo

Fuente: confección propia

4.4.4 Eliminación de un archivo

Una vez ingresada la contraseña de solo un uso para verificar la solicitud, se comprobará que el solicitante sea dueño del recurso. Una vez aprobada esta confrontación, se buscarán los metadatos, y se le solicitará al proveedor de almacenamiento la eliminación del documento. Cuando se confirme que esta acción

fue realizada con éxito, si el documento estuviese compartido, se enviará una notificación de la eliminación del archivo a todos los usuarios que tenían acceso al mismo. Realizada dicha notificación, se eliminarán los metadatos del documento.

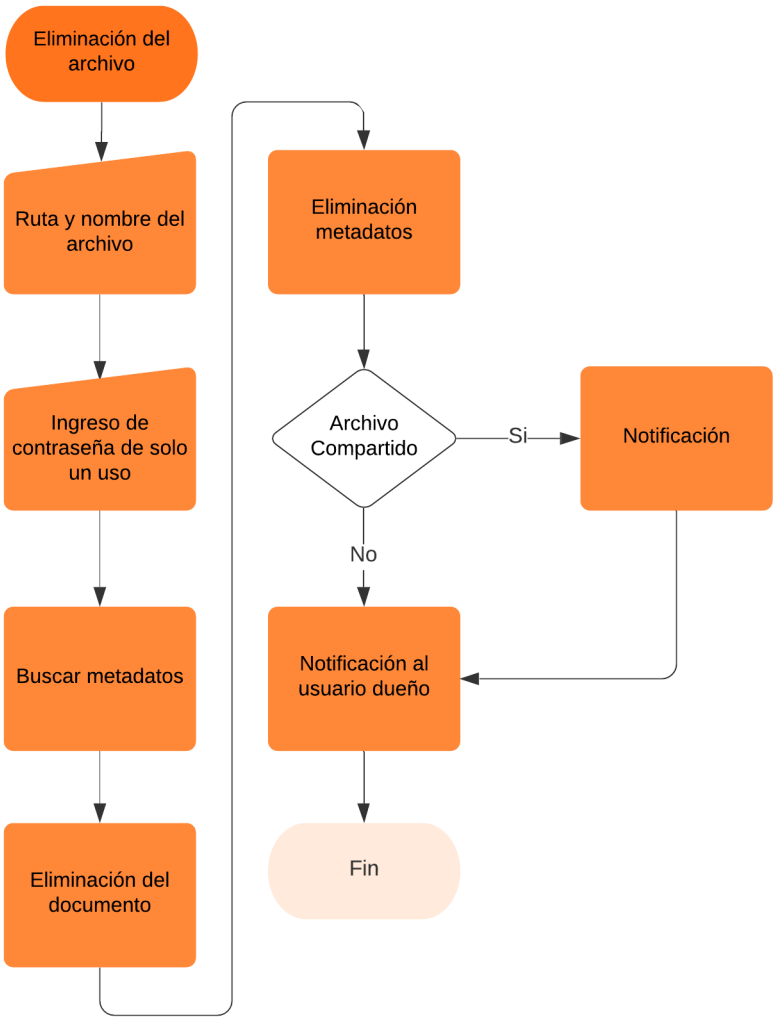

Figura 14 Diagrama de proceso: Eliminación de un archivo

Fuente: confección propia

Capítulo 5. Conclusiones y Recomendaciones

5.1 Conclusiones

En el diseño de este sistema se consideró, en todo momento, la triada de la seguridad de la información, la cual está compuesta por la confidencialidad, integridad y disponibilidad. Para cada uno de estos componentes se adoptaron medidas que garantizaran el cumplimiento estos tres aspectos.

La disponibilidad se garantiza al utilizar tecnologías y técnicas para asegurar el acceso a la información y operación del sistema el mayor tiempo posible. Emplear proveedores de la nube como Google para el almacenamiento de los datos asegura una disponibilidad del 99.99% y 99.999999999% de durabilidad anual (*Clases de almacenamiento* | *Cloud Storage*, s/f), el uso de tecnologías como *etcd* ayudan a que el sistema pueda correr en múltiples servidores y clústeres, de forma efectiva y fácil, ya que la tecnología de *etcd* ya está probada y aceptada por la industria en proyectos como Kubernetes (*Operating Etcd Clusters for Kubernetes*, s/f) y la utilizan de forma similar o igual a la manera en que propone este proyecto el uso de *etcd*.

La confidencialidad se logra principalmente mediante la encriptación de los archivos mediante tecnologías confiables y estándares de la industria como AES y GCM, este es uno de los puntos más críticos del sistema ya que, es donde más desviación podría existir, ya que para efectos de esta investigación solo se compararon tres de los algoritmos de encriptación más reconocidos, pero existen más, e inclusive propietarios, sin embargo, como se menciona desde el inicio, uno de los objetivos del trabajo es proponer el uso de tecnologías estándares de la industria y validadas por las agencias reguladoras. El uso de criptografía de llave pública y privada para guardar la llave de encriptación ayuda a resguardarla, como uno de los componentes más importantes del sistema.

La integridad se logra de forma indirecta mediante la encriptación. Al estar encriptada la información es imposible para un tercero modificar el archivo sin corromperlo, a la hora de desencriptar. El uso de modos de encadenamiento de bloques con autenticación también certifica que no exista ninguna manipulación indebida en el proceso de desencriptación. Se justifica el uso de SHA-3 como

nombres de archivo para implementar una forma de seguridad por oscuridad, tener todos los nombres del mismo tamaño, y en lo que de forma práctica son caracteres aleatorios, ayuda a que no sea posible conocer información del archivo. Otra de las utilidades del uso de SHA-3 es la implementación de un modelo de control de versiones, pues al cambiar cualquier bit del archivo, se cambia el resultado de la encriptación porque por definición el SHA-3 también cambiará. Al llevar un registro de la historia de las firmas se puede usar el mapeo para encontrar una versión anterior del documento y reconstruirla.

5.2 Recomendaciones

Si bien la especificación que aquí se documentó cumple con todos los objetivos planteados, existen mejoras que se podrían implementar en trabajos futuros. Uno de los puntos más cruciales en la implementación de criptografía aplicada es el manejo de las llaves, en este diseño las llaves privadas se mantienen de forma segura y oculta de los usuarios mediante el uso de *etcd*. La centralización de estas llaves podría verse como algo peligroso, lo cual podría ser mitigado mediante el uso de HSM o aún mejor, llaves USB criptográficas o “smartcards”, siempre en cumplimento con las normas FIPS 140-2. La implementación de esta medida le daría una mayor seguridad al sistema, ya que delegaría la parte más delicada a sistemas diseñados y certificados para este uso. Ahora bien, el diseño de este sistema no impide la implementación de esta mejora, sin embargo, en esta versión del proyecto no se realiza, ya que está fuera de los objetivos y alcances originales. El diseño de este sistema está enfocado en compartir documentos e información, y en ningún caso como repositorio a largo plazo; si bien, en otras iteraciones de la propuesta se pueden subsanar las debilidades para este tipo de uso, pues de igual forma existe el uso de llaves criptografías físicas, que tampoco forman parte de este trabajo.

La mayor parte del procesamiento y de los procesos claves se dan en el servidor, el uso de protocolos abiertos como HTTPS y gRPC es recomendado para poder escalar las interfaces con las cuales el usuario interactúa con el software, esto como aplicaciones móviles nativas o aplicaciones de escritorio, entre otras posibilidades.

Bibliografía

Aoki, K., Ichikawa, T., Kanda, M., Matsui, M., Moriai, S., Nakajima, J., & Tokita, T. (s/f). *Speci cation of Camellia | a 128-bit Block Cipher*. 35.

Arne Osvik, D. (2002). *Serpent Cipher Algorithm Implementation Linux Kernel* [C]. https://github.com/torvalds/linux/blob/e22ce8eb631bdc47a4a4ea7ecf4e4ba499db4f93/crypto/serpent_generic.c (Original work published 2011)

Bae, D., Kim, J., Park, S., & Song, O. (2005). Design and Implementation of IEEE 802.11i Architecture for Next Generation WLAN. En D. Feng, D. Lin, & M. Yung (Eds.), *Information Security and Cryptology* (Vol. 3822, pp. 346–357). Springer Berlin Heidelberg. https://doi.org/10.1007/11599548_30

Barker, E. B. (2016a). *Guideline for Using Cryptographic Standards in the Federal Government: Cryptographic Mechanisms* (NIST SP 800-175B; p. NIST SP 800-175B). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-175B

Barker, E. B. (2016b). *Guideline for Using Cryptographic Standards in the Federal Government: Cryptographic Mechanisms* (NIST SP 800-175B; p. NIST SP 800-175B). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-175B

Bellare, M., Kohno, T., & Namprempre, C. (2006). *The Secure Shell (SSH) Transport Layer Encryption Modes* (Núm. RFC4344; p. RFC4344). RFC Editor. https://doi.org/10.17487/rfc4344

Biryukov, A., Dinu, D., & Khovratovich, D. (2016). Argon2: New Generation of Memory-Hard Functions for Password Hashing and Other Applications. *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, 292–302. https://doi.org/10.1109/EuroSP.2016.31

Bluetooth SIG, Inc. (2010). *Core Specification 4.0*.

Brown, D. R. L. (s/f). *SEC 2: Recommended Elliptic Curve Domain Parameters*. 37.

Calculadora de precios de Google Cloud Platform | Google Cloud. (s/f). Recuperado el 21 de marzo de 2021, de https://cloud.google.com/products/calculator#id=a292e32a-7794-4f29-ad3a-b9e27ba59024

Callas, J., Donnerhacke, L., Finney, H., Shaw, D., & Thayer, R. (2007). *OpenPGP Message Format* (Núm. RFC4880; p. RFC4880). RFC Editor. https://doi.org/10.17487/rfc4880

Chavarría-González, M. (2011). La dicotomía cuantitativo/cualitativo, falsos dilemas en la investigación social. *Actualidades en Psicología*, 25(112). https://doi.org/10.15517/ap.v25i112.70

Chuah, C. W., Dawson, E., & Simpson, L. (2013). Key Derivation Function: The SCKDF Scheme. En L. J. Janczewski, H. B. Wolfe, & S. Shenoi (Eds.), *Security and Privacy Protection in Information Processing Systems* (Vol. 405, pp. 125–138). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-39218-4_10

Clases de almacenamiento | Cloud Storage. (s/f). Google Cloud. Recuperado el 22 de agosto de 2021, de https://cloud.google.com/storage/docs/storage-classes?hl=es

Contreras, D. V. V., Gonzalez, D. C., & Jaramillo, C. B. (s/f). *Revista Technology Inside by CPIC IV Volumen Noviembre, 2019 Dirección: San José, Costa Rica.*

Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., & Polk, W. (2008). *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile* (Núm. RFC5280; p. RFC5280). RFC Editor. https://doi.org/10.17487/rfc5280

Dansimp. (s/f-a). *Descripción general del cifrado de dispositivos de BitLocker en Windows 10—Microsoft 365 Security*. Recuperado el 22 de agosto de 2021, de https://docs.microsoft.com/es-es/windows/security/information-protection/bitlocker/bitlocker-device-encryption-overview-windows-10

Dansimp. (s/f-b). *Overview of BitLocker Device Encryption in Windows 10—Microsoft 365 Security*. Recuperado el 27 de marzo de 2021, de https://docs.microsoft.com/en-us/windows/security/information-protection/bitlocker/bitlocker-device-encryption-overview-windows-10

Dierks, T., & Rescorla, E. (2008). *The Transport Layer Security (TLS) Protocol Version 1.2* (Núm. RFC5246; p. RFC5246). RFC Editor. https://doi.org/10.17487/rfc5246

Diffie, W., & Hellman, M. (1976). New directions in cryptography. *IEEE Transactions on Information Theory*, *22*(6), 644–654. https://doi.org/10.1109/TIT.1976.1055638

/Docs/man1.1.1/man1/openssl-speed.html. (s/f). Recuperado el 25 de abril de 2021, de https://www.openssl.org/docs/man1.1.1/man1/openssl-speed.html

Dworkin, M. (s/f). Recommendation for block cipher modes of operation: The CCM mode for authentication and confidentiality. *COMPUTER SECURITY*, 27.

ECRYPT CSA. (2018). *Algorithms, Key Size and Protocols Report (2018)*.

Escudero, T. (2016). La investigación evaluativa en el siglo XXI: Un instrumento para el desarrollo educativo y social cada vez más relevante. *RELIEVE - Revista Electrónica de Investigación y Evaluación Educativa*, *22*(1). https://doi.org/10.7203/relieve.22.1.8164

European Network and Information Security Agency. (2013). *Algorithms, key size and parameters: Report – 2014.* Publications Office. https://data.europa.eu/doi/10.2824/36822

FIPS 197, Advanced Encryption Standard (AES). (s/f). 51.

Floyd, C. (1984). A Systematic Look at Prototyping. En R. Budde, K. Kuhlenkamp, L. Mathiassen, & H. Züllighoven (Eds.), *Approaches to Prototyping* (pp. 1–18). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-69796-8_1

Frankel, S., Glenn, R., & Kelly, S. (2003). *The AES-CBC Cipher Algorithm and Its Use with IPsec* (Núm. RFC3602; p. RFC3602). RFC Editor. https://doi.org/10.17487/rfc3602

Fruhwirth, C. (s/f). *TKS1—An anti-forensic, two level, and iterated key setup scheme*. 7.

Fruhwirth, C. (2018). *LUKS1 On-Disk Format Specification Version 1.2.3*. 15.

Garfinkel, S. (2017). *Anti-Forensics: Techniques, Detection and Countermeasures*. 9.

Google Cloud Free Program. (s/f). Recuperado el 21 de marzo de 2021, de https://cloud.google.com/free/docs/gcp-free-tier?hl=en

Grassi, P. A., Garcia, M. E., & Fenton, J. L. (2017). *Digital identity guidelines: Revision 3* (NIST SP 800-63-3; p. NIST SP 800-63-3). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-63-3

Guerrero-Bote, V. P., & Moya-Anegón, F. (2012). A further step forward in measuring journals' scientific prestige: The SJR2 indicator. *Journal of Informetrics*, *6*(4), 674–688. https://doi.org/10.1016/j.joi.2012.07.001

Hauer, B. (2015). Data and Information Leakage Prevention Within the Scope of Information Security. *IEEE Access*, *3*, 2554–2565. https://doi.org/10.1109/ACCESS.2015.2506185

Hernández Sampieri, R., Fernández Collado, C., & Pilar Baptista Lucio, M. (2014). *Metodología de la investigación*. McGraw-Hill.

Hirsch, J. E. (2005). An index to quantify an individual's scientific research output. *Proceedings of the National Academy of Sciences*, *102*(46), 16569–16572. https://doi.org/10.1073/pnas.0507655102

IBM Corporation. (2020). *Cost of a Data Breach Report 2020*. https://www.ibm.com/security/digital-assets/cost-data-breach-report/#/pdf.

IEEE Standard for Information technology--Telecommunications and information exchange between systems--Local and metropolitan area networks--Specific requirements-Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 3: Enhancements for Very High Throughput in the 60 GHz Band. (s/f). IEEE. https://doi.org/10.1109/IEEESTD.2012.6392842

ISO/IEC, I. (2010). *Information technology—Security techniques—Encryption algorithms—Part 3: Block ciphers*.

IThemes Security Pro Feature Spotlight – Password Requirements. (2021, marzo 5). IThemes. https://ithemes.com/ithemes-security-pro-feature-spotlight-password-requirements/

Kaliski, B. (2000). *PKCS #5: Password-Based Cryptography Specification Version 2.0* (Núm. RFC2898; p. RFC2898). RFC Editor. https://doi.org/10.17487/rfc2898

Kanno, S., & Kanda, M. (2011). *Addition of the Camellia Cipher Suites to Transport Layer Security (TLS)* (Núm. RFC6367; p. RFC6367). RFC Editor. https://doi.org/10.17487/rfc6367

KeyserSosa. (2018, agosto 1). *We had a security incident. Here's what you need to know.* [Reddit Post]. r/announcements. www.reddit.com/r/announcements/comments/93qnm5/we_had_a_security_incident_heres_what_you_need_to/

Kuechler, B., & Vaishnavi, V. (2008). On theory development in design science research: Anatomy of a research project. *European Journal of Information Systems*, *17*(5), 489–504. https://doi.org/10.1057/ejis.2008.40

Mian, P., Conte, T., Natali, A., Biolchini, J., & Travassos, G. (s/f). *A Systematic Review Process for Software Engineering*. 7.

Ministerio de Ciencia, Tecnología y Telecomunicaciones. (2013). *Política de Certificados para la Jerarquía Nacional de Certificadores Registrados*.

http://www.firmadigital.go.cr/Documentos/CPSistemaNacionalCertificaci%C3%B3nDigitalversi%C3%B3n1.pdf

Naranjo-Zeledón, L., José, S., & Rica, C. (2020). Investigación en Informática: El enfoque alternativo. *Revista Technology Inside*, *5*, 15.

Nechvatal, J., Barker, E., Bassham, L., Burr, W., Dworkin, M., Foti, J., & Roback, E. (2001). Report on the development of the Advanced Encryption Standard (AES). *Journal of Research of the National Institute of Standards and Technology*, *106*(3), 511. https://doi.org/10.6028/jres.106.023

Nippon Telegraph and Telephone Corporation. (2001, abril 17). *Announcement of Royalty-free Licenses for Essential Patents of NTT Encryption and Digital Signature Algorithms*. https://www.ntt.co.jp/news/news01e/0104/010417.html

Oiwa, Y., Watanabe, H., Takagi, H., Maeda, K., Hayashi, T., & Ioku, Y. (2017). *Mutual Authentication Protocol for HTTP* (Núm. RFC8120; p. RFC8120). RFC Editor. https://doi.org/10.17487/RFC8120

Operating etcd clusters for Kubernetes. (s/f). Kubernetes. Recuperado el 22 de agosto de 2021, de https://kubernetes.io/docs/tasks/administer-cluster/configure-upgrade-etcd/

Ortiz, C. (2021). *Genera un "word- cloud" basado en los resúmenes guardados en Zotero*. Gist. https://gist.github.com/cortiz/28dcbadee88e50109a07df8367413c41

Percival, C., & Josefsson, S. (2016). *The scrypt Password-Based Key Derivation Function* (Núm. RFC7914; p. RFC7914). RFC Editor. https://doi.org/10.17487/RFC7914

Performance Analysis of Data Encryption Algorithms. (s/f). Recuperado el 25 de abril de 2021, de https://www.cs.wustl.edu/~jain/cse567-06/ftp/encryption_perf/index.html

Pomberger, G., Bischofberger, W., Kolb, D., Pree, W., & Schlemm, H. (1998). *Prototyping-Oriented Software Development—Concepts and Tools*. 20.

Poulton, N. (2019). *Docker Deep Dive: Zero to Docker in a single book* (3a ed.). Shroff Publisher.

Regenscheid, A. (2019). *Recommendations for Discrete-Logarithm Based Cryptography: Elliptic Curve Domain Parameters* [Preprint]. https://doi.org/10.6028/NIST.SP.800-186-draft

Rescorla, E. (2018). *The Transport Layer Security (TLS) Protocol Version 1.3* (Núm. RFC8446; p. RFC8446). RFC Editor. https://doi.org/10.17487/RFC8446

Retina, E. P. (2020, junio 17). *Fuga de información: Cómo evitar poner en riesgo los activos de las empresas con el teletrabajo*. EL PAÍS RETINA. https://retina.elpais.com/retina/2020/06/15/innovacion/1592217009_535570.html

Robles-Sandoval, S. (2019). *Adaptación de la metodología de ciencia de diseño en el desarrollo de luminarias Adapting Design Science Methodology in luminaires development*. 6.

Schaad, J., Ramsdell, B., & Turner, S. (2019). *Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 4.0 Message Specification* (Núm. RFC8551; p. RFC8551). RFC Editor. https://doi.org/10.17487/RFC8551

Software Engineer Salary | *PayScale*. (s/f). Recuperado el 20 de marzo de 2021, de https://www.payscale.com/research/US/Job=Software_Engineer/Salary

Stadler, M. (1996). Publicly Verifiable Secret Sharing. En U. Maurer (Ed.), *Advances in Cryptology—EUROCRYPT '96* (Vol. 1070, pp. 190–199). Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-68339-9_17

Stallings, W. (2017). *Cryptography and network security: Principles and practice* (Seventh edition). Pearson.

The 3-Clause BSD License | *Open Source Initiative*. (s/f). Recuperado el 21 de marzo de 2021, de https://opensource.org/licenses/BSD-3-Clause

Top 200 Most Common Passwords of 2020 | *NordPass*. (s/f). Recuperado el 27 de marzo de 2021, de https://nordpass.com/most-common-passwords-list/

Trusted Computing Group. (2015, agosto 5). *TCG Storage Security Subsystem Class: Opal*. https://trustedcomputinggroup.org/wp-content/uploads/TCG_Storage-Opal_SSC_v2.01_rev1.00.pdf

U6.3 Esquema general de un sistema de cifrado simétrico. (s/f). Recuperado el 27 de marzo de 2021, de https://virtual.itca.edu.sv/Mediadores/cms/u63_esquema_general_de_un_sistema_de_cifrado_simtrico.html

U6.6 Esquema general de un sistema de cifrado asimétrico. (s/f). Recuperado el 27 de marzo de 2021, de https://virtual.itca.edu.sv/Mediadores/cms/u66_esquema_general_de_un_sistema_de_cifrado_asimtrico.html

Wang, S. (2006). *An Architecture for the AES-GCM Security Standard*. 129.