Universidad Autónoma de Querétaro

Facultad de Informática

Algoritmo de encriptado de llave privada para seguridad en la comunicación de los satélites GALILEO, compatible con el sistema OSNMA

Tesis

Que como parte de los requisitos
para obtener el Grado de
Maestro en Ciencias de la computación

Presenta

Ing. Erick Gómez Nieto

Dirigido por:

Dr. José Alberto Ramírez Aguilar

Co-Director:

Dr. Ricardo Chaparro Sánchez

Querétaro, Qro. a 22 de agosto de 2024

Universidad Autónoma de Querétaro

Facultad de Informática

Maestría en Ciencias de la Computación

Algoritmo de encriptado de llave privada para seguridad en la comunicación de los satélites GALILEO, compatible con el sistema OSNMA

Tesis

Que como parte de los requisitos para obtener el Grado

Maestro en Ciencias de la Computación

Presenta

ING. Erick Gómez Nieto

Dirigido por:

Dr. José Alberto Ramírez Aguilar

Co-dirigido por:

Dr. Ricardo Chaparro Sánchez

Dr. José Alberto Ramírez Aguilar
Presidente

Dr. Ricardo Chaparro Sánchez
Secretario

M.I.T. Dulce Carolina Sánchez Hernández
Vocal

Dra. Diana Margarita Córdova Esparza
Suplente

Dr. Juvenal Rodríguez Reséndiz
Suplente

Centro Universitario, Querétaro, Qro.
Agosto, 2024
México

1. Dedicatorias

A todos aquellos que han sido parte de este viaje académico, desde mis profesores hasta mi pareja, amigos y familiares. Gracias por su apoyo y confianza en mí.

También, para todos aquellos que sueñan con alcanzar el espacio...

2. Agradecimientos

A mis padres, que siempre mostraron su apoyo y constantemente estuvieron al pendiente de mis necesidades dentro y fuera de la maestría.

A mis profesores, que nunca nos mintieron de lo que era la maestría y me mantuvieron siempre consciente de lo que era estudiar una maestría.

A mis asesores, al Doctor Alberto Ramírez por permitirme conocer el proyecto Galileo y adentrarme al mundo de los GNSS, y al Doctor Ricardo Chaparro por el apoyo brindado y darme guía cuando lo necesitaba.

A mis amigos, que me brindaron su apoyo y empatía durante este proceso y me motivaron constantemente.

A mi pareja, quien vivió este proceso a mi lado y no dudo en seguirme motivando para culminarlo.

Finalmente, a todos aquellos que, de alguna manera, contribuyeron a que este proyecto se hiciera realidad, mi más sincero agradecimiento.

2.1 Agradecimientos institucionales

Quiero expresar mi agradecimiento a Telespazio por su apoyo y colaboración en el desarrollo de este trabajo.

TELESPAZIO
a LEONARDO and THALES company

Además, quiero agradecer a la Comisión Europea la cual dio apertura hacia México para que estudiantes como yo tengamos la posibilidad de participar en un proyecto de esta magnitud como lo es el sistema satelital europeo Galileo.

European Commission

Agradezco a la Unidad de Alta Tecnología (UAT) de la Facultad de Ingeniería de la Universidad Nacional Autónoma de México (UNAM) por permitirme colaborar con el proyecto Galileo y GIC México.

UAT - FI
Unidad de Alta Tecnología - Facultad de Ingeniería

También, dar especial agradecimiento a la Universidad Autónoma de Querétaro por el apoyo brindado por durante todo el proceso de investigación.

UNIVERSIDAD AUTONOMA DE QUERETARO
EDUCO EN LA VERDAD Y EN EL HONOR

Por ultimo dar un agradecimiento al Consejo Nacional de Humanidades Ciencias y Tecnologías (CONHACYT) por el apoyo brindado a lo largo de la investigación.

Índice general

Índice de cuadros

Índice de figuras

3. Abreviaturas y siglas

1. *XP* - Programación Extrema (*Extreme Programming*).
2. *OSNMA* - Autenticación de Mensajes de Navegación de Servicio Abierto (*Open Service Navigation Message Authentication*).
3. *AES* - Estándar de Encriptación Avanzada (*Advanced Encryption Standard*).
4. *DES* - Estándar de Encriptación de Datos (*Data Encryption Standard*).
5. *3DES* - Triple DES.
6. *IDEA* - Algoritmo de Encriptación de Datos Internacional (*International Data Encryption Algorithm*).
7. *RC5* - Cipher Block Chaining (Unidad de cifrado por bloques).
8. *RSA* - Rivest, Shamir, Adleman (sistema de cifrado asimétrico) (*Rivest, Shamir, Adelman*).
9. *CFB* - Retroalimentación de Cifrado (*Cipher Feedback*).
10. *GNSS* - Sistema de Navegación Satelital Global (*Global Navigation Satellite System*).
11. *TESLA* - Autenticación Tolerante a la Pérdida de Flujo Eficiente Temporizado (*Timed Efficient Stream Loss tolerant Authentication*).
12. *LORAN* - Del inglés *LOng RAnge Navigation*, navegación de largo alcance.
13. *TRANSIT* - También conocido como NAVSAT (Navy Navigation Satellite System).
14. *GPS* - Sistema de Posicionamiento Global (*Global Positioning System*).
15. *GLONASS* - Sistema de Navegación Global por Satélite (Globalnaya Navigatsionnaya Sputnikovaya Sistema).
16. *UAT* - Unidad de Alta Tecnología.

17. *UNAM* - Universidad Nacional Autónoma de México.
18. *WLAN* - Red de Área Local Inalámbrica (*Wireless Local Area Network*).
19. *MAC* - Código de Autenticación de Mensajes (*Message Authentication Code*).
20. *CDMA* - Acceso Múltiple por División de Código (*Code Division Multiple Access*).
21. *RHCP* - Polarización Circular a la Derecha (*Right-Hand Circular Polarization*).
22. *ARNS* - Servicios de Navegación por Radio en la Aviación (*Aviation Radio Navigation Services*).
23. *OS* - Servicio Abierto (*Open Service*).
24. *C/A* - Código Asistido (*Coarse/Acquisition*).
25. *P* - Código de Precisión (*Precision code*).
26. *GHz* - Gigahertz.
27. *MHz* - Megahertz.
28. *RAM* - Memoria de Acceso Aleatorio (*Random Access Memory*).
29. *GUI* - Interfaz Gráfica de Usuario.
30. *PBKDF2* - Función de Derivación de Clave Basada en Contraseña 2 (*Password-Based Key Derivation Function 2*)
31. *HMAC* - Código de Autenticación de Mensaje Hash (*Hash Message Authentication Code*)
32. *PBKDF2HMAC* - Función de Derivación de Clave Basada en Contraseña 2 con Código de Autenticación de Mensaje Hash (*Password-Based Key Derivation Function 2 with Hash-Based Message Authentication Code*)
33. *IV* - Vector de Inicialización (*Initialization Vector*)
34. *SHA* - Algoritmo de Hash Seguro (*Secure Hash Algorithm*)

35. *PKCS7* - Estándares de Criptografía de Clave Pública #7 (*Public-Key Cryptography Standards #7*)
36. *CDMX* - Ciudad de México
37. *KB* - Kilobyte
38. *SNR* - Relación Señal a Ruido (*Signal-to-Noise Ratio*)
39. *BER* - Tasa de Error de Bits (*Bit Error Rate*)
40. *dBi* - Ganancia de Antena en dB sobre un Radiador Isotrópico (*Antenna Gain in dB over an Isotropic Radiator*)
41. *dB* - Decibelio

4. Resumen

El proyecto que se presenta a continuación se centra en el diseño y desarrollo de un algoritmo de encriptado de llave privada para garantizar la seguridad y fidelidad de los datos transmitidos a través de las comunicaciones en el sistema de geoposicionamiento Galileo, con especial atención al sistema de autenticación OSNMA. Se adoptó la metodología XP (eXtreme Programming) debido a su adaptabilidad, lo que permitió un desarrollo ágil y concurrente del algoritmo con otros componentes del sistema. El algoritmo implementado, basado en AES en modo CFB, ofrece un nivel elevado de seguridad en la transmisión de datos.

Además, se llevó a cabo una evaluación exhaustiva de la eficiencia del algoritmo tanto en el proceso de cifrado como en el de descifrado, mediante pruebas de tiempo utilizando archivos de distintos tamaños. Esta investigación profundiza en la comprensión de la criptografía, los sistemas GNSS y la crítica necesidad de proteger la información en entornos como el sistema satelital Europeo Galileo.

Para validar la viabilidad del algoritmo propuesto, se realizó una simulación detallada de la constelación satelital Galileo, resaltando la importancia de considerar factores externos que puedan afectar la integridad de las comunicaciones satelitales en entornos reales.

Palabras clave: Criptografía, Cifrado, Descifrado, Galileo, GNSS, Seguridad.

5. Abstract

This work focuses on the design and development of a private key encryption algorithm to ensure the security and integrity of data transmitted through communications in the Galileo positioning system, with special attention to the OSNMA authentication system. The XP methodology (eXtreme Programming) was adopted due to its adaptability, allowing for agile and concurrent development of the algorithm alongside other system components. The implemented algorithm, based on AES in CFB mode, provides a high level of security in data transmission.

Furthermore, a comprehensive evaluation of the algorithm's efficiency was conducted in both the encryption and decryption processes, through timing tests using files of varying sizes. This research delves into the understanding of cryptography, GNSS systems, and the critical need to protect information in environments such as the Galileo satellite system.

To support the viability of the proposed algorithm, a detailed simulation of the Galileo satellite constellation was performed, highlighting the importance of considering external factors that may affect the integrity of satellite communications in real-world environments.

Keywords: Cryptography, Encryption, Decryption, Galileo, GNSS, Security.

6. Introducción

6.1 Justificación

El avance de las tecnologías de comunicación en los sistemas GNSS ha mejorado significativamente la accesibilidad y la rapidez en la transmisión de información. Sin embargo, esto también ha incrementado el riesgo de vulneraciones de seguridad y la obtención de información confidencial. Esto se debe a su participación activa en la comunicación y envío de datos en tiempo real. Ejemplos de estas actividades incluyen el geoposicionamiento en tiempo real, aplicaciones de generación de rutas, conducción autónoma, recomendaciones basadas en la ubicación y redes de comunicación bancarias. Otros usos destacados son el monitoreo y gestión de flotas de transporte, la agricultura de precisión, la navegación marítima y aérea, las operaciones de rescate y emergencia, y el seguimiento de mercancías y logística. Entre los distintos sistemas satelitales que facilitan estas actividades se encuentran el GPS, BeiDou, GLONASS y el sistema europeo Galileo.

El sistema de geoposicionamiento Europeo GALILEO cuenta con una técnica de encriptado llamada *Open Service Navigation Message Authentication* (OSNMA), basado en el algoritmo *Timed Efficient Stream Loss tolerant Authentication* (TESLA). Esta técnica de encriptado, se puede utilizar de forma gratuita por cualquier usuario, utiliza un algoritmo de llave pública que está en fase de prueba (Galileo, 2021a).

No obstante, el sistema OSNMA actual no carece de la falta de un algoritmo de llave privada oficial, lo que conlleva un aumento del riesgo de ataques maliciosos y compromete la seguridad de la información que circula a través del sistema. Por ende, se propone programar un algoritmo aplicable de encriptación del tipo llave privada, fundamentado en la criptografía moderna con el objetivo de salvaguardar la integridad de la información transmitida en el sistema GALILEO.

La introducción de una llave privada posibilitaría que la entidad de confianza, en este caso, la fuente, firmara digitalmente las firmas temporizadas generadas. Posteriormente, los receptores podrían utilizar la clave pública correspondiente para verificar la autenticidad de dichas firmas.

La programación de este algoritmo contribuirá a disminuir el nivel de riesgo al que se expone la información, protegiéndola de posibles ataques maliciosos, tales como la suplantación de señal y la interferencia intencional. Es importante garanti-

zar la integridad, autenticidad y confidencialidad de la información transmitida en el sistema Galileo, y la adopción de un algoritmo de encriptación eficaz desempeñará un papel crucial en la reducción del riesgo potencial de ataques a la seguridad.

6.2 Planteamiento del problema

El sistema de geoposicionamiento y radionavegación Galileo emplea múltiples técnicas para asegurar la precisión y seguridad de su señal de navegación. Sin embargo, existe una deficiencia en el ámbito del Sistema Abierto OSNMA. Debido a que esta tecnología es accesible de forma gratuita para cualquier usuario a nivel mundial y proporciona cierto nivel de seguridad al salvaguardar la integridad de la información recibida, carece actualmente de un algoritmo de encriptación de llave privada oficial. Esta ausencia representa una vulnerabilidad en la ciberseguridad y la criptografía.

La carencia de un sistema de encriptación de llave privada en el OSNMA podría facilitar interferencias intencionales y técnicas de suplantación de señal *spoofing*, lo que incrementaría el riesgo de alteración o falsificación de la información. Estas amenazas potenciales afectarían directamente a la información y a la confiabilidad del sistema satelital europeo. Por lo tanto, implementar un sistema de encriptación de llave privada en el OSNMA se vuelve crucial para prevenir tales ataques y asegurar la autenticidad e integridad de la señal de navegación en el sistema de radionavegación GALILEO.

La introducción de un algoritmo de encriptación de llave privada permitiría que la entidad de confianza, es decir, la fuente, firmara digitalmente las señales temporizadas generadas. Posteriormente, los receptores podrían utilizar la clave pública correspondiente para verificar la autenticidad de dichas firmas. Este enfoque fortalecería significativamente la seguridad del OSNMA al mitigar el riesgo de interferencias maliciosas y garantizar la autenticidad de la información transmitida.

7. Antecedentes

Las siguientes secciones son relevantes para explicar la importancia de la criptografía en la seguridad de la información en los sistemas satelitales de posicionamiento global (GNSS) y su implementación en el sistema GALILEO:

- La primera sección se explorara y se establecerán las bases históricas y técnicas de los sistemas de navegación global por satélite, como la evolución desde sistemas terrestres como LORAN y OMEGA hasta la llegada de sistemas espaciales como TRANSIT, sentando las bases para los modernos Sistemas de Navegación por Satélite, como el GPS y GLONASS.
- La segunda sección se enfoca en el sistema GALILEO, adentrándonos en la creación del Sistema Galileo por parte de la Unión Europea, destacando su enfoque civil, precisión y su papel internacional. Además, examinaremos la colaboración de la UNAM como Centro de GALILEO en México.
- En la tercera sección introduce el concepto de criptografía y sus tipos, explorando los fundamentos de la Criptografía, desde su papel en la seguridad de la información hasta las dos ramas esenciales: criptografía simétrica y asimétrica. Así como el desarrollo histórico desde la Segunda Guerra Mundial hasta la criptografía moderna y sus hitos clave.

En conjunto, estas secciones ofrecen una visión completa de los sistemas de navegación por satélite y la importancia de la seguridad en la transmisión de datos en dichos sistemas.

7.1 Sistemas de posicionamiento global

Dos sistemas precursores de los actuales sistemas globales satelitales de radionavegación y posicionamiento fueron los sistemas terrestres LORAN y OMEGA, los cuales utilizaban señales de radiofrecuencia desde una posición bien conocida como referencia, complementándose con otras provenientes de estaciones secundarias. Así, el retraso entre la recepción y la emisión de la señal de referencia en las estaciones secundarias ayudaba a conocer las distancias entre estas, lo cual contribuía a conocer la ubicación. Los sistemas LORAN y OMEGA no usaban satélites,

por ello en los años 60 aparece el sistema TRANSIT, el cual ya incorporaba el uso del segmento espacial. Las órbitas de los satélites y sus frecuencias emitidas eran bien conocidas, de tal forma que monitorizando las diferencias en frecuencia de la señal emitida y recibida daban como resultado la posición del receptor del usuario (Berné Valero et al., 2023).

El sistema satelital de navegación global (GNSS - Global Navigation Satellite System) es el nombre genérico que engloba a los Sistemas de Navegación por Satélite que proporcionan un posicionamiento geoespacial, y se concibe como un servicio de localización las 24 horas de día en cualquier lugar del globo sin importar las condiciones climáticas, dicho servicio es brindado por satélites que circundan la Tierra montados en una órbita geoestacionaria fija (Berné Valero et al., 2014, Leick et al., 2015).

Para 1978 Estados Unidos pone en órbita el primer satélite GPS, declarándolo oficialmente operativo para fines estrictamente militares. El sistema GPS o Sistema de Posicionamiento Global está conformado por 24 satélites que orbitan la tierra a una altitud aproximada de 20,200 km, con un periodo de 12 horas. Esta constelación de satélites permite ubicar de manera georreferenciada a través de receptores con latitud y longitud al dispositivo que se esté utilizando, como por ejemplo un tractor, una cosechadora, una pulverizadora, etc. (Álvarez, 2008, et al, 2014.

A la par de GPS, en 1982 aparece GLONASS (Sistema satelital de navegación global o *Global Navigation Stellite System*). Siendo el sistema de posicionamiento global por parte de la Unión Soviética y actualmente administrado por la Federación Rusa. Conformado por una red satelital conformada por 24 satélites distribuidos en 3 órbitas separadas por 120° sobre el ecuador. Sus satélites circundan la tierra a una altura de 19,130 Km de altura con una frecuencia de 11:12 horas. (Jerez y Alves, 2018, Kumar et al., 2021).

7.2 Sistema de posicionamiento global Europeo Galileo

Por su parte y en búsqueda de crear un sistema de posicionamiento global de uso civil y aprovechando los alcances tecnológicos que el continente europeo estaba desarrollando, se desarrolla GALILEO, un sistema de posicionamiento global propuesto por la unión europea que promete una alta precisión, garantizando un margen de error casi nulo y brindando un servicio de posicionamiento global bajo control civil a diferencia de los demás. Este proyecto comenzó en el año 2011

hasta llegar a contar con 30 satélites que orbitan a 23,222 Km de altura, situados en 3 planos orbitales con una inclinación de 56° respecto al ecuador, con un periodo de 14:04 horas. Es completamente compatible con los demás sistemas GNSS (GPS, GLONAS),ofreciendo servicios de navegación en tiempo real con un sesgo de aproximadamente cuatro metros de margen de error, asegurando poder brindar su servicio en condiciones extremas con tan solo segundos de error (Cakir, 2021).

El Centro de control de GALILEO en México se encuentra en la Unidad de Alta Tecnología (UAT) de la Facultad de Ingeniería de la Universidad Nacional Autónoma de México (UNAM), siendo parte del marco de la colaboración internacional entre la UNAM y empresas europeas especializadas en el sector de los sistemas GNSS, siendo México una de las instancias en las cuales el sistema satelital europeo GALILEO se encuentra trabajando en América.

7.3 Criptografía y tipos de encriptado

La Criptología, derivada del griego *krypto* (oculto) y *logos* (estudio), constituye la disciplina que se dedica a asegurar la confidencialidad, integridad, autenticidad y no rechazo en el intercambio de mensajes entre emisor y receptor. Esta ciencia comprende dos vertientes complementarias: la criptografía y el criptoanálisis. La criptografía se centra en el desarrollo de técnicas para escribir mensajes de forma secreta, persiguiendo el propósito de ocultar el significado de la información transmitida. Por otro lado, el criptoanálisis se ocupa de analizar y descifrar mensajes cifrados, desvelando su contenido original. En conjunto, la Criptología se erige como un pilar fundamental en la seguridad de la información, ofreciendo un marco integral para salvaguardar la privacidad y la autenticidad en las comunicaciones digitales (Fernández Acevedo, 2004, R., 2005, Paar y Pelzl, 2009, Rubí et al., 2011).

La criptografía se puede clasificar en dos categorías principales: simétrica y asimétrica. En el caso de la criptografía simétrica, también conocida como de clave privada, se utiliza una única clave tanto para cifrar como para descifrar datos. Los algoritmos utilizados en este tipo de cifrado suelen ser simples y rápidos, ya que se basan en operaciones matemáticas programables en computadoras.

Dentro de la criptografía simétrica, existen dos modos de cifrado principales: el cifrado de bloque, que divide los datos en bloques y aplica la clave en cada ronda de procesamiento, y el cifrado de flujo, que opera en bits individuales. Por otro lado, la criptografía asimétrica, o de clave pública, utiliza un par de claves diferentes: una

pública, que se comparte abiertamente y se usa para cifrar datos, y una privada, que se mantiene en secreto y se utiliza para descifrar. Los mensajes cifrados con la clave pública solo pueden descifrarse con la clave privada correspondiente, garantizando así la seguridad de la comunicación (Whitman y Mattord, 2021, Bisht y Singh, 2015, Estupiñán-Ortiz y Bone-Obando, 2018, Rani y Kaur, 2017).

A partir de la Segunda Guerra Mundial se ha impulsado el desarrollo de tecnología de cifrado, jugando un rol muy importante en las comunicaciones y en la forma de enviar y recibir información. En el año 1918, aparece uno de los artículos con mayor influencia en el área de la criptografía del siglo 20, la monografía de William F. Friedman: *The Index of Coincidence and Its Applications in Cryptography*, mostrando su investigación privada por parte de Riverbank Laboratories. En ese mismo año Edward H. Hebern de Oakland, California, registra la primera patente para una máquina de rotor, este dispositivo estaría destinado a ser un pilar de la criptografía militar para principios de los años 50 (Nielson y Monson, 2019).

Dentro del ámbito científico, la criptografía se origina como una disciplina dentro de las matemáticas, específicamente como parte de la "Teoría de la Información", iniciada por el matemático Claude Elwood Shannon en 1948. Esta rama del conocimiento se divide a su vez en dos ramas principales: la "Teoría de Códigosz la Çriptología". La Criptología, por su parte, se subdivide en Criptoanálisis y Criptografía, cada una con enfoques distintos dentro del estudio y aplicación de técnicas de seguridad y protección de la información (Paredes et al., 2006).

A los inicios del año 1970, gracias a Hosrt Feistel, quien trabajó en los sistemas de identificación para la división de Fuerza Aérea de Estados Unidos, Gracias a su pasión por la Criptografía y con el apoyo de IBM, desarrollaría lo que hoy en día se conoce como el *U.S Data Encryption Standard* (Nielson y Monson, 2019).

La criptografía moderna puede considerarse que comenzó después de tres acontecimientos importantes: primero, la publicación de la "Teoría de la Información"por Claude Shannon; segundo, la introducción del estándar de cifrado DES (Data Encryption Standard) en 1974; y finalmente, la investigación pionera realizada por Whitfield Diffie y Martin Hellman en 1976 sobre la aplicación de funciones matemáticas de un solo sentido al desarrollo del cifrado de clave pública. Estos hitos marcaron un punto de inflexión en la historia de la criptografía, estableciendo fundamentos teóricos y prácticos para el desarrollo de sistemas de seguridad de la información modernos (Paredes et al., 2006).

8. Fundamento teórico

8.1 Fundamentos de seguridad en sistemas de navegación

Si describimos lo que es la seguridad por encriptado de mensajes de navegación nos referimos a implementar métodos para verificar la identidad tanto del emisor como del remitente, a su vez, proteger la integridad de los datos mientras se transmiten los datos a través del canal elegido. Para lograr esto se hace uso de algoritmos de encriptación y técnicas de firma digital. Esta seguridad donde la geolocalización y radionavegación dependen de las señales transmitidas por satélites, expone a estos sistemas a una serie de amenazas de seguridad que podrían comprometer la integridad de la información.

La creciente dependencia de los sistemas de geolocalización y radionavegación nos enfrenta a desafíos constantes, particularmente a las vulnerabilidades a los sistemas satelitales que nos exponen a posibles ataques. Entre estos, el "*spoofing*"destaca por su capacidad para suplantar señales GNSS, lo que conduce a cálculos erróneos de posición sin que el receptor lo detecte. Además, la manipulación de señales, conocida como "*man-in-the-middle*", permite a los atacantes interceptar, modificar o leer mensajes entre partes que intentan comunicarse. Asimismo, la interferencia con las señales durante su trayectoria, conocida como interferencia, también representa una amenaza al canal de comunicación, al introducir señales no deseadas. Este tipo de ataques son especialmente dañinos ya que pueden comprometer la integridad de la información sin ser detectados por los usuarios del sistema GNSS (Conti et al., 2016, Jeya et al., 2019, Zapata, 2013).

Para asegurar la integridad y autenticidad de las señales de navegación y prevenir alteraciones o accesos no autorizados, se implementan técnicas de autenticación y cifrado. Entre las técnicas comunes se encuentra la firma digital, un dato electrónico que verifica la autenticidad e integridad de otro dato, generada mediante un procedimiento criptográfico que establece una relación única entre el dato y el firmante. Otras técnicas incluyen la autenticación basada en desafíos, que involucra un diálogo interactivo entre el verificador y el pretendiente, y la autenticación basada en encriptación, que puede ser simétrica (una clave compartida para cifrar y descifrar) o asimétrica (dos claves diferentes para cada usuario: una pública para cifrar y otra privada para descifrar). Permitiéndonos garantizar la seguridad e integridad de

la información transmitida (Cuno, 2015, Franchi, 2012, Hecht, 2011).

La elección de una técnica de autenticación y cifrado depende de varios factores, como el grado de seguridad necesario, la tolerancia a la pérdida de paquetes, la eficiencia computacional y la infraestructura disponible. Algunas aplicaciones pueden requerir una mayor seguridad a expensas de la eficiencia, mientras que otras pueden priorizar la velocidad de procesamiento.

A continuación, se presentan los elementos clave que respaldarán la fundamentación de la hipótesis, ver Fig. 8.1. Este mapa mental detalla los aspectos fundamentales relacionados con la seguridad en sistemas de navegación, abordando desde el fundamento teórico hasta la implementación de sistemas de cifrado específicos. Cada sección aborda temas cruciales, como la definición de seguridad por encriptado de mensajes de navegación, los desafíos y amenazas en sistemas de geolocalización, los tipos de ataques comunes, las técnicas de autenticación y cifrado, así como la comparación de algoritmos de encriptación simétrica y asimétrica. Además, se explora en detalle el protocolo OSNMA en el contexto de la constelación de satélites GALILEO y se analizan las vulnerabilidades del sistema GPS, contrastando su seguridad con OSNMA en términos de autenticación y cifrado. Este conjunto de temas forma la base integral para sustentar la tesis, ofreciendo una comprensión profunda de la seguridad en sistemas de navegación.

Figura 1
Mapa Mental: Fundamento Teórico y Sistemas de Cifrado

Figura 8.1. Mapa Mental: Fundamento Teórico y Sistemas de Cifrado

8.2 Sistemas de cifrado de navegación

La seguridad en sistemas de cifrado de navegación global es una preocupación constante, dada la creciente dependencia de la tecnología para los servicios de geolocalización y la radionavegación en diversas aplicaciones. A lo largo del tiempo, se han desarrollado y evolucionado diversas técnicas de cifrado para abordar los desafíos de seguridad asociados con la protección de las señales de navegación contra amenazas como el *spoofing* y la manipulación. En esta sección, exploramos la evolución y la comparativa de los sistemas de cifrado en navegación global, considerando tanto encriptación simétrica como asimétrica, y cómo han influido en la creación de soluciones más seguras y eficientes. Para comprender el significado de un encriptado del tipo simétrico, se debe de entender por principio que deben de existir por lo menos dos usuarios que necesiten comunicarse entre sí por algún canal no seguro, por ejemplo: Internet, WLAN, entre otros. Ya comprendiendo esto, si existe un tercer usuario "malicioso" que se encuentre navegando de manera libre conectado a un Router común o escuchando una señal de radio de alguna comunicación inalámbrica, a esta actividad no autorizada se le llama "*eavesdropping*". Los algoritmos criptográficos simétricos tienen dos versiones: cifrado en bloque y cifrado en flujo. Una cifra es una palabra para describir un algoritmo de cifrado. Los cifradores en bloque codifican datos en bloques pequeños de longitud fija de 64 bits de longitud (Franchi, 2012, Mendoza, 2008).

Una vez comprendido el funcionamiento en papel, el siguiente paso es comprender que este tipo de criptografía se basa en utilizar la misma contraseña para el cifrado como para el descifrado del mensaje, lo cual significa que para poder ver el contenido del mensaje los usuarios deben de tener la clave secreta, de tal modo si no tuviesen dicha clave, entre los principales algoritmos de encriptado simétrico, no podrían desencriptar el mensaje y ver su contenido. Los principales algoritmos para este tipo de cifrado son: *Data Encryption standard* (DES), 3DES, *Advanced Encryption Standard* (AES), *International Data Encryption Algorithm* (IDEA), RC5 y *Twofish* (Cabrera Serrano, 2023).

Por otro lado, la criptografía asimétrica se diferencia significativamente de los algoritmos mencionados previamente, ya que su objetivo no radica en establecer una correspondencia matemática directa entre la entrada y la salida. Mientras que la criptografía simétrica se basa en la sustitución y permutación de símbolos, la criptografía asimétrica opera bajo el principio de que tanto el texto plano como el texto

cifrado son representados como números, y el cifrado y descifrado son funciones matemáticas aplicadas a estos números para obtener otros números.

Para comprender su funcionamiento, es esencial destacar que la comunicación entre el emisor y el receptor debe llevarse a cabo a través de un canal seguro, que por sí solo no lo es. Por lo tanto, no es factible transmitir la clave y, por ende, la información directamente a través de dicho canal. Para abordar esta limitación, se requiere la generación de una clave que permita restablecer la seguridad del canal. Uno de los desafíos principales en la criptografía asimétrica radica en la cantidad de claves que deben ser generadas. Es importante destacar que, en este tipo de cifrado, tanto el emisor como el receptor poseen su propio conjunto de claves de cifrado y descifrado respectivamente, que se generan en pares.

Por lo tanto, cada vez que se crea una clave privada, se generan automáticamente las claves correspondientes para ambos participantes. Por ejemplo, en una red de n usuarios tendremos:

$$\frac{(n*(n-1))}{2}$$

Cada usuario en un sistema de cifrado de clave pública debe almacenar de forma segura su propia clave privada, así como las claves públicas de todos los demás usuarios con los que desea comunicarse de manera segura. Por lo tanto, para una red corporativa de 1000 personas, cada usuario necesitaría almacenar las claves públicas de los otros 999 usuarios, lo que totaliza 999 claves públicas además de su propia clave privada. Esto significa que cada usuario debe almacenar un total de n-1 claves de forma segura (Franchi, 2012).

En una red corporativa de 1000 personas, se requerirían un total de 999 * 1000 = 999,000 claves públicas en total. Esto es debido a que cada usuario necesita almacenar las claves públicas de los otros 999 usuarios en la red.

8.3 Encriptación Simétrica VS. Asimétrica

Una de las principales comparaciones entre ambos sistemas de encriptado es la longitud de la clave, la cual es dramáticamente diferente, por ejemplo, una clave simétrica de 80 bits proporciona la misma seguridad que una clave de 1024 bits en RSA., dándonos a entender que la cantidad de bits que se utilicen será determinante en la cantidad e tiempo en la cual esta clave podría ser vulnerada por algún tipo de ataque, obteniendo una relación entre la longitud de la clave y el tiempo estimado

de vulneración, donde tendremos claves de 56 a 64 bits los cuales serán de corto plazo, 112 a 128 bits, largo plazo en ausencia de computadoras cuánticas y 256 bits a largo plazo aun con computadoras cuánticas. (Franchi, 2012).

Ahora bien, analizando la información obtenida se puede resumir que la técnica de cifrado simétrica se utiliza una sola clave secreta compartida entre el emisor y el receptor para cifrar y descifrar los mensajes. Esta clave es esencial para el proceso de cifrado y debe mantenerse en secreto para garantizar la seguridad del sistema. La encriptación simétrica es eficiente en términos de procesamiento, lo que la hace adecuada para aplicaciones que requieren un alto rendimiento. Sin embargo, la principal limitación de la encriptación simétrica es la necesidad de distribuir la clave de manera segura, lo que puede ser un desafío en entornos donde la seguridad no está garantizada.

En el cuadro 8.1 podremos encontrar una comparación de ambos sistemas de cifrado mostrando las características más importantes de ambos tipos de cifrado.

Cuadro 1

Comparación entre cifrado simétrico y asimétrico

	Simétrica	**Asimétrica**
Número de claves	Una clave única, compartida entre los involucrados	Se establecen dos claves para cada usuario
Seguridad	Baja, al enviar la clave sin protección	Alta, el mensaje se cifra para cada participante
Velocidad	Dada la cantidad de información transmitida, la cual es poca al ser una sola clave, es alta	Al requerir de dos claves por mensaje, tiende a ser lenta
Cantidad de Bits	80 bits	1024 bits

Cuadro 8.1. *Comparación entre cifrado simétrico y asimétrico*

En lugar de ser una elección que determine cuál es superior a la otra, la selección del sistema de cifrado se basa en el ajuste a los requisitos y las necesidades específicas de la información que se va a transmitir. Esto implica que, al analizar las ventajas y desventajas de diferentes sistemas de cifrado, se puede determinar cuál se adapta mejor a las características particulares de los datos que deben protegerse y garantizar su seguridad.

Gracias a estos sistemas de cifrado, se puede observar una contribución significativa en el ámbito de la navegación, ya que se han aplicado diversas técnicas como: la encriptación por sustitución, por bloque, flujo y la asimétrica. Empezando por la encriptación por sustitución, en particular el cifrado de César. Este método consiste en transformar un texto legible en uno ilegible mediante un desplazamiento predefinido de letras o símbolos. Por ejemplo, al aplicar un desplazamiento de 3 posiciones en el cifrado de César, la frase "FLIUDGRV FHVDU DÑHDWRULRV"se convierte en el texto cifrado, y al revertir el proceso, obtenemos el texto claro: ÇIFRADOS CESAR ALEATORIOS". La cual nosotros lo ejemplificamos con nuestro alfabeto mediante la aplicación:

$$f : \Omega \to \Omega$$
$$f(n) = (n+24) mod 27$$
$$f : \mathbb{N} \to \mathbb{Z}_{27}$$

Se puede utilizar cualquier conjunto completo de residuos módulo 27 al realizar operaciones con números enteros en aritmética modular, aunque aquí hemos seleccionado el conjunto [0,26] $\{n \in \mathbb{N} \mid 0 \leq n \leq 26\}$, que representa los menores residuos no negativos módulo 27 (Delgado Pineda, 2010).

Con la evolución de la criptografía, se adoptaron métodos más avanzados como el cifrado por bloques y el de flujo. El cifrado por bloques divide el texto en bloques fijos y cifra cada uno independientemente mediante una transformación dependiente de una clave. Este método sigue rondas repetidas de una operación llamada "función de ronda", cada una dependiendo de subclaves derivadas de la clave secreta a través de un programa de claves. Estas subclaves aumentan la seguridad y complejidad del cifrado.

Por otro lado, el cifrado por flujo cifra un flujo continuo de texto según un estado interno en evolución. El programa de claves, o "*key schedule*.ªsegura que las subclaves sean únicas y adecuadas para cada ronda, garantizando la unicidad en el descifrado. La función de ronda debe ser biyectiva para permitir al receptor descifrar el texto cifrado de manera única. Estas técnicas avanzadas mejoran la seguridad y robustez de los sistemas criptográficos (Sánchez et al., 2014).

Por su parte tenemos el cifrado por flujo, este cifrado se aplica a textos mediante la combinación de caracteres con una secuencia de bits secreta llamada secuencia cifrante a través de operaciones como el operador producto y el operador *Xor*. El descifrado se realiza utilizando operaciones como producto, *Xor* y desplazamientos

de letras del texto cifrado con la secuencia cifrante para obtener el texto original. La secuencia cifrante se genera mediante un Generador de Registros por Retroalimentación (LFSR) en criptografía. Los cifradores en flujo se dividen en dos categorías: cifradores asíncronos y cifradores síncronos. Los cifradores asíncronos dependen solo del estado anterior y no de los caracteres del mensaje original, lo que implica que la función de transición de estados es independiente del mensaje. La pérdida de un símbolo en el criptograma puede llevar a la pérdida de sincronización entre el emisor y el receptor, lo que requiere que ambos sincronicen sus generadores de clave antes de continuar (Sánchez et al., 2014).

Es importante mencionar que no son los únicos sistemas de cifrado aplicados históricamente, como por ejemplo el de Scytale de Esparta, que fue una de las herramientas criptográficas más antiguas que se utilizaba para realizar un cifrado por trasposición. Que consistía en un cilindro con una tira de pergamino; el pergamino se enrollaba alrededor del cilindro y luego el mensaje se escribía a lo largo del pergamino. La clave en este caso sería el radio del propio cilindro. Si el pergamino se enrollaba alrededor de un cilindro de un radio diferente, las letras no se alinearían de la misma manera, lo que haría que el mensaje fuera ilegible (Bray, 2020).

A medida que se reconoció la importancia de abordar la distribución segura de claves, la encriptación asimétrica ganó prominencia. El sistema de cifrado asimétrico, también conocido como criptografía de clave pública, ofrece varias ventajas importantes. En primer lugar, garantiza la seguridad en la comunicación al utilizar un par de claves, una pública y una privada, donde la clave pública se emplea para cifrar datos y la clave privada correspondiente se utiliza para descifrarlos, lo que asegura un alto nivel de seguridad. Además, permite la autenticación de identidad al cifrar mensajes con la clave privada, lo que ayuda a prevenir la suplantación de identidad.

La distribución segura de claves se simplifica, ya que cada usuario solo debe mantener su propia clave privada en secreto, eliminando la necesidad de compartir claves secretas. La escalabilidad se facilita, ya que cada usuario puede tener su propio par de claves, y la confidencialidad a largo plazo está respaldada por problemas matemáticos difíciles de resolver. Además, la capacidad de crear firmas digitales garantiza la autenticidad e integridad de documentos electrónicos, siendo esencial en aplicaciones como la banca en línea y la autenticación de documentos legales. En resumen, la criptografía asimétrica es una elección valiosa para una amplia gama de aplicaciones de seguridad dentro de los sistemas de radionavegación y geoposi-

cionamiento (Mendoza, 2008).

Es importante comprender el funcionamiento del cifrado asimétrico debido a que es la base de la arquitectura OSNMA, en el sistema satelital europeo de radionavegación y geoposicionamiento GALILEO.

8.4 Implementación y funcionamiento de OSNMA en Galileo

Antes de comenzar a indagar en el funcionamiento del protocolo *Open Service Navigation Message Authentication* (OSNMA) es importante mencionar la bases del mismo, principalmente el protocolo de Autenticación Eficiente Temporizada Tolerante a la Pérdida de Flujo (*Timed Efficient Stream Loss-tolerant Authentication*), mejor conocido como TESLA por sus siglas en inglés, es un algoritmo de autenticación mediante *broadcast*, que consta en que un solo emisor tiene la capacidad de enviar información a múltiples receptores de manera simultánea. Haciendo un híbrido entre las técnicas de llaves simétricas y asimétricas. Destacando sus principales características, las cuales son: los requisitos de recursos de computación y de comunicación para su funcionamiento sin bajos; alta resistencia ante la perdida de datos, como es el caso de receptores GNSS en entornos con visibilidad reducida; óptimo para transmisiones *multicast*, como es el caso de GNSS (Iglesias Fernández, 2019, Gimenez et al., 2019).

El algoritmo se compone de dos componentes clave: en primer lugar, se emplea el Código de Autenticación de Mensajes (MAC, por sus siglas en inglés) para autenticar el mensaje de navegación en texto plano. Por otro lado, se implementa la transmisión demorada de la clave utilizada para calcular el MAC. Esta clave forma parte de una secuencia de claves denominada cadena de claves TESLA. En esta cadena, cada clave se genera a partir de la anterior mediante una función unidireccional. La generación de esta cadena, que tiene una longitud N, comienza con una clave secreta aleatoria k_n y concluye con una clave raíz pública k_0, la cual está certificada como auténtica. La revelación de las claves en la cadena sigue un proceso inverso.

El segundo mecanismo nos permite asegurarnos de que la revelación de la llave no sea conocida hasta después de que el mensaje de la MAC ha sido recibido, esto nos ayuda a evitar que algún intruso en la comunicación sea capaz de suplantar y generar mensajes. El proceso de verificación del remitente se lleva a cabo gracias a la implementación de un tres pasos básicos, los cuales son: la clave raíz se certifica

como auténtica a través de un esquema asimétrico basado en la verificación de la firma digital transmitida por el SIS y la clave pública disponible en el receptor; La clave TESLA actualmente recibida se autentica utilizando la clave raíz mediante la realización de la función unidireccional $k_j \rightarrow k_j - 1$ el número de veces requerido (j veces hasta llegar a 0)

Alternativamente, si ya se han realizado una o varias verificaciones de autenticación con éxito (por ejemplo, $k_j \rightarrow k_j - 1 \rightarrow ...k_0$ con $p < j$), la clave actual también puede autenticarse con una clave anterior de la cadena, más cercana que la clave raíz en la cadena de claves (es decir, la función unidireccional se calcula desde k_j a k_p, durante $j - p$ veces). Por último, en el receptor se genera el MAC utilizando la clave actual y los datos de navegación. Si coincide con el MAC contenido en el SIS, los datos de navegación quedan autenticados (Nicola et al., 2022).

Ahora bien, comprendiendo que TESLA es la base del protocolo de comunicación OSNMA y que es sumamente necesario para la autenticación de las claves, se puede comenzar por definir lo que es OSNMA. OSNMA es un servicio protección a la información que se mueve a través del sistema de radionavegación y geolocalización Europeo GALILEO, es parte de los Servicios abiertos a través de Mensaje de Navegación (I/NAV) el cual por medio de la difusión sobre el canal civil E1-B, el cual, junto a los canales E5 y E6, forman parte de las tres señales que transmiten la comunicación de forma permanente utilizada en los satélites GALILEO, independientes del Acceso Múltiple por División de Código (CDMA) y de las señales polarizadas a la derecha (RHCP). Por su parte el Canal E5 esta subdividida en dos señales llamadas E5a y E5b, ver Fig. 8.2. Esto pone a disposición del Sistema de radionavegación y geoposicionamiento GALILEO de 5 bandas de frecuencia que proporcionan un ancho de banda amplio a la disposición del sistema satelital para transmitir y recibir información, en el siguiente cuadro se puede apreciar como se divide el *Carrier* correspondiente a las frecuencias y la referencia de los anchos de banda de las señales de GALILEO (Galileo, 2021a).

Se puede comprender un poco más de las frecuencias que utiliza GALILEO en sus señales quedará establecido que para el Carrier E1 será la frecuencia de 1575.40 MHZ con 24.552 MHz de ancho de banda, E6 con 1278.750 MHz de frecuencia con 40.920 MHz de ancho de banda, E5 con 1191.795 MHz de frecuencia con 51.150 de ancho de banda, E5a 1176.450 MHz de frecuencia con 20.460 de ancho de banda y E5b con 1207.140 MHz de frecuencia con 20.460 de ancho de banda. Las bandas de frecuencia de Galileo han sido seleccionadas en el espec-

Figura 2

Plan de frecuencias de GALILEO.

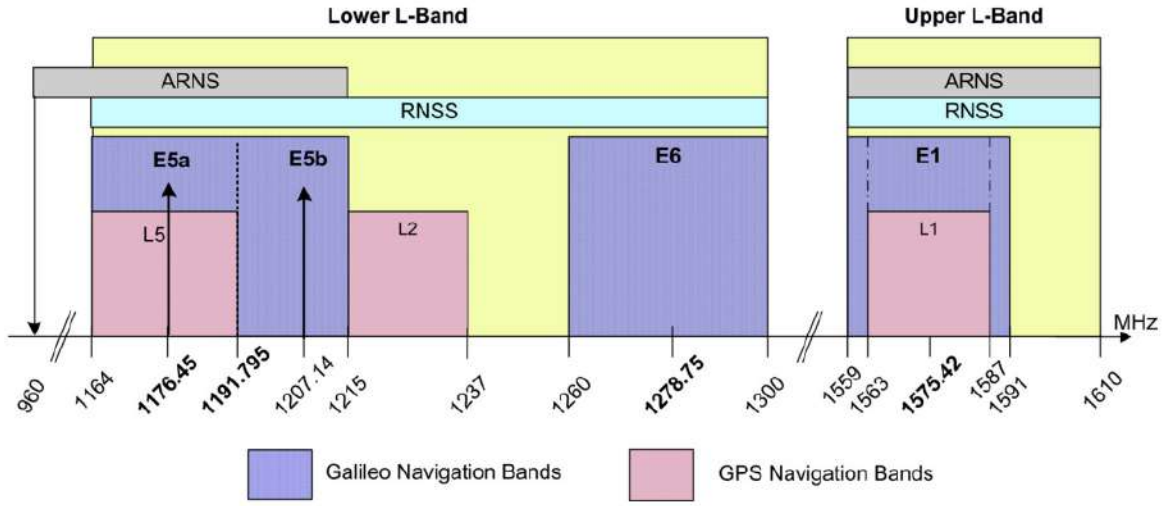


Figura 8.2. Plan de frecuencias de GALILEO.

Nota. Plan de frecuencias de los satélites Galileo en la banda L utilizado para la comunicación de los servicios de navegación recuperado de: Galileo, 2021b.

tro asignado para los Servicios de Satélite de Navegación por Radio (RNSS, por sus siglas en inglés) y, además de eso, las bandas E5a, E5b y E1 están incluidas en el espectro asignado para los Servicios de Navegación por Radio en la Aviación (ARNS, por sus siglas en inglés), empleadas por usuarios de la aviación civil y permitiendo aplicaciones dedicadas críticas para la seguridad (Galileo, 2021a).

Una vez comprendida la frecuencia en la que se mueve y algunos protocolos utilizados en la fase de transmisión de la información, se puede concluir que OSNMA no introduce ninguna interferencia en el sistema, lo que preserva el rendimiento de la navegación del Servicio Abierto (OS) de forma intacta. Además, este diseño asegura una completa compatibilidad con las versiones anteriores, permitiendo que los receptores OS estándar sigan funcionando sin problemas, ignorando los campos específicos de OSNMA en el mensaje I/NAV. Solo los receptores preparados para OSNMA podrán decodificar estos campos y autenticar los datos de navegación de Galileo. Además, para garantizar una implementación segura del protocolo TESLA, los receptores de OSNMA necesitan realizar una sincronización de tiempo al arranque con una referencia de tiempo con una precisión que varía entre 18 segundos y cinco minutos, dependiendo del modo de operación. Las fuentes de tiempo de referencia aproximado pueden ser, por ejemplo, un reloj en tiempo real interno o una

conexión segura a una red con capacidad de transferencia de tiempo (Götzelmann et al., 2023).

El proceso en el que OSNMA trabaja para transmitir información a través de los satélites GALILEO comienza por identificar si un satélite no forma parte del subconjunto de la constelación de satélites o no, si un satélite no pertenece a un subconjunto específico, su mensaje I/NAV OSNMA contendrá una secuencia de 40 bits de ceros, y este subconjunto de satélites que distribuyen los datos OSNMA cambia dinámicamente sin previo aviso para el usuario. Los receptores OSNMA deben estar preparados para esta variabilidad. El protocolo OSNMA está diseñado para permitir la autenticación de los datos de navegación de todos los satélites, incluso si solo un subconjunto los transmite, gracias a la autenticación cruzada, donde los satélites que transmiten emiten OSNMA pueden autenticar los datos de otros satélites mediante etiquetas específicas (Cucchi et al., 2022).

El proceso de los datos OSNMA se puede describir a alto nivel mediante los siguientes pasos, se reciben los datos de navegación junto con datos de OSNMA que incluyen una etiqueta, una clave de cadena TESLA y una clave raíz TESLA. La etiqueta autentica los datos de navegación transmitidos previamente y se recibe antes que su clave de cadena TESLA asociada. La clave raíz TESLA se autentica mediante una firma digital utilizando una clave pública disponible en el receptor. El receptor autentica la clave de cadena TESLA con la clave raíz TESLA o una clave previamente autenticada de la cadena TESLA. Luego, el receptor regenera localmente la etiqueta con la clave de cadena TESLA verificada y los datos, comprobando si coincide con la etiqueta recibida, ver Fig. 8.3. (Götzelmann et al., 2023).

8.5 Sistemas de seguridad implementados en otros servicios de GNSS

El Sistema de Posicionamiento Global, más conocido como GPS, es un sistema de localización desarrollado por el Departamento de Defensa de los Estados Unidos. Su propósito original era proporcionar estimaciones precisas de posición, velocidad y tiempo para fines militares. Desde su operatividad en 1995, el GPS se ha convertido en un sistema de geoposicionamiento ampliamente utilizado y, a diferencia de sistemas de geoposicionamiento de uso libre, el GPS está constantemente monitorizado por su entidad creadora. El sistema funciona calculando la posición de un receptor GPS en un espacio tridimensional (x, y, z) a partir de la medición de distancias a al menos tres satélites cuyas ubicaciones son conocidas. La distancia se

Figura 3

Diagrama de flujo de transición de información en el sistema satelital europeo Galileo.

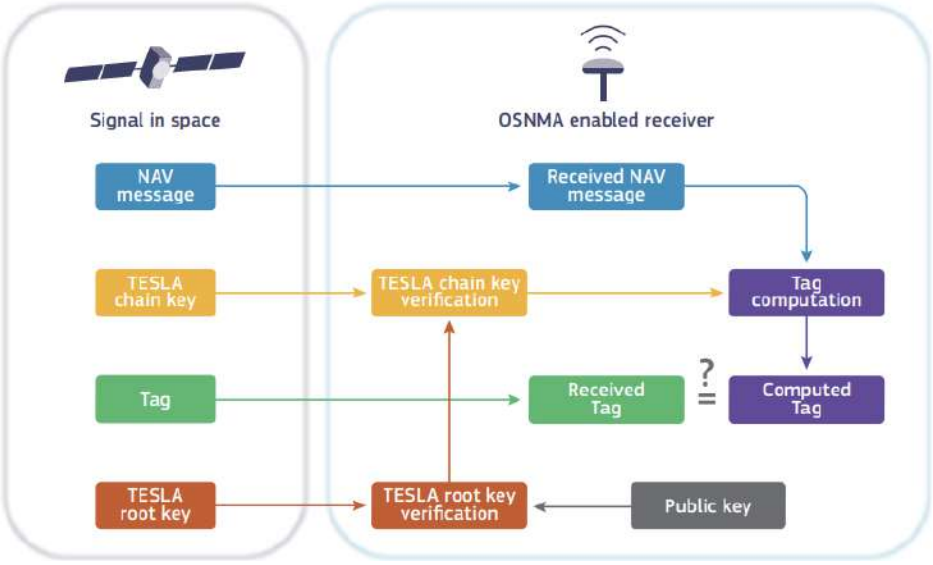


Figura 8.3. Diagrama de flujo de transición

Nota. Diagrama de flujo en el cual se representa el comportamiento de la información a su envió y recepción, recuperado de: Galileo, 2021a.

calcula multiplicando el tiempo que la señal de un satélite tarda en llegar al receptor por la velocidad de la luz. LA sincronización precisa de los relojes de los satélites y del receptor es crucial para este cálculo (Berné Valero et al., 2019).

Sin embargo, los relojes de los receptores son menos precisos que los de los satélites, lo que introduce errores conocidos como pseudodistancias. La variabilidad en los relojes del receptor requiere la utilización de al menos cuatro satélites para estimar con precisión la posición del receptor. Para calcular las pseudodistancias, se tiene en cuenta que las señales GPS son débiles y se encuentran en un entorno de ruido de fondo. Los receptores generan un código pseudoaleatorio que se ajusta a la señal transmitida por el satélite. El receptor calcula la distancia ajustando el código hasta que coincida con el código recibido, lo que representa el tiempo de vuelo de la señal (Pozo-Ruz et al., 2000).

Es fundamental señalar que, a diferencia de GALILEO, el sistema GPS no emplea un sistema de autenticación y cifrado similar a OSNMA, y sus señales son de acceso público, lo que implica que estas señales no están cifradas ni autenticadas, lo que las hace susceptibles a amenazas como el *spoofing* y la manipulación de señales. Para abordar esta vulnerabilidad, se recurre a códigos pseudoaleato-

rios que controlan el acceso al sistema de satélites. Estos códigos pseudoaleatorios permiten regular el acceso al sistema, ya que en situaciones conflictivas se puede alterar el código, forzando a los satélites a emplear una única banda de frecuencia sin interferencias, dado que cada satélite posee su propio código GPS

No obstante, esta solución presenta limitaciones, como un mayor consumo de recursos, lo que conlleva tiempos de respuesta más prolongados y la saturación de los canales utilizados, especialmente en la banda L1 destinada al uso civil. A pesar de la alta velocidad de los satélites (alrededor de 4 km/s), la posición instantánea de los mismos puede estimarse con un error de unos pocos metros basándose en predicciones de sus posiciones anteriores en un período de 24 a 48 horas. Las estaciones terrestres revisan periódicamente los relojes atómicos de los satélites, dos de cesio y dos de rubidio, enviando efemérides y correcciones de relojes.

La precisión y estabilidad de los relojes, junto con la trayectoria de los satélites, son críticas para el funcionamiento del sistema GPS. El sistema de código pseudoaleatorio transmitido en el GPS incluye tres tipos de cadenas: el código C/A (Coarse/Acquisition) a 1.023 MHz, destinado a usuarios civiles; el código P (Precision code) a un poco más de 10 MHz, utilizado por usuarios militares; y el código Y, que se envía encriptado en lugar del código P cuando se activa el modo de operación antiengaños. Los satélites de GPS transmiten en dos frecuencias: la portadora L1 a 1575.42 MHz, donde se transmiten los códigos C/A y P, y la portadora L2 a 1227.6 MHz, que transporta información militar modulada en el código P (Pozo-Ruz et al., 2000).

Es importante mencionar que una efeméride, también conocidas como radiodifundidas o "*Broadcast ephemeris*", consiste en un conjunto de entre 16 y 20 elementos que definen la trayectoria elipsoidal de cada satélite. Éstos envían en el Bloque 2 del mensaje de navegación. Dichas efemérides solo son capaces de posicionar al satélite que las envía (Pou Peña, 2016).

8.6 Análisis de algoritmos de encriptado usados en sistemas GNSS

La seguridad de la información y la protección de datos son aspectos cruciales en el ámbito de la informática y las comunicaciones. En este contexto, los algoritmos de cifrado desempeñan un papel fundamental al garantizar la confidencialidad y la integridad de la información sensible. Entre los algoritmos más prominentes, se encuentran el *Data Encryption Standard* (DES), el *Triple DES* (3DES), el *Advanced*

Encryption Standard (AES), el *International Data Encryption Algorithm* (IDEA), el RC5 y *Twofish*. Estos algoritmos representan hitos significativos en la evolución de las técnicas de cifrado, abordando desafíos específicos y proporcionando soluciones robustas para proteger la información contra posibles amenazas.

8.6.1 *Data Encryption Standard* (DES)

El algoritmo DES (*Data Encryption Standard*) es un sistema de cifrado por bloque tipo producto que opera en bloques de información de 64 bits, generando un bloque de texto cifrado del mismo tamaño. La longitud de la clave es de 56 bits, expresada comúnmente como un bloque de 64 bits, donde cada octavo bit se utiliza para verificar la paridad.

En su nivel más básico, DES combina dos conceptos fundamentales de cifrado: la confusión y dispersión. La estructura principal de DES consiste en una combinación de sustitución y permutación. En términos generales, el proceso de DES comienza con una permutación inicial en un bloque de texto plano de 64 bits. Luego, el bloque permutado se divide en mitades de 32 bits cada una y se somete a 16 rondas de una función tipo Feistel, que mezcla la información con la clave original. Después de la décimo sexta ronda, las mitades se vuelven a unir y se aplica una permutación final, finalizando el algoritmo.

La generación de subllaves implica recorrer cierto número de posiciones de los bits de la clave original para formar 16 nuevas subllaves de 48 bits cada una. Estas subllaves se utilizan en cada ronda de la función de Feistel, donde la mitad derecha del bloque se expande, combina con la subllave correspondiente mediante XOR, pasa por cajas de sustitución y permutaciones, y se combina con la mitad izquierda mediante XOR para formar la nueva mitad derecha. Este proceso se repite en 16 rondas, finalizando con una permutación final (Méndez, 2011).

8.6.2 *Triple DES* (3DES)

Triple DES (3DES) fue desarrollado como una mejora de DES para abordar sus fallas evidentes sin requerir la creación de un nuevo criptosistema. La solución consiste en extender el tamaño de la clave de DES mediante la aplicación del algoritmo tres veces en sucesión, utilizando tres llaves diferentes. Esto resulta en un tamaño de clave combinada de 168 bits (3 veces 56 bits), lo que va más allá de las capacidades de las técnicas de fuerza bruta, como las empleadas por el FEP DES

Cracker.

Aunque el uso de Triple DES ha sido mirado con cierta sospecha debido a que el algoritmo original no fue diseñado específicamente para esta aplicación, no se han descubierto defectos graves en su diseño. En la actualidad, Triple DES es un criptosistema ampliamente utilizado en numerosos protocolos de Internet, demostrando su eficacia y resistencia en el ámbito de la seguridad de la información (Vargas y Mnedez, 2015).

Para poder observar las características principales de cada uno de estos algoritmos se presenta en el cuadro 8.2:

Cuadro 2

Comparación entre DES y 3DES

Factor	DES	3DES
Creador	IBM en 1974	IBM en 1978
Tamaño en bits	56	168
Seguridad	No seguro	Medianamente seguro
Tipo de clave	Única	1 dividida en 3 partes
Rendimiento de descifrado	60 %	80 %
Resistencia al criptoanálisis	Vulnerable	Vulnerable
Tipo de cifrado	Simétrico	Simétrico
Tiempo para descifrar	800 días	400 días

Cuadro 8.2. *Comparación entre DES y 3DES. Recuperado de:* Tole et al., 2021

8.6.3 *International Data Encryption Algorithm* (IDEA)

El cifrado de bloque IDEA opera con bloques de texto claro y cifrado de 64 bits, controlado por una clave de 128 bits. La innovación fundamental en el diseño de este algoritmo es el uso de operaciones de tres grupos algebraicos diferentes. Se evitan por completo las cajas de sustitución y las búsquedas de tablas asociadas utilizadas en los cifrados de bloque disponibles hasta la fecha. La estructura del algoritmo se elige de manera que, con la excepción de que se utilizan subbloques de clave diferentes, el proceso de cifrado es idéntico al proceso de descifrado.

Los 52 subbloques de clave de 16 bits, generados a partir de la clave de 128 bits, se obtienen mediante la siguiente secuencia: en primer lugar, la clave de 128 bits se divide en ocho subbloques de 16 bits, que se utilizan directamente como los primeros ocho subbloques de clave. Posteriormente, la clave de 128 bits experimenta un desplazamiento cíclico hacia la izquierda de 25 posiciones. El bloque resultante de 128 bits se divide nuevamente en ocho subbloques de 16 bits, que se emplean directamente como los siguientes ocho subbloques de clave. Este proceso de desplazamiento cíclico se repite hasta que se hayan generado los 52 subbloques de clave de 16 bits requeridos (Chang, 2004).

8.6.4 RC5

El algoritmo RC5, desarrollado por Ronald Rivest en 1994, está diseñado para ser eficiente en el uso de memoria y proporcionar un alto nivel de seguridad. Sus características clave incluyen un tamaño de bloque de 32, 64 o 128 bits, una clave criptográfica variable de 0 a 2040 bits (0 a 255 bytes) y un número de rondas que varía de 0 a 255. El nivel de seguridad se establece mediante dos parámetros, siendo una configuración común el RC5 64/12/16. Sin embargo, para asegurar la resistencia contra posibles ataques, se recomienda que el número de rondas sea mayor a 16 (Cavalcante et al., 2011).

8.6.5 *Twofish*

Twofish es un algoritmo criptográfico que opera en modo de cifrado por bloques y fue finalista en la competición del Estándar de Cifrado Avanzado (AES) organizada por el Instituto Nacional de Normas y Tecnología (NIST). Es un cifrador de bloque de 128 bits que puede aceptar claves de hasta 256 bits de longitud. Utiliza una red

Feistel que se ejecuta 16 veces y comprende blanqueo de entrada, cajas S, Transformación Pseudo Hadamard y blanqueo de salida. Las cajas S son matrices que contienen sustituciones simples mapeando bits de entrada a bits de salida, y *Twofish* emplea cuatro cajas S biyectivas, dependientes de la clave y de 8x8 bits. Además, utiliza la transformación Pseudo Hadamard, la cual divide bloques en dos partes iguales y aplica operaciones aritméticas simples. La clave se interpreta mediante la matriz Máxima Distancia Separable (MDS) y la transformación Pseudo Hadamard se realiza mediante operaciones aritméticas específicas (Haryono, 2020).

8.6.6 *Advanced Encryption Standard* (AES)

El Estándar de Cifrado Avanzado (AES), también conocido como Rijndael, es una especificación para la encriptación de datos electrónicos establecida por el Instituto Nacional de Estándares y Tecnología (NIST) de EE. UU. en 2001. Desarrollado por Joan Daemen y Vincent Rijmen, AES es una versión específica del cifrado Rijndael, seleccionada por el NIST para su uso generalizado. Utiliza tamaños de bloque de 128 bits y ofrece tres longitudes de clave: 128, 192 y 256 bits. Reemplazando al Estándar de Encriptación de Datos (DES), AES es ampliamente adoptado a nivel mundial, siendo un algoritmo de clave simétrica, lo que significa que la misma clave se utiliza para cifrar y descifrar los datos.

AES se basa en un principio de diseño conocido como una red de sustitución-permutación, una combinación de sustitución y permutación, y es rápido tanto en *software* como en *hardware*. A diferencia de su predecesor DES, AES no utiliza una red de Feistel. AES es una variante de Rijndael que tiene un tamaño de bloque fijo de 128 bits y un tamaño de clave de 128, 192 o 256 bits. En contraste, la especificación de Rijndael en sí se especifica con tamaños de bloque y clave que pueden ser cualquier múltiplo de 32 bits, con un mínimo de 128 y un máximo de 256 bits (Rijmen y Daemen, 2001).

AES opera en una matriz de orden de columna principal de 4 × 4 de bytes, llamada estado, aunque algunas versiones de Rijndael tienen un tamaño de bloque más grande y tienen columnas adicionales en el estado. La mayoría de los cálculos de AES se realizan en un campo finito especial. Por ejemplo, si hay 16 bytes, b_0, b_1, ..., b_{15}, estos bytes se representan como esta matriz:

$$\begin{bmatrix} b_0 & b_4 & b_8 & b_{12} \\ b_1 & b_5 & b_9 & b_{13} \\ b_2 & b_6 & b_{10} & b_{14} \\ b_3 & b_7 & b_{11} & b_{15} \end{bmatrix} \tag{8.1}$$

El tamaño de la clave utilizado para un cifrado AES especifica el número de repeticiones de rondas de transformación que convierten la entrada, llamada texto plano, en la salida final, llamada texto cifrado. El número de ciclos de repetición es el siguiente:

- 10 ciclos de repetición para llaves de 128 bits.
- 12 ciclos de repetición para llaves de 192 bits.
- 14 ciclos de repetición para llaves de 256 bits.

Cada ronda cuenta con 4 pasos:

1. SubBytes:

   - Cada byte $a_{i,j}$ en la matriz de estado se reemplaza con un SubByte S($a_{i,j}$) utilizando una caja de sustitución de 8 bits, la S-box de Rijndael. Esta operación proporciona la no linealidad en el cifrado. La S-box utilizada se deriva de la inversa multiplicativa sobre GF (2^8), conocida por tener buenas propiedades de no linealidad.

     Dada la matriz:

     $$\text{Matriz de estado} = \begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix}$$

     Después de aplicar el paso de SubBytes, cada byte en la matriz se reemplazaría según la SubByte correspondiente de la S-box de Rijndael. Supongamos que los nuevos valores se obtienen de la siguiente manera:

$$\text{Matriz de salida } \textit{S-box} = \begin{bmatrix} S(a_{0,0}) & S(a_{0,1}) & S(a_{0,2}) & S(a_{0,3}) \\ S(a_{1,0}) & S(a_{1,1}) & S(a_{1,2}) & S(a_{1,3}) \\ S(a_{2,0}) & S(a_{2,1}) & S(a_{2,2}) & S(a_{2,3}) \\ S(a_{3,0}) & S(a_{3,1}) & S(a_{3,2}) & S(a_{3,3}) \end{bmatrix}$$

$$\text{Matriz resultante} = \begin{bmatrix} b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} \\ b_{1,0} & b_{1,1} & b_{1,2} & b_{1,3} \\ b_{2,0} & b_{2,1} & b_{2,2} & b_{2,3} \\ b_{3,0} & b_{3,1} & b_{3,2} & b_{3,3} \end{bmatrix}$$

Durante el proceso de descifrado, se utiliza el paso de SubBytes inverso, que primero requiere realizar la transformación afín y luego encontrar el inverso multiplicativo (simplemente invirtiendo los pasos utilizados en el paso de SubBytes) (Rijmen y Daemen, 2001).

2. Intercambio de filas (*ShiftRows*)

   - Este paso opera en las filas del estado; desplaza cíclicamente los bytes en cada fila por un cierto desplazamiento. Para AES, la primera fila permanece sin cambios. Cada byte de la segunda fila se desplaza una posición hacia la izquierda. De manera similar, las tercer y cuarta filas se desplazan por dos y tres posiciones respectivamente. Para bloques de 128 bits y 192 bits, el patrón de desplazamiento es el mismo. La fila n se desplaza circularmente a la izquierda por n-1 bytes (Rijmen y Daemen, 2001).

     Dada la matriz resultante del paso anterior:

$$\text{Matriz de entrada} = \begin{bmatrix} b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} \\ b_{1,0} & b_{1,1} & b_{1,2} & b_{1,3} \\ b_{2,0} & b_{2,1} & b_{2,2} & b_{2,3} \\ b_{3,0} & b_{3,1} & b_{3,2} & b_{3,3} \end{bmatrix}$$

     Se aplica el paso ShiftRows, obtendríamos la siguiente matriz de estado:

$$\text{Matriz resultante} = \begin{bmatrix} b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} \\ b_{1,1} & b_{1,2} & b_{1,3} & b_{1,0} \\ b_{2,2} & b_{2,3} & b_{2,0} & b_{2,1} \\ b_{3,3} & b_{3,0} & b_{3,1} & b_{3,2} \end{bmatrix}$$

De acuerdo con las reglas mencionadas anteriormente, cada fila se desplaza cíclicamente hacia la izquierda según el número de fila. La primera fila permanece sin cambios, la segunda fila se desplaza una posición hacia la izquierda, la tercera fila se desplaza dos posiciones hacia la izquierda, y la cuarta fila se desplaza tres posiciones hacia la izquierda.

3. Combinación de columnas (*MixColumns*)

   - En este paso, los cuatro bytes de una columna del estado se combinan usando una transformación lineal invertible. Dicha función, toma cuatro bytes como entrada y produce cuatro bytes como salida, donde cada byte de entrada afecta a los cuatro bytes de salida. Junto con *ShiftRows*, *MixColumns* proporciona difusión en el cifrado. Durante esta operación, cada columna se transforma utilizando una matriz fija (la multiplicación de la matriz por la columna da como resultado la nueva columna)

     Cada byte de la columna se considera como un coeficiente en un polinomio y se manipula utilizando operaciones aritméticas especiales en el campo finito GF(2^8). Esta manipulación implica multiplicar cada columna por una matriz fija en GF(2^8), lo que garantiza que la transformación sea reversible y contribuya a la seguridad del algoritmo AES (Rani y Kaur, 2017).

     Dada la siguiente matriz de entrada (procesada por el paso anterior):

$$\text{Matriz de entrada} = \begin{bmatrix} b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} \\ b_{1,1} & b_{1,2} & b_{1,3} & b_{1,0} \\ b_{2,2} & b_{2,3} & b_{2,0} & b_{2,1} \\ b_{3,3} & b_{3,0} & b_{3,1} & b_{3,2} \end{bmatrix}$$

     Aplicamos la operación MixColumns multiplicando cada columna por la siguiente matriz fija en GF(2^8):

$$\text{Matriz de entrada} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix}$$

Después de realizar la multiplicación y adición adecuadas, obtenemos la matriz resultante que representa el estado después del paso MixColumns.

En un sentido más general, cada columna se trata como un polinomio sobre GF(2^8) y luego se multiplica módulo (x^4) + 1 con un polinomio fijo c(x) = 0x03·(x^3)+(x^2)+x+0x02. Los coeficientes se muestran en su equivalente hexadecimal de la representación binaria de polinomios de bits de GF (Rijmen y Daemen, 2001).

4. Agregar ronda de clave (*AddRoundKey*)

   - Para cada ronda, se deriva una subclave de la clave principal utilizando el programa de claves de Rijndael; cada subclave tiene el mismo tamaño que el estado. La subclave se suma combinando cada byte del estado con el byte correspondiente de la subclave utilizando XOR a nivel de bits(Rijmen y Daemen, 2001).

9. Planteamiento Teórico

9.1 Pregunta de investigación

¿Es factible implementar un algoritmo de cifrado de llave privada compatible con el sistema de cifrado OSNMA, que garantice la seguridad y fiabilidad en el intercambio de información del canal y la frecuencia del sistema?

10. Hipótesis

Si se utiliza un algoritmo de cifrado de llave privada compatible con el sistema de cifrado OSNMA, tomando en cuenta las condiciones reales del sistema GALILEO, como los datos de autenticación de navegación y la generación de la MAC, entonces será posible simular la órbita y establecer una conexión cifrada punto a punto que garantice la seguridad y fiabilidad en el intercambio de información del canal y la frecuencia del sistema.

11. Objetivos

11.1 Objetivo General

Desarrollar un algoritmo de encriptado del tipo llave privada, de acuerdo con el sistema de autenticación de mensajes OSNMA, con el propósito de garantizar la seguridad de los canales de comunicación del sistema de geoposicionamiento GALILEO con la finalidad de generar conexiones seguras y dar certidumbre a la información transmitida.

11.1.1 Objetivos Específicos

1. Identificar los fundamentos teóricos de la criptografía para la creación de llaves públicas y privadas, así como la criptografía clásica y moderna.
2. Programar el algoritmo de llave privada que sea compatible con el protocolo OSNMA y sea funcional para el sistema satelital europeo y que permita garantizar la seguridad en la comunicación.
3. Simular la constelación satelital Galileo con ayuda de este *software*: *FreeFlyer*, para probar la fiabilidad de la propuesta.

Comité de Bioética

En el ámbito de la ética en la informática, se han realizado esfuerzos especialmente importantes. Esto se debe al impacto que tiene en la sociedad actual, dando lugar a comportamientos que implican una estrecha relación entre la ética y la informática, como la confidencialidad de los datos personales y la proliferación de publicidad no solicitada a través de(*spamming*).

Debido a esto, tenemos que considerar todos los aspectos del proceso de desarrollo, desde la recopilación de datos, su transferencia, aislamiento y desglose, dentro de un marco que proteja los derechos humanos y la integridad científica. Esto implica evaluar las condiciones en las que se lleva a cabo este proceso para garantizar el cumplimiento de los preceptos bioéticos, desde la codificación del software hasta la implementación de estas tecnologías, con el fin de asegurar que contribuyan a mejorar la calidad de vida y evitar la creación de nuevos mercados que exploten a las personas (Carmona y Cruz, 2024).

En este sentido se hace mención de que en esta investigación no habrá participación de ninguna persona por lo tanto en el caso de este trabajo no hay consideraciones de tipo bioético, sin embargo, como en toda investigación que llegue a solicitar la participación de personas, en caso de que se requiera información alguna, se tendrá el cuidado de avisar a los posibles sujetos participantes en el estudio acerca de los pormenores de la investigación y avisando que la información recabada será para uso exclusivo de este trabajo.

Estableciendo en general el concepto de ética informática como una herramienta teórico-práctico de orientación de los seres humanos hacia actitudes coherentes con la dignidad humana (Arencibia, 2008).

12. Metodología

Como se evidenció en la introducción y en la exposición del problema, se destaca la importancia de garantizar la seguridad y confiabilidad en la transmisión de información mediante el cifrado. Aunque se reconoce que la tecnología de Galileo es relativamente nueva en el ámbito de las comunicaciones satelitales y que su método de cifrado es seguro, pero no infalible, se hace necesario continuar investigando en el campo del cifrado. En este sentido, el presente trabajo tiene como objetivo principal la formulación de una propuesta que añada una capa adicional de seguridad. Además, se enfocará en el desarrollo y evaluación de un método de cifrado simétrico, tomando como referencia el enfoque del cifrado tipo AES. Utilizando un enfoque en la inteligencia artificial, dado a su gran potencial que tiene en el área de procesamiento de información y de señales.

Para esta investigación, se adopta el modelo de metodología XP, también conocida como eXtreme Programming, una metodología de desarrollo de software creada por Kent Beck en su libro .Extreme Programming Explained: Embrace Change"(1999). Esta elección se basa en la forma en la que se acopla de la metodología XP para el desarrollo de un algoritmo que debe ser altamente adaptable y compatible con el protocolo OSNMA en el contexto del proyecto GALILEO. La flexibilidad de XP permite desarrollar el algoritmo de manera concurrente con el protocolo OSNMA, lo cual es crucial debido a las restricciones de tiempo y la falta de información específica sobre el desarrollo de aplicaciones para geoposicionamiento en el proyecto GALILEO.

Las variables independientes para este desarrollo abarcan aspectos como el algoritmo de llave privada, datos de autenticación de navegación, generación de la MAC, y dependientes como parámetros de órbita, configuración de seguridad adicional, información de autenticación de usuarios, frecuencia y canal de comunicación, protocolos de comunicación y políticas de seguridad. Todos estos elementos son fundamentales para garantizar la seguridad y la confiabilidad en la comunicación cifrada punto a punto en el sistema GALILEO.

A continuación se puede comprender el ciclo de vida de la metodología XP, el cual se divide en 5 etapas, las cuales son: planificación, diseño, codificación, pruebas y lanzamiento, ver Fig. 12.1. Teniendo gran importancia el concepto de “finalizado”, ya que el objetivo de cada iteración no es agregar toda la funcionalidad para

justificar el lanzamiento del producto al mercado, sino incrementar el valor por medio de “software que funciona” (Joskowicz, 2008).

Figura 4
Ciclo de vida de la metodología XP

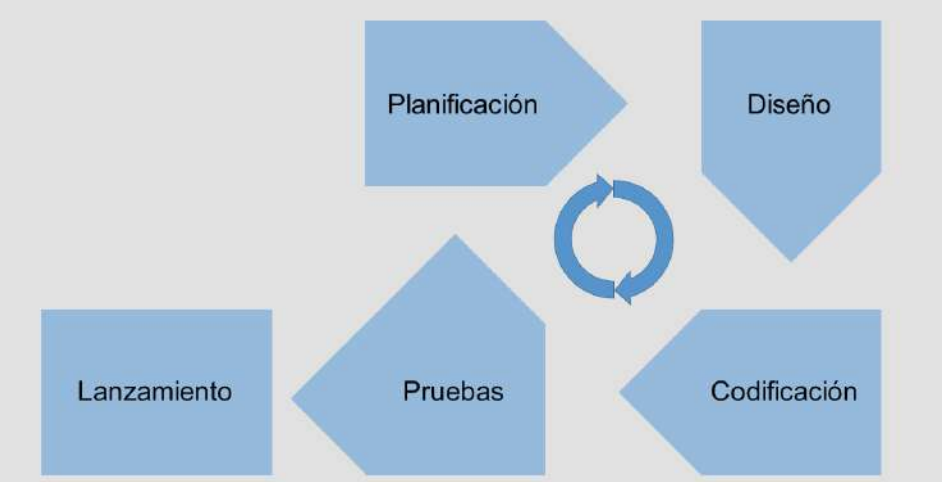


Figura 12.1. *Metodología XP.*

Donde cada etapa se compone por:

1. Planeación: Comienza por llevar a cabo una sesión de escucha activa para recopilar los requisitos, lo que facilita que los miembros técnicos del equipo XP comprendan el contexto empresarial del software. Esto les ayuda a captar la esencia, las características principales y las funcionalidades necesarias para el producto.

2. Diseño: El diseño XP se rige por el principio de mantenerlo sencillo (MS). Se prioriza un diseño simple sobre uno más complejo. El diseño guía la implementación de una 12 historia y si surge un problema de diseño difícil, XP recomienda crear un prototipo de esa parte del diseño de inmediato. Este prototipo, llamado solución en punta, se implementa y evalúa para reducir el riesgo antes de comenzar la implementación real. El objetivo es evaluar las estimaciones originales y disminuir el riesgo asociado a la historia que contiene el problema de diseño.

3. Codificación: el equipo no comienza la codificación inmediatamente después de desarrollar las historias y realizar el diseño preliminar. En su lugar, se desarrollan pruebas unitarias para cada historia que se incluirá en la entrega actual.

Estas pruebas unitarias permiten que los desarrolladores se centren en implementar lo necesario para pasar las pruebas. Después de completar el código, se lleva a cabo una prueba unitaria de manera inmediata, lo que ofrece una retroalimentación instantánea para los desarrolladores. Esto les proporciona un medio para abordar los problemas de manera inmediata. De esta manera, la estrategia de integración continua contribuye a prevenir problemas de compatibilidad e interfaces, y crea un entorno de "prueba de humo"que ayuda a identificar los errores a tiempo.

4. Pruebas: Las pruebas unitarias previas al inicio de la codificación son un componente fundamental de la metodología XP. Estas pruebas unitarias deben ser desarrolladas utilizando una estructura que permita su automatización, lo que posibilita su ejecución de manera repetida y sencilla. Por otro lado, las pruebas de aceptación, también conocidas como pruebas del cliente, son definidas por el cliente y se centran en las características y funcionalidades generales del sistema que son visibles y revisables por parte del cliente. Estas pruebas de aceptación se derivan de las historias de usuario que se han implementado como parte de la entrega del software.

5. Lanzamiento: Si hemos alcanzado este punto, indica que hemos probado todas las historias de usuario con éxito, cumpliendo con los requisitos del cliente. Contamos con un software funcional y estamos listos para integrarlo en el producto final.

12.1 Equipo experimental

Al ser una propuesta de un algoritmo de cifrado, el alcance de esta propuesta será solamente simulada, para este propósito se utilizara de un equipo de computo de tipo LapTop, a continuación se mencionaran sus características principales:

- Procesador 11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz 2.42 GHz
- 12 Gb de RAM
- Tarjeta de vídeo integrada iRIS X^e

12.2 Lenguaje de programación

Existen innumerables lenguajes de programación para desarrollar aplicaciones de cifrado y descifrado, pero es crucial señalar que estamos trabajando en un sistema específico donde los satélites de Galileo operan, y el lenguaje de elección es **Python**. Este se convierte en un requisito esencial para el desarrollo de ambos algoritmos debido a su compatibilidad inherente con los satélites de Galileo, al estar codificados en el mismo lenguaje.

En cuanto a **Python**, es un lenguaje de alto nivel que, aunque no es recomendado para usuarios novatos o con conocimientos limitados de programación, destaca por su sintaxis simple y legible, facilitando la escritura y mantenimiento del código. La versatilidad de **Python** se manifiesta en su capacidad para admitir múltiples aplicaciones, especialmente en el área de inteligencia artificial. Lo que realmente distingue a **Python** es su extensa comunidad de desarrolladores y su rica documentación en línea, lo que fomenta el crecimiento de información y bibliotecas disponibles para su uso, proporcionando un respaldo sólido para resolver problemas y obtener ayuda.

Además, **Python** ofrece una amplia biblioteca que abarca desde manipulación de datos hasta desarrollo web, simplificando tareas comunes con soluciones integradas. Su adopción en diversos campos, como inteligencia artificial, análisis de datos, desarrollo web y automatización, subraya su versatilidad y aplicabilidad en una amplia gama de escenarios. **Python** destaca por su desarrollo rápido, así como su capacidad de integración con otras tecnologías. La portabilidad entre plataformas, combinada con su adopción generalizada en la industria, contribuye a la demanda constante de habilidades en **Python** en el mercado laboral, consolidando su posición como un lenguaje de programación líder y atractivo tanto para profesionales como para principiantes (González Duque, 2011).

12.3 Librerías

En este apartado se dará una breve descripción de aquellas librerías de **Python** que podrán ser utilizadas para el desarrollo del algoritmo de acuerdo a las necesidades de mismo:

- **tkinter**
  Es la biblioteca estándar de interfaz gráfica de usuario (GUI) de **Python**, destaca por su simplicidad y facilidad de uso, siendo especialmente amigable para

aquellos nuevos en el desarrollo de interfaces. Integrado de forma nativa en **Python**, Tkinter es multiplataforma y ofrece una amplia gama de widgets incorporados, como botones y cuadros de texto, que facilitan la creación rápida de interfaces interactivas. Además, proporciona manejo de eventos, personalización de la apariencia de los widgets, y soporte para estilos temáticos a través de ttk. Cuenta con bastante documentación abundante y sigue siendo apoyada por desarrolladores especialistas en este lenguaje,a su vez, Tkinter se mantiene como una opción sólida y accesible para proyectos de desarrollo de GUI en **Python**, especialmente para aplicaciones más simples o en entornos educativos (Moore, 2018).

Al hacer uso de esta biblioteca se utilizaran dos funciones en especifico las cuales son:

1. filedialog
   Al utilizar esto, no necesitamos diseñar cuadros de diálogo estándar por nosotros mismos. Los cuadros de diálogo de archivos ayudan en la apertura y guardado de archivos e incluso directorios. Nos permite abrir un solo archivo, un directorio, guardar como archivo y mucho más de manera sustancial (Bajpai y Jain, 2021).
2. simpledialog
   Contiene clases y funciones convenientes para crear diálogos modales simples para obtener un valor de entrada del usuario (Feng-Cong et al., 2010).

- cryptography
  Python cuenta con un paquete de Criptografía que ofrece recetas y primitivas criptográficas a los desarrolladores de **Python**. Está diseñado para ser su "biblioteca estándar criptográfica". El paquete de Criptografía incluye tanto recetas de alto nivel como interfaces de bajo nivel para algoritmos criptográficos comunes, como cifradores simétricos, resúmenes de mensajes y funciones de derivación de claves (Bray, 2020).

12.4 Ejecución del proyecto

12.4.1 Planificación del proyecto

En esta etapa crucial, se delinean los objetivos y las prioridades del proyecto destinado a desarrollar el algoritmo propuesto. Este proceso implica la definición precisa de las características a implementar, así como la asignación de prioridades a cada tarea. Además, se forja una hoja de ruta detallada (planificación) que facilita el seguimiento del progreso y permite adaptaciones ágiles en respuesta a cambios y necesidades emergentes durante el desarrollo.

La planificación se inicia con una cuidadosa consideración de los objetivos fundamentales, basándose en las investigaciones previas expuestas en el estado del arte de este documento. Un punto crucial a destacar en este contexto es la contribución singular de esta tesis: proponer un algoritmo que no solo sea técnicamente sólido, sino también accesible y adaptable a la arquitectura OSNMA empleada en los satélites Galileo.

12.4.1.1 Establecimiento de las historias de Usuario

Ejemplo de historia de usuario

A lo largo del proyecto se realizaron reuniones con el cliente para establecer los requerimientos que tendrá el algoritmo. Se analizó cada caso mediante las Historias de Usuario. Las cuales se detallaron mediante los siguientes cuadros de historias de usuario.

Cuadro 3

Historia de usuario - Seleccionar un archivo de texto (TXT).

	Historia de Usuario
Código: H1	**Usuarios:** Todos
Nombre de historia: Seleccionar un archivo de texto (TXT)	
Prioridad: Alta	**Riesgo de desarrollo:** Alta
Puntos estimados: 5	**Iteración asignada:** 2
Programador asignado: Erick Gómez	
Descripción: Los usuarios deben tener la capacidad de elegir el archivo de texto que desean cifrar.	
Observación: Esencial para el propósito del cifrador.	

Cuadro 12.1. *Historia de Usuario 1*

Donde cada celda tiene un objetivo, los cuales son:

- **Código (H1):** Este es el código único asignado a esta historia de usuario para identificarla dentro del sistema de gestión de proyectos.
- **Usuarios (Todos):** Indica qué usuarios están involucrados o se beneficiarán de esta historia de usuario. En este caso, se especifica que todos los usuarios están incluidos.
- **Nombre de historia:** Es el título o nombre descriptivo de la historia de usuario, en este caso, "Seleccionar un archivo de texto (TXT)".
- **Prioridad (Alta):** Indica la importancia relativa de esta historia de usuario en comparación con otras historias. En este caso, se considera de alta prioridad.
- **Riesgo de desarrollo (Alta):** Evalúa el nivel de riesgo asociado con el desarrollo de esta historia de usuario. Aquí se indica que tiene un alto riesgo.
- **Puntos estimados (5):** Representa una estimación del esfuerzo necesario para completar esta historia de usuario, en términos de tiempo, complejidad, etc.
- **Iteración asignada (2):** Indica en qué iteración del proceso de desarrollo se planea abordar esta historia de usuario.
- **Programador asignado (Erick Gómez):** Especifica qué miembro del equipo de desarrollo está asignado para trabajar en esta historia de usuario.
- **Descripción:** Proporciona una descripción detallada de la funcionalidad que se debe implementar como parte de esta historia de usuario. En este caso, se trata de permitir a los usuarios seleccionar un archivo de texto que deseen cifrar.
- **Observación:** Proporciona información adicional o comentarios relevantes sobre la historia de usuario. Aquí se destaca que esta funcionalidad es esencial para el propósito del cifrador.

El resto de las historias de usuario se encuentran en el Anexo I.

12.4.1.2 Tareas

Conforme a los fundamentos teóricos establecidos en este documento, se hace importante definir las actividades específicas dedicadas al desarrollo del algoritmo propuesto. Se proponen las siguientes tareas para comprender en profundidad el funcionamiento tanto del algoritmo de cifrado como del de descifrado.

Cuadro 4

Actividad 1 - Historia 1 - Diseñar interfaz gráfica.

Tarea	
Código: T1	**Código de historia:** H1
Nombre de historia: Diseñar interfaz gráfica	
Tipo de tarea: Desarrollo	**Puntos estimados:** 3
Tiempo: 4 días	
Programador asignado: Erick Gómez	
Descripción: ▪ Crear elementos de interfaz gráfica (etiqueta y botón) que permitan a los usuarios seleccionar un archivo de texto para cifrar. ▪ Implementar la lógica para manejar la selección de archivos utilizando la biblioteca *filedialog* de *Tkinter*.	

Cuadro 12.2. Actividad 1 - Historia 1

Cuadro 5

Actividad 2 - Historia 1 - Validar función de interfaz gráfica.

Tarea	
Código: T2	**Código de historia:** H1
Nombre de historia: Validar función de interfaz gráfica	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Tiempo: 2 días	
Programador asignado: Erick Gómez	
Descripción: ▪ Asociar una función a la acción de clic en el botón para manejar la selección de archivos. ▪ Almacenar la ruta del archivo seleccionado en una variable para su uso posterior en el proceso de cifrado.	

Cuadro 12.3. Actividad 2 - Historia 1.

Cuadro 6

Actividad 3 - Historia 2 - Generar Contraseña.

Tarea	
Código: T3	**Código de historia:** H2
Nombre de historia: Generar Contraseña	
Tipo de tarea: Desarrollo	**Puntos estimados:** 5
Tiempo: 4 días	
Programador asignado: Erick Gómez	
Descripción: ▪ Crear una función que genere automáticamente una contraseña de cifrado. ▪ Utilizar caracteres alfanuméricos y especiales para aumentar la complejidad de la contraseña.	

Cuadro 12.4. Actividad 3 - Historia 2.

Cuadro 7

Actividad 4 - Historia 2 - Mostrar Contraseña Generada.

	Tarea
Código: T4	**Código de historia:** H2
Nombre de historia: Mostrar Contraseña Generada	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Tiempo: 4 días	
Programador asignado: Erick Gómez	
Descripción: ▪ Integrar un elemento de interfaz gráfica para mostrar la contraseña generada automáticamente.	

Cuadro 12.5. Actividad 4 - Historia 2.

Cuadro 8

Actividad 5 - Historia 3 - Cifrar un archivo.

	Tarea
Código: T5	**Código de historia:** H3
Nombre de historia: Cifrar un archivo	
Tipo de tarea: Desarrollo	**Puntos estimados:** 5
Tiempo: 4 días	
Programador asignado: Erick Gómez	
Descripción: ▪ Utilizar la contraseña generada y la biblioteca *cryptography* para cifrar el contenido del archivo seleccionado. ▪ Aplicar un esquema de cifrado simétrico, como AES en modo CFB.	

Cuadro 12.6. Actividad 5 - Historia 3.

Cuadro 9

Actividad 6 - Historia 3 - Guardar El Archivo Cifrado.

	Tarea
Código: T6	**Código de historia:** H3
Nombre de historia: Guardar El Archivo Cifrado	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Tiempo: 1 días	
Programador asignado: Erick Gómez	
Descripción: ▪ Utilizar la biblioteca *filedialog* para solicitar al usuario la ubicación y el nombre del archivo cifrado de destino. ▪ Guardar el archivo cifrado en la ubicación especificada.	

Cuadro 12.7. Actividad 6 - Historia 3.

Cuadro 10

Actividad 7 - Historia 4 - Guardar El Archivo Cifrado.

	Tarea
Código: T7	**Código de historia:** H4
Nombre de historia: Mostrar Información en la Interfaz Gráfica	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Tiempo: 1 días	
Programador asignado: Erick Gómez	
Descripción: ▪ Crear elementos de interfaz gráfica para mostrar información relevante sobre el cifrado, como la ubicación del archivo cifrado y otros detalles.	

Cuadro 12.8. Actividad 7 - Historia 4.

Cuadro 11

Actividad 8 - Historia 5 - Validación del archivo cifrado.

Tarea	
Código: T8	**Código de historia:** H5
Nombre de historia: Validación del archivo cifrado	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Tiempo: 1 días	
Programador asignado: Erick Gómez	
Descripción: ▪ Desarrollar la lógica para validar la integridad del archivo cifrado y asegurarse de que sea un archivo válido antes de proceder con el descifrado.	

Cuadro 12.9. Actividad 8 - Historia 5.

Cuadro 12

Actividad 9 - Historia 6 - Seleccionar archivo cifrado.

Tarea	
Código: T9	**Código de historia:** H6
Nombre de historia: Seleccionar archivo cifrado	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Tiempo: 1 días	
Programador asignado: Erick Gómez	
Descripción: ▪ Implementar la funcionalidad para abrir una ventana de selección de archivos. ▪ Integrar la lógica para filtrar y permitir la selección de archivos cifrados. ▪ Conectar la interfaz gráfica con el código que manejará el archivo seleccionado.	

Cuadro 12.10. Actividad 9 - Historia 6.

Cuadro 13

Actividad 10 - Historia 7 - Ingresar contraseña para descifrar.

	Tarea
Código: T10	**Código de historia:** H7
Nombre de historia: Ingresar contraseña para descifrar	
Tipo de tarea: Desarrollo	**Puntos estimados:** 3
Tiempo: 2 días	
Programador asignado: Erick Gómez	
Descripción: ▪ Diseñar y agregar un campo de entrada de contraseña en la interfaz gráfica. ▪ Implementar la lógica para capturar la contraseña ingresada por el usuario. ▪ Integrar la contraseña en el proceso de descifrado del archivo seleccionado.	

Cuadro 12.11. Actividad 10 - Historia 7.

Cuadro 14

Actividad 11 - Historia 8 - Ver datos descifrados.

	Tarea
Código: T11	**Código de historia:** H8
Nombre de historia: Ver datos descifrados	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Tiempo: 1 días	
Programador asignado: Erick Gómez	
Descripción: ▪ Desarrollar la interfaz gráfica para mostrar los datos descifrados. ▪ Conectar la interfaz con la lógica que procesa y presenta los datos descifrados. ▪ Asegurarse de que la visualización de datos sea clara y comprensible para el usuario.	

Cuadro 12.12. Actividad 11 - Historia 8.

Cuadro 15

Actividad 12 - Historia 9 - Guardar datos descifrados como archivo TXT.

Tarea	
Código: T12	**Código de historia:** H9
Nombre de historia: Guardar datos descifrados como archivo TXT	
Tipo de tarea: Desarrollo	**Puntos estimados:** 3
Tiempo: 5 días	
Programador asignado: Erick Gómez	
Descripción: ▪ Crear un botón o funcionalidad para permitir al usuario guardar los datos descifrados. ▪ Implementar la lógica de guardado que tome los datos descifrados y los almacene en un archivo TXT. ▪ Manejar situaciones de error o excepciones durante el proceso de guardado.	

Cuadro 12.13. Actividad 12 - Historia 9.

12.4.2 Diseño

12.4.2.1 Tarjetas CRC (Clase, Responsabilidad y Colaboración)

Las tarjetas CRC, integradas en la metodología XP para el diseño de software orientado a objetos, son fundamentales para vincular directamente las funcionalidades del sistema con las necesidades del cliente. Cada tarjeta representa una Clase, identificando personas, cosas o eventos, junto con sus Responsabilidades (tareas realizadas y conocimientos adquiridos) y Colaboradores (otras clases con las que interactúa). Esta práctica ayuda a organizar el código de manera efectiva, asegurando una implementación que refleje claramente los requisitos de las Historias de Usuario, fortaleciendo así la conexión entre las metas del negocio y el desarrollo del software.

Cuadro 16

CRC para Selector De Archivos.

Clase	**Selector De Archivos**
Responsabilidades	- Crear elementos de interfaz gráfica (etiqueta y botón) para seleccionar un archivo de texto. - Asociar una función a la acción de clic para manejar la selección de archivos. - Almacenar la ruta del archivo seleccionado en una variable para su uso posterior en el proceso de cifrado.
Colaboradores	- Interfaz Grafica - Tkinter (filedialog)

Cuadro 12.14. CRC 1.

Cuadro 17

CRC para Generador de contraseñas.

Clase	**Generador de Contraseña**
Responsabilidades	- Crear una función que genere automáticamente una contraseña de cifrado utilizando caracteres alfanuméricos y especiales. - Integrar un elemento de interfaz gráfica para mostrar la contraseña generada automáticamente.
Colaboradores	- Interfaz Gráfica - Tkinter

Cuadro 12.15. CRC 2.

Cuadro 18

CRC para Cifrador de Archivos.

Clase	**Cifrador de Archivos**
Responsabilidades	- Utilizar la contraseña generada y la biblioteca cryptography para cifrar el contenido del archivo seleccionado. - Aplicar un esquema de cifrado simétrico, como AES en modo CFB. - Utilizar la biblioteca filedialog para solicitar al usuario la ubicación y el nombre del archivo cifrado de destino. - Guardar el archivo cifrado en la ubicación especificada.
Colaboradores	- Selector De Archivos - Tkinter - cryptography

Cuadro 12.16. CRC 3.

Cuadro 19

CRC para Interfaz de Información de Cifrado.

Clase	**Interfaz de Información de Cifrado**
Responsabilidades	- Crear elementos de interfaz gráfica para mostrar información relevante sobre el cifrado, como la ubicación del archivo cifrado y otros detalles.
Colaboradores	- Tkinter

Cuadro 12.17. CRC 4.

Cuadro 20

CRC para Valor de Integridad.

Clase	**Valor de Integridad**
Responsabilidades	- Desarrollar la lógica para validar la integridad del archivo cifrado y asegurarse de que sea un archivo válido antes de proceder con el descifrado.
Colaboradores	- Cifrador De Archivos - Tkinter

Cuadro 12.18. CRC 5.

Cuadro 21

CRC para Seleccionar Archivos.

Clase	**Seleccionar Archivos**
Responsabilidades	-Implementar la funcionalidad para abrir una ventana de selección de archivos. - Integrar la lógica para filtrar y permitir la selección de archivos cifrados. - Conectar la interfaz gráfica con el código que manejará el archivo seleccionado.
Colaboradores	- Interfaz Gráfica - Tkinter

Cuadro 12.19. CRC 6.

Cuadro 22

CRC para Ver Datos Descifrados.

Clase	**Ver Datos Descifrados**
Responsabilidades	-Desarrollar la interfaz gráfica para mostrar los datos descifrados. - Conectar la interfaz con la lógica que procesa y presenta los datos descifrados. - Asegurarse de que la visualización de datos sea clara y comprensible para el usuario.
Colaboradores	- Interfaz Gráfica - Tkinter

Cuadro 12.20. CRC 7.

Cuadro 23

CRC para Guardar Datos.

Clase	**Guardar Datos**
Responsabilidades	- Crear un botón o funcionalidad para permitir al usuario guardar los datos descifrados. - Implementar la lógica de guardado que tome los datos descifrados y los almacene en un archivo TXT. - Manejar situaciones de error o excepciones durante el proceso de guardado.
Colaboradores	- Interfaz Gráfica - Sistema de Archivos - Tkinter

Cuadro 12.21. CRC 8.

12.4.2.2 Modelo conceptual

Aquí se identificará un modelo preliminar del algoritmo en forma de diagrama de flujo

Diagrama de flujo de encriptado

El primer paso en nuestro enfoque es la creación del algoritmo de encriptado, ver Fig. 12.2. Este diagrama de flujo presenta una secuencia lógica de bloques que representan acciones específicas en el proceso de cifrado. Desde la selección del archivo a cifrar hasta la obtención de la clave privada, cada bloque cumple una función esencial en el flujo de trabajo.

Figura 5
Diagrama de flujo preliminar del algoritmo de cifrado

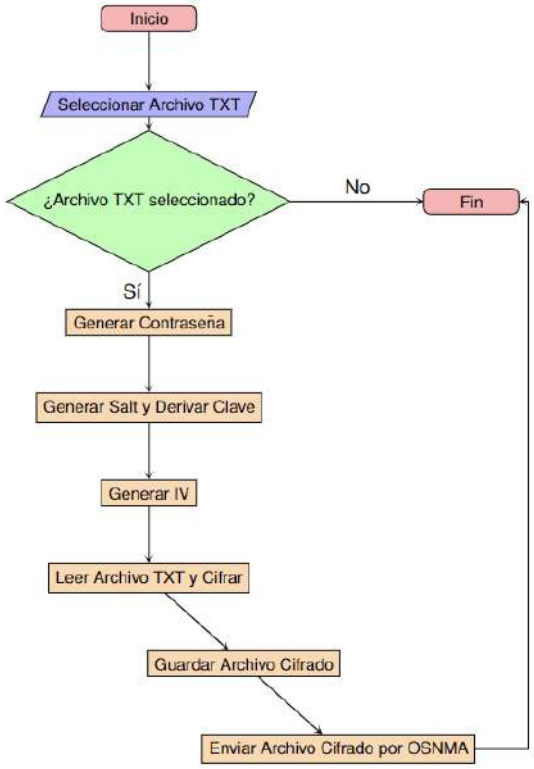


Figura 12.2. Diagrama de flujo encriptado

Diagrama de flujo de desencriptado

El diagrama de flujo para el proceso de desencriptado presenta una secuencia

lógica de operaciones que permiten la recuperación del archivo original a partir del contenido cifrado. Este diagrama sirve como guía para entender las fases clave del desencriptado, ver Fig. 12.3.

Figura 6

Diagrama de flujo preliminar del algoritmo de descifrado

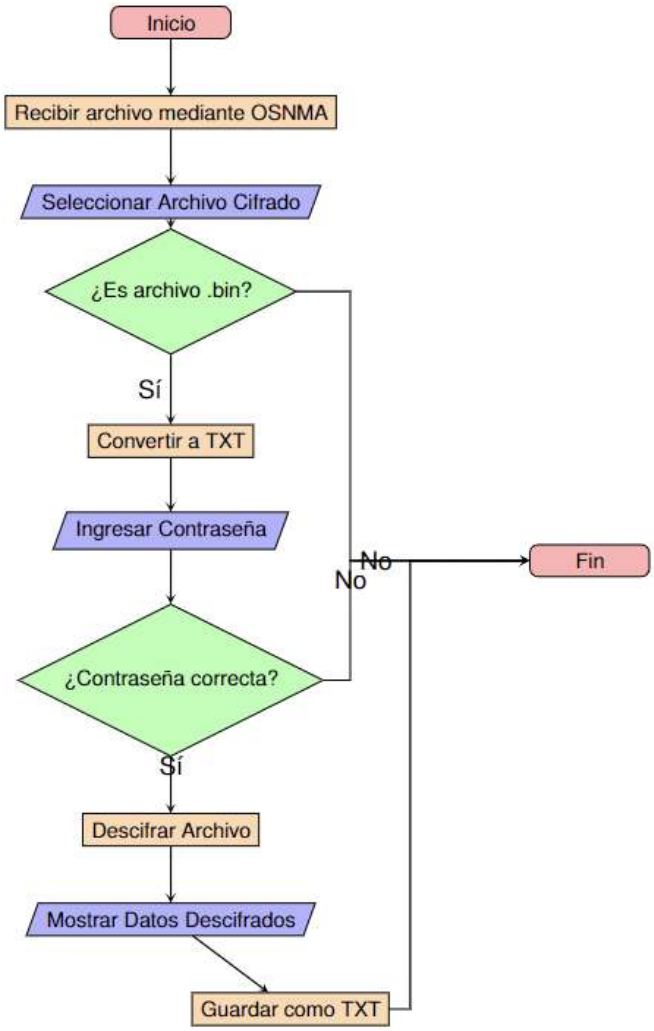


Figura 12.3. Diagrama de flujo desencriptado.

12.4.2.3 Diseño de interfaz de usuario

Las interfaces de la aplicación fueron desarrolladas de acuerdo a los requerimientos dados por el cliente, con el propósito de ofrecer un entorno amigable y fácil de manejar para el usuario.

Ventana de cifrado

Esta pantalla se usara cada vez que se requiera seleccionar un archivo y cifrarlo, es un requisito obligatorio y funcional. Ver Fig. 12.4.

Figura 7
Ventana de cifrado

Figura 12.4. Ventana de cifrado

Ventana de descifrado

Esta pantalla se usara cada vez que se requiera seleccionar y descifrar un archivo, es un requisito obligatorio y funcional. Ver Fig. 12.5.

Figura 8
Ventana de descifrado

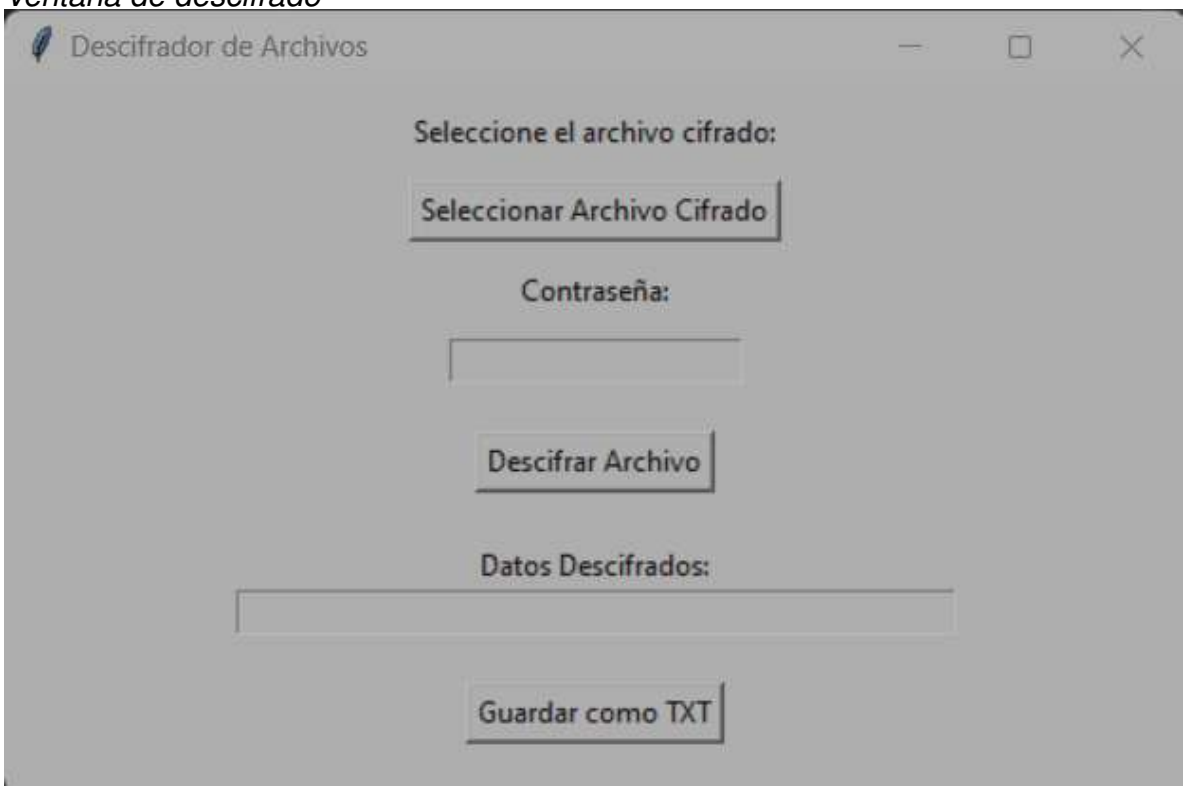


Figura 12.5. Ventana de descifrado

En cada caso, tanto de cifrado como de descifrado, esta ventana aparecerá de manera emergente al momento de que se ejecuta el algoritmo. En la situación de se interactué con los botones de *Seleccionar archivo* o *Seleccionar archivo cifrado*, se abrirá una ventana con acceso a los documentos del equipo.

12.4.3 Codificación

Llamar librerías necesarias para para ejecutar el algoritmo - Se conformo por las siguientes librerías necesarias para poder ejecutar, visualizar, referenciar y utilizar sus atributos y tipos de datos disponibles que nos ofrece Python. Ver Fig. 12.6.

Figura 9

Librerias de Python utilizadas en los algoritmos

```
import os
import secrets
import string
from cryptography.hazmat.primitives.kdf.pbkdf2 import PBKDF2HMAC
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives import padding
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes
from cryptography.hazmat.backends import default_backend
import tkinter as tk
from tkinter import filedialog
```

Figura 12.6. Librerías

Ventana de cifrado - Se creo el modelo de ventana que nos permitirá interactuar con el algoritmo de cifrado, así como seleccionar el archivo a cifrar deseado y la opción de almacenar el archivo cifrado en una carpeta especifica. A su vez, se crearon los modelos de las clases, conformadas por las necesidades, atributos, el tipo de datos y necesarios para el desarrollo del algoritmo. Ver Fig. 12.7.

Figura 10

Código de Interfaz de usuario del algoritmo de cifrado

```
class CifradorApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Cifrador de Archivos")
        self.root.geometry("400x200")  # Ajusta el tamaño de la ventana

        self.file_to_encrypt = tk.StringVar()
        self.generated_password = tk.StringVar()

        tk.Label(root, text="Seleccione el archivo a cifrar (TXT):").pack(pady=10)
        tk.Button(root, text="Seleccionar Archivo", command=self.seleccionar_archivo).pack()

        tk.Label(root, text="Contraseña generada automáticamente:").pack(pady=10)
        tk.Entry(root, textvariable=self.generated_password, state='readonly').pack()

        tk.Button(root, text="Cifrar Archivo", command=self.cifrar_archivo).pack(pady=20)

    def seleccionar_archivo(self):
        file_path = filedialog.askopenfilename(filetypes=[("Archivos TXT", "*.txt")])
        self.file_to_encrypt.set(file_path)

    def cifrar_archivo(self):
        file_to_encrypt = self.file_to_encrypt.get()
        if not file_to_encrypt:
            print("No se seleccionó un archivo TXT.")
            return
```

Figura 12.7. Interfaz de usuario del algoritmo de cifrado

Función de cifrado - Seguido, se desarrollo esta aplicación que tiene como objetivo principal ofrecer a los usuarios la capacidad de cifrar archivos de texto, brindándoles una capa adicional de seguridad para sus datos sensibles, ver Fig. 12.8. La funcionalidad clave se centra en la selección de un archivo de texto, generación automática de una contraseña robusta, y la aplicación de cifrado utilizando el algoritmo *Advanced Encryption Standard* (AES) en modo *Cipher Feedback* (CFB). Además, se integran medidas de seguridad como la derivación de claves mediante *Password-Based Key Derivation Function 2* (PBKDF2) con *Hash Message Authentication Code* (HMAC) utilizando el algoritmo SHA256.

Figura 11

Código de algoritmo de cifrado

```
    # Cifrar el archivo TXT
    self.cifrar(file_to_encrypt)

def cifrar(self, file_to_encrypt):
    # Solicitar al usuario la ubicación y nombre donde se guardara el archivo cifrado (.bin)
    encrypted_file = filedialog.asksaveasfilename(defaultextension=".bin", filetypes=[("Archivos Binarios", "*.bin")])

    # Generar una contraseña aleatoria de 16 caracteres
    contrasena_aleatoria = generar_contrasena_aleatoria(16)  # Aumenta la longitud de la contraseña
    password = contrasena_aleatoria.encode('utf-8')

    # Mostrar la contraseña generada al usuario
    self.generated_password.set(contrasena_aleatoria)  # Muestra la contraseña generada

    # Generar una sal aleatoria de 16 bytes
    salt = os.urandom(16)

    # Derivar una clave a partir de la contraseña y la sal utilizando PBKDF2HMAC con SHA256
    kdf = PBKDF2HMAC(
        algorithm=hashes.SHA256(),
        iterations=100000,
        salt=salt,
        length=32,
        backend=default_backend()
    )
    key = kdf.derive(password)

    # Generar un vector de inicialización (iv) aleatorio de 16 bytes
    iv = os.urandom(16)

    # Leer el contenido del archivo a cifrar en modo binario
    with open(file_to_encrypt, 'rb') as file:
        data = file.read()

    # Configurar el cifrador AES en modo CFB
    cipher = Cipher(algorithms.AES(key), modes.CFB(iv), backend=default_backend())
    encryptor = cipher.encryptor()
    # Aplicar relleno PKCS7 al contenido del archivo
    padder = padding.PKCS7(128).padder()
    padded_data = padder.update(data) + padder.finalize()
    # Realizar el cifrado y obtener el texto cifrado
    ciphertext = encryptor.update(padded_data) + encryptor.finalize()

    # Escribir la sal, el vector de inicialización y el texto cifrado en el archivo de salida
    with open(encrypted_file, 'wb') as file:
        file.write(salt + iv + ciphertext)

    print("Archivo cifrado guardado en:", encrypted_file)
```

Figura 12.8. Algoritmo de cifrado

Clases primarias - Se crearon las siguientes funciones que son las que se encargarán de generar la contraseña aleatoria con caracteres alfanuméricos y crea una instancia de la clase CifradorApp y lanza la interfaz gráfica de la aplicación, ver Fig. 12.9.

Figura 12

Código del generador de contraseña y el lanzador de GUI

```
def generar_contrasena_aleatoria(longitud):
    # Esta función genera y devuelve una contraseña aleatoria de la longitud especificada.
    caracteres = string.ascii_letters + string.digits + string.punctuation
    contrasena = ''.join(secrets.choice(caracteres) for _ in range(longitud))
    return contrasena

if __name__ == "__main__":
    # Crea una instancia de la clase CifradorApp y lanza la aplicación GUI.
    root = tk.Tk()
    app = CifradorApp(root)
    root.mainloop()
```

Figura 12.9. Contraseña y main

Ventana de descifrado - El siguiente código presenta una implementación de un "Descifrador de Archivos"mediante una interfaz gráfica construida en Python con la biblioteca Tkinter, ver Fig. 12.10. Este programa proporciona una herramienta visual para seleccionar, descifrar y guardar archivos descifrados, brindando una solución amigable y accesible para los usuarios.

Figura 13

Código Interfaz gráfica de descifrado

```
# Definición de la clase DescifradorApp
class DescifradorApp:
    def __init__(self, root):
        # Inicialización del objeto DescifradorApp con la ventana root proporcionada
        self.root = root
        self.root.title("Descifrador de Archivos")  # Título de la ventana
        self.root.geometry("500x300")  # Ajuste del tamaño de la ventana

        # Variables de control para almacenar rutas y datos cifrados/descifrados
        self.encrypted_file = tk.StringVar()
        self.decrypted_data = tk.StringVar()

        # Elementos de la interfaz gráfica
        tk.Label(root, text="Seleccione el archivo cifrado:").pack(pady=10)
        tk.Button(root, text="Seleccionar Archivo Cifrado", command=self.seleccionar_archivo).pack()

        tk.Label(root, text="Contraseña:").pack(pady=10)
        tk.Entry(root, show="*", textvariable=self.encrypted_file, state='readonly').pack()

        tk.Button(root, text="Descifrar Archivo", command=self.descifrar_archivo).pack(pady=20)

        tk.Label(root, text="Datos Descifrados:").pack()
        tk.Entry(root, textvariable=self.decrypted_data, state='readonly', width=50).pack()

        tk.Button(root, text="Guardar como TXT", command=self.guardar_como_txt).pack(pady=20)

    # Método para seleccionar un archivo cifrado
    def seleccionar_archivo(self):
        file_path = filedialog.askopenfilename(filetypes=[("Archivos TXT", "*.bin")])
        self.encrypted_file.set(file_path)
```

Figura 12.10. Interfaz gráfica de descifrado

Seleccionar archivo - El bloque mencionado nos permitirá escoger el archivo

en bin, para proceder a descifrar, ver Fig. 12.11.

Figura 14

Código Selección de archivo

```
# Método para seleccionar un archivo cifrado
def seleccionar_archivo(self):
    file_path = filedialog.askopenfilename(filetypes=[("Archivos TXT", "*.bin")])
    self.encrypted_file.set(file_path)
```

Figura 12.11. Selección de archivo

Descifrar archivo - Los siguientes pasos involucran la separación del contenido cifrado en componentes esenciales, como la sal, el vector de inicialización y el texto cifrado. Luego, mediante la función PBKDF2HMAC, se deriva una clave de cifrado utilizando la contraseña proporcionada y la sal. Con la clave derivada, se configura un cifrador AES en modo CFB para llevar a cabo el proceso de descifrado, ver Fig. 12.12.

Figura 15

Código de descifrado de archivo

```
# Método para descifrar un archivo
def descifrar_archivo(self):
    encrypted_file = self.encrypted_file.get()

    # Solicitar contraseña al usuario
    password = simpledialog.askstring("Contraseña", "Ingrese la contraseña:")

    # Leer el contenido del archivo cifrado
    with open(encrypted_file, 'rb') as file:
        file_content = file.read()

    # Verificar que hay suficientes bytes en el contenido del archivo
    if len(file_content) < 32:
        print("Error: Archivo cifrado no válido.")
        return

    # Separar la sal, el vector de inicialización y el texto cifrado
    salt = file_content[:16]
    iv = file_content[16:32]
    ciphertext = file_content[32:]

    # Derivar una clave a partir de la contraseña y la sal utilizando PBKDF2HMAC
    kdf = PBKDF2HMAC(
        algorithm=hashes.SHA256(),
        iterations=100000,
        salt=salt,
        length=32,
        backend=default_backend()
    )
    key = kdf.derive(password.encode('utf-8'))

    # Configurar el cifrador AES en modo CFB y descifrar el archivo
    cipher = Cipher(algorithms.AES(key), modes.CFB(iv), backend=default_backend())
    decryptor = cipher.decryptor()
    decrypted_padded_data = decryptor.update(ciphertext) + decryptor.finalize()

    # Deshacer el relleno PKCS7 y obtener los datos originales
    unpadder = padding.PKCS7(128).unpadder()
    original_data = unpadder.update(decrypted_padded_data) + unpadder.finalize()

    # Establecer los datos descifrados en la interfaz gráfica
    self.decrypted_data.set(original_data.decode('utf-8'))
```

Figura 12.12. Descifrado de archivo

Función de descifrado y guardado - Cuando esta función se invoca, primero recupera los datos descifrados de la interfaz. A continuación, solicita al usuario la ubicación y el nombre del archivo donde desea guardar la información descifrada. Utilizando la biblioteca *filedialog*, se abre una ventana de diálogo que facilita la selección o creación de un archivo de texto, ver Fig. 12.13.

Posteriormente, los datos descifrados se escriben en el archivo .txt seleccionado, asegurando que el contenido se almacene de manera segura y legible.

Figura 16

Código de guardado de archivo descifrado

```
    # Método para guardar los datos descifrados como archivo TXT
    def guardar_como_txt(self):
        decrypted_data = self.decrypted_data.get()

        # Preguntar al usuario dónde guardar el archivo TXT
        txt_file_path = filedialog.asksaveasfilename(defaultextension=".txt", filetypes=[("Archivos de Texto", "*.txt")])

        # Escribir los datos descifrados en el archivo TXT
        with open(txt_file_path, 'w', encoding='utf-8') as txt_file:
            txt_file.write(decrypted_data)

        print(f"Datos descifrados guardados en: {txt_file_path}")

# Bloque principal para ejecutar la aplicación si se ejecuta el script de manera independiente
if __name__ == "__main__":
    root = tk.Tk()
    app = DescifradorApp(root)
    root.mainloop()
```

Figura 12.13. Guardado de archivo descifrado

12.4.4 Pruebas

Las pruebas no solo son esenciales para cumplir con los requisitos de las historias de usuario, sino que también ofrecen una valiosa oportunidad para evaluar si la implementación se alinea verdaderamente con las expectativas. Durante este proceso, se llevan a cabo pruebas específicas centradas en la funcionalidad de cada iteración previamente definida. Estas pruebas no solo aseguran la conformidad con los objetivos establecidos, sino que también proporcionan retroalimentación crucial sobre la efectividad y la precisión de la implementación en relación con las necesidades reales del usuario.

Cuadro 24

Prueba de aceptación H1.

Pruebas de aceptación	
Código: P1	**Código de historia:** H1
Descripción	- Verificar que los usuarios puedan seleccionar un archivo de texto (TXT). - Asegurarse de que el sistema permita la selección de archivos válidos.
Condición de ejecución	- Confirmar que la funcionalidad de selección de archivo esté activa. - Validar que el sistema maneje adecuadamente errores de selección de archivo.
Entrada	- Seleccionar un archivo de texto existente. - Intentar seleccionar un archivo que no sea de texto (por ejemplo, un archivo binario).
Resultado	- Verificar que el archivo seleccionado se cargue correctamente. - Confirmar que el sistema informe de manera adecuada si hay un problema con la selección del archivo.
Evaluación de prueba	- Asegurarse de que la selección de archivos sea intuitiva para los usuarios. - Verificar que el sistema maneje correctamente diferentes casos de entrada.

Cuadro 12.22. Prueba H1

Cuadro 25

Prueba de aceptación H2.

Pruebas de aceptación	
Código: P2	**Código de historia:** H2
Descripción	- Confirmar que el sistema pueda generar una contraseña aleatoria. - Asegurarse de que la contraseña generada sea segura.
Condición de ejecución	- Verificar que la generación de contraseña ocurra de manera automática. - Asegurarse de que la contraseña cumpla con criterios de seguridad establecidos.
Entrada	- No se requiere entrada directa del usuario para la generación de la contraseña.
Resultado	- Verificar que se genere una contraseña aleatoria y segura.
Evaluación de prueba	- Confirmar que la contraseña generada cumpla con estándares de seguridad. - Asegurarse de que la generación automática sea rápida y eficiente.

Cuadro 12.23. Prueba H2

Cuadro 26

Prueba de aceptación H3.

Pruebas de aceptación	
Código: P3	**Código de historia:** H3
Descripción	- Verificar que los usuarios puedan visualizar la contraseña generada automáticamente antes de cifrar un archivo.
Condición de ejecución	- Asegurarse de que la contraseña sea visible antes de proceder con el cifrado. -Confirmar que la visualización sea segura y no permita la copia no autorizada.
Entrada	- No se requiere entrada directa del usuario para la visualización de la contraseña.
Resultado	- Confirmar que la contraseña se muestra claramente y de forma segura.
Evaluación de prueba	- Asegurarse de que la visualización de la contraseña sea intuitiva y comprensible para los usuarios.

Cuadro 12.24. Prueba H3

Cuadro 27

Prueba de aceptación H4.

Pruebas de aceptación	
Código: P4	**Código de historia:** H4
Descripción	- Verificar que los usuarios puedan cifrar el archivo TXT seleccionado utilizando la contraseña generada automáticamente.
Condición de ejecución	- Confirmar que la opción de cifrado esté activa después de seleccionar un archivo y generar la contraseña.
Entrada	- Seleccionar un archivo de texto. - Utilizar la contraseña generada automáticamente.
Resultado	- Asegurarse de que el archivo se cifre correctamente. - Confirmar que el archivo cifrado sea seguro y no sea accesible sin la contraseña.
Evaluación de prueba	-Verificar que el proceso de cifrado sea transparente y fácil de entender para los usuarios.

Cuadro 12.25. Prueba H4

Cuadro 28

Prueba de aceptación H5.

Pruebas de aceptación	
Código: P5	**Código de historia:** H5
Descripción	- Verificar que el archivo cifrado se guarde en un formato bin.
Condición de ejecución	- Confirmar que la opción de guardar en formato binario esté activa después de cifrar el archivo.
Entrada	- Seleccionar la opción de guardar en formato binario después de cifrar un archivo.
Resultado	- Asegurarse de que el archivo cifrado se guarde correctamente en un formato binario.
Evaluación de prueba	-Confirmar que la opción de guardar en formato binario garantiza la integridad y seguridad del archivo cifrado.

Cuadro 12.26. Prueba H5

Cuadro 29

Prueba de aceptación H6.

Pruebas de aceptación	
Código: P6	**Código de historia:** H6
Descripción	- Verificar que los usuarios puedan seleccionar un archivo cifrado para descifrar su contenido.
Condición de ejecución	- Confirmar que la opción de selección de archivo cifrado esté activa. - Asegurarse de que el sistema maneje correctamente archivos cifrados válidos.
Entrada	- Seleccionar un archivo cifrado existente. - Intentar seleccionar un archivo que no esté cifrado.
Resultado	- Verificar que el archivo cifrado seleccionado sea reconocido por el sistema. - Confirmar que el sistema informe si el archivo seleccionado no es válido.
Evaluación de prueba	- Asegurarse de que la selección de archivos cifrados sea clara para los usuarios. - Verificar la manejo adecuado de casos de entrada no válidos.

Cuadro 12.27. Prueba H6

Cuadro 30

Prueba de aceptación H7.

Pruebas de aceptación	
Código: P7	**Código de historia:** H7
Descripción	- Validar que los usuarios puedan ingresar la contraseña necesaria para descifrar el archivo seleccionado.
Condición de ejecución	- Confirmar que la interfaz permita la entrada de la contraseña para descifrar. - Asegurarse de que el sistema maneje la validación de contraseñas de manera segura.
Entrada	- Ingresar la contraseña correcta para descifrar el archivo. - Intentar descifrar con una contraseña incorrecta.
Resultado	- Verificar que el archivo se descifre correctamente con la contraseña correcta. - Confirmar que el sistema informe si la contraseña ingresada es incorrecta.
Evaluación de prueba	- Asegurarse de que la interfaz de entrada de contraseña sea intuitiva y segura. - Verificar la correcta validación de contraseñas.

Cuadro 12.28. Prueba H7

Cuadro 31

Prueba de aceptación H8.

Pruebas de aceptación	
Código: P8	**Código de historia:** H8
Descripción	- Validar que los usuarios puedan ver los datos descifrados en la interfaz gráfica después de descifrar un archivo.
Condición de ejecución	- Confirmar que la interfaz muestre los datos descifrados de manera clara y legible. - Asegurarse de que la visualización sea segura y no permita acceso no autorizado.
Entrada	- No se requiere entrada directa del usuario para la visualización de datos descifrados.
Resultado	- Verificar que los datos descifrados se muestren correctamente en la interfaz.
Evaluación de prueba	- Asegurarse de que la visualización de datos descifrados sea fácil de entender para los usuarios.

Cuadro 12.29. Prueba H8

Cuadro 32

Prueba de aceptación H9.

Pruebas de aceptación	
Código: P9	**Código de historia:** H9
Descripción	- Validar que los usuarios tengan la opción de guardar los datos descifrados como un archivo de texto (TXT).
Condición de ejecución	- Confirmar que la opción de guardar como archivo TXT esté activa después de descifrar los datos. - Asegurarse de que el sistema maneje correctamente la operación de guardado.
Entrada	- Seleccionar la opción de guardar como archivo TXT después de descifrar los datos.
Resultado	- Verificar que los datos descifrados se guarden correctamente como un archivo TXT.
Evaluación de prueba	- Confirmar que la opción de guardar como archivo TXT sea intuitiva para los usuarios. - Verificar que el sistema maneje adecuadamente la operación de guardado.

Cuadro 12.30. Prueba H9

13. Análisis de Resultados

13.1 Análisis de los algoritmos propuestos

13.1.1 Análisis del algoritmo de cifrado

El algoritmo de enciptado desarrollado para este caso de estudio nació de implementar el cifrado simétrico, específicamente el AES (*Advanced Encryption Standard*) en modo CFB (*Cipher Feedback*). Este algoritmo se diseño para agregar una capa adicional de seguridad a la comunicación de los satélites Galileo, especifícamele para el sistema OSNMA, proporcionando un alto nivel de seguridad para la transición de datos de importancia para el usuario final.

Para lograr su implementación se utilizó el lenguaje de programación Python y la librería *cryptography*. Buscando desarrollar una aplicación de archivos con una interfaz gráfica de usuario para facilitar su uso y demostrar su funcionalidad práctica.

La aplicación permite al usuario seleccionar un archivo de texto plano (.txt) para ser cifrado. Una vez seleccionado el archivo, se genera automáticamente una contraseña aleatoria de 16 caracteres y se muestra al usuario. Esta contraseña se utiliza como clave para el algoritmo de cifrado.

Para generar la contraseña aleatoria se utiliza el modulo *secrets* de el lenguaje de programación antes mencionado, este módulo nos proporciona una serie de funciones seguras para generar números de manera aleatoria. Dichos números generan la contraseña con una combinación de letras mayúsculas, minúsculas, dígitos y caracteres especiales, lo que garantiza una alta entropía y una resistencia hacia los ataques de fuerza bruta.

Esta contraseña generada se convierte en una secuencia de bytes y se utiliza junto con una sal aleatoria de 16 bits para derivar una clave de cifrado utilizando el algoritmo PBKDF2HMAC (*Password-Based Key Derivation Function 2 with Hash-Based Message Authentication Code*). Dicho proceso de derivación de clave nos garantiza una mayor seguridad contra ataques de diccionario.

Para describir el proceso de encriptado, una vez derivada la clave de cifrado, se genera un vector de inicialización (IV) aleatorio de 16 bits. El contenido del archivo seleccionado se lee en modo binario y se le aplica un relleno PKCS7 (*Public Key Cryptography Standards 7*), que es un estándar de criptografía que define el formato

para la encriptación y desencriptación de mensajes. Especifica un método de relleno (*padding*) para asegurar que los mensajes tengan una longitud adecuada para el algoritmo de cifrado utilizado, para asegurar que tenga una longitud compatible con el bloque de cifrado AES.

Por ultimo el cifrador AES en modo CFB se configura con la clave y el IV generados, y se utiliza para cifrar el contenido del archivo. El texto cifrado junto con la sal y el IV se escribe en un nuevo archivo binario con extensión .bin, que representa el archivo cifrado resultante.

13.1.2 Análisis del algoritmo de descifrado

El algoritmo de descifrado desarrollado en este estudio es una contraparte del algoritmo de cifrado presentado anteriormente. Está diseñado para desencriptar archivos cifrados utilizando el algoritmo AES en modo CFB, utilizando la contraseña proporcionada por el usuario durante el proceso de descifrado.

La implementación del algoritmo de descifrado se lleva a cabo en una aplicación de descifrado de archivos con una interfaz gráfica de usuario (GUI). La aplicación permite al usuario seleccionar un archivo cifrado (.bin) y proporcionar la contraseña necesaria para el proceso de descifrado.

El proceso de descifrado comienza con la selección del archivo cifrado y la introducción de la contraseña por parte del usuario. La contraseña proporcionada se utiliza junto con la sal almacenada en el archivo cifrado para derivar la clave de descifrado utilizando el algoritmo PBKDF2HMAC con SHA256.

Donde SHA-256 es un algoritmo de hash criptográfico que pertenece a la familia de funciones *Secure Hash Algorithm* (SHA). El propósito principal de un algoritmo de hash como SHA-256 es tomar una entrada (o "mensaje") de cualquier tamaño y producir una salida de longitud fija (256 bits o 32 bytes en el caso de SHA-256) que es única para cada entrada. Este resultado es conocido como el "hash.º resumen criptográfico"del mensaje.

Una vez que se ha derivado la clave de descifrado, se utiliza para configurar el cifrador AES en modo CFB. El vector de inicialización (IV) y el texto cifrado se extraen del archivo cifrado, y se utiliza el cifrador configurado para descifrar el texto cifrado y obtener los datos originales.

Los datos descifrados pueden contener relleno PKCS7, que se aplica durante el proceso de cifrado para asegurar una longitud de bloque adecuada. Antes de pre-

sentar los datos descifrados al usuario, se deshace el relleno PKCS7 para obtener los datos originales.

13.2 Análisis de tiempo

13.2.1 Análisis del tiempo de cifrado

Para evaluar la eficiencia del algoritmo de cifrado implementado en la aplicación, se realizó un análisis de tiempo durante el proceso de cifrado de archivos. La prueba consistió en medir el tiempo que tarda el algoritmo en cifrar un archivo de texto plano de tamaño variable. Se utilizaron archivos de texto con extensiones .txt de diferentes tamaños, desde pequeños archivos hasta archivos más grandes, representando distintos escenarios de uso.

El experimento arrojo los siguientes resultados:

- Los tamaños de los conjuntos de datos varían desde 2 KB hasta 27,204 KB, lo que indica una amplia gama de complejidades en los datos procesados. Buscando la eficiencia del algoritmo en distintos casos de aplicación.

- Los tiempos de cifrado varían desde 5.41 segundos hasta 13.93 segundos. Esto sugiere que el tiempo de cifrado no está directamente correlacionado con el tamaño del conjunto de datos, ya que algunos conjuntos de datos más pequeños pueden tener tiempos de cifrado más largos que algunos conjuntos de datos más grandes.

- Aunque generalmente se podría esperar que los conjuntos de datos más grandes requieran más tiempo de cifrado, esto no siempre es el caso en este cuadro. Por ejemplo, "Plan institucional"(26 KB) tiene un tiempo de cifrado de 6.68 segundos, mientras que "Weather archive from 2020 to 2023"(746 KB) tiene un tiempo de cifrado de 8.72 segundos.

- Algunos conjuntos de datos, como "Tiraderos-Clandestinos"(13.93 segundos) y "Servicios turisticos en CDMX"(13.90 segundos), tienen tiempos de cifrado significativamente más largos en comparación con otros conjuntos de datos de tamaño similar. Estos podrían ser considerados como valores atípicos que merecen una investigación adicional para comprender las razones detrás de estos tiempos de cifrado más largos.

Los resultados de la prueba se muestran en el cuadro 13.1, donde se puede observar que el análisis de los datos de tiempo de cifrado revela una variedad significativa en los tiempos registrados para diferentes conjuntos de datos. Aunque se esperaría que los conjuntos de datos más grandes requieran más tiempo de cifrado, esto no siempre es el caso. Factores adicionales, como el tipo de datos dentro del conjunto de datos, también juegan un papel crucial en el tiempo necesario para cifrar o descifrar la información.

Por ejemplo, los conjuntos de datos que contienen solo datos numéricos pueden tener tiempos de cifrado más cortos en comparación con aquellos que contienen una variedad de caracteres, como letras o caracteres especiales. La presencia de datos diversos puede afectar negativamente el tiempo de cifrado debido al procesamiento adicional requerido.

Además, el tamaño del conjunto de datos influye en el tiempo de procesamiento, siendo más grande el dataset, mayor será el procesamiento necesario. Este factor puede limitar la capacidad de experimentar con conjuntos de datos más grandes debido a las limitaciones técnicas del equipo utilizado para realizar las pruebas. Así, se destaca la importancia de considerar múltiples factores, además del tamaño del conjunto de datos, al evaluar y comparar los tiempos de cifrado en diferentes escenarios.

Cuadro 33

Análisis de Tiempo de Cifrado.

Dataset	Tamaño (KB)	Tiempo de Cifrado (segundos)
Python Future	2	6.91
Populated countries	3	10.31
Iris	5	6.78
Gendergapin average new	10	6.36
Diabetes	24	5.41
Plan institucional	26	6.68
RAW	27	10.23
Stock Price	30	6.63
Onlinefoods	35	5.92
Fresher jobs	59	6.11
Medical insurance	113	7.55
Tiraderos-Clandestinos	151	10.37
Number of internet users	158	8.67
Redmet	174	9.47
Imdb top 2000 movies	198	10.61
Laptops	199	11.35
Fighter stats	203	7.38
Servicios turisticos en CDMX	273	6.27
Bollywood Movie List (1920-2024)	320	8.97
Areas naturales protegidas	492	12.46
Dataset Python Question Answer	705	12.81
Weather archive from 2020 to 2023	746	8.72
Salaries	783	8.38
Suelo de conservación	789	10.49
Data Science salaries 2024	831	7.54
INT KenPom Summary	878	8.81
Afluencia rtp desglosado	936	9.02
Gemma dataset prompt recover 2	3,605	7.44
Fakenews	22,812	9.51
Hechos de tránsito 2023	27,204	10.15

Cuadro 13.1. Análisis de los tiempos de Cifrado

Figura 17

Correlación entre tamaño y tiempo de cifrado

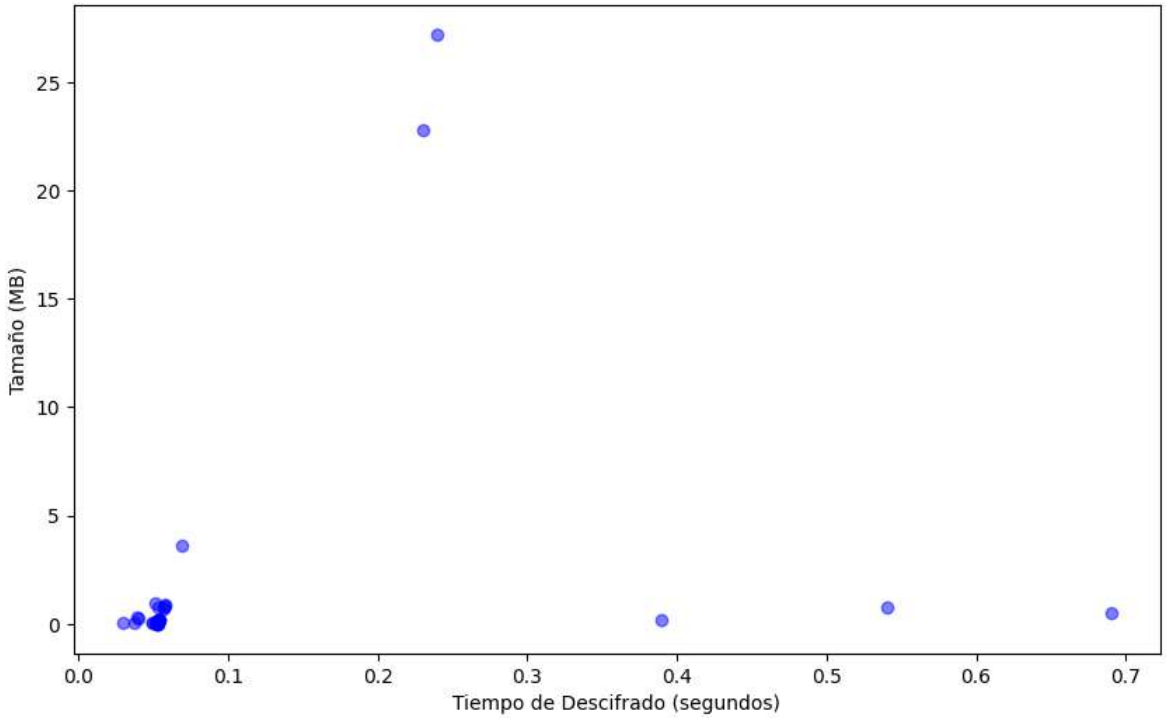


Figura 13.1. Correlación de cifrado

Con los datos arrojados por el cuadro 13.1, se puede calcular los siguientes resultados, ver Fig. 13.1:

En donde de acuerdo a los del análisis estadístico del tiempo de cifrado revelan una distribución significativa en los datos. La mediana, que representa el valor central de la muestra, se sitúa en 8.765 segundos, lo que indica que el 50 % de los tiempos de cifrado son superiores a este valor.

La media aritmética, que representa el promedio de los tiempos, es de 8.95 segundos, evidenciando una leve tendencia al alza en comparación con la mediana.

La moda, que identifica el valor más frecuente en la muestra, se registra en 5.41 segundos, sugiriendo que este tiempo es el más recurrente en los datos observados. El rango, que representa la diferencia entre el valor máximo y mínimo en la muestra, es de 8.52 segundos, indicando una variabilidad moderada en los tiempos de cifrado.

Por otro lado, el índice de correlación entre el tamaño del dataset y el tiempo de cifrado es de aproximadamente 0.107, lo que indica una correlación débil entre estas

dos variables. Estos datos nos indican que si bien existe una relación positiva entre el tamaño del dataset y el tiempo de cifrado, esta relación no es significativamente fuerte, lo que sugiere que otros factores pueden influir como el tipo de dato que se cifre o la capacidad de procesamiento del equipo en el que se realizo el experimento, con en el tiempo de cifrado además del tamaño del dataset.

13.2.2 Análisis del tiempo de descifrado

Los resultados del experimento de descifrado mostrados en el cuadro 13.2 nos revelan que:

- Se observó una variación significativa en los tiempos de descifrado entre diferentes conjuntos de datos, con valores que oscilan entre 0.03 y 0.69 segundos.
- Los conjuntos de datos más pequeños, como Python Future, Iris y Populated countries, tienden a tener tiempos de descifrado más bajos, alrededor de 0.05 segundos o menos, pero con sus excepciones, como el dataset "Laltops", con un tamaño de 199 KB y tan solo 0.39 segundos de tiempo de descifrado.
- Los tiempos de descifrado tienden a aumentar a medida que el tamaño del conjunto de datos aumenta, pero no de manera lineal.
- A mayor número de datos, mayor tiempo de descifrado como Fakenews y Hechos de tránsito 2023, tienen tiempos de descifrado significativamente más altos, de 0.23 a 0.24 segundos, respectivamente.
- El conjunto de datos Gemma dataset prompt recover 2 destaca como una excepción extrema, con un tamaño de 3,605 KB y un tiempo de descifrado de 0.069 segundos, mucho más alto que otros conjuntos de datos de tamaño similar.

Los datos revelan una variabilidad considerable en los tiempos de descifrado, donde conjuntos pequeños como Python Future muestran tiempos breves, mientras que otros, como Gemma dataset prompt recover 2, muestran tiempos más largos, sugiriendo una complejidad inherente. Esta complejidad puede atribuirse al tipo de caracteres presentes en los conjuntos de datos y a las limitaciones de procesamiento del equipo.

La presencia de diversos tipos de caracteres, como letras y caracteres especiales, puede impactar negativamente en el tiempo de descifrado debido al procesamiento adicional requerido. Además, la cantidad de datos a descifrar también influye significativamente. Conjuntos de datos más grandes necesitan más tiempo de procesamiento, lo que puede limitar la capacidad de experimentar con archivos aún más grandes debido a las restricciones técnicas del equipo utilizado en las pruebas. Esta complejidad subyacente resalta la importancia de considerar tanto el tamaño como la naturaleza de los datos al evaluar el rendimiento del algoritmo de descifrado.

Cuadro 34

Análisis de Tiempo de Descifrado.

Dataset	Tamaño (KB)	Tiempo de Descifrado (segundos)
Python Future	2	0.052
Populated countries	3	0.053
Iris	5	0.052
Gendergapin average new	10	0.053
Diabetes	24	0.03
Plan institucional	26	0.037
RAW	27	0.05
Stock Price	30	0.052
Onlinefoods	35	0.05
Fresher jobs	59	0.051
Medical insurance	113	0.052
Tiraderos-Clandestinos	151	0.054
Numberofinternetusers new	158	0.054
Redmet	174	0.053
Imdb top 2000 movies	198	0.053
Laptops	199	0.39
Fighter stats	203	0.054
Servicios turisticos en CDMX	273	0.04
Bollywood Movie List (1920-2024)	320	0.039
Áreas naturales protegidas	492	0.69
Dataset Python Question Answer	705	0.057
Weather archive from 2020 to 2023	746	0.54
Salaries	783	0.057
Suelo de conservación	789	0.053
Data Science salaries 2024	831	0.058
INT KenPom Summary	878	0.058
Afluencia rtp desglosado	936	0.051
Gemma dataset prompt recover 2	3,605	0.069
Fakenews	22,812	0.23
Hechos de tránsito 2023	27,204	0.24

Cuadro 13.2. Análisis de Tiempo de Descifrado

Con los datos del cuadro 13.2, se puede realizar la siguiente correlación entre el tiempo de descifrado y el tamaño del archivo, ver Fig. 13.2:

Figura 18
Correlación entre tamaño y tiempo de descifrado

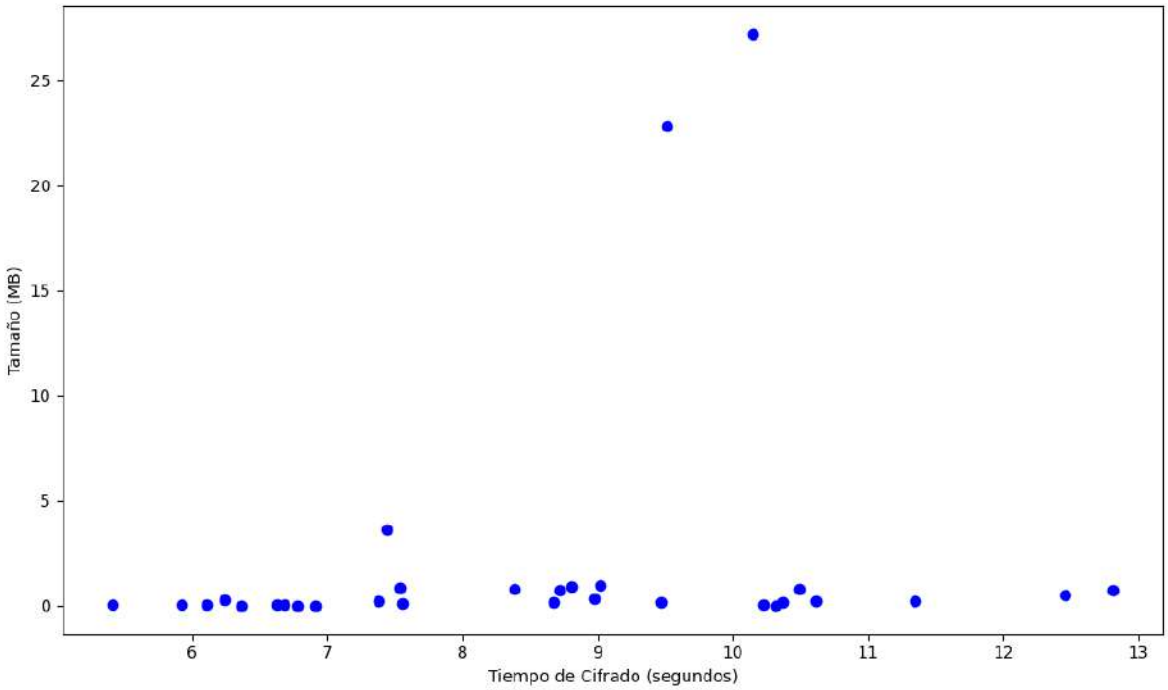


Figura 13.2. Correlación entre tamaño y tiempo de descifrado

Donde al realizar el análisis estadístico se revela que la información importante sobre el tiempo de descifrado de los datos evaluados. empezando con los de tendencia central empezando con la mediana, que al ser una medida de tendencia central robusta, se sitúa en 0.053 segundos, lo que indica que la mitad de los tiempos de descifrado se encuentran por encima de este valor y la otra mitad por debajo. Respaldado por la moda, que también es de 0.053 segundos, sugiriendo que este es el tiempo de descifrado más comúnmente observado en los datos.

La media del tiempo de descifrado es de aproximadamente 0.1124 segundos, ofreciendo una idea del tiempo de descifrado promedio para los diferentes conjuntos de datos. Sin embargo, es importante recordar que la media puede estar sesgada de acuerdo al tipo de dato con el que se este trabajando.

El rango del tiempo de descifrado es de aproximadamente 0.66 segundos, lo que indica la variabilidad en los tiempos de descifrado observados. Este valor representa la diferencia entre el tiempo de descifrado más largo y el más corto registrado entre

los conjuntos de datos evaluados.

Finalmente, el índice de correlación entre el tamaño del conjunto de datos y el tiempo de descifrado es de aproximadamente 0.218. Esta correlación positiva débil sugiere que, en general, a medida que aumenta el tamaño del conjunto de datos, tiende a aumentar también el tiempo de descifrado. Sin embargo, la relación no es muy fuerte, lo que indica que otros factores pueden influir en los tiempos de descifrado además del tamaño del conjunto de datos.

13.3 Eficiencia por tiempo de ejecución

Existen dos métodos mencionados dentrpo del fundamento teórico, RC5 y IDEA, de los cuales no se han desarrollado las bibliotecas necesarias para su uso en Python. Esto limita su implementación en este experimento, principalmente porque uno de los requisitos es que sea compatible con el sistema OSNMA, el cual está programado en Python.

Se realizaron pruebas utilizando tres de los algoritmos de cifrado presentados dentro de los fundamentos teóricos: DES, 3DES y AES. De estos, AES fue seleccionado para el desarrollo de la propuesta, demostrando superioridad en el tiempo de cifrado. Aunque su tiempo de descifrado fue similar al de 3DES, al promediar los tiempos, AES resultó ser mejor que los otros dos.

Como se puede observar en el cuadro 13.3, se seleccionó un conjunto de datos para realizar esta comparación. El conjunto de datos utilizado fue "Servicios turísticos en CDMX", con un tamaño de 273 KB, arrojando los siguientes resultados:

Cuadro 35

Comparación de tiempos de cifrado y descifrado entre DES, 3DES y AES-CFB.

Algoritmo de cifrado	Tiempo de cifrado (seg)	Tiempo de descifrado (seg)
DES	16.3	0.04
3DES	14.8	0.027
AES CFB	6.27	0.04

Cuadro 13.3. Comparación de tiempos

Al comparar los algoritmos DES, 3DES y AES-256 como se muestra en el cuadro 13.4:

Cuadro 36

Comparación de longitudes de clave y rondas entre DES, 3DES y AES-CFB.

Algoritmo	Longitud de Clave	Rondas de Cifrado
DES	56 bits	16 rondas de Feistel
3DES	168 bits (3 claves de 56 bits)	48 rondas (3 rondas de 16)
AES-256	256 bits	14 rondas

Cuadro 13.4. Comparación entre DES, 3DES y AES

Podemos decir que de acuerdo a los resultados obtenidos puedo decir que en términos de seguridad, AES, al contar con un nivel superior debido a su longitud de clave la cual es de 256 bits, significa que es más robusta contra ataques de fuerza bruta en comparación con los de 56 y 168 bits respectivamente.

En cuanto a eficiencia aun cuando AES cuenta con menos rondas, la estructura del diseño permite un proceso de cifrado y descifrado más rápidos y eficaces en comparación con 3DES, que aun que es seguro, es considerablemente mas lento debido a sus 48 rondas.

Dándonos como resultado que AES ha sido el seleccionado para realizar la propuesta en este trabajo de investigación, siendo este la mejor opción entre los algoritmos comparados, brindando seguridad y eficiencia, contando también que al ser el algoritmo más utilizado actualmente por organizaciones a nivel global, eso le asegura relevancia y soporte continuo en un futuro.

13.4 Evaluación de la Integridad de los Datos Descifrados

Para llevar a cabo la evaluación de similitud, es esencial comprender que lo que se cifra y descifra es el contenido mismo, es decir, los caracteres internos del documento. Con este fin, se ha desarrollado un algoritmo diseñado para medir la similitud entre las longitudes de palabras y sus medidas de tendencia central. Los resultados de esta evaluación se presentan de manera detallada en el cuadro 13.5. Este cuadro proporciona una visión general de la integridad de los datos mediante la comparación de la similitud, la cantidad de palabras y las características estadísticas de los archivos analizados.

Cuadro 37

Análisis de similitud, número de palabras y medias de tendencia central de los datos.

Dataset	Similitud%	# de palabras	Media	Moda	Mediana
Python Future	100	101	10.74	11	11
Populated countries	100	264	8.42	9	9
Iris	100	750	4.88	3	3
Gendergapin average new	100	1723	4.61	4	4
Diabetes	100	6921	2.34	2	2
Plan institucional	100	3859	5.66	2	5
RAW	100	3646	6.47	6	6
Stock Price	100	2520	10.82	13	12
Online foods	100	5898	4.99	6	6
Fresher jobs	100	7634	6.57	9	7
Medical insurance	100	19411	4.79	2	4
Tiraderos-Clandestinos	100	20368	6.19	3	5
Numberofinternetusers new	100	26522	4.84	4	4
Redmet	100	15370	10.35	11	11
Imdb top 2000 movies	100	30988	5.47	3	5
Laptops	100	38394	4.28	4	4
Fighter stats	100	41343	3.93	4	4
Servicios turisticos en CDMX	100	39811	5.47	7	6
Bollywood Movie List (1920-2024)	100	52024	5.09	4	5
Áreas naturales protegidas	100	21976	19.21	20	20
Dataset Python Question Answer	100	107964	5.61	3	5
Weather archive from 2020 to 2023	100	122887	4.96	4	4
Salaries	100	170812	3.61	2	3
Suelo de conservación	100	31473	18.93	19	19
Data Science salaries 2024	100	181415	3.64	2	3
INT KenPom Summary	100	148021	5.01	7	4
Afluencia rtp desglosado	100	125679	6.42	9	6
Gemma dataset prompt recover	100	658923	4.49	3	4
Fakenews	100	3308420	4.91	3	4
Hechos de tránsito 2023	100	4476444	5.15	2	4

Cuadro 13.5. Análisis de Integridad de datos

Además, en el cuadro 13.5 se puede observar que todos los conjuntos de datos mostraron una similitud del 100 % lo que indica que no existe perdida alguna ni diferencia significativa entre los datos analizados. Dando como resultado que los conjuntos de datos son consistentes entre sí en términos de las palabras que contienen.

Es de suma importancia mencionar que en la interpretación de el cuadro también aborda la consideración de la similitud del 100 % entre los conjuntos de datos, lo que se refiere a la naturaleza de la simulación realizada en un entorno controlado. Sin embargo, se enfatiza que esta similitud perfecta puede verse comprometida en un entorno real debido a diversos factores externos, como la transmisión a través de sistemas satelitales, interferencias electromagnéticas, variaciones en el tiempo, o errores en la transmisión que podrían alterar la integridad de los archivos.

El objetivo de utilizar 30 Dataset de distintos tamaños permitió tener en cada uno un número diferente de palabras, lo que ayudo a encontrar una variación significativa entre los conjuntos de datos, que van desde unos pocas 101 palabras a evaluar, hasta aproximadamente las 44 millones.

Otra forma de evaluar la eficacia del algoritmo fue calcular las medidas de tendencia central en cuestión a los caracteres que aparecen entre el archivo original y el archivo descifrado, calculando media, moda y mediana. Donde en ambos documentos estas medidas deben ser las mismas para darle una veracidad certera al funcionamiento de la propuesta.

Para abordar estas preocupaciones y garantizar la fiabilidad de los resultados en un entorno operativo, se desarrolló un algoritmo específico capaz de medir la semejanza entre los archivos originales antes de ser cifrados y enviados, y los archivos descifrados recibidos posteriormente. Este algoritmo (anexo II y IV), proporcionará una evaluación más precisa de la integridad de los datos, permitiendo así una mayor confiabilidad en las conclusiones y recomendaciones presentadas en la tesis.

Este análisis de integridad de datos proporciona una visión general de la consistencia y las características de los conjuntos de datos analizados, lo que contribuye a una comprensión más profunda de la calidad y la fiabilidad de la información utilizada en la investigación. Estos hallazgos son fundamentales para garantizar la validez y la robustez de los resultados obtenidos en el estudio.

13.5 Análisis de seguridad en la contraseña

El algoritmo de cifrado genera una contraseña de 16 caracteres por ejemplo ç$<Mxi=obXGtuJe7ƶ para verificar su seguridad y con base en los principios de combinatoria y la teoría de la probabilidad, se puede resaltar que:

- La contraseña incluye una combinación de caracteres alfabéticos en mayúsculas y minúsculas, dígitos numéricos y caracteres especiales. Para calcular el número total de combinaciones posibles para cada carácter en la contraseña, considerando cada conjunto de datos en las siguientes categorías:
  - Letras mayúsculas (26 caracteres)
  - Letras minúsculas (26 caracteres)
  - Dígitos numéricos (10 caracteres)
  - Caracteres especiales (32 caracteres)
- Se utilizó la fórmula de combinatoria para calcular el número total de combinaciones posibles para cada carácter en la contraseña. Luego, se elevó este valor a la longitud de la contraseña para obtener el número total de combinaciones posibles para la contraseña completa.
- Según la teoría de probabilidad nos indica que para adivinar correctamente la contraseña en un intento es igual a la inversa del número total de combinaciones posibles. Por lo tanto, cuanto mayor sea el número de combinaciones posibles, menor será la probabilidad de adivinar la contraseña en un solo intento.

Ahora bien, para poder romper una contraseña de 16 dígitos caracteres como el ejemplo antes mencionado, tomando como referencia los puntos anteriores, se puede calcular que el número de intentos necesarios para encontrar la contraseña es de aproximadamente 37,157,429,083,410,091,685,945,089,785,856. Esto significa que, en promedio, se necesitarían este número de intentos para adivinar correctamente la contraseña utilizando un ataque de fuerza bruta.

Esto demuestra que la contraseña es extremadamente segura y difícil de adivinar debido a su longitud y a la combinación de diferentes tipos de caracteres. Principalmente porque se necesitaría de un equipo muy sofisticado y recursos muy altos para poder generar esa cantidad de procesamiento de información.

13.6 Simulación del algoritmo en el software FreeFlyer

13.6.1 Margen de error en la trasmisión

La siguiente simulación servirá como referencia para entender el funcionamiento, principalmente para comprender que en el sistema existirán perdidas de información por situaciones ajenas al algoritmo, empezando por la calidad del enlace, la cual sera evaluada en base a la tasa de error de bit, BER (*Bit Error Rate*), el cual representa la probabilidad de error en la recepción digital. Para calcular el nivel de BER, deberemos de obtener un cierto nivel de señal del receptor definido por el parámetro C/N_0 o *Carrier to Noise density ratio*. Con este parametro y conociendo el *bitrate* asociado a la transmisión, también se puede caracterizar la calidad e la transmisión de la señal en recepción en términos de energía por bit o E_b/N_0 (Méndez Sampedro, 2023).

Otro parámetro también calculado para comprender la perdida es la relación señal a ruido (SNR) dado que es un un indicador de qué tan comprometido esta el mensaje por el ruido (Bautista, 2023).

Para calcular los valores de BER (Bit Error Rate) y SNR (Signal-to-Noise Ratio), nos basaremos en los datos obtenidos de las especificaciones del satélite y, especialmente, de la antena que se utiliza. Esta antena, diseñada para operar en banda L, tiene la capacidad de transmitir en diversas frecuencias, incluyendo las utilizadas por los satélites Galileo, como 1176.45 MHz (L5), 1227.60 MHz (L2), 1381.05 MHz (L3) y 1575.42 MHz (L1). Estas frecuencias se distribuyen en un total de 10 bandas operativas, divididas en:

- 4 frecuencias en el rango de 1164-1215 MHz (E5A-E5B).
- 3 frecuencias en el rango de 1260-1300 MHz (E6).
- 3 frecuencias en el rango de 1559-1591 MHz (L1).

En nuestro caso práctico, nos enfocaremos en el canal E1 (1559-1591 MHz). La potencia de transmisión se ha registrado en -157.25 dBW. Se recomienda una ganancia mínima de 12 dBi para la antena de banda L, con un ángulo de elevación no inferior a 90°. La distancia promedio entre los satélites y la tierra es de alrededor de 23222 Km. Además, se establece un ancho de banda de 24.552 MHz para el canal

E1. Estos datos proporcionados sientan las bases para el diseño y la implementación de sistemas de comunicación por satélite en el canal E1, y serán fundamentales en el cálculo de la tasa de errores de bits y la relación señal a ruido.

Las formulas a utilizar para calcular los valores de errror aproximado serán las siguientes (CARLOS y ALAN, 2007, Bautista, 2023):

1. **Potencia de transmisión convertida a Watts:**

$$P_{\text{Watts}} = 10^{(P_{\text{dBW}}/10)}$$

2. **Ganancia de la antena del satélite en veces:**

$$G_{\text{antena}} = 10^{(G_{\text{dBi}}/10)}$$

3. **Potencia de la señal recibida en la estación terrena:**

$$\text{Potencia recibida (Watts)} = \text{Potencia de transmisión} \times \left(\frac{\lambda_{\text{sat}}}{4\pi \text{distancia}}\right)^2$$

   - Donde λ_{sat} es la longitud de onda en metros.

4. **Densidad espectral de potencia del ruido:**

$$\text{Potencia del ruido (Watts/Hz)} = kT$$

   - Donde k es la constante de Boltzmann y T es la temperatura en Kelvin.

5. **Potencia del ruido en el ancho de banda:**

$$\text{Potencia del ruido en el ancho de banda (Watts)} = \text{Potencia del ruido} \times \text{Ancho de banda}$$

6. **Relación señal a ruido (SNR):**

$$\text{SNR} = \frac{\text{Potencia recibida}}{\text{Potencia del ruido en el ancho de banda}}$$

7. **Error de transmisión (BER):**

$$\text{BER} = \frac{1}{2} \cdot \text{erfc}\left(\sqrt{\text{SNR}}\right)$$

- Donde erfc es la función de error complementaria.

De acuerdo a las formulas y datos proporcionados se puede hacer un análisis de la señal el cual con una potencia alta de potencia considerablemente alta de aproximadamente $9{,}03 \times 10^{27}$ Watts. Esta alta potencia de señal nos da a entender que es una transmisión robusta y confiable desde el satélite hasta la estación terrena, lo cual es crucial para mantener una comunicación efectiva.

Por otro lado, la potencia del ruido en el ancho de banda es extremadamente baja, alrededor de $9{,}83 \times 10^{-14}$ Watts. Esta potencia es deseable ya que indica una relación señal a ruido (SNR) muy alta, de hecho, la relación señal a ruido (SNR) calculada es extraordinariamente alta, aproximadamente $9{,}19 \times 10^{40}$. Una SNR tan alta indica que la señal es muchas órdenes de magnitud más fuerte que el ruido, lo que garantiza una excelente calidad de la señal y una comunicación sin errores.

El error de transmisión (BER) calculado es de $0{,}0$, lo que indica que no se produjeron errores de bits durante la transmisión. Esto confirma la robustez del sistema de comunicación y la alta calidad de la señal recibida.

Sin embargo, es importante no confiarse completamente en estos resultados. Existen otros factores que pueden influir en la comunicación o la transmisión de información, y que no han sido considerados en este análisis. Por ejemplo, condiciones atmosféricas adversas, interferencias electromagnéticas, o incluso errores en el equipo de recepción podrían afectar la calidad de la señal y la fiabilidad de la transmisión. Por lo tanto, es fundamental realizar una evaluación exhaustiva que considere todos estos posibles factores para garantizar una comunicación confiable y efectiva en entornos reales.

13.6.2 Simulación de la constelación satelital en FreeFlyer

Para el desarrollo de nuestra simulación, hemos seleccionado como herramienta principal el software FreeFlyer, desarrollado por A.I. Solutions. Este software está especialmente diseñado para modelar escenarios satelitales y constelaciones, permitiendo de manera intuitiva modelar escenarios de comunicación por satélite en ubicaciones y momentos específicos. FreeFlyer facilita la simulación en tiempo real de las órbitas de la constelación, así como las operaciones de cada satélite, ofreciendo una representación visual de la posición de los satélites y estaciones terrenas.

Gracias a su lenguaje de programación similar a C++, FreeFlyer proporciona

un entorno de programación sencillo y flexible. Además, permite realizar configuraciones y programar mediante bloques predefinidos dentro de su interfaz gráfica de usuario (GUI). El código utilizado para simular la constelación de los satélites Galileo se ha desarrollado utilizando datos reales, como la altitud, excentricidad e inclinación de los satélites con respecto al ecuador, así como las coordenadas de las estaciones terrenas que forman parte del sistema Galileo.

Resultados de simulación

Los datos empleados en la simulación se basan en los parámetros proporcionados en la sección de fundamentos teóricos. Estos incluyen una altitud de 23222 km sobre la Tierra, una inclinación de 56° respecto al ecuador, una excentricidad de 0.2 y la disposición de 30 satélites distribuidos en 3 órbitas, con 10 satélites por órbita. También se consideran las estaciones terrenas que permiten el enlace y monitoreo continuo de los satélites, ubicadas en Laranca (Chipre), Spitsbergen (Svalbard, Noruega), La Reunión (Francia) y Maspalomas (Gran Canaria, España).

Se resalta el comportamiento de la constelación de los satélites Galileo alrededor de la Tierra y el movimiento elíptico que siguen al completar su trayectoria diaria. Esto demuestra la cobertura continua que ofrecen sobre la Tierra y asegura la constancia y seguridad de la señal gracias a su avanzada tecnología, ver Fig. 13.3.

Figura 19

Simulación en Freeflyer de vista 3D

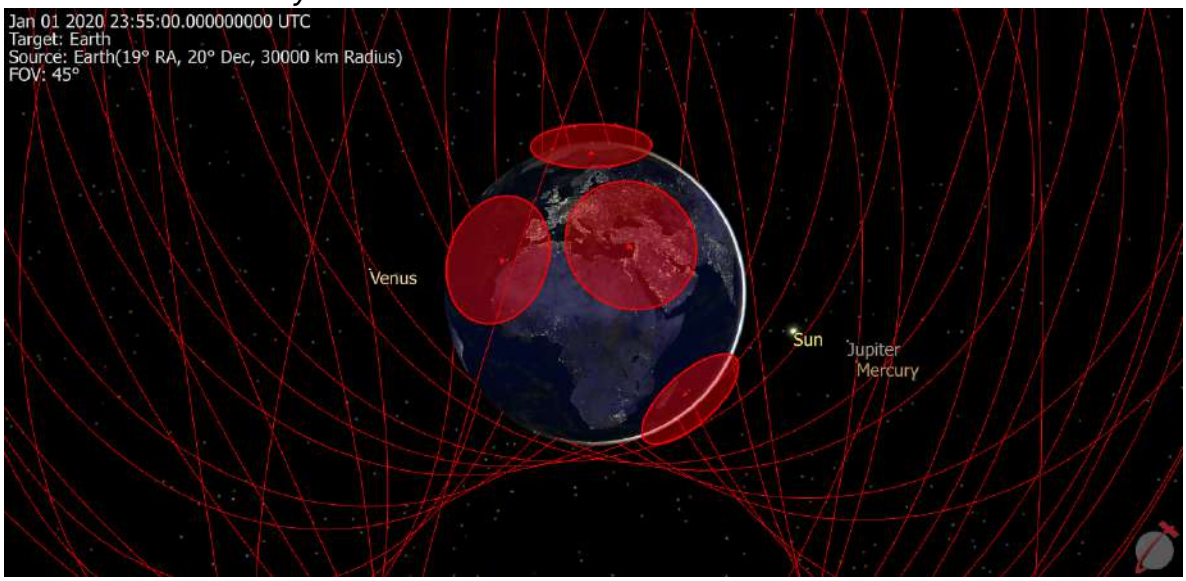


Figura 13.3. Simulación vista 3D

La simulación nos permite recrear una representación plana en 2 ejes, de la

trayectoria recorrida por los satélites, proporcionando una comprensión completa de su trayectoria sobre la Tierra, ver Fig. 13.4.

Figura 19

Simulación en Freeflyer de vista 2D

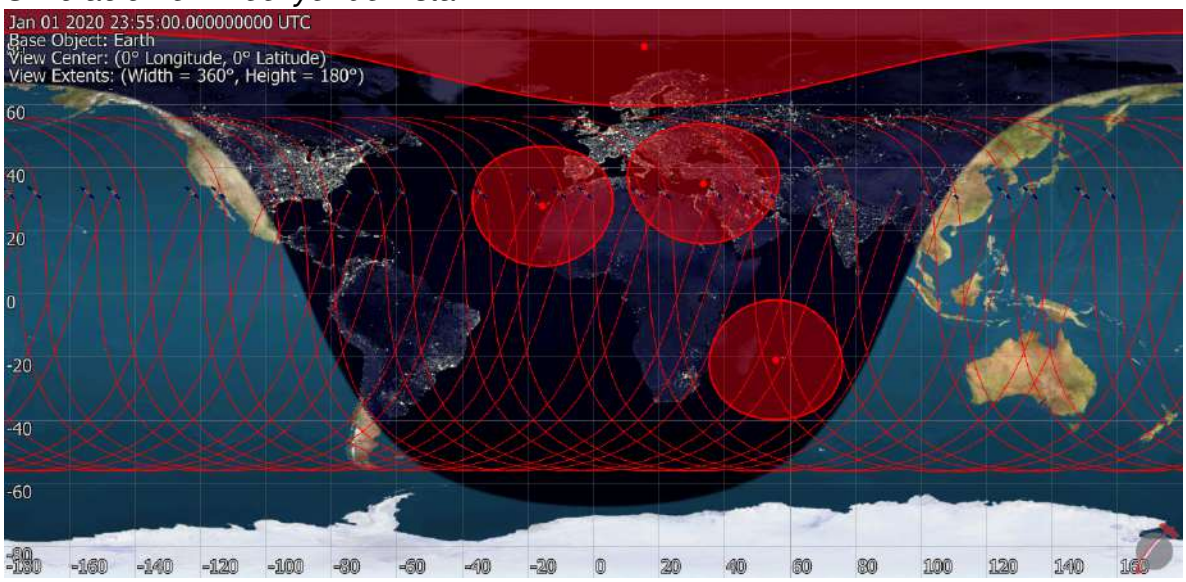


Figura 13.4. Simulación en 2D

Las estaciones terrenas generan informes de seguimiento de los satélites en intervalos regulares a lo largo del día. Ver Fig. 13.5.

Figura 20

Reporte de ubicación de satélites con respecto a las estaciones terrenas

"Ground Station:"	Galileocons[0].ElapsedTime	Segment.Azimuth(Galileocons[0].Epoch) (deg)	Segment.Elevation(Galileocons[0].Epoch) (deg)
Ground Station:	0.000000000	80.126170129	47.032191539
Ground Station:	0.003472222	84.315490262	47.339565501
Ground Station:	0.006944444	88.651897406	47.408384373
Ground Station:	0.010416667	93.086338271	47.215540312
Ground Station:	0.013888889	97.560542576	46.741976551
Ground Station:	0.017361111	102.010986054	45.974011067
Ground Station:	0.020833333	106.373821628	44.904244595
Ground Station:	0.024305556	110.589866077	43.531902491
Ground Station:	0.027777778	114.608695963	41.862598610
Ground Station:	0.031250000	118.391209792	39.907623450
Ground Station:	0.034722222	121.910472578	37.682938148
Ground Station:	0.038194444	125.151087166	35.208051475
Ground Station:	0.041666667	128.107565056	32.504944443

Figura 13.5. Reporte de ubicación de satélites

Por último, es importante señalar que, debido a las limitaciones del simulador, no es posible enviar archivos externos. Sin embargo, se puede simular el envío de archivos internos que generan un informe de envío, donde se muestra el momento

del envío del mensaje, el tiempo que tardó en regresar a la estación terrena y la velocidad a la que se movió, ver Fig. 13.7. A su vez, se muestra el mensaje enviado por la simulación para confirmar la entrega y la creación del informe. Ver Fig. 13.6.

Figura 21

Notificación de validación de envió de mensaje

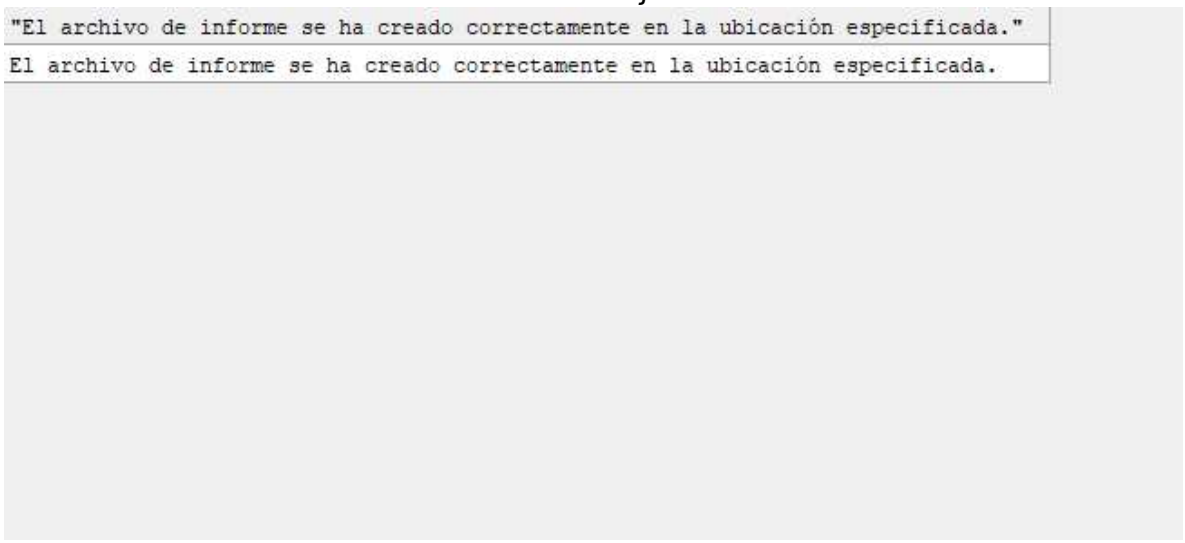
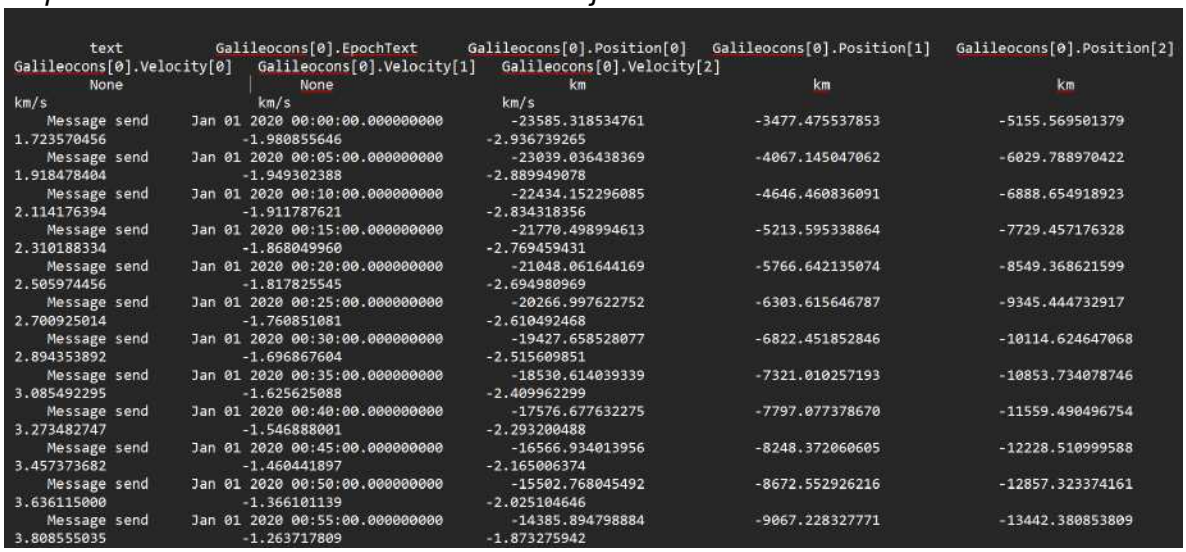


Figura 13.6. Notificación de validación

Figura 22

Reporte de validación de envió de mensaje



text	Galileocons[0].EpochText	Galileocons[0].Position[0]	Galileocons[0].Position[1]	Galileocons[0].Position[2]	Galileocons[0].Velocity[0]	Galileocons[0].Velocity[1]	Galileocons[0].Velocity[2]
None	None	km	km	km	km/s	km/s	km/s
Message send	Jan 01 2020 00:00:00.000000000	-23585.318534761	-3477.475537853	-5155.569501379	1.723570456	-1.980855646	-2.936739265
Message send	Jan 01 2020 00:05:00.000000000	-23039.036438369	-4067.145047062	-6029.788970422	1.918478404	-1.949302388	-2.889949078
Message send	Jan 01 2020 00:10:00.000000000	-22434.152296085	-4646.460836091	-6888.654918923	2.114176394	-1.911787621	-2.834318356
Message send	Jan 01 2020 00:15:00.000000000	-21770.498994613	-5213.595338864	-7729.457176328	2.310188334	-1.868049960	-2.769459431
Message send	Jan 01 2020 00:20:00.000000000	-21048.061644169	-5766.642135074	-8549.368621599	2.505974456	-1.817825545	-2.694980969
Message send	Jan 01 2020 00:25:00.000000000	-20266.997622752	-6303.615646787	-9345.444732917	2.700925014	-1.760851081	-2.610492468
Message send	Jan 01 2020 00:30:00.000000000	-19427.658528077	-6822.451852846	-10114.624647068	2.894353892	-1.696867604	-2.515609851
Message send	Jan 01 2020 00:35:00.000000000	-18530.614039339	-7321.010257193	-10853.734078746	3.085492295	-1.625625088	-2.409962299
Message send	Jan 01 2020 00:40:00.000000000	-17576.677632275	-7797.077378670	-11559.490496754	3.273482747	-1.546888001	-2.293200488
Message send	Jan 01 2020 00:45:00.000000000	-16566.934013956	-8248.372060605	-12228.510999588	3.457373682	-1.460441897	-2.165006374
Message send	Jan 01 2020 00:50:00.000000000	-15502.768045492	-8672.552926216	-12857.323374161	3.636115000	-1.366101139	-2.025104646
Message send	Jan 01 2020 00:55:00.000000000	-14385.894798884	-9067.228327771	-13442.380853809	3.808555035	-1.263717809	-1.873275942

Figura 13.7. Reporte de validación

14. Conclusiones

Una vez concluido el presente trabajo podemos destacar las siguientes conclusiones. Se logro el desarrollo de un algoritmo innovador diseñado para garantizar la seguridad e integridad de la información transmitida a través del sistema satelital europeo Galileo, no solo representa un avance técnico significativo, sino que también abre la puerta a futuras colaboraciones en proyectos de similar o mayor relevancia con empresas europeas.

Además, el proyecto ha fortalecido la colaboración entre la Facultad de Informática de la Universidad Autónoma de Querétaro con la Unidad de Alta Tecnología de la Facultad de Ingeniería de la Universidad Nacional Autónoma de México, siendo el principal medio de comunicación con el sistema Galileo en México, quien ha jugado un papel crucial en este proyecto. Esta colaboración no solo ha facilitado el desarrollo del algoritmo, sino que también ha creado oportunidades para futuras asociaciones y proyectos de investigación. Al involucrar a la UNAM, hemos asegurado un canal continuo para la transferencia de conocimientos y tecnología, beneficiando a ambas instituciones y a la comunidad académica en general.

Con el desarrollo de este proyecto, se logró realizar una revisión exhaustiva de los fundamentos teóricos de la criptografía, lo que permitió comprender la necesidad de proteger la información y reconocer diversas soluciones viables. Además, se recopiló información en nuestro idioma sobre el funcionamiento de Galileo, su modo de operación, características y especificaciones, así como su sistema de cifrado OSNMA y sus principales características. Esta recopilación proporciona una valiosa oportunidad para futuras investigaciones, facilitando el acceso a información crucial que permitirá continuar avanzando en este campo. Este proceso cumplió con el primer objetivo específico, estableciendo un criterio y un marco adecuados para que la solución propuesta cumpliera con los requisitos históricos y las tendencias pertinentes para los sistemas GNSS.

Por otro lado, se logró la programación de un algoritmo de encriptado de llave privada compatible con el protocolo OSNMA. Este proceso reveló la complejidad de programar un algoritmo de cifrado, requiriendo una comprensión profunda tanto del funcionamiento teórico como de las habilidades técnicas necesarias. Se experimentó y se comparó con diversas estructuras y procedimientos para garantizar el correcto funcionamiento del algoritmo, logrando cifrar y descifrar datos con precisión.

La selección del modelo de cifrado AES demostró ser la correcta para la propuesta debido a sus características superiores en comparación con DES y 3DES. AES ofrece una mayor seguridad gracias a su longitud de clave variable (256 bits), mayor eficiencia y menor vulnerabilidad a ataques de fuerza bruta, así como una fácil implementación en Python. Esto cumplió con el segundo objetivo establecido, validando la elección de AES como el cifrado más adecuado para la solución propuesta.

Finalmente, se programó una simulación de la constelación satelital europeo Galileo usando FreeFlyer. Esta etapa evaluó la viabilidad del sistema satelital europeo Galileo en nuestro país, validando que la propuesta no solo es útil para la región geográfica de origen del sistema, sino que también puede ser de gran utilidad para las empresas mexicanas y cualquier usuario interesado en esta tecnología. Los usuarios finales, que ya cuentan con dispositivos compatibles con Galileo, tendrán la oportunidad de experimentar esta tecnología de primera mano. Además, se puso a prueba la fiabilidad del algoritmo en un entorno simulado, subrayando la importancia de abordar factores como fenómenos atmosféricos e interferencias electromagnéticas que pueden impactar la integridad de la información transmitida. Este proceso cumplió con el tercer objetivo específico establecido.

En conclusión, se programó con éxito un algoritmo de encriptado AES en modo CFB basado en llave privada para el sistema de autenticación de mensajes OSNMA, cumpliendo con el objetivo general de fortalecer la seguridad de los canales de comunicación del sistema Galileo. El algoritmo ha sido diseñado y probado para cumplir con los estándares de seguridad exigidos, representando un avance significativo en la protección de la información en este contexto. Su implementación contribuirá a evitar posibles vulnerabilidades y a asegurar la integridad y confidencialidad de la información transmitida a través del sistema satelital europeo Galileo.

15. Trabajos Futuros

Afortunadamente, el tema de los sistemas de GNSS es sumamente amplio, lo que nos brinda la oportunidad de adentrarnos aún más en el fascinante mundo del espacio exterior. Esto es especialmente relevante al hablar del sistema de geoposicionamiento y radiolocalización Galileo, principalmente porque es relativamente nuevo, especialmente para nuestra región, México. Este sistema nos abre un abanico de posibilidades considerable, sobre todo porque Galileo nos permite contribuir a su desarrollo continuo.

Considerando esto, los futuros trabajos propuestos incluyen:

- Implementación del algoritmo en hardware específico.
- Validación en un entorno de bajo nivel del método propuesto.
- Experimentación con archivos de formatos distintos a los utilizados en este trabajo de investigación.
- Creación de librería de algoritmo de cifrado IDEA para Python.
- Creación de librería de algoritmo de cifrado RC5 para Python.
- Futuras colaboraciones con la UAT de la Facultad de Ingenieria de la UNAM.

Bibliografía

Álvarez, D. A. G. (2008). Sistema GNSS (global navigation satellite system). *Madrid: Universidad Autónoma de Madrid*.

Arencibia, M. G. (2008). Ética aplicada a la informática: un reto para el desarrollo social. *Revista Cubana de Ciencias Informáticas*, *2*(1-2), 45-54.

Bajpai, A., & Jain, N. (2021). *Python GUI for Image Processing Applications* [Tesis doctoral, Jaypee University of Information Technology, Solan, HP].

Bautista, F. J. (2023). ESTIMADOR DE RELACIÓN SEÑAL A RUIDO USANDO REDES NEURONALES PARA RECONOCEDORES AUTOMÁTICOS DE VOZ EN USO HOSPITALARIO. *Jornadas de Acústica, Audio y Sonido, UNTREF*.

Berné Valero, J. L., Anquela Julián, A. B., & Garrido Villén, N. (2014). GNSS. GPS: fundamentos y aplicaciones en Geomática. *Colección Académica. Editorial UPV*.

Berné Valero, J. L., Garrido Villén, N., & Capilla Romá, R. (2019). GNSS: GPS, Galileo, Glonass, Beidou. Fundamentos y métodos de posicionamiento. *Colección Académica*.

Berné Valero, J. L., Garrido Villén, N., & Capilla Romá, R. (2023). GNSS. Geodesia espacial y Geomática. *Colección Manual de referencia*.

Bisht, N., & Singh, S. (2015). A comparative study of some symmetric and asymmetric key cryptography algorithms. *International Journal of Innovative Research in Science, Engineering and Technology*, *4*(3), 1028-1031.

Bray, S. W. (2020). *implementing cryptography using Python*. John Wiley & Sons.

Cabrera Serrano, X. A. (2023). *Análisis comparativo de los modelos de encriptación simétrica y asimétrica.* [B.S. thesis]. Babahoyo: UTB-FAFI. 2023.

Cakir, T. (2021). International cooperation as an essential part of the Galileo programme. En *Legal Aspects Around Satellite Constellations: Volume 2* (pp. 161-177). Springer.

CARLOS, C. J. R., & ALAN, C. G. E. (2007). *Fundamentos, infraestructura, diseño e implementación de un enlace punto a punto de microondas dentro de la Jerarquía Digital Plesiócrona (PDH) en México* [Tesis de maestría, INSTITUTO POLITECNICO NACIONAL].

Carmona, M. E. R., & Cruz, M. A. C. (2024). Salud digital y la necesidad de su legislación en México. *Medicna en la era digital: Alcances y perspectivas*.

Cavalcante, T. M., Garcia, F. P., Gomes, D. G., & Andrade, R. M. (2011). Avaliação de desempenho dos algoritmos criptográficos skipjack e rc5 para redes de sensores sem fio. *Simpósio Brasileiro de Computação Ubíqua e Pervasiva*, 1103-1112.

Chang, H.-S. (2004). International data encryption algorithm. *jmu. edu, googleusercontent. com, Fall*.

Conti, M., Dragoni, N., & Lesyk, V. (2016). A survey of man in the middle attacks. *IEEE communications surveys & tutorials*, *18*(3), 2027-2051.

Cucchi, L., Damy, S., Paonni, M., Nicola, M., & Motella, B. (2022). Receiver testing for the galileo E1 OSNMA and I/NAV improvements. *Proceedings of the 35th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2022)*, 808-819.

Cuno, A. (2015). Conceptos de Firma Digital. *Identidad digital. La identificación desde los registros parroquiales al DNI electrónico*.

Delgado Pineda, M. (2010). Taller y Laboratorio: laboratorio de matemáticas: Cifrados Cesar aleatorios: La encriptación por sustitución desde los números racionales a los irracionales. *100cias@uned N.° 3*.

Estupiñán-Ortiz, B. L., & Bone-Obando, C. C. (2018). La Criptografía como elemento de la seguridad informática. *Polo del Conocimiento*, *3*(2), 118-131.

et al, V. (2014). El posicionamiento satelital y sus sistemas de corrección. *Manfredi*.

Feng-Cong, L., Yi-Nan, Z., & Xiao-Lin, Q. (2010). Secondary Development of Visual DSP++ Using Python for Debugging Peripherals of DSP. *2010 First International Conference on Pervasive Computing, Signal Processing and Applications*, 899-902. https://doi.org/10.1109/PCSPA.2010.222

Fernández Acevedo, F. J. (2004). El documento electrónico en el derecho civil chileno: Análisis de la Ley 19.799. *Ius et Praxis*, *10*(2), 137-167.

Franchi, M. R. (2012). *Algortimos de encriptación de clave asimétrica* [Tesis doctoral, Universidad Nacional de La Plata].

Galileo, O. (2021a). Open service: Service definition document v1.

Galileo, O. (2021b). SIGNAL-IN-SPACE INTERFACE CONTROL DOCUMENT V.2.

Gimenez, J. J., Carcel, J. L., Fuentes, M., Garro, E., Elliott, S., Vargas, D., Menzel, C., & Gomez-Barquero, D. (2019). 5G new radio for terrestrial broadcast: A forward-looking approach for NR-MBMS. *IEEE Transactions on Broadcasting*, *65*(2), 356-368.

González Duque, R. (2011). *Python para todos*. Creative Commons Reconocimiento.

Götzelmann, M., Köller, E., Viciano-Semper, I., Oskam, D., Gkougkas, E., & Simon, J. (2023). Galileo open service navigation message authentication: Preparation phase and drivers for future service provision. *NAVIGATION: Journal of the Institute of Navigation*, *70*(3).

Haryono, W. (2020). Comparison encryption of how to work caesar cipher, hill cipher, blowfish and twofish. *Data Science: Journal of Computing and Applied Informatics*, *4*(2), 100-110.

Hecht, P. (2011). A Zero-Knowledge authentication protocol using non commutative groups. *Actas del VI Congreso Iberoamericano de Seguridad Informática CIBSI*, *11*, 96-102.

Iglesias Fernández, J. L. (2019). Técnicas de autenticación criptográfica en señales abiertas GNSS: el fenómeno del spoofing y métodos de defensa. *Repositori Institucional (O2)*.

Jerez, G. O., & Alves, D. B. M. (2018). GLONASS: Revisão teórica e estado da arte. *Revista Brasileira de Geomática*, *6*(2), 155-173.

Jeya, R., Amutha, B., Nikhilesh, N., & Immaculate, R. R. (2019). Signal interferences in wireless communication-an overview. *spectrum*, *2*, 3.

Joskowicz, J. (2008). Reglas y prácticas en eXtreme Programming. *Universidad de Vigo*, *22*.

Kumar, P., Srivastava, P. K., Tiwari, P., & Mall, R. (2021). Application of GPS and GNSS technology in geosciences. En *GPS and GNSS Technology in Geosciences* (pp. 415-427). Elsevier.

Leick, A., Rapoport, L., & Tatarnikov, D. (2015). *GPS satellite surveying*. John Wiley & Sons.

Méndez, P. S. G. (2011). descripcion polinomial de los sistemas de cifrado DES y AES.

Méndez Sampedro, J. (2023). *Galileo Search and Rescue (SAR): Análisis del sistema y simulación del balance de enlace* [Tesis de maestría, Universitat Oberta de Catalunya (UOC)].

Mendoza, J. C. (2008). Demostración de cifrado simétrico y asimétrico. *Ingenius: Revista de Ciencia y Tecnología*, (3), 46-53.

Moore, A. D. (2018). *Python GUI Programming with Tkinter: Develop responsive and powerful GUI applications with Tkinter*. Packt Publishing Ltd.

Nicola, M., Motella, B., Pini, M., & Falletti, E. (2022). Galileo OSNMA public observation phase: Signal testing and validation. *IEEE Access*, *10*, 27960-27969.

Nielson, S. J., & Monson, C. K. (2019). *Practical Cryptography in Python: Learning Correct Cryptography by Example*. Apress.

Paar, C., & Pelzl, J. (2009). *Understanding cryptography: a textbook for students and practitioners*. Springer Science & Business Media.

Paredes, G. G., et al. (2006). Introducción a la criptografía. *Revista Digital Universitaria Vol. 7*.

Pou Peña, G. (2016). Comparativa de diferentes tipos de efemérides GNSS en cálculo de grandes líneas base con Magnet Office. *E.T.S.I. en Topografía, Geodesia y Cartografía (UPM)*.

Pozo-Ruz, A., Ribeiro, A., García-Alegre, M., García, L., Guinea, D., & Sandoval, F. (2000). Sistema de posicionamiento global (gps): descripción, análisis de errores, aplicaciones y futuro. *ETS ingenieros de Telecomunicaciones. Universidad de Malaga*, 1-9.

R., Á. S. (2005). *Aplicación de las matrices por bloques de criptosistemas de cifrado en flujo* [Tesis doctoral, Universidad de Alicante].

Rani, S., & Kaur, H. (2017). Technical Review on Symmetric and Asymmetric Cryptography Algorithms. *International Journal of Advanced Research in Computer Science*, *8*(4).

Rijmen, V., & Daemen, J. (2001). Advanced encryption standard. *Proceedings of federal information processing standards publications, national institute of standards and technology*, *19*, 22.

Rubí, M. A. C., de la Cruz, N. S., Lobaton, A. M. D., Rodríguez, G. A., & Rubí, F. C. (2011). Teoría de números en criptografía y su debilidad ante la posible era de las computadoras cuánticas. *CIENCIA ergo-sum, Revista Científica Multidisciplinaria de Prospectiva*, *18*(3), 264-273.

Sánchez, J. A. V., Padrón, A., Prieto, R., Herrera, A., & Calva, G. (2014). Cifrador de flujo para tecnología GSM: una comparación entre hardware y software. *SOMI: Congreso de instumentación*.

Tole, D. C., Tique, D. A., Calero, C. C., & González, I. V. (2021). Aunando e Investigando Métodos De Encriptación DES y 3DES. *Ingenio Magno*, *12*(2), 21-32.

Vargas, Y. T. M., & Mnedez, H. A. M. (2015). Comparación de algoritmos basados en la criptografía simétrica DES, AES y 3DES. *Mundo Fesc*, *5*(9), 14-21.

Whitman, M. E., & Mattord, H. J. (2021). *Principles of information security*. Cengage learning.

Zapata, F. G. (2013). *Detección de ataques de spoofing a los sistemas de navegación global.* UNIVERSIDAD DE SEVILLA.

Anexos

15.1 Anexo I - Historias de usuario

Historia de Usuario	
Código: H2	**Usuarios:** Todos
Nombre de historia: Generar contraseña aleatoria	
Prioridad: Alta	**Riesgo de desarrollo:** Media
Puntos estimados: 3	**Iteración asignada:** 1
Programador asignado: Erick Gómez	
Descripción: El sistema debería generar una contraseña segura de forma automática para garantizar la seguridad del cifrado.	
Observación: Mejora la seguridad y comodidad para el usuario.	

Cuadro 15.1. Historia de usuario - Generar contraseña aleatoria

Historia de Usuario	
Código: H3	**Usuarios:** Todos
Nombre de historia: Ver la contraseña generada automáticamente	
Prioridad: Media	**Riesgo de desarrollo:** Media
Puntos estimados: 3	**Iteración asignada:** 2
Programador asignado: Erick Gómez	
Descripción: Antes de proceder con el cifrado, los usuarios deben poder visualizar la contraseña generada automáticamente.	
Observación: Facilita la interacción y la comprensión del usuario.	

Cuadro 15.2. Historia de usuario - Ver la contraseña generada automáticamente

	Historia de Usuario
Código: H4	**Usuarios:** Todos
Nombre de historia: Cifrar un archivo TXT	
Prioridad: Alta	**Riesgo de desarrollo:** Alta
Puntos estimados: 5	**Iteración asignada:** 3
Programador asignado: Erick Gómez	
Descripción: Los usuarios deberían poder cifrar el archivo TXT seleccionado utilizando la contraseña generada automáticamente.	
Observación: Funcionalidad principal del cifrador.	

Cuadro 15.3. Historia de usuario - Cifrar un archivo TXT

	Historia de Usuario
Código: H5	**Usuarios:** Todos
Nombre de historia: Guardar el archivo en un formato bin	
Prioridad: Alta	**Riesgo de desarrollo:** media
Puntos estimados: 4	**Iteración asignada:** 2
Programador asignado: Erick Gómez	
Descripción: El archivo cifrado debe guardarse en un formato binario para garantizar la integridad y seguridad de los datos.	
Observación: Asegura la integridad y seguridad del archivo cifrado.	

Cuadro 15.4. Historia de usuario - Guardar el archivo en un formato bin

	Historia de Usuario
Código: H6	**Usuarios:** Todos
Nombre de historia: Seleccionar archivo cifrado	
Prioridad: Alta	**Riesgo de desarrollo:** Baja
Puntos estimados: 3	**Iteración asignada:** 1
Programador asignado: Erick Gómez	
Descripción: Como usuario, quiero tener la capacidad de seleccionar un archivo cifrado para poder descifrar su contenido.	
Observación: Tener certeza de es el archivo deseado.	

Cuadro 15.5. Historia de usuario - Seleccionar archivo cifrado

	Historia de Usuario
Código: H7	**Usuarios:** Todos
Nombre de historia: Ingresar contraseña para descifrar	
Prioridad: Media	**Riesgo de desarrollo:** Baja
Puntos estimados: 2	**Iteración asignada:** 1
Programador asignado: Erick Gómez	
Descripción: Como usuario, quiero ingresar una contraseña para descifrar el archivo seleccionado y acceder a los datos originales.	
Observación: Validar que la contraseña sea la correcta y logre descifrar el archivo seleccionado.	

Cuadro 15.6. Historia de usuario - Ingresar contraseña para descifrar

Historia de Usuario	
Código: H8	**Usuarios:** Todos
Nombre de historia: Ver datos descifrados	
Prioridad: Alta	**Riesgo de desarrollo:** Baja
Puntos estimados: 5	**Iteración asignada:** 1
Programador asignado: Erick Gómez	
Descripción: Ver los datos descifrados en la interfaz gráfica después de descifrar un archivo, para verificar que el proceso fue exitoso.	
Observación: Validar que los datos sean correctos y coincidan con el original.	

Cuadro 15.7. Historia de usuario - Ver datos descifrados

Historia de Usuario	
Código: H9	**Usuarios:** Todos
Nombre de historia: Guardar datos descifrados como archivo TXT	
Prioridad: Media	**Riesgo de desarrollo:** Media
Puntos estimados: 4	**Iteración asignada:** 2
Programador asignado: Erick Gómez	
Descripción: Como usuario, quiero tener la opción de guardar los datos descifrados como un archivo de texto (TXT) para poder acceder a ellos posteriormente.	
Observación: Tener la posibilidad de almacenar el archivo descifrado en el equipo del destinatario.	

Cuadro 15.8. Historia de usuario - Guardar datos descifrados como archivo TXT

15.2 Anexo II - Imágenes comparativas

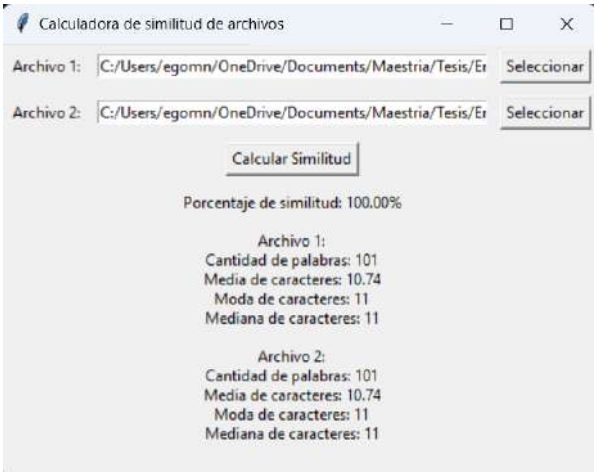


Figura 15.1. Comparación de archivos: 1

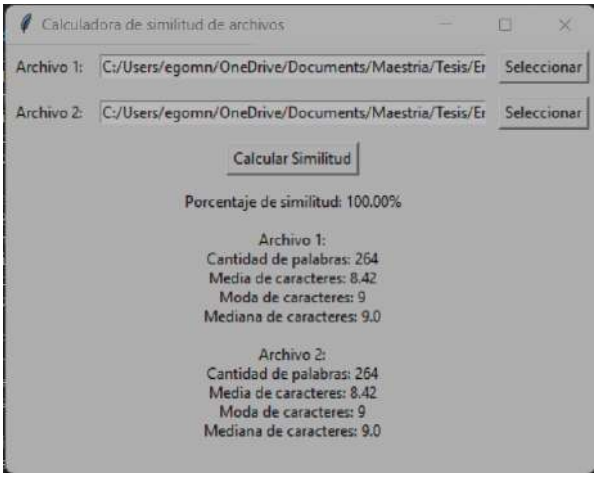


Figura 15.2. Comparación de archivos: 2

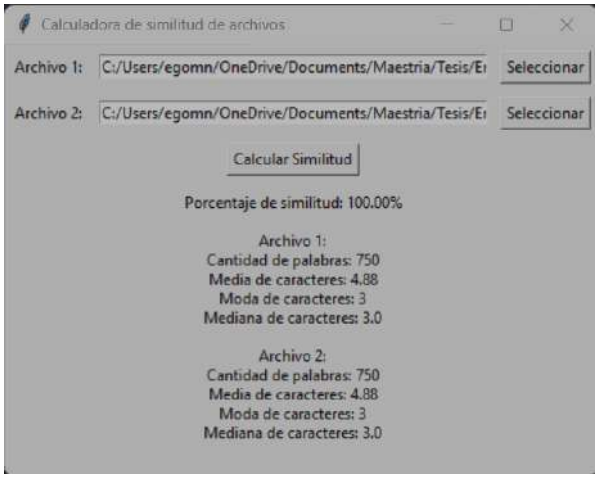


Figura 15.3. Comparación de archivos: 3

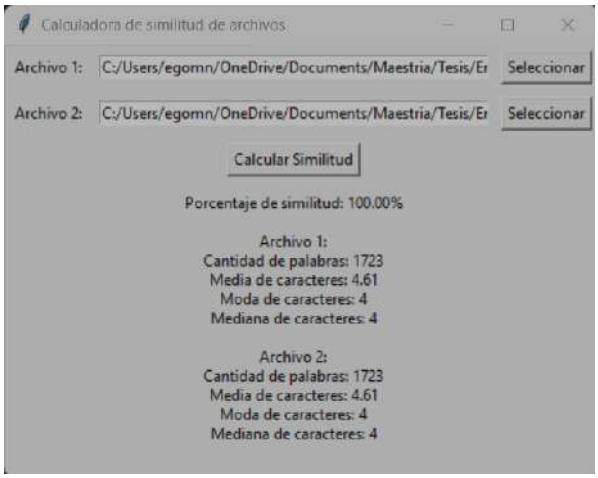


Figura 15.4. Comparación de archivos: 4

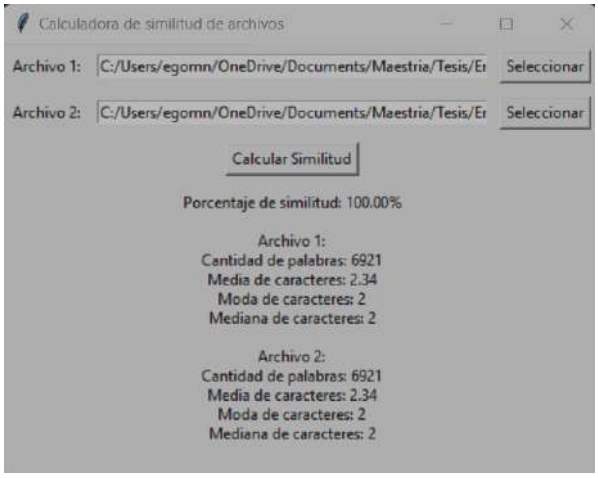


Figura 15.5. Comparación de archivos: 5

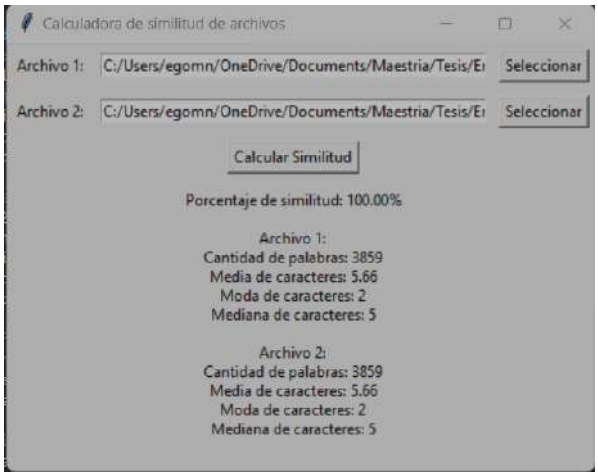


Figura 15.6. Comparación de archivos: 6

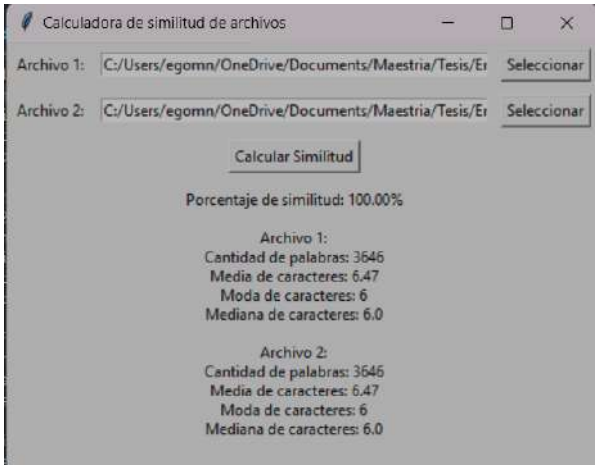


Figura 15.7. Comparación de archivos: 7

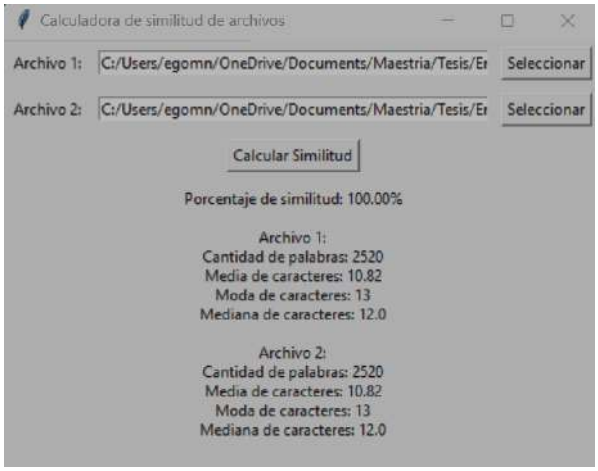


Figura 15.8. Comparación de archivos: 8

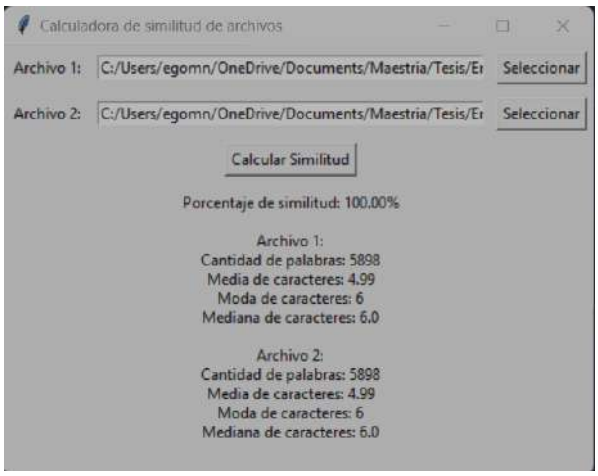


Figura 15.9. Comparación de archivos: 9

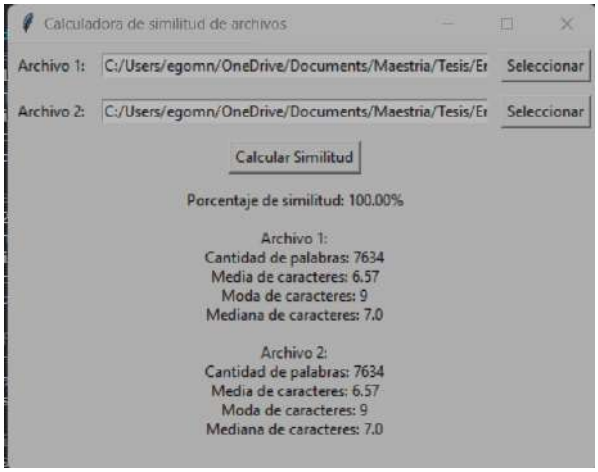


Figura 15.10. Comparación de archivos: 10

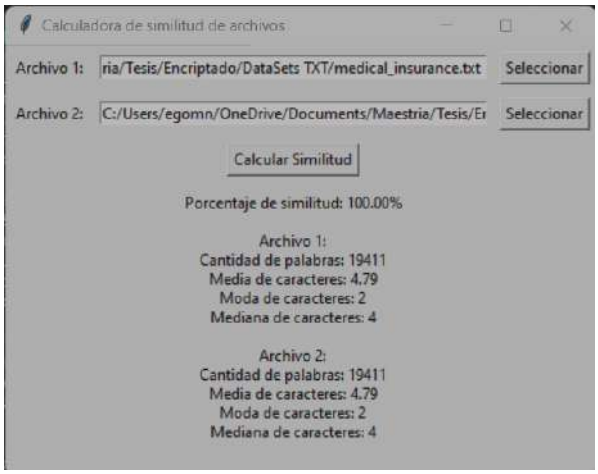


Figura 15.11. Comparación de archivos: 11

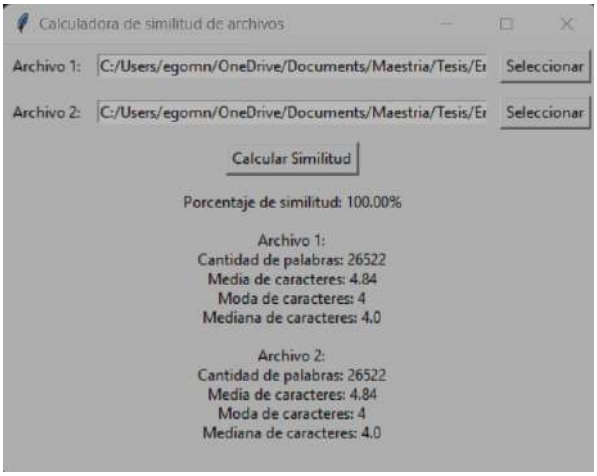


Figura 15.12. Comparación de archivos: 12

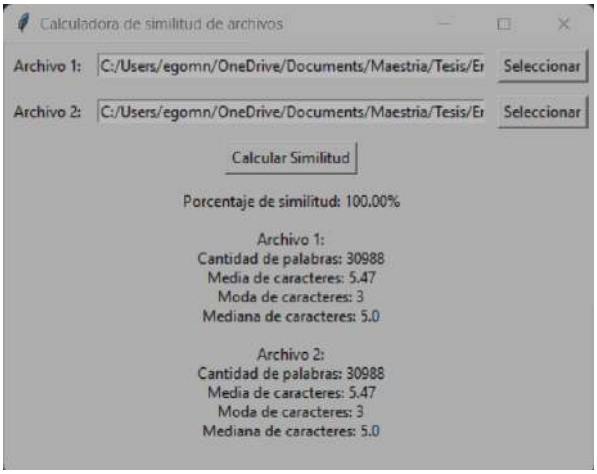


Figura 15.13. Comparación de archivos: 13

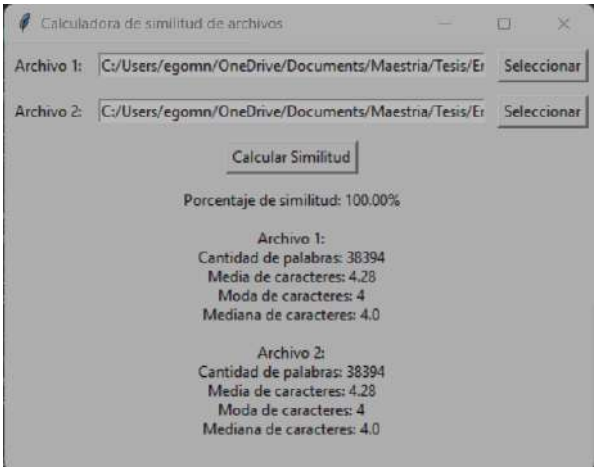


Figura 15.14. Comparación de archivos: 14

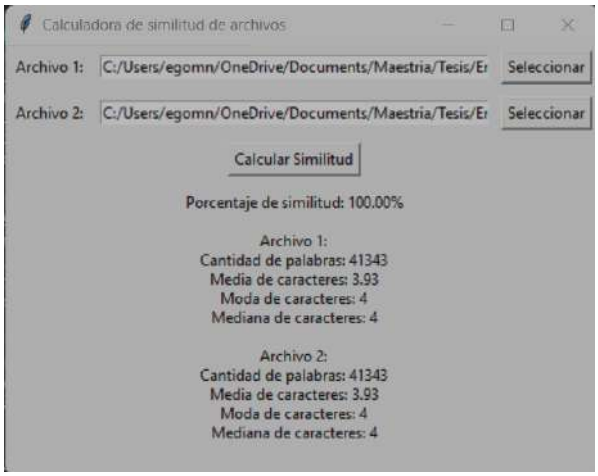


Figura 15.15. Comparación de archivos: 15

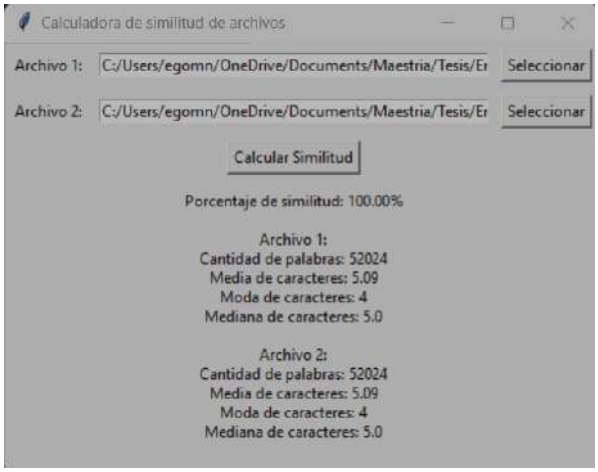


Figura 15.16. Comparación de archivos: 16

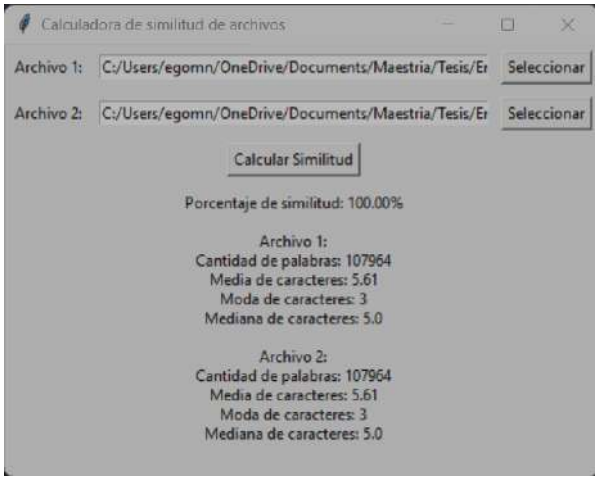


Figura 15.17. Comparación de archivos: 17

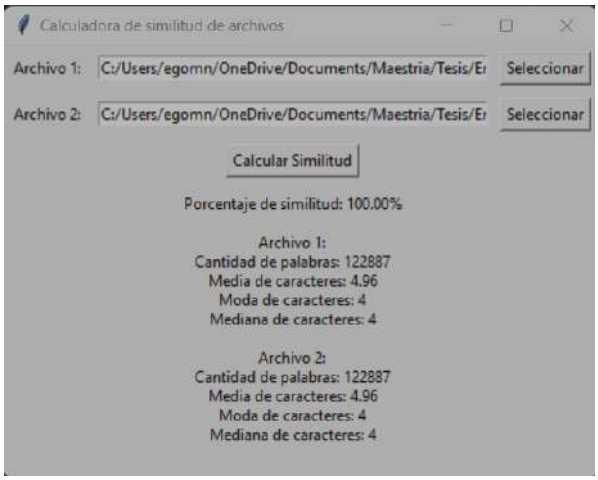


Figura 15.18. Comparación de archivos: 18

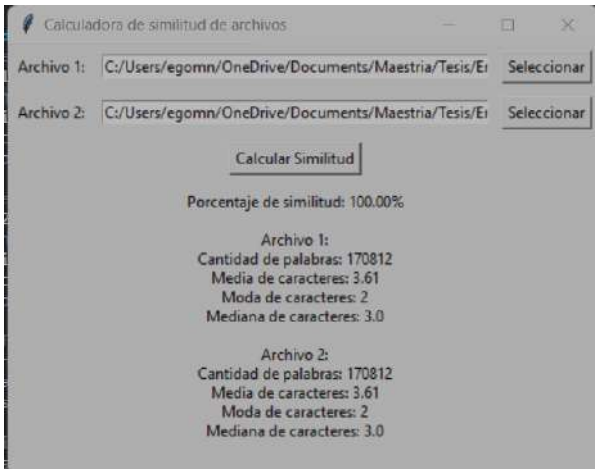


Figura 15.19. Comparación de archivos: 19

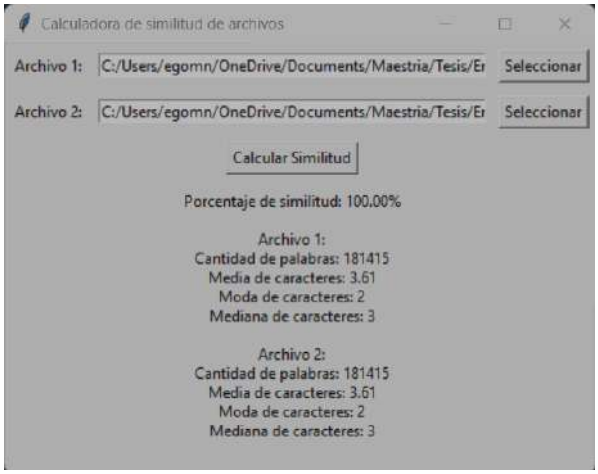


Figura 15.20. Comparación de archivos: 20

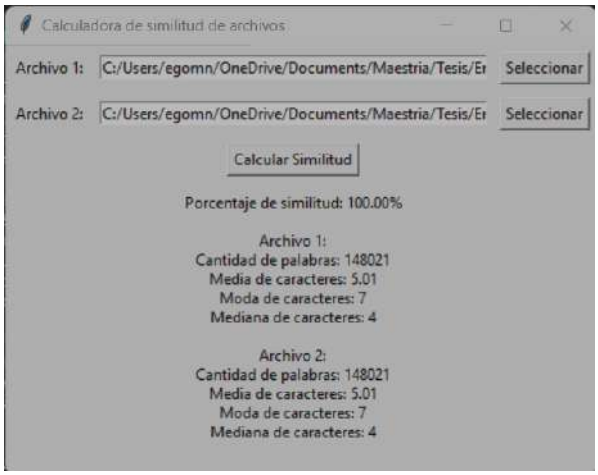


Figura 15.21. Comparación de archivos: 21

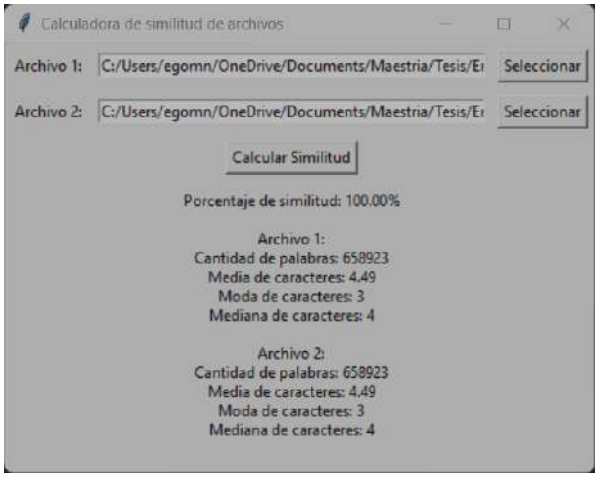


Figura 15.22. Comparación de archivos: 22

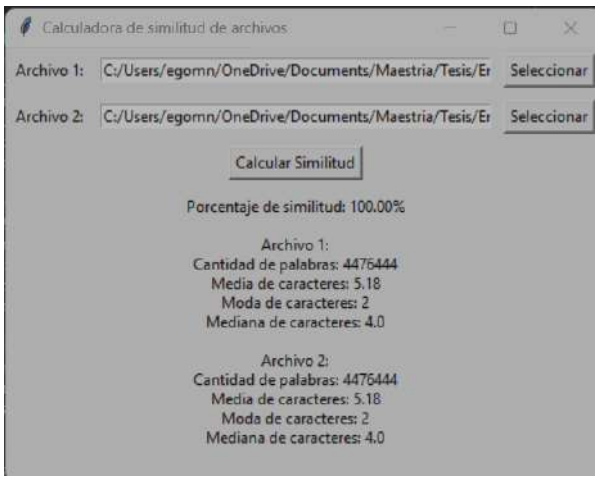


Figura 15.23. Comparación de archivos: 23

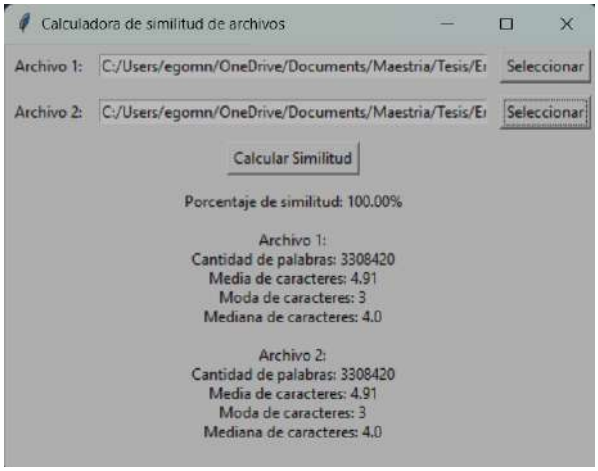


Figura 15.24. Comparación de archivos: 24

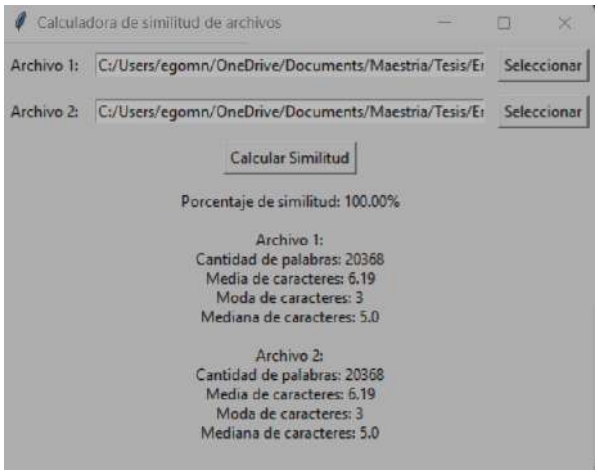


Figura 15.25. Comparación de archivos: 25

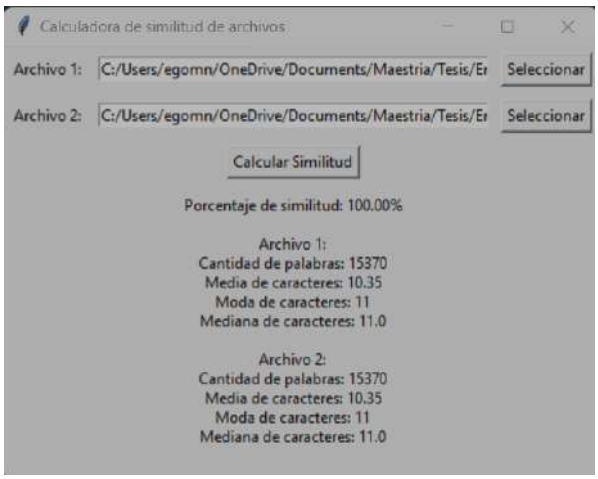


Figura 15.26. Comparación de archivos: 26

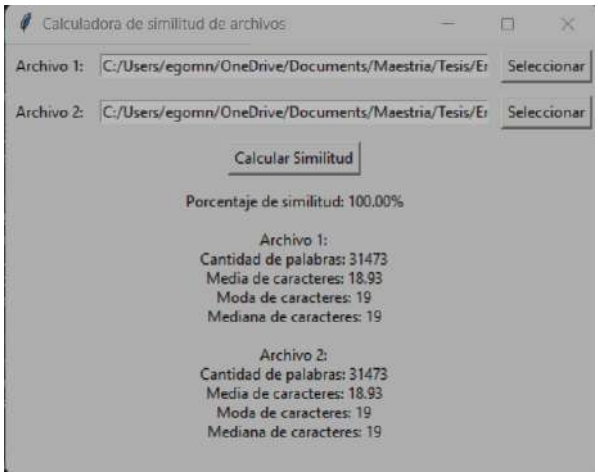


Figura 15.27. Comparación de archivos: 27

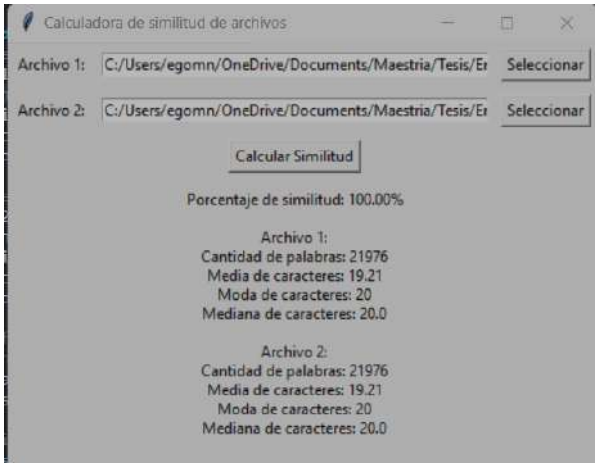


Figura 15.28. Comparación de archivos: 28

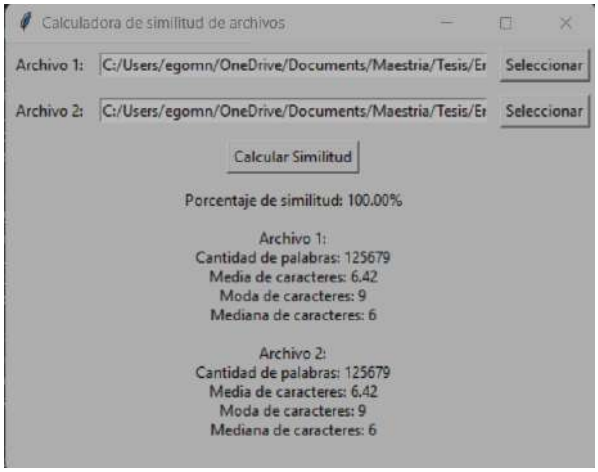


Figura 15.29. Comparación de archivos: 29

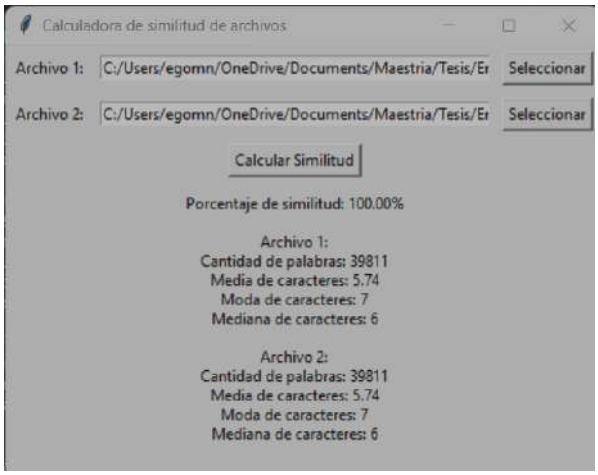


Figura 15.30. Comparación de archivos: 30 v

15.3 Anexo III - Código simulación en FreeFlyer

Listing 15.1: Simulacion ConstelaciÃşn Galileo

```
Formation Galileocons;
ReportInterface ReportInterface1;

Galileocons.Count = 30;
Galileocons.ViewAsGroup = 0;
Galileocons.GroupPointSize = 8;

Galileocons[0].A = 23222;
Galileocons[0].E = 0.2;
Galileocons[0].I = 56; // InclinaciÃşn en grados
Variable altitude = 23222;
Variable inclination = 56;
Variable i;


For i = 0 to Galileocons.Count-1;
    Galileocons[i].A = 23222;
    Galileocons[i].E = 0.2;
    Galileocons[i].I = 56;
    Galileocons[i].RAAN = 56*i;
End;

// EstaciÃşn terrena
GroundStation maspalomas;
maspalomas.Longitude = -15.6348;
maspalomas.Latitude = 27.7614;
maspalomas.Height = 0.65;
maspalomas.MaskShape = "Cone";

// EstaciÃşn terrena Laranca, Chipre
GroundStation laranca;
laranca.Longitude = 33.3838;
```

```
laranca.Latitude = 34.8651;
laranca.Height = 0.65;
laranca.MaskShape = "Cone";

// EstaciÃşn terrena Spitsbergen, Svalbard, Noruega
GroundStation spitsbergen;
spitsbergen.Longitude = 15.3707;
spitsbergen.Latitude = 78.2305;
spitsbergen.Height = 0.65;
spitsbergen.MaskShape = "Cone";

// EstaciÃşn terrena La ReuniÃşn, Francia
GroundStation laReunion;
laReunion.Longitude = 55.5156;
laReunion.Latitude = -20.9097;
laReunion.Height = 0.65;
laReunion.MaskShape = "Cone";


// Configurar segmento de visibilidad
VisibilitySegment Segment;
Segment.SetObserver(maspalomas);
Segment.SetObserver(laranca);
Segment.SetObserver(spitsbergen);
Segment.SetObserver(laReunion);
Segment.SetTarget(Galileocons[0]);


ReportInterface1.Filename = "Ruta␣del␣archivo";

// Ejecutar el bucle principal
While (Galileocons[0].ElapsedTime < TIMESPAN(1 days));
    If (Segment.Visibility(Galileocons[0].Epoch));
        Report "Ground␣Station:", Galileocons[0].ElapsedTime,
        Segment.Azimuth(Galileocons[0].Epoch),
        Segment.Elevation(Galileocons[0].Epoch);
        Report "Message␣send", Galileocons[0].EpochText,
```

```
        Galileocons[0].Position,
        Galileocons[0].Velocity to ReportInterface1;
    End;
        Map Galileocons, maspalomas, laranca, spitsbergen, laReunion;
        View Galileocons, maspalomas, laranca, spitsbergen, laReunion;
    Step Galileocons;
End;

Close ReportInterface1;
// Verificar si el archivo se ha creado correctamente
Variable fileExists;
fileExists = FileSystem.Exists(ReportInterface1.Filename);

If (fileExists);
    Report "El␣archivo␣de␣informe␣se␣ha
␣␣␣␣creado␣correctamente␣en␣la␣ubicaciÃşn␣especificada.";
Else;
    Report "Error:␣No␣se␣pudo␣crear␣el␣archivo
␣␣␣␣de␣informe␣en␣la␣ubicaciÃşn␣especificada.";
End;

// Mostrar los objetos en un mapa y vista

FileSystem.Copy(ReportInterface1.Filename, "Ruta␣del␣archivo");
```

15.4 Anexo IV - Verificador de archivos

Listing 15.2: Verificador de archvios

```
import tkinter as tk
from tkinter import filedialog
from tkinter import messagebox
import statistics

def calcular_similitud(archivo1, archivo2):
    # Leer palabras del primer archivo
    with open(archivo1, 'r', encoding='utf-8') as file1:
        contenido_archivo1 = file1.read()
        palabras_archivo1 = contenido_archivo1.split()
        caracteres_archivo1 = [len(palabra) for
            palabra in palabras_archivo1]
        cantidad_palabras_archivo1 = len(palabras_archivo1)

    # Leer palabras del segundo archivo
    with open(archivo2, 'r', encoding='utf-8') as file2:
        contenido_archivo2 = file2.read()
        palabras_archivo2 = contenido_archivo2.split()
        caracteres_archivo2 = [len(palabra) for palabra in
            palabras_archivo2]
        cantidad_palabras_archivo2 = len(palabras_archivo2)

    interseccion = len(set(palabras_archivo1).intersection
    (set(palabras_archivo2)))
    union = len(set(palabras_archivo1).union(set(palabras_archivo2)))
    similitud = interseccion / union

    media_archivo1 = statistics.mean(caracteres_archivo1)
    moda_archivo1 = statistics.mode(caracteres_archivo1)
    mediana_archivo1 = statistics.median(caracteres_archivo1)

    media_archivo2 = statistics.mean(caracteres_archivo2)
```

```
    moda_archivo2 = statistics.mode(caracteres_archivo2)
    mediana_archivo2 = statistics.median(caracteres_archivo2)

    return (similitud, media_archivo1, moda_archivo1,
    mediana_archivo1, cantidad_palabras_archivo1,
            media_archivo2, moda_archivo2,
            mediana_archivo2, cantidad_palabras_archivo2)

def seleccionar_archivo(num_archivo):
    archivo = filedialog.askopenfilename(filetypes=[
    ("Archivos de texto", "*.txt")])
    if num_archivo == 1:
        archivo1_entry.delete(0, tk.END)
        archivo1_entry.insert(tk.END, archivo)
    else:
        archivo2_entry.delete(0, tk.END)
        archivo2_entry.insert(tk.END, archivo)

def calcular_similitud_y_mostrar():
    archivo1 = archivo1_entry.get()
    archivo2 = archivo2_entry.get()

    if archivo1 == "" or archivo2 == "":
        messagebox.showwarning("Advertencia",
        "Por favor seleccione ambos archivos.")
        return

    try:
        similitud, media_archivo1, moda_archivo1,
        mediana_archivo1, cantidad_palabras_archivo1, \
        media_archivo2, moda_archivo2,
        mediana_archivo2, cantidad_palabras_archivo2 =
        calcular_similitud(archivo1, archivo2)

        resultado_label.config(text=
        "Porcentaje de similitud: {:.2f}%\n\n"
                "Archivo 1:\n"
```

```
                    "Cantidad de palabras: {}\n"
                    "Media de caracteres: {:.2f}\n"
                    "Moda de caracteres: {}\n"
                    "Mediana de caracteres: {}\n\n"
                    "Archivo 2:\n"
                    "Cantidad de palabras: {}\n"
                    "Media de caracteres: {:.2f}\n"
                    "Moda de caracteres: {}\n"
                    "Mediana de caracteres: {}".format(similitud * 100,
                            cantidad_palabras_archivo1,
                            media_archivo1, moda_archivo1,
                            mediana_archivo1,
                            cantidad_palabras_archivo2,
                            media_archivo2, moda_archivo2,
                            mediana_archivo2))
    except Exception as e:
        messagebox.showerror("Error",
        f"OcurriÃş un error al calcular
        la similitud:\n{str(e)}")

# Crear la ventana principal
ventana = tk.Tk()
ventana.title("Calculadora de similitud de archivos")

# Crear widgets
archivo1_label = tk.Label(ventana, text="Archivo 1:")
archivo1_entry = tk.Entry(ventana, width=50)
archivo1_boton = tk.Button(ventana, text="Seleccionar",
command=lambda: seleccionar_archivo(1))

archivo2_label = tk.Label(ventana, text="Archivo 2:")
archivo2_entry = tk.Entry(ventana, width=50)
archivo2_boton = tk.Button(ventana, text="Seleccionar",
command=lambda: seleccionar_archivo(2))

calcular_boton = tk.Button(ventana, text="Calcular Similitud",
command=calcular_similitud_y_mostrar)
```

```
resultado_label = tk.Label(ventana,
text="Porcentaje de similitud:\n\nArchivo 1:\n\nArchivo 2:")

# Ubicar widgets en la ventana
archivo1_label.grid(row=0, column=0, padx=5, pady=5, sticky="e")
archivo1_entry.grid(row=0, column=1, padx=5, pady=5)
archivo1_boton.grid(row=0, column=2, padx=5, pady=5)

archivo2_label.grid(row=1, column=0, padx=5, pady=5, sticky="e")
archivo2_entry.grid(row=1, column=1, padx=5, pady=5)
archivo2_boton.grid(row=1, column=2, padx=5, pady=5)

calcular_boton.grid(row=2, column=1, padx=5, pady=5)
resultado_label.grid(row=3, column=1, padx=5, pady=5)

# Ejecutar la ventana
ventana.mainloop()
```