UNIVERSIDAD CÉSAR VALLEJO

FACULTAD DE INGENIERÍA Y ARQUITECTURA

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

Algoritmo de encriptación de imágenes basado en AES, SHA y PHASH

TESIS PARA OBTENER EL TÍTULO PROFESIONAL DE:
Ingeniero de Sistemas

AUTOR:

Ramirez Melgarejo Gabriel Gonzalo (ORCID: 0000-0001-8445-463X)

ASESOR:

Mag. Milner David Liendo Arévalo (ORCID: 0000-0002-7665-361X)

LÍNEA DE INVESTIGACIÓN:

Sistema de Información y Comunicaciones

LIMA – PERÚ

2020

Dedicatoria

El proyecto de investigación está dedicado a mis padres quienes me brindaron la motivación, el sustento y apoyo incondicional. A mis familiares, quienes me impulsaron a cumplir mis metas durante mis estudios de mi carrera profesional.

Agradecimiento

Agradecer a mi familia por el apoyo de todo el periodo académico que tomó mi carrera profesional, por seguir a mi lado en cada día de mi vida. A mis asesores de investigación, los ingenieros Milner David Liendo Arevalo.

A mis docentes y a mis compañeros por el apoyo, disposición, el tiempo brindado, paciencia durante el transcurso en el desarrollo de mi proyecto de tesis, para conseguir los resultados deseados.

Índice de contenidos

Índice de tablas

Índice de figuras

Resumen

El problema de la investigación fue, No se ha encontrado estudios que expliquen el impacto del uso de un algoritmo de encriptación de imágenes basado en AES, SHA y PHASH en función al tiempo de encriptación, en la integridad y en el consumo de recursos de hardware. lo que ha limitado el uso de los algoritmos de criptografía simétrica para la encriptación de imágenes. El objetivo de la investigación fue determinar el impacto del uso de un algoritmo de encriptación de imágenes basado en AES, SHA y PHASH. El algoritmo está probado en un sistema diseñado en JavaScript usando un gestor de base de datos, MongoDB. Se utilizo la metodología SCRUM para desarrollar el software. Para finalizar, los datos obtenidos demostraron un aumento de 16,47% en el tiempo de encriptación de imágenes, además, se garantizó la integridad de la imagen con un promedio de 98,96% y para concluir, un promedio de 72,06% y 69,34% con respecto al consumo de CPU y RAM. Por ende, se comprobó la integridad de la imagen encriptada y un consumo de hardware óptimo.

Palabras clave: Algoritmos criptográficos, Criptografia simétrica, Análisis Imágenes, Reconocimiento de imagen.

Abstract

The problem of the research was, studies not have been found that explain the impact of the use of an image encryption algorithm based on AES, SHA and PHASH based on the encryption time, on the integrity and on the consumption of hardware resources. which has limited the use of symmetric cryptography algorithms for image encryption. The objective of the research was to determine the impact of the use of an image encryption algorithm based on AES, SHA and PHASH. The algorithm is tested in a system designed in JavaScript using a database manager, MongoDB. The SCRUM methodology was used to develop the software. Finally, the data obtained showed an increase of 16.47% in the time of image encryption, in addition, the integrity of the image was guaranteed with an average of 98.96% and to conclude, an average of 72.06% and 69.34% with respect to CPU and RAM consumption. Hence, the integrity of the encrypted image and optimal hardware consumption were checked.

Keywords: Cryptographic algorithms, Symmetric cryptography, Image analysis, Image recognition.

I. INTRODUCCIÓN

Actualmente, la encriptación de imágenes se ha convertido un factor fundamental debido al constante manejo de imágenes y a la transmisión de dichas imágenes. según Kumar y Chauhan (2018) confirmo que la imagen digital es el formato más utilizado para compartir información con cualquier persona. para enviar este formato de forma segura, es necesario un enfoque de criptografía. La criptografía es una técnica utilizada para evitar el acceso no autorizado a los datos. Tiene dos componentes principales: Algoritmo de cifrado y Clave. Varios algoritmos criptográficos son disponibles en el mercado como DES, AES, RC5, TDES y RSA. (Kumar y Chauhan, 2018, p. 60). La encriptación de imágenes es el tema principal de esta investigación, la cual se irá explicando por medio de los algoritmos criptográficos buscados en trabajos relacionados al tema. Al respecto Sangeeta y Arpneek (2017) explicó que Algunos cifrados son mejores que otros en algunos aspectos, Cada algoritmo tiene su propias ventajas y deficiencias. En resumen, todos los Las técnicas son útiles para el cifrado en tiempo real. Cada técnica es única a su manera, que podría ser adecuado para diferentes aplicaciones (Sangeeta y Arpneek, 2017, p. 360). Con el desarrollo constante del internet y las nuevas tecnologías, la transmisión de contenidos multimedia se ha vuelto un factor muy importante debido a que enviamos información personal. Las transmisiones de información adoptan mecanismos de seguridad como la Criptografia para asegurar la integridad de la información. según Pan, Lei, y Chen (2018) indicaron que, En los últimos años, junto con la rápida promoción y popularización de la tecnología de red y la tecnología de comunicación digital en el mundo, las imágenes digitales y los video digital se han convertido en un medio importante para él envió y almacenamiento de información por medio de internet en el ámbito civil o Campos militares, Sin embargo, los problemas de seguridad de la red han sido durante mucho tiempo un factor importante que ha afectado y restringido el desarrollo de la tecnología de red (p. 1).

En esta investigación se recopilo información sobre la encriptación imágenes, lo cual se podrá observar por medio de los estudios previos o antecedentes referentes al tema de investigación como: técnicas criptográficas, sistemas de encriptación, algoritmos criptográficos, procesamiento imágenes, etc. Además, se describirán los resultados obtenidos de los estudios previos mencionados.

Se encontraron trabajos previos como el de Rani y Kaur (2017) “Algoritmos de criptografía simétrica y asimétrica” el propósito de este artículo es conceptualizar y proponer algoritmos criptográficos para mantener la información segura y en secreto de una red, usando la criptografía asimétrica o Criptografia simétrica (p. 182).

Así también Nisar, Hafiz, y Saleem (2016) “Un punto de referencia para la evaluación del rendimiento Y la evaluación de la seguridad de los esquemas de cifrado de imágenes” En este estudio se diseñó y desarrolló un benchmark basado en todos los parámetros necesarios para un buen cifrado de imágenes. Se han realizado extensos estudios a las categorías de todos los parámetros utilizados por diferentes investigadores para evaluar sus (p. 18).

Por su parte Liang, Yan, Tengfei, Hongtu y Jianfeng (2016) “El esquema mejorado de eficiencia para el control de acceso seguro de la distribución de video digital” proponen un esquema para aumentar la eficacia en el control de acceso de video digital en decodificadores principalmente de tres enfoques. En primer lugar, con la comparación específica entre AES y RC6. En segundo lugar, se proponer la mejora de la eficiencia del proceso de códecs mediante la adopción de la codificación vp9, que es un códec de video abierto y libre (p. 12645).

Tambien Cevallos (2019) “Algoritmo RC5 mejorado que utiliza computación paralela para redes de comunicación” presento un diseño del algoritmo de criptografía RC5 son una seguridad mejorada que se logra a través de modificación de la clave secrete por medio de procesos de Computación Paralela. Si bien el diseño fue desarrollado para el algoritmo RC5 la lógica principal podría ser aplicada a otros algoritmos de cifrado de bloque utilizando el mismo esquema propuesto (p. 104).

Por otro lado, Wang y Liu (2017) en su estudio “Un algoritmo de cifrado de imágenes novedoso y efectivo basado en caos y codificación de ADN” proponen un algoritmo de cifrado de imágenes eficaz basado en caos y las reglas de codificación del ADN. Los resultados del experimento y Los análisis indican que el algoritmo propuesto es capaz de resistir ataques típicos y tiene buen carácter de seguridad (p. 6229).

Asi mismo Gan, Chai, Yuan y Lu (2018) “Algoritmo de cifrado de imágenes basado en S-boxes y caos basados en LFT” en su artículo se propuso un algoritmo de cifrado de imágenes eficiente basado en el sistema caótico usando S-boxes, SHA 256, etc. además de garantizar la eficiencia, seguridad y la resistencia en ataques de texto plano (p. 8760).

Por otro lado, Li, Sun y Wang (2019) “Modelo Memcapacitor y su aplicación para el algoritmo de cifrado de imágenes” proponen el diseño de un circuito caótico con un condensador de memoria, que tiene abundantes comportamientos dinámicos. Como potencial aplicación, combinando el modo de unión del ADN y las características del sistema caótico, se crea un nuevo esquema de cifrado de imágenes propuesto. Además, el algoritmo propuesto tiene una mejor capacidad anti interferente para el cifrado de imágenes e información (p. 1).

Tambien Liu y Miao (2016) “Algoritmo de cifrado de imágenes basado en un mapa caótico logístico con parámetros variables” en su artículo se propuso un nuevo algoritmo de cifrado de imágenes basado en un mapa caótico logístico variable y un algoritmo dinámico. Se usa el mapa logístico de parámetros variados para mezclar la imagen simple, y luego use un algoritmo dinámico para cifrar la imagen. Se realizaron experimentos como el análisis de Histograma, análisis de entropía de información, análisis de sensibilidad, análisis de espacio clave, análisis de correlación y complejidad computacional para evaluar su actuación. Los resultados del experimento muestran que este algoritmo es de alta seguridad. y puede ser competitivo para el cifrado de imágenes (p. 1).

Actualmente la Criptografía es una técnica que se utiliza para evitar el acceso a los datos. Los datos se pueden cifrar mediante algoritmos criptográficos por ello Algunos estudios específicos han demostrado que no es posible determinar que algoritmos a utilizar o combinar son factibles para la encriptación de imágenes, tomando como base para esta investigación los algoritmos AES, SHA y PHASH, debe destacarse que es necesario dar a conocer esta información a las entidades que trabajan con imágenes. Al respecto Wang y Liu (2017) menciono Las imágenes digitales tienen una alta resolución y ocupan más espacio de almacenamiento. Debido a ello los algoritmos típicos de encriptación de imágenes como Data Encryption Standard (DES), Advanced Encryption Standard (AES), Rivest-Shamir-

Adleman (RSA) no son competentes para cifrar tales datos digitales. Asu ves Nisar et al. (2016) indicaron que otros algoritmos brindan una mejor seguridad no significa que los datos cifrados por AES sean vulnerables a los ataques ya qué se puede aumentar su seguridad agregando más rondas o una llave más larga (p. 6230). Por otro lado, Hu, Li, Li, Li y Chu (2016) mencionaron que el algoritmo AES se utiliza generalmente en el cifrado de contenido de vídeo y RC6 ha demostrado que su rendimiento es al menos igual o mejor al algoritmo AES con excelentes credenciales de seguridad dependiendo el entorno de desarrollo (p. 12646). Por otro lado, Gan et al. (2018) mencionaron que los sistemas caóticos con la combinación con el algoritmo SHA garantiza la eficiencia, seguridad y la resistencia de la imagen encriptada (p. 8760).

La finalidad de esta investigación es proponer un algoritmo de encriptación de imágenes en función al tiempo, integridad y consumo de recursos de hardware mediante la combinación de algoritmos que mejoren el proceso de encriptación de imágenes. De acuerdo con Rani y Kaur (2017) La criptografía juega un papel importante en la seguridad de mantener la confidencialidad, autenticación, integridad y el no repudio de la información (p. 182). Por otro lado, Cevallos (2019) indicó que Para mejorar el rendimiento de un algoritmo de clave simétrica se pueden aplicar varias técnicas y métodos; no solo implementado en hardware, pero también utilizando software e incluso sistemas integrados con asociaciones entre hardware y software. Aunque no todas estas técnicas permiten elevar el nivel de seguridad, se pueden aplicar junto con otros mecanismos para lograr un mejor nivel de seguridad con un desempeño aceptable (p. 107).

Algunas consecuencias con respecto a la utilización inadecuado de los algoritmos criptográficos pueden generar problemas con respecto al tiempo, seguridad e integridad de la información encriptada. De acuerdo con Cevallos (2019) las claves simétricas tienen un tiempo de vida más corto que claves asimétricas, lo que significa que es necesario aplicar más mantenimiento y administración de claves para las primeras (p. 104). Además, en la Criptografia asimétrica es necesario que remitente y el objetivo necesitan tener la clave antes de que se establezca la comunicación esto es una gran desventaja. Por otro lado, Liu y Miao (2016) mencionaron que algunas debilidades comunes del mapa logístico es que incluyen un espacio de claves relativamente pequeño y una

distribución desigual de secuencias esto conllevan riesgos de seguridad para el cifrado de la información (p. 1).

Esta investigación tiene una justificación teórica ya que en el desarrollo de esta investigación se realiza la descripción, explicación y análisis sobre la encriptación de imágenes por medio de algoritmos criptográficos, además se realizó la revisión de artículos científicos relacionados al tema. Esta investigación se realizó con el propósito de generar conocimiento sobre la encriptación de imágenes por medio de algoritmos criptográficos además de mostrar los conocimientos existentes sobre el tema. Al respecto, Cevallos (2019) describió un nuevo diseño del algoritmo RC5 que pueda tener una longitud de la clave hasta 2040 bits, a diferencia de otros algoritmos como AES que funciona con solo tres longitudes de clave. Esto permite comparar los algoritmos tradicionales con el mejorado. Para poder medir la duración del tiempo de encriptación (p. 111). Por otro lado, Nisar et al. (2016) menciono que el cifrado basado en bloques (AES) es bueno para seguridad criptográfica en cuanto a la comunicación del canal está libre de distorsiones y la imagen no es necesaria para ser comprimido, además los esquemas basados en el caos es un cifrado de elección, ya que tiene una alta seguridad criptográfica y rendimiento junto con cierta tolerancia a la compresión y el ruido de la imagen (p. 19).

La presente investigación se justifica tecnológicamente debido a que en el desarrollo de esta investigación se utilizará tecnologías como software tales como: lenguaje de programación, algoritmos de encriptación, hardware, etc. donde se desarrolla e implementará el algoritmo propuesto para la encriptación de imágenes. Algunas recomendaciones sobre la encriptación de imágenes Mencionadas en los artículos científicos abarcan sobre la codificación ADN, donde se recomienda utilizar otros métodos que ayuden a optimizar el cifrado de la imagen. Al respecto, Wang y Liu (2017) indicaron que su algoritmo propuesto puede implementarse mediante un método paralelo, esto puede ahorrar mucho tiempo para obtener imágenes de cifrado. Con respecto a otras recomendaciones, Se recomienda considerar el consumo de energía de encriptación de imágenes (p. 6242). Al respecto, Fei, Li y Yang (2016) combinan el paralelismo AES-NI y CPU. El algoritmo AES se utiliza ampliamente para mejorar el consumo energía y su eficiencia

también es muy significativa. Por otro lado, se recomienda usar un circuito integrado que permita encriptar imágenes (p. 1106). Al respecto, Gan et al. (2018) mencionaron que es posible diseñar un circuito de hardware de algoritmo de encriptación que utiliza FPGA para la comunicación segura en tiempo real (p. 8780).

Sobre la realidad problemática presentada se planteó el problema general y los problemas específicos de la investigación. El problema general de la investigación fue, No se ha encontrado estudios que expliquen el impacto del uso de un algoritmo de encriptación de imágenes basado en AES, SHA y PHASH en función al tiempo de encriptación, en la integridad y en el consumo de recursos de hardware. lo que ha limitado el uso de los algoritmos de criptografía simétrica para la encriptación de imágenes. Los problemas específicos de la investigación fueron los siguientes:

- **PE1:** No se ha encontrado estudios que expliquen el impacto del uso de un algoritmo de encriptación de imágenes basado en AES, SHA y PHASH en función al tiempo de encriptación de imágenes.
- **PE2:** No se ha encontrado estudios que expliquen el impacto del uso de un algoritmo de encriptación de imágenes basado en AES, SHA y PHASH en función a la integridad de las imágenes.
- **PE3:** No se ha encontrado estudios que expliquen el impacto del uso de un algoritmo de encriptación de imágenes basado en AES, SHA y PHASH en función al consumo de recursos de hardware.

El objetivo general fue Determinar el impacto del uso de un algoritmo de encriptación de imágenes basado en AES, SHA y PHASH. Los objetivos específicos fueron los siguientes:

- **OE1:** Determinar el impacto del uso de un algoritmo de encriptación de imágenes basado en AES, SHA y PHASH en función al tiempo de encriptación de imágenes**.**
- **OE2:** Determinar el impacto del uso de un algoritmo de encriptación de imágenes basado en AES, SHA y PHASH en función a la integridad de las imágenes**.**

- **OE3:** Determinar el impacto del uso de un algoritmo de encriptación de imágenes basado en AES, SHA y PHASH en función al consumo de recursos de hardware.

La hipótesis general de la investigación fue El algoritmo de encriptación de imágenes basado en AES, SHA y PHASH reduce el tiempo de encriptación de imágenes, brinda integridad de imágenes y tiene un consumo de hardware aceptable.

De acuerdo con Bai (2015), el algoritmo AES es rompible. Sin embargo, el tamaño de la clave de AES hace que sea difícil. Se desarrollaron nuevos algoritmos basados en AES. Sin embargo, esos algoritmos eran más sencillos lo cual no brindaban una seguridad aceptable en comparación de AES (p. 1). En el estudio realizado por Rani, y Kaur (2017) La criptografía es una tecnología, que es esencial para la seguridad de la red. Hay dos formas de enviar dichos datos en el usando la clave asimétrica y criptografía simétrica. (p. 185).

Las hipótesis específicas fueron los siguientes:

- **HE1:** El algoritmo de encriptación de imágenes basado en AES, SHA y PHASH reduce el tiempo de encriptación de imágenes**.**

De acuerdo con Liu and Miao (2016) mencionaron que el algoritmo propuesto encripta una imagen en menor tiempo que los algoritmos DES y AES, y es bastante aceptable para el cifrado de imágenes (p. 11). En el estudio realizado por Aqeel, Liao, Kulsoom y Ulla (2016), el algoritmo propuesto basado en sha-256 reduce el tiempo de encriptación de imágenes comparado con otros algoritmos (p. 11256)

- **HE2:** El algoritmo de encriptación de imágenes basado en AES, SHA y PHASH garantiza la Integridad de las imágenes**.**

De acuerdo con Sonia y Grewal (2016), Los algoritmos existentes permiten la integridad del mensaje, los mensajes tienen una alta probabilidad de que les ocurra cualquier cambio, lo cual se realiza Códigos de autenticación y verificación de firmas digitales para verificar la integridad del mensaje (p. 45).

- **HE3:** El algoritmo de encriptación de imágenes basado en AES, SHA y PHASH brinda un consumo de hardware óptimo**.**

El uso del algoritmo propuesto de Aqeel et al. (2016), consiguió realizar la encriptación de imágenes con un consumo de hardware mínimo según las especificaciones de los componentes a utilizar. lo cual fueron, Intel® Pentium® Dual Core de 2,3 GHz con 2048 MB de RAM en computadora personal (p. 11256).

II. MARCO TEÓRICO

En este capítulo se sintetizan los antecedentes y teorías relacionadas Sobre los trabajos previos relacionados al tema de esta investigación, se encontró investigaciones referentes a algoritmos de encriptación, sistemas, técnicas criptográficas, etc., las cuales todas las mencionadas se orientan a la encriptación de imágenes.

Para poder precisar más este importante tema revisamos los trabajos de Liu, Meehan, Tong, Wu, Xu y Xu (2020) donde estudiaron de un enfoque de aprendizaje profundo sobre la identificación de etiquetas del medicamento mediante imagen y texto incrustado con el objetivo de desarrollar una etiqueta de identificación de medicamentos a través del modelo de incrustación de imágenes y texto (DLI-IT) para modelar patrones basados en texto de datos históricos para la detección de drogas sospechosas. Como resultados se concluyó que, al combinar el análisis de incorporación de imágenes y texto en un marco de aprendizaje profundo, El enfoque DLI-IT logró un desempeño competitivo en el avance de la identificación de etiquetas de medicamentos, además el modelo fue entonces probado en 300 imágenes externas de etiquetas de medicamentos, el resultado demostró que nuestro modelo alcanza hasta el 88% de la precisión en la identificación de etiquetas de medicamentos, que supera al método de identificación basado en imágenes o texto anterior con una mejora de hasta un 35% (p. 1).

Así mismo, en la investigación de Li et al. (2019) estudiraon un nuevo modelo de Memcapacitor y su aplicación en los algoritmos de cifrados de imágenes con el objetivo de diseñar un circuito caótico novedoso con un condensador de memoria que soporte abundantes comportamientos dinámicos. Como resultado se concluyó que el algoritmo tiene mejores efectos de cifrado, mayor espacio de claves y mayor sensibilidad a una llave. Comparado con otros algoritmos de cifrado, el tiempo de cálculo se reduce en un 30% y tiene mejor rapidez y conveniencia (p. 1). Además, el algoritmo puede resistir ataques exhaustivos, ataques estadísticos y ataques diferenciales. Todas estas características muestran que el algoritmo es muy adecuado para el cifrado de imágenes digitales. Asimismo, Li et al. (2019) recomendó que en el futuro se lleve a cabo el cifrado de imágenes por medio de aplicaciones específicas o algoritmos criptográficas (p. 15).

A su vez, Cevallos (2019) estudio el algoritmo RC5 empleando computación paralela para esquemas de comunicaciones con el objetivo de presentar un esquema alternativo de algoritmo de criptografía con un nivel de seguridad aumentado que se obtiene mediante mecanismos de expansión de clave con procesos de Computación Paralela, además de poder ser aplicada en diferentes diseños de algoritmos de cifrado en bloque. Como resultados se concluyó que usar la Computación Paralela fue factible para modificar el algoritmo RC5 para mejorar la seguridad mediante una técnica de expansión clave manteniendo un rendimiento aceptable para las redes de comunicaciones (p. 121). Dos factores importantes en el rendimiento del RC5 mejorado algoritmo son el número de filas y la longitud del bloque utilizado para el proceso de cifrado. A medida que aumenta el número de filas, el cifrado El proceso también necesita más tiempo, aunque el nivel de seguridad aumenta. Asimismo, Cevallos (2019) recomendó aplicar los conceptos e ideas expuestas a otros algoritmos de clave simétrica (p. 121).

Por otra parte, Gan et al. (2018) presentaron un algoritmo de cifrado de imágenes basado en LFT, S-boxes y sistema de caos con el objetivo de presentar un algoritmo que puede resistir ataques de texto plano conocido y texto plano utilizando S-boxes, LFT, FSP, 2D-LASM, el caos correlacionado y la sustitución correlacionada para mejorar el nivel de seguridad. Como resultado se concluyó que el algoritmo corre rápido, pero también resiste varios ataques comunes, como ataque diferencial, ataque de fuerza bruta, ataque estadístico, ataques de texto sin formato conocido y de texto sin formato elegido (p. 8780). Asimismo, Gan et al. (2018) recomendaron que se podría diseñar el circuito de hardware del algoritmo de cifrado utilizando FPGA, y hacerlo aplicable en la comunicación segura en tiempo real (p. 8780).

A su vez, Kumar y Chauhan (2018) estudiaron la criptografía de imágenes con Matrix Array Clave simétrica utilizando un enfoque basado en los sistemas de caos con el objetivo de proponer un algoritmo con Matrix Array Symmetric Key (MASK) para la generación de claves y un enfoque basado en el Caos para el cifrado de imágenes. La función principal de MASK es generar claves se cretas. El proceso de cifrado implica la generación de claves secretas, se consideró la clave de MASK-256 para el cifrado que tiene 16 rondas. El concepto

basado en el caos ha sido considerado para el cifrado de imagen. Como resultados se concluyó que el algoritmo propuesto es mejor en comparación con AES y MAES (p. 65).

Así mismo, Pan et al. (2018) estudiaron el cifrado de imágenes digitales sobre un algoritmo basado en un mapa logístico caótico con el objetivo de desarrollar una tecnología que permite mantener la información segura utilizando las tecnologías de caos ya que da la posibilidad de descifrar la información sin las claves correctas es muy reducida. Como resultados se concluyó que el algoritmo propuesto no pierde la información a la hora de encriptar y desencriptar una imagen. Además, el cifrado tradicional no es adecuado para el cifrado de imágenes, por lo que es necesario buscar una nueva solución como la teoría del caos para la investigación de las imágenes cifradas (p. 9).

Aportando a lo mencionado, Wang y Liu (2017) estudiaron un algoritmo de cifrado de imágenes basado en sistemas de caos y la codificación ADN con el objetivo de proponer un algoritmo de cifrado de imágenes utilizando Mapa caótico lineal por partes (PWLCM) y mapas logísticos para generar todos los parámetros que necesita el algoritmo presentado y la tecnología de codificación de ADN funciona como una herramienta auxiliar. Como resultados se concluyó que El algoritmo propuesto tiene la capacidad de cifrar imágenes de forma segura (p. 6243). Asimismo, Cevallos (2019) recomendó aplicar experimento utilizando GPU para evaluar el rendimiento del algoritmo propuesto (p. 121).

Por otro lado, Méndez, Cisneros y Villa (2017) estudiaron la mejoría de algoritmos criptográficos mediante la union de la esteganografía en imágenes con la finalidad de aumentar la seguridad de la información, utilizando tecnologías como AES, WinHex, ImageDiff, LSB y 2NAES. Como resultados se concluyó que la propuesta de mejora del algoritmo criptográfico 2NAES con la combinación de la técnica LSB, incremento la seguridad comparado con el algoritmo AES (p. 20).

En la investigación de Sangeeta y Arpneek (2017) se estudiaron los algoritmos de criptografía de clave simétrica con el objetivo de dar a conocer el funcionamiento de Criptografía de clave simétrica en diferentes aspectos como la seguridad, integridad y rendimiento que ofrecen los diferentes algoritmos

criptográficos. Como resultados se concluyó que los algoritmos de Criptografia simétrica tienen mayor rendimiento y escalabilidad que los algoritmos de Criptografia asimétrica (p. 360).

Así mismo, Liu y Miao (2016) estudiaron un nuevo algoritmo de cifrado de imágenes basado en un mapa caótico logístico variable y un algoritmo dinámico con el objetivo resolver las debilidades que tiene el mapa logístico y resistir el ataque de reconstrucción del espacio de fase se usa el mapa logístico de parámetros variados para mezclar la imagen simple, y luego use un algoritmo dinámico para cifrar la imagen. Como resultados se concluyó que el algoritmo propuesto tiene una alta seguridad y puede ser competitivo con algún otro algoritmo de cifrados de imágenes (p. 11).

En la investigación Guesmi, Farah, Kachouri y Samet (2016) se estudió el Cifrado de imágenes basado en claves hash mediante crossover operador y caos con el objetivo de proponer un esquema de cifrado de imágenes en color que utiliza claves basadas sobre el operador cruzado, el sistema de caos y el algoritmo SHA-2. Como resultados se concluyó que el esquema propuesto puede lograr un buen resultado de cifrado a través de un solo proceso rondas en el cifrado, el espacio de la clave es lo suficientemente grande como para resistir ataques, por lo que el esquema es confiable y seguro para ser aplicado en cifrado de imágenes (p. 4766).

Así mismo, Aqeel et al. (2016) estudiaron una técnica de fusión modificada (dual) para el cifrado de imágenes usando hash SHA-256 y múltiples mapas caóticos con el objetivo de proponer una técnica de encriptación de imágenes de fusión dual modificada (MDF) además Se aplica una técnica novedosa de codificación de ADN mediante mapas caótica a nivel de píxel y el hash SHA-256 de la imagen que se utiliza para generar claves secretas para evitar ataque de texto plano. Como resultados se concluyó que el criptosistema propuesto tiene un efecto de cifrado bastante bueno que el esquema de fusión original, pero también tiene la capacidad de sostener el ruido que se agrega durante la transmisión a través de un canal ruidoso (p. 11263). Además, el factor primordial del esquema mejorado es adecuado para las aplicaciones en tiempo real.

Por otro lado, Bhute y Arjaria (2016) estudiaron la seguridad de datos de usuarios en la nube de basada en los algoritmos AES y RC6 con el objetivo de presentar un marco eficiente para la seguridad de los datos de texto, los datos están totalmente controlados por usuario y la seguridad del usuario está controlada por los servidores se utilizan los algoritmos AES y RC6 para cifrar la información. Como resultados se concluyó que el enfoque presentado es mejor en términos de uso de la clave privada, aleatorización de claves y proporciona apoyo al usuario en comparación con el enfoque tradicional (p. 116).

En la investigación de Nisar et al. (2016) se estudió la evaluación del desempeño y seguridad de los algoritmos para el cifrado de imágenes con el objetivo de diseñar y desarrollar un benchmark basado en todos los parámetros necesarios para un buen cifrado de imágenes. Como resultados se concluyó que el cifrado basado en bloques (AES) es bueno para seguridad criptográfica en cuanto a la comunicación el canal esté libre de distorsiones y la imagen no es necesaria para ser comprimido, aunque el Cifrado tolerante a la compresión además Se ha demostrado que el esquema basado en el caos es un cifrado de elección, ya que tiene una alta seguridad criptográfica y rendimiento que ha sido demostrado junto con cierta tolerancia a la compresión o ruido (p. 28).

Finalmente, en la investigación de Ma y Ye (2015) se estudió un esquema de cifrado de imágenes basado en un híbrido de sistemas hipercaóticos con el objetivo de presentar un esquema de cifrado basado en un diferencial ordinario de sistema de ecuaciones. Como resultados se concluyó que el algoritmo propuesto tiene un alto nivel de seguridad y se podría utilizar para la transmisión de imágenes vía Internet. Además, se evaluó el rendimiento utilizando técnicas estadísticas como análisis de histogramas, Análisis de correlación de píxeles adyacentes, Correlación entre imágenes simples e imágenes cifradas, Análisis de entropía de información, etc. las cuales salieron favorables (p. 32).

En esta sección de la investigación se colocó las teorías relacionadas a la investigación con los sustentos teóricos encontrados en los artículos científicos relacionados con la variable, dimensiones e indicadores de esta investigación.

Unos de los algoritmos más populares actualmente es el algoritmo AES, es utilizado en la Criptografía simétrica debido a la seguridad que brinda y al

rendimiento que ofrece en el cifrado de información, Al respecto Rani y Kaur (2017) explicaron:

> AES pertenece a los algoritmos de cifrado de bloques de claves simétrico algoritmo diseñado por Vincent Rijmen y Joan Daemen en 1998. Se basa en la red Feistel y admite 128 tamaño de bloque de bits y longitud de clave 128, 192 y 256 bits. AES realiza 10, 12 o 14 rondas y el número de rondas depende de la clave. Significa para la longitud de cifra de 128 bits AES realiza 10 rondas, para la cifra de 192 bits realiza 12 rondas y para cifra de 256 bits realiza 14 rondas. (p. 183)

Con respecto a lo anterior, el algoritmo AES es un algoritmo estándar para la encriptación de información, debido a su complejidad, rendimiento y seguridad que ofrece. Al respecto, Fei et al. (2016) mencionaron:

> AES es un algoritmo de criptografía simétrica con alta seguridad y rendimiento, por lo que fue elegido como el algoritmo de criptografía estándar y se usa ampliamente en la actualidad, como Secure Socket Layer (SSL), etc. Sin embargo, la implementación de AES basada en software tiene el problema de la baja eficiencia; mientras que la implementación de AES basada en hardware necesita comprar dispositivos de propósito especial. El método de adoptar conjuntos de instrucciones y recursos de múltiples núcleos de la CPU para acelerar el algoritmo AES puede resolver los dos problemas. (p. 1099)

Se ha implementado y propuesto mejoras al rendimiento AES como, por ejemplo: AES-NI. Esta mejora brinda un mayor rendimiento en cuanto al tiempo de cifrado por que utiliza un CPU dedicado a Criptografia. Al respecto, Fei et al. (2016) agregaron:

> Debido a la base y universalidad de las operaciones de cifrado / descifrado AES, Intel agrega AES-New Instructions (AES-NI) para sus procesadores después de 2010. Por lo tanto, este documento propone un algoritmo de cifrado paralelo rápido llamado NIPAES, que se basa en AES NI y CPU multinúcleo. NIPAES adopta la propiedad de bloque de AES y la propiedad de paralelismo del modo Contador (CTR), y

> emplea OpenMP para distribuir cargas de trabajo de cifrado a cada subproceso de manera uniforme, lo que realizará AES-NI para completar el cifrado / descifrado. (p. 1100)

Con respecto al algoritmo AES, se menciona que fue creados en estados unidos para el remplazo del algoritmo DES que fue muy popular en la década de los 70. Al respecto, Bai (2015) explicó:

> AES es un algoritmo de cifrado simétrico de nueva generación. Propuesto para reemplazar el DES. Fue adoptado por el Gobierno de USA y ahora usado en todo el mundo. AES Encripta los datos en agrupación de 128 bits, mientras que tiene flexibilidad. Tamaños de cifra de 128, 256 y 192 bits, y sus números de Las rondas de repetición son 10, 14 y 12 respectivamente. SAES es la versión simplificada de AES. (p. 2)

De acuerdo con Chi-Wu, Ying-Hao, Shih-Hao, y Hsing-Chang (2011), mostraron el algoritmo AES (p. 3)

PseudoCódigo AES

Inicio

//Introducimos el mensaje a descifrar

Leer Estado

//Generación de subclaves

RondaLlave [NumeroRondas + 1] = Expansionclave (Clave);

//Primera subclave

AgregarRondaLlave (Estado, RondaLlave [NumeroRondas]);

//Rondas

PARA i= NumeroRondas mientras i<9 hacer

contador = Contador+1;

SistitucionBytes (estado); // este método realiza una sustitución no lineal de bytes (AesCaja).

TurnoFilas (estado); // método permite realizar una rotación cíclica hacia la izquierda

CombinacionColumans (estado); // en este método el estado se multiplica con una matriz definida

AgregarRondaLlave (estado, RondaLlave [i]); // la combinación de la clave con el estado

FINPARA

//Ronda final

SistitucionBytes (State);

TurnoFilas (State);

AgregarRondaLlave (estado, RondaLlave [i]);

Fin

TurnoFilas: este método permite realizar una rotación cíclica hacia la izquierda de forma que la fila inicial no cambie, la siguiente fila se mueve una posición a la izquierda en, la tercera fila se mueve a la izquierda en dos posiciones y, por último, la última fila se mueve en tres posiciones a la izquierda.

TurnoFilas (Estado)

INICIO

Definir variable temporal t[]

PARA i=1 mientras i<4

Contar ciclo i=i+1

PARA c=0 mientras c<4

Contar ciclo c=c+1

Obtener t[c] = Estado[i][(c+i)]; //el valor cambia a la variable temporal

FINPARA

PARA c=0 mientras c<4

Contar ciclo c=c+1

Obtener Estado[i][c] = t[c]; //el valor se copia a al estado

FINPARA

FINPARA

retornar Estado

FIN

SistitucionBytes: este método realiza una sustitución no lineal de bytes. Dicha sustitución se realiza utilizando la tabla s-box

SistitucionBytes (ESTADO)

INICIO

PARA r=0 mientras r<4

Contar ciclo r=r+1

PARA c=0 mientras c<Nb

Contar ciclo c=c+1

Obtener ESTADO[r][c] = **AesCaja** [ESTADO[r][c]];

FINPAR

FINPAR

retornar ESTADO

FIN

CombinacionColumans: en este método el estado se multiplica con una matriz definida

CombinacionColumans (ESTADO)

INICIO

PARA c=0 mientras c<Nb

Contar ciclo c=c+1

PARA r=0 mientras r<4

Contar ciclo r=r+1

Obtener ESTADO [r][c] = ESTADO [r][c] Xor MATRIZ[a][b]

FINPARA

Definir MATRIZ [0][0]=2; Define MATRIZ [0][1]=3; Define MATRIZ [0][2]=1; Define MATRIZ [0][3]=1;

Definir MATRIZ [1][0]=1; Define MATRIZ [1][1]=2; Define MATRIZ [1][2]=3; Define MATRIZ [1][3]=1;

Definir MATRIZ [2][0]=1; Define MATRIZ [2][1]=1; Define MATRIZ [2][2]=2; Define MATRIZ [2][3]=3;

Definir MATRIZ [3][0]=3; Define MATRIZ [3][1]=1; Define MATRIZ [3][2]=1; Define MATRIZ [3][3]=2;

FINPARA

retornar ESTADO;

FIN

AgregarRondaLlave: este método consiste en la combinación de la clave con el estado mediante una operación XOR

AgregarRondaLlave (ESTADO, RondaLlave)

INICIO

PARA r=0 mientras r<4

Contar ciclo r=r+1

PARA c=0 mientras r<Nb

Contar ciclo c=c+1

Obtener ESTADO [r][c] = ESTADO [r][c] Xor RONDALLAVE[r][c] ;

FINPARA

FINPARA

retornar ESTADO;

FIN

Definir AesCaja = **[** 0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5, 0x30, 0x01, 0x67, 0x2b, 0xfe, 0xd7, 0xab, 0x76, 0xca, 0x82, 0xc9, 0x7d, 0xfa, 0x59, 0x47, 0xf0, 0xad, 0xd4, 0xa2, 0xaf, 0x9c, 0xa4, 0x72, 0xc0, 0xb7, 0xfd, 0x93, 0x26, 0x36, 0x3f, 0xf7, 0xcc, 0x34, 0xa5, 0xe5, 0xf1, 0x71, 0xd8, 0x31, 0x15, 0x04, 0xc7, 0x23, 0xc3, 0x18, 0x96, 0x05, 0x9a, 0x07, 0x12, 0x80, 0xe2, 0xeb, 0x27, 0xb2, 0x75, 0x09, 0x83, 0x2c, 0x1a, 0x1b, 0x6e, 0x5a, 0xa0, 0x52, 0x3b, 0xd6, 0xb3, 0x29, 0xe3, 0x2f, 0x84, 0x53, 0xd1, 0x00, 0xed, 0x20, 0xfc, 0xb1, 0x5b, 0x6a, 0xcb, 0xbe, 0x39, 0x4a, 0x4c, 0x58, 0xcf, 0xd0, 0xef, 0xaa, 0xfb, 0x43, 0x4d, 0x33, 0x85, 0x45, 0xf9, 0x02, 0x7f, 0x50, 0x3c, 0x9f, 0xa8, 0x51, 0xa3, 0x40, 0x8f, 0x92, 0x9d, 0x38, 0xf5, 0xbc, 0xb6, 0xda, 0x21, 0x10, 0xff, 0xf3, 0xd2, 0xcd, 0x0c, 0x13, 0xec, 0x5f, 0x97, 0x44, 0x17, 0xc4, 0xa7, 0x7e, 0x3d, 0x64, 0x5d, 0x19, 0x73, 0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90, 0x88, 0x46, 0xee, 0xb8, 0x14, 0xde, 0x5e, 0x0b, 0xdb, 0xe0, 0x32, 0x3a, 0x0a, 0x49, 0x06, 0x24, 0x5c, 0xc2, 0xd3, 0xac, 0x62, 0x91, 0x95, 0xe4, 0x79, 0xe7, 0xc8, 0x37, 0x6d, 0x8d, 0xd5, 0x4e, 0xa9, 0x6c, 0x56, 0xf4, 0xea, 0x65, 0x7a, 0xae, 0x08, 0xba, 0x78, 0x25, 0x2e, 0x1c, 0xa6, 0xb4, 0xc6, 0xe8, 0xdd, 0x74, 0x1f, 0x4b, 0xbd, 0x8b, 0x8a, 0x70, 0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6, 0x0e, 0x61, 0x35, 0x57, 0xb9, 0x86, 0xc1, 0x1d, 0x9e, 0xe1, 0xf8, 0x98, 0x11, 0x69, 0xd9, 0x8e, 0x94, 0x9b, 0x1e, 0x87, 0xe9, 0xce, 0x55, 0x28, 0xdf, 0x8c, 0xa1, 0x89, 0x0d, 0xbf, 0xe6, 0x42, 0x68, 0x41, 0x99, 0x2d, 0x0f, 0xb0, 0x54, 0xbb, 0x16,**];**

Otro algoritmo más usado es el SHA ya que trabaja con un conjunto de funciones de hash. el algoritmo convierte una cadena de texto en un texto con valores hexadecimales. Al respecto, Sonia y Grewal (2016) mencionaron:

> En agosto 2002 agrega tres miembros (SHA-384, SHA-256, y SHA-512) a la familia de funciones SHA, seguidas por uno más (SHA-224) en 2004. Las conexiones entre las funciones aprobadas por el NIST. son los siguientes: Tanto SHA-256 como SHA-512 tienen similares diseños SHA-256 funciona con palabras de 32 bits, mientras que SHA512 opera con palabras de 64 bits. Ambos diseños llevan fuerte similar a SHA-1, aunque están mucho más cerca de entre sí que a su predecesor común. (p. 45)

El algoritmo SHA también se utiliza como complemento a los Sistemas caóticos y a los algoritmos criptográficos, de acuerdo con Gan et al. (2018) mencionaron:

> La función hash SHA 256 se utiliza para generar el parámetro del sistema y los valores iniciales del sistema caótico. Un conjunto de

> cajas S se construye con la permutación en zigzag [17] de la caja S original, y la posición inicial para la confusión del zigzag depende de la imagen plana. Luego, cada píxel de imagen se sustituye por el elemento elegido de la caja S elegida de acuerdo con la imagen. (p. 8763)

Con respecto a lo anterior el algoritmo se complementa a los sistemas caóticos, De acuerdo con Li et al. (2019) agregaron:

> Sobre el cifrado caótico, hay algunas ventajas que hacen que nuestra investigación sea más atractiva y significativa. Al principio, la imagen original se procesa y genera un conjunto de valores hash mediante el algoritmo SHA-3. El valor de hash generado y la imagen original se realizan en la operación XOR. los valores hash se procesan para generar el valor inicial del sistema caótico memcapacitor mediante la distancia de Hamming. (p. 2)

De acuerdo con Guesmi, Farah, Kachouri y Samet (2016), mostraron el algoritmo SHA-256 (p. 3).

pseudo Código SHA256

Función SHA-256

```
INICIO
    Leer m=Entrada;
    //Variables
    Definir hash [8]; // Valor de hash
    Definir w [63];
    Definir a, b, c, d, e, f, g, h; //Variables de estado.
    Definir k [63]; // Vector para la constante K
    Definir s0, s1;
    Definir S0, S1, maj, ch; //Variables auxiliares para la actualización de variables de estado
    Definir temp1, temp2;
```

Definir k ={0x428A2F98, 0x71374491, 0xB5C0FBCF, 0xE9B5DBA5, 0x3956C25B, 0x59F111F1, 0x923F82A4, 0xAB1C5ED5, 0xD807AA98, 0x12835B01, 0x243185BE, 0x550C7DC3, 0x72BE5D74, 0x80DEB1FE, 0x9BDC06A7, 0xC19BF174,0xE49B69C1, 0xEFBE4786, 0x0FC19DC6, 0x240CA1CC, 0x2DE92C6F, 0x4A7484AA, 0x5CB0A9DC, 0x76F988DA, 0x983E5152, 0xA831C66D, 0xB00327C8, 0xBF597FC7, 0xC6E00BF3, 0xD5A79147, 0x06CA6351, 0x14292967, 0x27B70A85, 0x2E1B2138, 0x4D2C6DFC, 0x53380D13, 0x650A7354, 0x766A0ABB, 0x81C2C92E, 0x92722C85, 0xA2BFE8A1, 0xA81A664B, 0xC24B8B70, 0xC76C51A3, 0xD192E819, 0xD6990624, 0xF40E3585, 0x106AA070, 0x19A4C116, 0x1E376C08, 0x2748774C, 0x34B0BCB5, 0x391C0CB3, 0x4ED8AA4A, 0x5B9CCA4F, 0x682E6FF3, 0x748F82EE, 0x78A5636F, 0x84C87814, 0x8CC70208, 0x90BEFFFA, 0xA4506CEB, 0xBEF9A3F7, 0xC67178F2}

// Inicialización del vector hash.

Definir hash [0] =0x6A09E667, hash [1] =0xBB67AE85, hash [2] =0x3C6EF372, hash [3] =0xA54FF53A, hash [4] =0x510E527F, hash [5] =0x9B05688C, hash [6] =0x1F83D9AB, hash [7] =0x5BE0CD19;

Para cada bloque b de 64 bytes en los datos m, hacer Hacer

Definer a = hash[0], b = hash[1] ,c = hash[2], d = hash[3], e = hash[4], f = hash[5], g = hash[6], h = hash[7]

Para i=0 hasta 63 hacer

Contar ciclo i = i +1;

// Procesamiento del vector w (Message Schedule).

Si (i < 16)

INICIOSI

Obtener w[i] = b[i]

FINSI

DELOCONTRARIO

INICIODELOCONTRARIO

Obtener s0 = (w[i − 15] , implica 7) XOR (w[i − 15] , implica 18) XOR (w[i − 15] 3)

Obtener s1 = (w[i − 2] , implica 17) XOR (w[i − 2] , implica 19) XOR (w[i − 2] 10)

Obtener w[i] = w[i − 16] + s0 + w[i − 7] + s1

// Actualización de las variables de estado a, b, c, d, f, g y h.

Obtener S1 = (e , implica 6) XOR (e , implica 11) XOR (e , implica 25)

Obtener ch = (e pero f) XOR ((¬e) pero g)

Obtener temp1 = h + S1 + ch + k[i] + w[i]

Obtener S0 = (a , implica 2) XOR (a , implica 13) XOR (a , implica 22)

Obtener maj = (a pero b) XOR (a pero c) XOR (b pero c)

Obtener temp2 = S0 + maj

Obtener h = g,

Obtener g = f,

Obtener f = e,

Obtener e = d + temp1,

Obtener d = c,

Obtener c = b,

Obtener b = a,

Obtener a = temp1 + temp2,

FINDELOCONTRARIO

FINPARACADA

//Actualización de Hash

Optener hash[0] = hash[0] + a, hash[1] = hash[1] + b, hash[2] = hash[2] + c, hash[3] = hash[3] + d, hash[4] = hash[4] + e, hash[5] = hash[5] + f, hash[6] = hash[6] + g, hash[7] = hash[7] + h

retornar hash,

FIN

El algoritmo PHASH se utiliza para encontrar similitudes entre hashes es usado mayormente para buscar infracciones de derechos de autor en línea, análisis de forense digital y la comparación de imágenes. Al respecto, Liu et al. (2020) indicaron:

> Perceptual Hash (Phash) se usa principalmente para la identificación de similitudes. Phash es una función hash que se puede utilizar en Crypto-hashing, marca de agua digital, y procesamiento de señales digitales. Hay cuatro tipos de Algoritmos Phash actualmente en uso: DCT (Discreto Transformada coseno) basado en hash; (2) Hash basado en el operador Marr-Hildreth; (3) Hash basado en varianza radial; (4) Bloquear valor medio basado en hash. (p. 2)

Por otro lado, para la comparación de imagen se debe tener en cuenta algunos aspectos como marcas de agua, filtros, marcos, colores, tamño de images, etc. Al respecto, Chamoso, Rivas, Sánchez y Rodríguez (2018) mencionaron:

> Al analizar si dos imágenes son iguales, es necesario realizar una serie de Verificaciones debido al hecho de que dos imágenes aparentemente idénticas pueden ser computacionalmente diferentes por su compresión, diferencia de calidad, número de colores o tamaño, ligeras modificaciones de la imagen con filtros o marcas de agua, cambios en la tonalidad, inserción de marcos o rotaciones. Existen diferentes metodologías que intentan dar solución a un similar problema, pero muchos de ellos no cubren todas las posibles transformaciones que se pueden hacer para imagen sin que parezca diferente al ojo humano. (p. 2)

Con respecto a lo anterior, Chamoso, Rivas, Sánchez y Rodríguez (2018) agregaron:

> Existen diferentes algoritmos propuestos basados en la técnica de generación de valor hash: PHASH (también llamado “Perceptive Hash”, con diferentes variaciones), aHash (también llamado Average Hash o Mean Hash) y dHash también llamado Difference Hash). El típico algoritmo basado en hash El diagrama de flujo de los algoritmos

se muestra en Sin embargo, todas las variaciones de esta metodología presentan diferentes problemas cuando se trata de una imagen que se ha girado o a la que se le ha añadido un marco. (p.10)

pseudo Código PHASH

Proceso PHASH (<u>imagen</u>)

```
Leer size = 32;
Leer smallerSize = 8;
Leer img = imagen;
Leer vals = []; //valores de los pizeles
//Calculo de los pixeles de la imagen
PARA (x = 0; x < img.width; x++)
        vals[x] = [];
        PARA (y = 0; y < img.height; y++)
                vals[x][y] = intToRGBA(img.getPixelColor(x, y)).b;
        FINPAR
FINPARA
//-------------------------------------------------------
//Calculo de la DCT 32*32 - comprension de imagenes  (transformada de coseno discreta)
Leer dctVals = applyDCT(vals, size); // Reducir la DCT.
//-------------------------------------------------------
//Calcule el valor promedio usando frecuencia 8x8
LEER total = 0;
 PARA (var _x = 0; _x < smallerSize; _x++) {
   PARA (var _y = 0; _y <smallerSize; _y++) {
     total += dctVals[_x][_y];
   FINPARA
FINPARA
 Leer avg = total / (this.smallerSize * this.smallerSize);
//-------------------------------------------------------
```

```
// Reducir aún más la DCT. Establecer hash en 0 o 1 dependiendo de si cada uno de los 64 valores está por encima o por debajo del valor medio.
Leer hash = '';
  PARA (_x2 = 0; _x2 <smallerSize; _x2++) {
    PARA (_y2 = 0; _y2 <smallerSize; _y2++) {
      hash += dctVals[_x2][_y2] > avg ? '1' : '0';
    FINPARA
  FINPARA
//-------------------------------------------------------------
//hash 0100100100000110010110001001011 1
return hash;
```

FinProceso

En esta sección se trató sobre los indicadores que se evaluaron en la investigación.

El Tiempo es una magnitud física con la que se medirá cuanto se demora el algoritmo propuesto en encriptar y desencriptar una imagen. En cuanto a la Dimensión Tiempo, se consideró el tiempo que demoro el algoritmo en cifrar una imagen, de acuerdo con Liu y Miao (2016) agregaron: “Ciertamente, Cuanto más tiempo se necesita y más seguro es el algoritmo. Por lo tanto, los usuarios pueden elegir una T adecuada para sus diferentes demandas de seguridad.” (p. 11)

De acuerdo con lo anterior, se muestra como fue la medición del tiempo del algoritmo propuesto en comparación con otro algoritmo, de acuerdo con Liu y Miao (2016) mencionaron: “el tiempo de nuestro algoritmo con T = 1 y 2 es menor que el de los algoritmos DES y AES, y es bastante aceptable para el cifrado de imágenes. “(p. 11)

Por otro lado, En otros trabajos previos se realizó el análisis de la complejidad del tiempo de codificación de la imagen, de acuerdo con Wang y Liu (2017) explicaron:

> Las direcciones ejecutadas con mayor frecuencia y que consumen más tiempo son la codificación de ADN y decodificación ADN.

> Supongamos que el tamaño de la imagen plana es M × N, lo que significa que hay M × N píxeles en una imagen plana. Aunque el algoritmo escanea una imagen simple fila por fila, en realidad, Cada píxel de la imagen plana se codifica o decodifica. Así que la complejidad básica del tiempo es O (M × N). Además, cuando se opera una imagen codificada sin formato y una imagen de clave codificada fila por fila, la complejidad del tiempo es O (4 × M × N). (p. 6241)

Por otro lado, Aqeel et al. (2016) mencionaron: “Comparación del tiempo de descifrado de encriptación de imágenes grises de 8 bits para diferentes tamaños (en segundos).” (p. 11256).

Otra dimensión evaluada en la integridad es fundamental en el desarrollo de un algoritmo de encriptación debido a que se busca es mostrar la imagen sin modificaciones y sin perdidas de datos.

Con respecto a la Dimensión integridad, algunos autores describen como mantener la integridad de las imágenes encriptadas, Al respecto Kumar y Chauhan, (2018) mencionaron:

> Algoritmos de cifrado fuertes y las técnicas de gestión de claves optimizadas siempre ayudan a conseguir la autenticación, confidencialidad e integridad de datos y reducir los gastos generales del sistema. El largo la longitud de la clave requiere más tiempo de cálculo para descifrar el código y se vuelve difícil para el hacker detectar el modelo criptográfico. (p. 60)

Con respecto a lo anterior, si un pixel tiene un valor diferente en la desencriptación de una imagen, el algoritmo mostrara una imagen diferente a la original, Al respecto Wang y Liu (2017) explicaron:

> Message Digest Algorithm 5 (MD5) esta función hash fue ampliamente utilizada, que genera un valor de hash de 128 bits que normalmente se presenta como un número hexadecimal de 32 dígitos literalmente [25]. Al respecto la Ref. [8, 20], debido a su buena

característica de seguridad, incluso un cambio de un bit puede llevar a una diferencia significativa entre dos imágenes. (p. 6232)

Por otro lado, para poder comprobar la integridad de las imágenes se compara la imagen original con la imagen desencriptada para observar si hubo un cambio de pixeles entre las imágenes. Al respecto, Pan et al. (2018) mencionaron:

> La Figura 4 muestra los efectos de cifrado y descifrado de Foto a color Se puede ver en los resultados de la Fig. 4 que La diferencia entre el histograma y la escala de grises. La imagen de la fotografía en color es mayor que la de Fig. 3. Sin embargo, a partir de la imagen en escala de grises, el rendimiento del gráfico encriptado y descifrado en escala de grises es no es la misma que la de la Fig. 3, y es relativamente suave en El histograma de la imagen cifrada en texto plano. Puede ser Visto desde el proceso de cifrado y descifrado de la vida. Gráfico que este trabajo ha logrado buenos resultados. (p. 8)

Finalmente, los Recursos de Hardware son un aspecto fundamental en un algoritmo debido a que cumplirá su tarea y mostrará resultados en un tiempo determinado dependiendo a la capacidad de Hardware.

Sobre la dimensión Recursos de hardware, los algoritmos criptográficos brindan una mayor eficiencia si se cuenta con el hardware adecuado para la encriptación de información, Al respecto Cevallos (2019) menciono:

> El uso de Computación de alto rendimiento es la mejor opción cuando hay recursos computacionales disponibles. En se presenta un mecanismo para mejorar significativamente el rendimiento de los algoritmos asimétricos la clave que utilizan un middleware de cuadrícula computacional de igual a igual esquema. Sin embargo, puede que no sea posible emplear esta técnica. en todos los entornos donde se requiera criptografía. (p. 108)

Con respecto a lo anterior, algunos trabajos previos muestran el tiempo de encriptación según los recursos de hardware utilizados. Al respecto, Liu y Miao (2016) agregaron:

> Comparamos la complejidad computacional de nuestro algoritmo con los algoritmos tradicionales DES y AES. Todos los algoritmos son experimentados por Matlab R2014a en la computadora con CPU a 3.6 GHz y 8 GB de memoria. Los resultados obtenidos se podrán encontrar en la Tabla 4. (p. 11)

Por otro lado, algunos autores muestran el proceso de encriptación de imágenes con respecto a los recursos de hardware utilizados. Al respecto, Fei et al. (2016) mencionaron: “Los experimentos se realizan en un servidor de alto rendimiento con dos CPU Intel Xeon E5-2640 v2., Hynix DDR3 RAM 62.9 GB y el sistema operativo Linux.” (p. 1104). Con respecto a, Liu y Miao (2016) mencionaron: “Todos los algoritmos son experimentados por Matlab R2014a en el Computadora con CPU a 3.6 GHz y 8 GB de memoria.” (p. 11).

De acuerdo con, Aqeel et al. (2016) indicaron:

> El análisis de complejidad de tiempo se logra con un CPU Intel® Pentium® de 2,3 GHz con 2048 MB de RAM en computadora personal. El algoritmo está codificado en Matlab R2012a y compilado por 7.14 en Windows 7 Home . (p. 11256).

En esta sección se describen algunos algoritmos criptograficos, herramientas de medición y la metodología de desarrollo que se va a utilizar.

El algoritmo RC6 es un cifrado por bloques y pertenece a la Criptografía asimétrica, además está basada en el algoritmo RC5. De acuerdo con Hu et al. (2016) mencionaron:

> RC6 tiene una estructura y descripción simples en relación con los otros cifrados de bloque propuestos para tomar el lugar de AES. RC6 posee un bloque de 128 bits y admite tamaños de clave de 128, 256 y 192 bits, pero, como RC5, puede parametrizarse para admitir una amplia variedad de tamaños de palabra, tamaños de clave y número de rondas. (p. 12648)

Con respecto a lo anterior, el algoritmo RC6 presenta una operación extra para realizar el cifrado en comparación de RC5. Al respecto, Hu et al. (2016) explicaron:

> RC6 es muy similar a su antecesor RC5 en su estructura, al usar rotaciones, adición modular y operaciones XOR; de hecho, RC6 se puede ver como interconectando dos procesos de encriptación RC5 paralelos, sin embargo, RC6 usa una operación de multiplicación extra 32 × 32 no presente en RC5 para hacer que la rotación sea dependiente en cada bit en una palabra, y no solo en los pocos bits menos significativos, que se ejecutan rápidamente en los procesadores modernos y aumentan la seguridad y la eficiencia. (p. 12649)

El algoritmo RSA es un algoritmo que utiliza clave pública y privada para encriptar mensajes o realizar firmas digitales. El algoritmo brinda una seguridad superior con respecto a la codificación de archivos. Al respecto Rani y Kaur (2017) agregaron:

> RSA es el algoritmo asimétrico más popular y probado se basa en el hecho matemático que es fácil de encontrar lo privado y claves públicas basadas en números primos muy grandes. Cifrado: calcular c = m ^ e mod n, donde e y n son la llave pública y m es el bloque de mensajes. La c es el mensaje cifrado. Descifrado: la clave privada d se utiliza para descifrar mensajes. Calcular: m = c ^ d mod n, donde n es el módulo y d es la clave privada. (p. 184)

El algoritmo RSA proporciona una llave publica que es compartida con todos, Sin embargo, también proporciona una clave privada que solo lo conoce el usuario. Al respecto, Cevallos (2019) explicaron:

> Los algoritmos de llave simétrica tienen llaves más cortas en comparación con las claves de algoritmos asimétricos como RSA (Rivest, Shamir y Adleman), El Gamal o ECC. Por ejemplo, AES usa una llave de un de 256 bits, mientras que RSA funciona con un tamaño de llave mínima de 1024 bits. (p. 104)

Se realizo una comparación entre algoritmos de encriptación para determinar que algoritmos utilizar y combinar para la encriptación de imágenes, lo cual se determinó que el algoritmo AES es más conveniente para realizar

encriptación utilizando los algoritmos SHA y PHASH como complemento. Ver Anexo 3.

Algunas herramientas de medición estudiadas se encuentran ionforge imagediff es un programa gratuito que tiene varias funciones para la administración de imágenes, Es muy fácil de utilizar y brinda un mayor rendimiento. Para la comparación de imágenes tiene 4 módulos los cuales son: Rayos X, Depredador, Térmico y Monocromo. De acuerdo con, Méndez et al. (2017) indico: “el programa WinHex ayuda en la comparación de forma visual las desigualdades del código HEX de imágenes y el programa IonForge se utiliza para encontrar diferencias pixel a pixel de imágenes.” (p. 12)

El programa Awesome Duplicate Photo FinderEs que permite encontrar y eliminar fotos duplicadas. Este programa es muy fácil de usar además muestra el porcentaje de similitud de las imágenes. Awesome Duplicate Photo finder puede comparar imágenes redimensionadas o incluso imágenes con colores corregidos (fotos en blanco y negro, Etc.). Admite todos los tipos de imagen principales: Jpg, Bmp, Gif, Png. (Duplicate finder, 2019)

En esta investigación se utilizará una marco de trabajo ágil para el desarrollo de software debido a que permite los cambios de los requerimientos y soluciones según las necesidades del proyecto.

Scrum es un framework para el desarrollo de software, pertenece al grupo de metodologías agiles. Esta metodología está enfocado a las buenas prácticas para tener un trabajo en equipo colaborativo para obtener los mejores resultados posibles. Al respecto, Ashraf, y Aftab (2017). indicaron:

> Con mayor productividad, menor tiempo de comercialización el producto, mejor colaboración, retroalimentación continua, y capacidad para aceptar el cambio, Scrum es reconocido como modelo ágil más utilizado actualmente. Según, 61% de los encuestados que eran de 76 países usan Melé. Scrum ofrece una amplia gama de prácticas de gestión. (p. 12)

Con respecto a lo anterior, los equipos evalúan la mejor forma de realizar las actividades en base a los conocimientos adquiridos. Al respecto. Ashraf, y Aftab (2017) explicaron:

> Se requiere la adaptación del modelo de proceso Scrum para realizar un proyecto con éxito y cumplir con los objetivos. Planificación, diseño, aseguramiento de la calidad del software, las pruebas, el rendimiento del equipo y la comunicación son algunas áreas clave donde los profesionales enfrentan desafíos que deben abordarse con prudencia a medida que las prácticas del proceso Scrum no los apoye formalmente. Diferentes investigadores y los practicantes han estado haciendo esfuerzos para evolucionar Scrum durante la última década para abordar estos desafíos. La mayoría de ellos traté de optimizar el Scrum tradicional con un Adopción pragmática de la filosofía ágil. (p. 12)

Por otro lado, Scrum ayuda aumentar la productividad del desarrollo del software en base a los requerimientos del cliente y la participación del equipo. Al respecto, Ashraf y Aftab (2017) índico: “Varios estudios secundarios se centraron principalmente en Scrum, su evolución, adopción y tendencias de uso en la industria. Alguna evidencia empírica consolidada para revelar la asociación entre Scrum y aumento productividad.” (p. 73).

Se realizo una comparación entre las metodologías scrum, xp y kanban para determinar que metodología utilizar para el uso del algoritmo, lo cual se determinó que la metodología SCRUM es la más conveniente para realizar softwares. Ver Anexo 4.

Esta metodología cuenta con fases de planificación del proyecto, fase de codificación, pruebas y diseño las cuales se mostrarán en este trabajo.

III. METODOLOGÍA

3.1 Tipo y Diseño de la investigación

El modelo de investigación realizado es cuantitativo. Para definirlos Hernández, Fernández y Baptista (2016) indicaron:

> Lo cuantitativo es probatorio y secuencial. Cada etapa es importante el orden es riguroso, sin embargo, se puede redefinir alguna fase. Se determinan variables y de las preguntas hipótesis; se realiza un plan para sustentar; se revisan las mediciones capturadas usando metodologías estadísticas, y se proporciona conclusiones respecto de la o las hipótesis. (p.124)

El diseño es experimental porque no hay un manejo intencional de variables para generar un resultado. Al respecto, Hernández et al. (2016) mencionaron:

> Un estudio no experimental no genera alguna situación, sino que se observan casos que existen, no generados en la investigación por el investigador. En el diseño no experimental las variables independientes suceden y no es manipulables, no hay control intencional sobre las variables ni se pueden modificar, porque ya pasaron en un tiempo. (p. 152)

A su vez esta investigación se encuentra en el diseño transversal descriptivo, debido a que indagamos los valores que se manifiestan en la variable y medimos un grupo de objetos. Al respecto, Hernández et al. (2016) mencionaron: “el diseño de investigación transversal recopila datos de un momento único. La finalidad es explicar variables y examinar su incidencia en un tiempo dado.” (p. 154).

Con respecto a lo anterior, Hernández et al. (2016) mencionaron:

> el diseño transeccional descriptivos tiene el objetivo de indagar la incidencia o niveles de las variables en una población. La técnica consiste encontrar una o múltiples variables a un conjunto objetos, situaciones, seres vivos, contextos, comunidades, fenómenos, etc., y facilitar su descripción. (p. 155)

3.2 Variables y Operacionalización

Variable Dependiente

Impacto de un algoritmo basado en AES, SHA y PHASH para la encriptación de imágenes.

3.3 Población, muestra y muestreo

Población

Para este proyecto se tomó como población las imágenes digitales mostradas en la página web Pixabay. Lo cual son de 2 millones de imágenes sin copyright (Pixabay, 2020).

Muestra

En esta investigación se realizó una muestra por conveniencia, para lo cual se tomó como muestra 300 imágenes digitales de la página web Pixabay las cuales pertenecen a las más populares. El muestreo por conveniencia.

La selección de la muestra se ha determinado según los recursos disponibles para esta investigación y el tiempo de desarrollo de esta investigación.

Muestreo

Se utilizo el muestreo no probabilista utilizando como tecnica la muestreo por conveniencia. De acuerdo con Otzen y Manterola (2017), la muestra por conveniencia consiste en seleccionar individuos de fácil accesibilidad y proximidad de los objetos para la investigación (p. 230).

Li et al. (2019), Pan et al. (2018), Kumar y chauhan(2018), escogieron sola 1 imagen aleatorias para las respectivas pruebas del algoritmos, Aqeel et al. (2016), utilizaron 30 imágenes aleatorias.

3.4 Técnicas e instrumentos de recolección de datos, validez y confiabilidad

En la investigación se utilizó el procedimiento de recolección de datos por medio de la visualización. Al respecto, Pulido (2015) indico:

> La observación es un procedimiento que admite la recopilación de información sobre el desarrolla de un objeto social. Este significado implica dos puntos principales: los datos se recopilan durante el suceso para su posterior análisis; por ultimo el suceso no es mantenido, finalizado o creado para la investigación, debido a que se refería a un método experimental (p. 1149)

Como instrumentos de recolección de datos se utilizó fichas de recolección de datos donde se detallan los resultados obtenidos con respecto a los indicadores planteados en esta investigación (Anexo 2).

Validez

La validez es la capacidad de medición de una herramienta además se refiere a la precisión del instrumento cuando mide.

Se usará la validez de criterio por medio de las dimensiones expuestas anteriormente, de acuerdo con, Hernández et al. (2016) indicaron: “validez de criterio es una herramienta de que se constituye al cotejar resultados con algún criterio que intenta medir lo mismo.” (p. 202).

Se usará validez contenido para la elaboración del instrumento de medición (ficha de recolección de datos), además se usará las pruebas estadísticas con un nivel de confianza del 95%. Al Respecto a la validez de Contenido, Hernández et al. (2016) mencionaron: “La validez de contenido es el nivel donde una herramienta refleja conocimientos sobre el contenido que se va a medir. Es el grado en el que la medición representa al concepto o variable medida.” (p. 201).

Para la validación de los indicadores mencionados se utilizó la validez de contenido y el de juicio de expertos. Los indicadores utilizados se obtuvieron de los trabajos previos relacionados al tema. Realizados por, Wang y Liu (2017, p. 6241), Liu y Miao (2016, p. 11) y Pan et al. (2018, p. 7)

3.5 Procedimientos

el modo de recolección de información se realiza por medio de una base de datos de imágenes pública. Las imágenes se obtuvieron de la página web Pixabay la cual nos permite copiar, modificar, distribuir y usar las imágenes, incluso para fines comerciales, sin solicitar permisos. (Pixabay, 2019).

3.6 Métodos de análisis de datos

El método de análisis de datos utilizados para la investigación es cuantitativo. Se utiliza la recolección de datos para sustentar la hipótesis basándose en análisis estadísticos

Pruebas de normalidad

Para la prueba de normalidad se utilizó la prueba de Kolgomorov Smirnov Si es que la muestra tiene más de 50 registros. De acuerdo con Aslam (2015), La prueba K-S bajo estadística clásica se aplica cuando todas las observaciones en los datos son determinadas, precisas y seguras. Sin embargo, en situaciones reales, puede suceder que los datos no estar representados por términos estadísticos o los datos pueden estar en un intervalo o datos imprecisos, varios autores desarrollaron el K-S prueba para el análisis de datos difusos.

Por otro lado, Romero (2016) indico: “se trata de un estudio de significación estadística para corroborar los datos de la muestra pertenecen a una distribución normal. Se utiliza para variables cuantitativas y si el tamaño muestral supera los 50”. (p. 36)

3.7 Aspectos éticos

En esta investigación se está considerando la autoría de los diversos artículos científicos con el adecuado citado y referenciado con estilo APA, por lo que la información mostrada es genuina y veraz. No se está violando los principios éticos establecidos en el Código de Ética de la Universidad César Vallejo.

IV. RESULTADOS

En el presente capítulo, se detallan los resultados por parte de la investigación tomando para el análisis los indicadores “Tiempo de encriptación”, “integridad de las imágenes” y “Consumo de Hardware”. De igual forma, se realiza el procesamiento de la información de la implementación del algoritmo para la encriptación de imágenes.

Contrastación de Hipótesis

Primer Indicador: Porcentaje de diferencia entre el tiempo de encriptación del algoritmo propuesto con respecto al algoritmo AES.

HIPÓTESIS ESPÉCIFICA

El algoritmo de encriptación de imágenes basado en AES, SHA y PHASH reduce el tiempo de encriptación de imágenes.

HIPÓTESIS ESTADÍSTICAS

HIPOTESIS NULA (H0): El algoritmo de encriptación de imágenes basado en AES, SHA y PHASH reduce el tiempo de encriptación de imágenes.

HIPOTESIS ALTERNATIVA (Ha): El algoritmo de encriptación de imágenes basado en AES, SHA y PHASH no reduce el tiempo de encriptación de imágenes.

NIVEL DE SIGNIFICANCIA

Se determina el margen de error con una confiabilidad del 95%. Usando un nivel de significancia del 5% (α = 0.05). Por lo tanto, el nivel de confianza será del 95% (1 − α = 0.95).

PRUEBA ESTADÍSTICA DE NORMALIDAD:

Como el número de muestra para este indicador es de 300 imágenes se emplea la prueba de normalidad de Kolmogórov-Smirnov mediante el uso del aplicativo IBM SPSS.

Tabla 1 *Prueba de Normalidad- Primer Indicador*

	Kolmogórov-Smirnov			Shapiro-Wilk		
	Estadístico	gl	Sig.	Estadístico	gl	Sig.
Porcentaje de diferencia entre el tiempo de encriptación del algoritmo AES-SHA-PHASH con respecto al algoritmo AES (segundos)	,139	300	,000	,895	300	,000

a. Corrección de significación de Lilliefors

Tabla 2 *Prueba de Normalidad Kolmogorov- Primer Indicador*

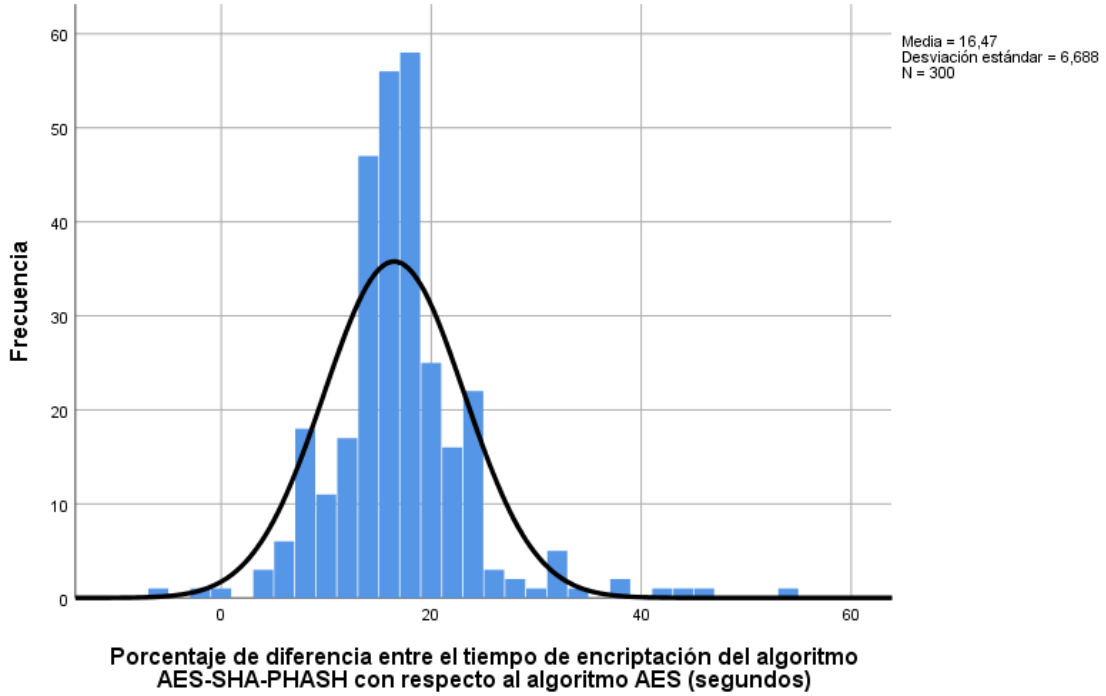
		Porcentaje de diferencia entre el tiempo de encriptación del algoritmo AES-SHA-PHASH con respecto al algoritmo AES (segundos)
N		300
Parámetros normales[a,b]	Media	16,47
	Desv. Desviación	6,688
Máximas diferencias extremas	Absoluto	,139
	Positivo	,139
	Negativo	-,116
Estadístico de prueba		,139
Sig. asintótica(bilateral)		,000[c]

a. La distribución de prueba es normal.

b. Se calcula a partir de datos.

c. Corrección de significación de Lilliefors.

Figura 1. Histograma del indicador 1



Al aplicar la prueba de normalidad en el programa IBM SPSS, nos detalla que las muestras no son normales, por ende, aplicamos pruebas no paramétricas. (wilcoxon)

Figura 2. Prueba Wilcoxon – Primer Indicador

Resumen de prueba de hipótesis

	Hipótesis nula	Prueba	Sig.	Decisión
1	La mediana de Porcentaje de diferencia entre el tiempo de encriptación del algoritmo AES-SHA-PHASH con respecto al algoritmo AES (segundos) es igual a 15.	Prueba de rangos con signo de Wilcoxon para una muestra	,000	Rechazar la hipótesis nula.

Se muestran significaciones asintóticas. El nivel de significación es de ,05.

Comprobamos el nivel de significación, si es menor que 0.05 la distribución no es normal, si es mayor que 0.05 la distribución es normal. Si P < 0.05 rechazamos la hipótesis nula y aceptamos la hipótesis alterna, por ende, la hipótesis aceptada es, El algoritmo de encriptación de imágenes basado en AES, SHA y PHASH No reduce el tiempo de encriptación de imágenes.

Segundo Indicador: Porcentaje de similitud de la imagen original y la imagen desencriptada.

HIPÓTESIS ESPÉCIFICA

El algoritmo de encriptación de imágenes basado en AES, SHA y PHASH reduce el tiempo de encriptación de imágenes.

HIPÓTESIS ESTADÍSTICAS

HIPOTESIS NULA (H0): El algoritmo de encriptación de imágenes basado en AES, SHA y PHASH garantiza la Integridad de las imágenes**.**

HIPOTESIS ALTERNATIVA (Ha): El algoritmo de encriptación de imágenes basado en AES, SHA y PHASH no garantiza la Integridad de las imágenes**.**

NIVEL DE SIGNIFICANCIA

Se define el margen de error con una confiabilidad del 95%. Usando un nivel de significancia del 5% (α = 0.05). Por lo tanto, el nivel de confianza será del 95% (1 − α = 0.95).

PRUEBA ESTADÍSTICA DE NORMALIDAD:

Como el número de muestra para este indicador es de 300 imágenes se emplea la prueba de normalidad de Kolmogórov-Smirnov mediante el uso del aplicativo IBM SPSS.

Tabla 3 *Prueba de Normalidad- Segundo Indicador*

	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Estadístico	gl	Sig.	Estadístico	gl	Sig.
Porcentaje de similitud entre la imagen original y la imagen desencriptada (segundos)	,530	300	,000	,073	300	,000

a. Corrección de significación de Lilliefors

Tabla 4 *Prueba de Normalidad Kolmogorov – Segundo Indicador*

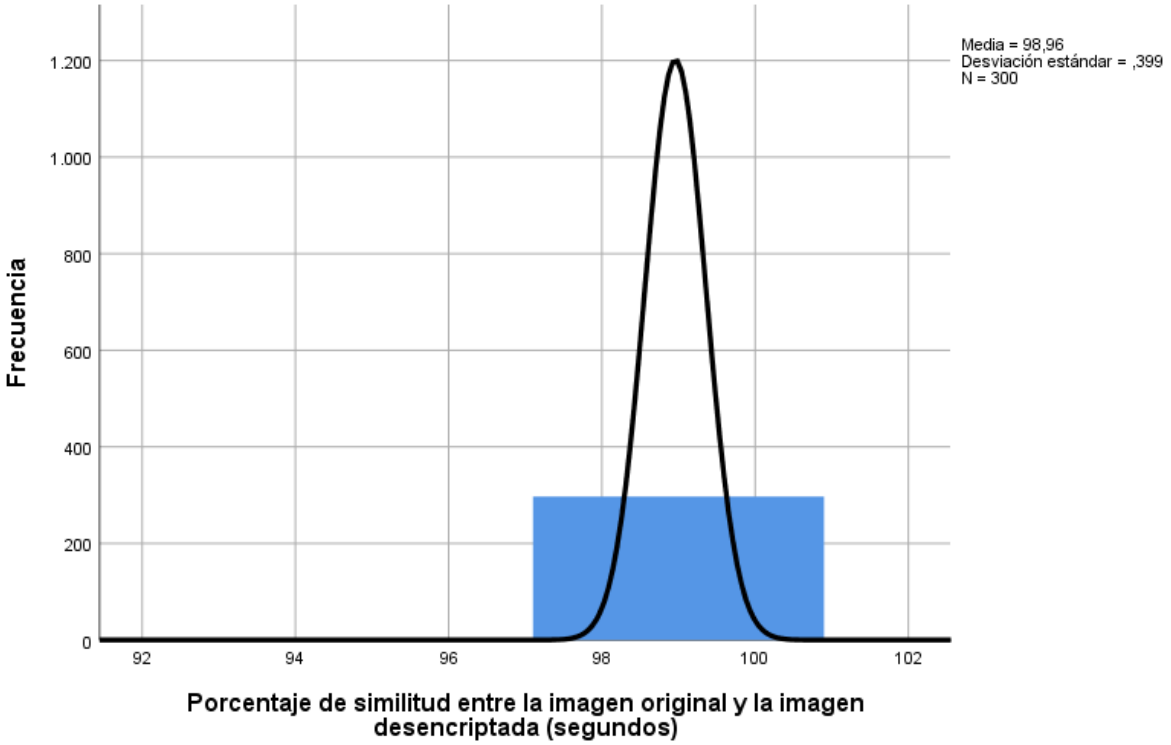
		Porcentaje de similitud entre la imagen original y la imagen desencriptada (segundos)
N		300
Parámetros normales[a,b]	Media	98,96
	Desv. Desviación	,399
Máximas diferencias extremas	Absoluto	,530
	Positivo	,460
	Negativo	-,530
Estadístico de prueba		,530
Sig. asintótica(bilateral)		,000[c]

a. La distribución de prueba es normal.

b. Se calcula a partir de datos.

c. Corrección de significación de Lilliefors.

Figura 3. Histograma del indicador 2



Al aplicar la prueba de normalidad en el programa IBM SPSS, nos detalla que las muestras no son normales, por ende, aplicamos pruebas no paramétricas. (wilcoxon)

Figura 4. Prueba Wilcoxon – Segundo Indicador

Resumen de prueba de hipótesis

	Hipótesis nula	Prueba	Sig.	Decisión
1	La mediana de Porcentaje de similitud entre la imagen original y la imagen desencriptada (segundos) es igual a 99.	Prueba de rangos con signo de Wilcoxon para una muestra	,083	Retener la hipótesis nula.

Se muestran significaciones asintóticas. El nivel de significación es de ,05.

Comprobamos el nivel de significación, si es menor que 0.05 la distribución no es normal, si es mayor que 0.05 la distribución es normal. Si P > 0.05 aceptamos la hipótesis nula y rechazamos la hipótesis alterna, por ende, la hipótesis aceptada es, El algoritmo de encriptación de imágenes basado en AES, SHA y PHASH garantiza la Integridad de las imágenes.

Tercer Indicador: Porcentaje de uso del CPU.

HIPÓTESIS ESPÉCIFICA

El algoritmo de encriptación de imágenes basado en AES, SHA y PHASH brinda un consumo de hardware óptimo.

HIPÓTESIS ESTADÍSTICAS

HIPOTESIS NULA (H0): El algoritmo de encriptación de imágenes basado en AES, SHA y PHASH brinda un consumo de hardware óptimo.

HIPOTESIS ALTERNATIVA (Ha): El algoritmo de encriptación de imágenes basado en AES, SHA y PHASH no brinda un consumo de hardware óptimo.

NIVEL DE SIGNIFICANCIA

Se define el margen de error con una confiabilidad del 95%. Usando un nivel de significancia del 5% (α = 0.05). Por lo tanto, el nivel de confianza será del 95% (1 − α = 0.95).

PRUEBA ESTADÍSTICA DE NORMALIDAD:

Como el número de muestra para este indicador es de 300 imágenes se emplea la prueba de normalidad de Kolmogórov-Smirnov mediante el uso del aplicativo IBM SPSS.

Tabla 5 *Prueba de Normalidad- Tercer Indicador*

	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Estadístico	gl	Sig.	Estadístico	gl	Sig.
Porcentaje de uso de CPU	,104	300	,000	,943	300	,000

a. Corrección de significación de Lilliefors

Tabla 6 *Prueba de Normalidad Kolmogorov – Tercer Indicador*

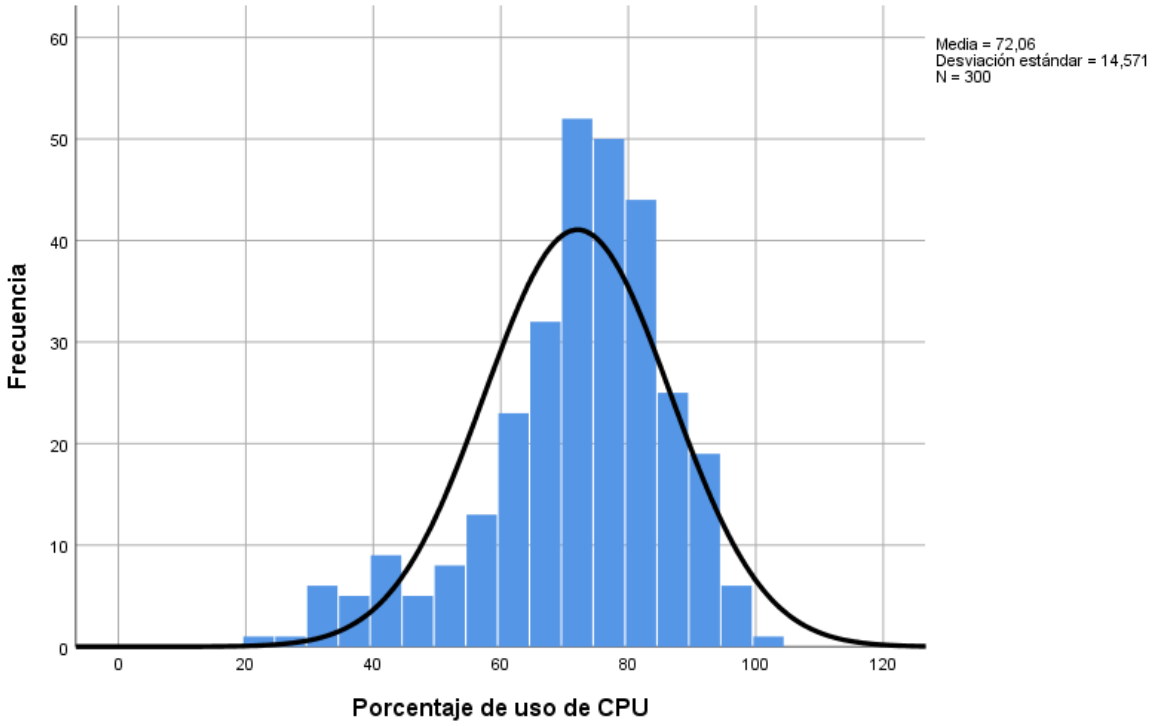
		Porcentaje de uso de CPU
N		300
Parámetros normales[a,b]	Media	72,06
	Desv. Desviación	14,571
Máximas diferencias extremas	Absoluto	,104
	Positivo	,047
	Negativo	-,104
Estadístico de prueba		,104
Sig. asintótica(bilateral)		,000[c]

a. La distribución de prueba es normal.

b. Se calcula a partir de datos.

c. Corrección de significación de Lilliefors.

Figura 5. Histograma del indicador 3



Al aplicar la prueba de normalidad en el programa IBM SPSS, nos detalla que las muestras no son normales, por ende, aplicamos pruebas no paramétricas. (wilcoxon)

Figura 6. Prueba Wilcoxon – Tercer Indicador

Resumen de prueba de hipótesis

	Hipótesis nula	Prueba	Sig.	Decisión
1	La mediana de Porcentaje de uso de CPU es igual a 70.	Prueba de rangos con signo de Wilcoxon para una muestra	,000	Rechazar la hipótesis nula.

Se muestran significaciones asintóticas. El nivel de significación es de ,05.

Comprobamos el nivel de significación, si es menor que 0.05 la distribución no es normal, si es mayor que 0.05 la distribución es normal. Si P < 0.05 rechazamos la hipótesis nula y aceptamos la hipótesis alterna, por ende, la hipótesis aceptada es, El algoritmo de encriptación de imágenes basado en AES, SHA y PHASH no brinda un consumo de hardware óptimo con respecto al CPU usado.

Cuarto Indicador: Porcentaje de uso de RAM.

HIPÓTESIS ESPÉCIFICA

El algoritmo de encriptación de imágenes basado en AES, SHA y PHASH brinda un consumo de hardware óptimo.

HIPÓTESIS ESTADÍSTICAS

HIPOTESIS NULA (H0): El algoritmo de encriptación de imágenes basado en AES, SHA y PHASH brinda un consumo de hardware óptimo.

HIPOTESIS ALTERNATIVA (Ha): El algoritmo de encriptación de imágenes basado en AES, SHA y PHASH no brinda un consumo de hardware óptimo.

NIVEL DE SIGNIFICANCIA

Se define el margen de error con una confiabilidad del 95%. Usando un nivel de significancia del 5% (α = 0.05). Por lo tanto, el nivel de confianza será del 95% (1 − α = 0.95).

PRUEBA ESTADÍSTICA DE NORMALIDAD:

Como el número de muestra para este indicador es de 300 imágenes se emplea la prueba de normalidad de Kolmogórov-Smirnov mediante el uso del aplicativo IBM SPSS.

Tabla 7 *Prueba de Normalidad- Cuarto Indicador*

	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Estadístico	gl	Sig.	Estadístico	gl	Sig.
Porcentaje de uso de RAM	,089	300	,000	,979	300	,000

a. Corrección de significación de Lilliefors

Tabla 8 *Prueba de Normalidad Kolmogorov – Cuarto Indicador*

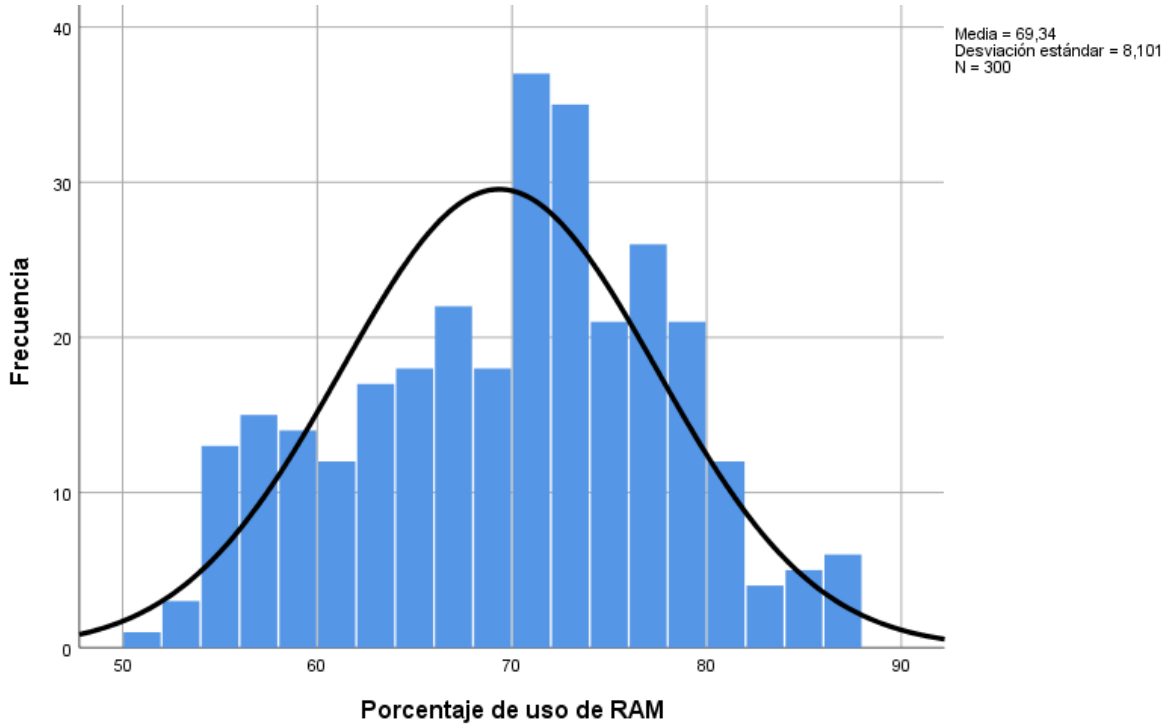
		Porcentaje de uso de RAM
N		300
Parámetros normales[a, b]	Media	69,34
	Desv. Desviación	8,101
Máximas diferencias extremas	Absoluto	,089
	Positivo	,059
	Negativo	-,089
Estadístico de prueba		,089
Sig. asintótica(bilateral)		,000[c]

a. La distribución de prueba es normal.

b. Se calcula a partir de datos.

c. Corrección de significación de Lilliefors.

Figura 7. Histograma del indicador 4



Al aplicar la prueba de normalidad en el programa IBM SPSS, nos detalla que las muestras no son normales, por ende, aplicamos pruebas no paramétricas. (wilcoxon)

Figura 8. Prueba Wilcoxon – Cuarto Indicador

Resumen de prueba de hipótesis

	Hipótesis nula	Prueba	Sig.	Decisión
1	La mediana de Porcentaje de uso de RAM es igual a 70.	Prueba de rangos con signo de Wilcoxon para una muestra	,307	Retener la hipótesis nula.

Se muestran significaciones asintóticas. El nivel de significación es de ,05.

Comprobamos el nivel de significación, si es menor que 0.05 la distribución no es normal, si es mayor que 0.05 la distribución es normal. Si P > 0.05 aceptamos la hipótesis nula y rechazamos la hipótesis alterna, por ende, la hipótesis aceptada es, El algoritmo de encriptación de imágenes basado en AES, SHA y PHASH brinda un consumo de hardware óptimo con respecto a la RAM usado.

V. DISCUSIÓN

A continuación, se analizaron los resultados adquiridos en los análisis y comparaciones realizadas con respecto a las dimensiones: Tiempo, Integridad y Recurso de Hardware.

Empezando con la hipótesis inicial propuesta, se concluyó que el promedio del porcentaje de diferencia con respecto al tiempo de encriptación de imágenes, contando con 300 imágenes como muestra, se mostró como resultado un valor promedio de 16,47%. Con dichos valores obtenidos se rechazó la hipótesis nula teniendo como conclusión un aumento en el tiempo de encriptación de imágenes en diferencia al algoritmo AES, dicho aumento se representa en un 16,47% tomando como referencia la investigación de realizada por Cevallos (2019, p. 121), quienes encontraron que tener un mayor clave implica el incremento de la seguridad debido a que la información cifrada será más robusta en cuantos a ataques de fuerza bruta. Sin embargo, calcular un mayor clave también significa más energía, tiempo y consumo de recursos, lo cual provoca una pérdida en el tiempo de encriptación. Los resultados del estudio fueron similares a las investigaciones descritos por qué, en el algoritmo propuesto se usa una nueva clave generada por el algoritmo SHA lo cual incrementa el tiempo de encriptación de imágenes. Sin embargo, los resultados fueron desiguales a las conclusiones de Li et al. (2019, p. 15) quienes encontraron que el algoritmo propuesto reduce un 30% en el tiempo de encriptación de imágenes. los resultados del presente estudio fueron diferentes a los estudios descritos por qué no se hace una manipulación de la clave secreta para encriptar la imagen.

Con respecto a la segunda hipótesis planteada, se determinó que el promedio del porcentaje de integridad de las imágenes encriptadas, contando con 300 imágenes como muestra, se mostró como resultado un valor promedio de 98,96 %. Con dichos valores obtenidos se aprueba la hipótesis nula teniendo como conclusión que el algoritmo garantiza la integridad de las imágenes desencriptadas tomando como referencia la investigación realizada por Pan et al. (2018, p. 8), quienes encontraron que para poder comprobar la integridad de las imágenes se compara la imagen original con la imagen desencriptada para observar si hubo un cambio de pixeles entre las imágenes. los resultados de este estudio fueron semejantes a los estudios descritos por qué, se usa el algoritmo PHASH para comparar la imagen original y la imagen desencriptada lo cual muestra si la imagen

ha tenido algún cambio en los valores de los pixeles. Sin embargo, los resultados fueron desiguales a las conclusiones de Kumar y Chauhan (2018, p. 65) por que se utiliza otro método de encriptación de imágenes como los mapas caóticos los cuales se utilizan histogramas para comparar la imagen original y desencriptada. Los resultados concluidos fueron desiguales a los estudios discretos debido a que utilizan algoritmos de función de hash para comprobar la integridad de la información.

Como ultima hipótesis planteada, se determinó que el promedio del porcentaje de consumo de hardware, contando con 300 imágenes como muestra, se mostró como resultado un valor promedio de 72,06 % con respecto al CPU usado y un 69,34% con respecto a la RAM usada. Con dichos valores obtenidos se admite la hipótesis nula teniendo como conclusión que el algoritmo realiza un consumo de hardware optimo, el valor se representa en un 70% tomando como referencia la investigación de realizada por Aqeel et al. (2016, p. 11263) quienes encontraron que la encriptación de imágenes tuvo un consumo de hardware mínimo según las especificaciones de los componentes a utilizar en una computadora personal. los resultados fueron parecidos a las conclusiones de la investigación descrita por qué, el algoritmo propuesto tiene un consumo del 70% de los recursos de hardware lo cual se considera óptimo para la encriptación de imágenes. Sin embargo, los resultados del estudio de Cevallos (2019) quienes encontraron que al utilizar un hardware específico para la encriptación de imágenes como los FPGA son más factibles con respecto al consumo de hardware debido a que tiene un mayor rendimiento. Los resultados fueron desiguales a las conclusiones de la investigación descrita porque en este estudio no se implementa un hardware específico para la encriptación de imágenes.

VI. CONCLUSIONES

Las conclusiones de la investigación fueron las siguientes:

1. El resultado porcentual en promedio del tiempo encriptación en diferencia al algoritmo AES teniendo como muestra 300 imágenes fue de 16,47% en aumento. Por ende, se demuestra que el algoritmo de encriptación de imágenes basado en AES, SHA y PHASH en función al tiempo de encriptación de no logra el objetivo de reducir el tiempo de encriptación de imágenes.
2. El resultado porcentual del promedio de la integridad de las imágenes desencriptadas, teniendo como muestra 300 imágenes fue de 98,96%. Por ende, se demuestra que el algoritmo de encriptación de imágenes basado en AES, SHA y PHASH en función a la integridad de las imágenes logra el objetivo de garantizar la integridad de las imágenes desencriptadas.
3. El resultado porcentual del promedio del consumo de hardware, teniendo como muestra 300 imágenes fue de 72,06 % con respecto al CPU usado y un 69,34% con respecto a la RAM usada teniendo como base un 70% de hardware consumido. Por ende, se demuestra que el algoritmo de encriptación de imágenes basado en AES, SHA y PHASH en función al consumo de recursos de hardware logro el objetivo de garantizar el algoritmo tenga un consumo de hardware óptimo.
4. Finalmente, según los resultados favorables de la investigación, se puede concluir que el algoritmo propuesto garantiza la integridad de las imágenes y el consumo de hardware óptimo dando por concluidos resultados favorables con respecto a los objetivos e hipótesis planteadas en la investigación.

.

VII. RECOMENDACIONES

Las recomendaciones para futuras investigaciones son las siguientes:

1. Ampliar y especificar la muestra con respecto a imágenes a evaluar. Por lo tanto, se podría evaluar grupos de imágenes con respecto al ancho o altura de las imágenes, para obtener resultados más precisos sobre la encriptación de imágenes.
2. Ampliar la cantidad de criterios de las comparaciones realizadas. Por lo tanto, se recomienda evaluar los criterios de complejidad del algoritmo, GPU consumido, compresión de imágenes.
3. Desarrollar investigaciones considerando nuevos algoritmos criptográficos o herramientas que ayuden a mejorar la encriptación de imágenes como por ejemplo mapas caóticos, codificación ADN, funciones HASH, FPGA.

REFERENCIAS

ALI, Raza, y ALI, Muhammad. A non-feistel symmetric key encryption algorithm. *Bahria University Journal of Information & Communication Technology*. 2017, 10(2), pp.1-7. ISBN: 1999-4974.

AQEEL Rehman, LIAO, Xiaofeng, KULSOOM, Ayesha y ULLAH, Sami. A Modified (Dual) Fusion Technique for Image Encryption Using Sha-256 Hash and Multiple Chaotic Maps. *Multimedia Tools and* Applications. 2016, 75(18), pp.11241-11266. ISBN: 1573-7721.

ASHRAF, Sara y AFTAB, Shabib. Latest transformations in scrum: A state of the art review. *International Journal of Modern Education and Computer Science.* 2017, 9(7), pp.12-22.

ASLAM, Muhammad. Introducing Kolmogorov−Smirnov Tests under Uncertainty: An Application to Radioactive Data, *ACS Omega.* 2015, 5(1), pp.914-917. ISBN: 2470-1343.

BAI, Lan. A Novel Image Cryptosystem Based On S-AES and Chaotic Map. *EDP Sciences.* 2015, *31*(2), pp.1-3.

BHUTE, Shristi y ARJARIA, Siddhartha. An Efficient AES And RC6 Based Cloud-User Data Security with Attack Detection Mechanism. *International Journal of Advanced Technology and Engineering Exploration.* 2016, *3*(21), pp. 110-117. ISBN: 2394-5443.

CEVALLOS, David. Enhanced RC5 algorithm using parallel computing for communication networks. *Ingeniería y Ciencia.* 2019, 15(29), pp.103-125. ISBN: 2256-4314.

CHAMOSO, Pablo, RIVAS, Alberto, SÁNCHEZ, Ramiro y RODRÍGUEZ, Sara. Social computing for image matching. *PLoS One*. 2018, 13(5), pp.1-23.

FEI, Xiongwei, LI, Kenli y YANG, Wangdong. A Fast-Parallel Cryptography Algorithm Based on AES-NE. *Journal of Intelligent and Fuzzy Systems.* 2016, *31*(2), pp.1099–1107.

GAN, Zhihua, CHAI, XXiuli, YUAN, Ke y LU, Yang. A Novel Image Encryption Algorithm Based on LFT Based S-Boxes and Chaos. *Multimedia Tools and Applications.* 2018, *77*(7), pp.8759-8783.

GUESMI, Ramzi, Ben, Mohamed, KACHOURI, Abdennaceur y SAMET, Mounir. Hash Key-Based Image Encryption Using Crossover Operator Y Chaos. *Multimedia Tools and Applications.* 2016, *75*(8), pp.4753-4769. ISBN: 1573-7721.

HERNÁNDEZ, Roberto, FERNÁNDEZ, Carlos y BAPTISTA, Pilar. Metodología de la investigación. 6ª ed. México, D.F.: McGraw-Hill. 2016, ISBN:978-1-4562-2396-0

HU, Liang, LI, Yan, LI, Tengfei, LI, Hongtu y CHU, Jianfeng. The Efficiency Improved Scheme for Secure Access Control of Digital Video Distribution. *Multimedia Tools and Applications*. 2016, *75*(20), pp.12645-12662. ISBN:1573-7721.

KUMAR, Tarun y CHAUHAN, Shikha. Image cryptography with matrix array symmetric key using chaos-based approach. *International Journal of Computer Network and Information Security.* 2018, 12(3), pp.60-66.

LI, Nan, SUN, Junwei y WANG, Yanfeng. A Novel Memcapacitor Model Y Its Application for Image Encryption Algorithm. *Journal of Electrical and Computer Engineering*. 2019, *19*(4): 1-16 pp. ISBN:2090-0147.

LIU, Lingfeng y MIAO, Suoxia. A New Image Encryption Algorithm Based on Logistic Chaotic Map with Varying Parameter. *SpringerPlus.* 2016, *5*(1), pp.1-12. ISBN: 2193-1801.

LIU, Xiangwen, MEEHAN, Joe, TONG, Weida, WU, Leihong, XU, Xiaowei y XU, Joshua. DLI-IT: A deep learning approach to drug label identification through image and text embedding. *BMC Medical Informatics and Decision Making*. 2020, 20, pp.1-9. ISBN:1472-6947.

MA, Junming y YE, Ruisong. An Image Encryption Scheme Based on Hybrid Orbit of Hyper-Chaotic Systems. *International Journal of Computer Network and Information Security.* 2015, *7*(5), pp.25-33. ISBN:1076-2787.

MÉNDEZ, Pablo, CISNEROS, Andrés y VILLA, Henry. Propuesta de mejora de un algoritmo criptográfico con la combinación de la esteganografía en imágenes*. Revista Cumbres.* 2017, *3* (2), pp.9-16. ISBN:1390-9541.

NISAR, Ahmed, HAFIZ, Shahzad y SALEEM, Gulshan. A Benchmark for Performance Evaluation Y Security Assessment of Image Encryption Schemes. *International Journal of Computer Network and Information Security,* 2016, *8*(12), pp.18.

OTZEN, Tamara y MANTEROLA, Carlos. Técnicas de Muestreo sobre una Población a Estudio. *International Journal of Morphology.* 2017, *35*(1), pp.227-232. ISBN:0717-9502.

PAN, Hailan, LEI, Yongmei y CHEN, Jian. Research on Digital Image Encryption Algorithm Based on Double Logistic Chaotic Map. *EURASIP Journal on Image and Video Processing.* 2018, *5(1)*, pp.1-10. ISBN:2470-1343.

PULIDO, Marta. Ceremonial and Protocol: Methods and Techniques for Scientific Research, *Opción*. 2015, 31(1), pp.1137 – 1156. ISBN:1012-1587.

RANI, Sonia y KAUR, Harpreet. Technical review on symmetric and asymmetric cryptography algorithms. *International Journal of Advanced Research in Computer Science*. 2017, 8(4), pp.182-186,. ISBN:0976-5697.

ROMERO, Manuel. Pruebas de bondad de ajuste a una distribución normal. *Revista Enfermería del Trabajo.* 2019, 6(3), pp.36-45. ISBN:2174-2510.

SANGEETA, virdi y ARPNEEK Kaur. A review on symmetric key cryptography algorithms. *International Journal of Advanced Research in Computer Science.* 2017, 8(4), pp.359-361. ISBN:0976-5697.

SINGH, Apoorva y PANDEY, Dhirendra. Implementation of requirement engineering in extreme programing and SCRUM. *International Journal of Advanced Research in Computer Science.* 2017, 8(5), pp.621-624. ISBN:0976-5697.

SONIA, Pic y GREWAL, Surender. K. Hashing Key Based Analysis of Polynomial Encryption Standard. *International Journal of Computer Network and Information Security.* 2016*, 8*(11), pp .44-51.

WANG, Xingyuan y LIU, Chuanming. A Novel Y Effective Image Encryption Algorithm Based on Chaos Y DNA Encoding. *Multimedia Tools and Applications.* 2017*,* 76(5), pp. 6229-6245. ISBN:1573-7721.

YOLFARIS, Fuertes y SEPÚLVEDA, Jorge. Scrum, Kanban y Canvas en el sector comercial, industrial y educativo - Una revisión de la literatura. *RACCIS.* 2016, 6(1), pp.46-50. ISBN: 2248-7441.

ANEXOS

Anexo 3: Matriz de operacionalización de variables

Tabla 9 *Matriz de operacionalización de las variables de la investigación*

Variable	Definición Conceptual	Definición Operacional	Dimensión	Indicador	Escala de Medición
Impacto de un algoritmo basado en AES, SHA y PHASH para la encriptación de imágenes. **(Wang y Liu, 2017, p. 6230;** **Liu y Miao, 2016, p. 11)**	La encriptación es un procedimiento que convierte la información ilegible. Resultados que se obtienen del uso de un algoritmo para la encriptación de imágenes (Sangeeta y Arpneek, 2017, p. 358; Rani, S., y Kaur, 2017, p. 182)	La variable se analiza en función al Tiempo, Integridad y Recursos para la encriptación de imágenes (Wang y Liu, 2017, p. 6241; Cevallos, 2019, p. 104; Liu y Miao, 2016, p. 11)	Tiempo encriptación imágenes. (Wang y Liu, 2017, p. 6241; Cevallos, 2019, p. 104; Liu y Miao, 2016, p. 11; Rani, S., y Kaur, 2017, p. 18)	Porcentaje de diferencia del tiempo de encriptación entre el tiempo de encriptación del algoritmo AES-SHA-PHASH con respecto al algoritmo AES. (Wang y Liu, 2017, p. 6241; Liu y Miao, 2016, p. 11)	Porcentual
			Integridad de las imágenes (Wang y Liu, 2017, p. 6232-6236; Cevallos, 2019, p. 104; Ali R., y Ali, M., 2017, p. 1; Pan, Lei y Chen, 2018, p. 8)	Porcentaje de similitud de la imagen original y la imagen desencriptada. (Wang y Liu, 2017, p. 6232-6236; Pan, Lei y Chen, 2018, p. 7)	Porcentual
			Recursos de hardware (Cevallos, 2019, p. 104 - 108; Liu, L., y Miao, S., 2016, p. 11; Aqeel-ur-Rehman, Liao, Kulsoom y Ullah, 2016, p. 11256)	Porcentaje de uso de CPU (Cevallos, 2019, p. 104 - 108; Liu, L., y Miao, S., 2016, p. 11) Porcentaje de uso de RAM (Cevallos, 2019, p. 104 - 108; Liu, L., y Miao, S., 2016, p. 11)	Porcentual

Dimensiones

Tiempo encriptación imágenes.

(Wang y Liu, 2017, p. 6241; Cevallos, 2019, p. 104; Liu y Miao, 2016, p. 11; Rani, S., y Kaur, 2017, p. 18)

Integridad de las imágenes

(Wang y Liu, 2017, p. 6232-6236; Cevallos, 2019, p. 104; Ali R., y Ali, M., 2017, p. 1; Pan, Lei y Chen, 2018, p. 8)

Recursos de hardware

(Cevallos, 2019, p. 104 - 108; Liu, L., y Miao, S., 2016, p. 11; Aqeel-ur-Rehman, Liao, Kulsoom y Ullah, 2016, p. 11256)

Indicadores

Porcentaje de diferencia del tiempo de encriptación entre el tiempo de encriptación del algoritmo AES-SHA-PHASH con respecto al algoritmo AES.

$$X= (TA2-TA1)/TA1 * 100\%$$

TA1=Tiempo del algoritmo 1

TA2= Tiempo del algoritmo 2

Tiempo de encriptación de imágenes. (Wang y Liu, 2017, p. 6232-6236; Pan, Lei y Chen, 2018, p. 7)

Porcentaje de similitud de la imagen original y la imagen desencriptada.

Similitud de imagen original y la imagen desencriptada. (Wang y Liu, 2017, p. 6232-6236; Pan, Lei y Chen, 2018, p. 7; Pan, Lei y Chen, 2018, p. 8)

Porcentaje de uso de CPU

Uso de CPU. (Cevallos, 2019, p. 104 - 108; Liu, L., y Miao, S., 2016, p. 11)

Porcentaje de uso de RAM

PR=MU * 100/TM

PR= porcentaje ram usado

MU= cantidad de memoria ram usada

TM= Total de memoria ram

Uso de RAM. (Cevallos, 2019, p. 104 - 108; Liu, L., y Miao, S., 2016, p. 11)

Anexo 4: Matriz de consistencia

Tabla 10 Matriz de consistencia

PROBLEMAS	OBJETIVOS	HIPOTESIS	VARIABLE	DIMENSIONES	INDICADORES
General	General	General			
¿Cuál será el impacto del uso de un algoritmo de encriptación de imágenes basado en AES, SHA y PHASH?	Determinar el impacto del uso de un algoritmo de encriptación de imágenes basado en AES, SHA y PHASH.	El uso de un algoritmo de encriptación de imágenes basado en AES, SHA y PHASH reduce el tiempo de encriptación de imágenes, brinda integridad de imágenes y tiene un consumo de hardware aceptable (Rani, S., y Kaur, 2017, p. 182; Bai, 2015, p.3; Wang y Liu, 2017, p. 6243)	-	-	-
Específicos	Específicos	Específicos			Indicadores
¿Cuál será el impacto del uso de un algoritmo de encriptación de imágenes basado en AES, SHA y PHASH con respecto al tiempo de encriptación de imágenes?	Determinar el impacto del uso de un algoritmo de encriptación de imágenes basado en AES, SHA y PHASH con respecto al tiempo de encriptación de imágenes.	El uso de un algoritmo de encriptación de imágenes basado en AES, SHA y PHASH reduce el tiempo de encriptación de imágenes. (Liu y Miao, 2016, p. 11; Wang y Liu, 2017, p. 6241)	Impacto de un algoritmo basado en AES, SHA y PHASH para la encriptación de imágenes. (Wang y Liu, 2017, p. 6230; Liu y Miao, 2016, p. 11)	Tiempo de encriptación imágenes (Wang y Liu, 2017, p. 6241; Cevallos, 2019, p. 104; Liu y Miao, 2016, p. 11; Rani, S., y Kaur, 2017, p. 18)	Porcentaje de diferencia entre el tiempo de encriptación del algoritmo propuesto con respecto al algoritmo AES (Wang y Liu, 2017, p. 6241; Liu y Miao, 2016, p. 11)

¿Cuál será el impacto del uso de un algoritmo de encriptación de imágenes basado en AES, SHA y PHASH respecto a la Integridad de las imágenes?	Determinar el impacto del uso de un algoritmo de encriptación de imágenes basado en AES, SHA y PHASH respecto a la integridad de las imágenes.	El uso de un algoritmo de encriptación de imágenes basado en AES, SHA y PHASH garantiza la integridad de las imágenes. (Wang y Liu, 2017, p. 6236 - 6232; Ali R., y Ali, M., 2017, p. 1;)		Integridad de las imágenes (Wang y Liu, 2017, p. 6232-6236; Cevallos, 2019, p. 104; Ali R., y Ali, M., 2017, p. 1; Pan, Lei y Chen, 2018, p. 8)	Porcentaje de similitud de la imagen original y la imagen desencriptada (Wang y Liu, 2017, p. 6232-6236; Pan, Lei y Chen, 2018, p. 7)
¿Cuál será el impacto del uso de un algoritmo de encriptación de imágenes basado en AES, SHA y PHASH respecto al consumo de recursos de hardware en la encriptación de imágenes?	Determinar el impacto del uso de un algoritmo de encriptación de imágenes basado en AES, SHA y PHASH respecto al consumo de recursos de hardware en la encriptación de imágenes.	El uso de un algoritmo de encriptación de imágenes basado en AES, SHA y PHASH brinda un consumo de hardware óptimo. (Aqeel-ur-Rehman, Liao, Kulsoom y Ullah, 2016, p. 11256; Liu, L., y Miao, S., 2016, p. 11;)		Recursos de hardware (Cevallos, 2019, p. 104 - 108; Liu, L., y Miao, S., 2016, p. 11; Aqeel-ur-Rehman, Liao, Kulsoom y Ullah, 2016, p. 11256)	Porcentaje de uso de CPU (Cevallos, 2019, p. 104 - 108; Liu, L., y Miao, S., 2016, p. 11) Porcentaje de uso de RAM (Cevallos, 2019, p. 104 - 108; Liu, L., y Miao, S., 2016, p. 11)

Anexo 5: Instrumento de recolección de datos

Tabla 11 *Ficha de recolección de datos*

Item	Width (px)	Height (px)	Porcentaje de diferencia entre el tiempo de encriptación del algoritmo AES-SHA-PHASH con respecto al algoritmo AES (segundos)	Porcentaje de similitud entre la imagen original y la imagen desencriptada (segundos)	Recursos de hardware	
					Porcentaje de uso de CPU	Porcentaje de uso de RAM
img 001.jpg	640	427	38	99	96	77
img 002.jpg	640	360	30	99	48	51
img 003.jpg	640	427	23	99	63	72
img 004.jpg	640	427	23	99	73	69
img 005.jpg	640	435	7	99	68	58
img 006.jpg	640	427	23	99	63	67
img 007.jpg	640	427	42	99	35	68
img 008.jpg	640	427	33	99	43	69
img 009.jpg	640	427	15	99	71	57
img 010.jpg	640	427	54	99	31	69
img 011.jpg	640	415	14	99	70	62
img 012.jpg	640	466	21	95	63	60
img 013.jpg	640	326	18	99	45	58
img 014.jpg	640	427	31	99	44	60
img 015.jpg	640	351	27	99	39	65
img 016.jpg	640	396	8	99	73	59
img 017.jpg	640	427	21	99	34	66
img 018.jpg	640	427	21	99	40	60
img 019.jpg	640	427	14	99	72	63
img 020.jpg	640	435	21	99	32	63
img 021.jpg	640	640	19	99	33	61

img 022.jpg	640	427	7	99	56	64
img 023.jpg	640	360	8	99	63	65
img 024.jpg	640	427	21	99	59	63
img 025.jpg	640	427	14	99	59	61
img 026.jpg	640	505	19	95	67	64
img 027.jpg	640	427	21	99	32	67
img 028.jpg	640	426	21	99	82	63
img 029.jpg	640	427	14	99	67	66
img 030.jpg	640	427	21	99	82	67
img 031.jpg	640	427	23	95	92	68
img 032.jpg	640	426	23	99	78	75
img 033.jpg	640	427	7	99	84	66
img 034.jpg	640	427	14	99	81	71
img 035.jpg	640	427	23	99	78	73
img 036.jpg	640	427	14	99	38	72
img 037.jpg	640	421	15	99	69	74
img 038.jpg	640	427	23	99	69	75
img 039.jpg	640	427	15	99	74	74
img 040.jpg	640	427	14	99	81	73
img 041.jpg	640	360	27	99	33	72
img 042.jpg	640	402	15	99	42	72
img 043.jpg	640	427	46	99	40	74
img 044.jpg	640	427	23	99	77	60
img 045.jpg	640	434	31	99	39	72
img 046.jpg	640	427	31	99	70	66
img 047.jpg	640	427	15	99	75	60
img 048.jpg	640	427	31	99	43	77

img 049.jpg	640	427	31	99	38	69
img 050.jpg	640	427	43	99	46	69
img 051.jpg	640	389	17	99	66	55
img 052.jpg	640	439	23	99	47	57
img 053.jpg	640	430	14	99	84	66
img 054.jpg	640	414	-6	99	89	63
img 055.jpg	640	399	25	99	91	63
img 056.jpg	640	427	14	99	95	63
img 057.jpg	640	405	15	99	92	60
img 058.jpg	640	427	7	99	89	60
img 059.jpg	640	427	14	99	90	58
img 060.jpg	640	427	7	99	89	61
img 061.jpg	640	397	0	99	58	53
img 062.jpg	640	427	7	99	52	53
img 063.jpg	640	427	15	99	54	54
img 064.jpg	640	422	23	99	59	54
img 065.jpg	640	360	18	99	54	56
img 066.jpg	640	287	22	99	62	57
img 067.jpg	640	270	11	99	57	59
img 068.jpg	640	423	7	99	65	53
img 069.jpg	640	427	14	99	58	56
img 070.jpg	640	427	23	99	66	54
img 071.jpg	640	427	15	99	48	56
img 072.jpg	640	360	18	99	56	54
img 073.jpg	640	426	7	99	58	55
img 074.jpg	640	426	15	99	53	57
img 075.jpg	640	426	15	99	56	55

img 076.jpg	640	425	23	99	58	57
img 077.jpg	640	427	23	99	67	56
img 078.jpg	640	427	7	99	64	54
img 079.jpg	640	426	15	99	68	56
img 080.jpg	640	427	14	99	66	54
img 081.jpg	640	480	20	99	68	58
img 082.jpg	640	427	14	99	65	54
img 083.jpg	640	427	38	99	66	55
img 084.jpg	640	480	13	99	65	55
img 085.jpg	640	432	15	99	67	59
img 086.png	640	427	15	99	69	54
img 087.jpg	640	427	23	99	71	59
img 088.jpg	640	457	14	99	70	58
img 089.jpg	640	390	8	99	77	62
img 090.jpg	640	427	23	99	74	58
img 091.jpg	640	427	7	99	70	58
img 092.jpg	640	427	23	99	69	58
img 093.jpg	640	360	18	99	73	57
img 094.jpg	640	427	15	99	71	57
img 095.jpg	640	360	8	99	80	57
img 096.jpg	640	427	23	99	74	58
img 097.jpg	640	418	14	99	78	60
img 098.jpg	640	427	14	99	88	57
img 099.jpg	640	427	15	99	81	58
img 100.jpg	640	388	17	99	91	57
img 101.jpg	1280	853	18	99	69	66
img 102.jpg	1280	853	21	99	70	64

img 103.jpg	1280	720	21	99	57	66
img 104.jpg	1280	853	19	99	41	66
img 105.jpg	1280	853	18	99	51	64
img 106.jpg	1280	855	5	99	68	63
img 107.jpg	1280	853	3	99	74	62
img 108.jpg	1280	800	-3	99	56	62
img 109.jpg	1280	853	15	99	70	64
img 110.jpg	1280	845	19	99	65	63
img 111.jpg	1280	853	11	99	67	63
img 112.jpg	1280	853	15	99	87	62
img 113.jpg	1280	960	17	99	71	66
img 114.jpg	1280	853	17	99	79	64
img 115.jpg	1280	853	17	99	73	64
img 116.jpg	1280	853	17	99	79	64
img 117.jpg	1280	853	17	99	80	65
img 118.jpg	1280	776	19	99	73	63
img 119.jpg	1280	1087	16	99	88	66
img 120.jpg	1280	853	23	99	53	64
img 121.jpg	1280	853	17	99	61	64
img 122.jpg	1280	853	19	99	62	64
img 123.jpg	1280	853	19	99	69	64
img 124.jpg	1280	719	16	99	70	63
img 125.jpg	1280	853	17	99	78	65
img 126.jpg	1280	853	20	99	78	66
img 127.jpg	1280	832	20	99	76	66
img 128.jpg	1280	853	17	99	86	66
img 129.jpg	1280	800	20	99	84	67

img 130.jpg	1280	850	15	99	84	67
img 131.jpg	1280	1280	17	99	81	74
img 132.jpg	1280	853	12	99	64	66
img 133.jpg	1280	853	9	99	67	65
img 134.jpg	1280	720	6	99	72	65
img 135.jpg	1280	854	16	99	83	74
img 136.jpg	1280	836	16	99	92	75
img 137.jpg	1280	960	25	99	93	68
img 138.jpg	1280	853	11	99	74	67
img 139.jpg	1280	1280	21	99	91	72
img 140.jpg	1280	853	24	99	81	67
img 141.jpg	1280	964	16	99	62	71
img 142.jpg	1280	1920	20	99	75	77
img 143.jpg	1280	719	23	99	84	70
img 144.jpg	1280	853	21	99	78	79
img 145.jpg	1280	855	25	99	78	70
img 146.jpg	1280	853	14	99	76	70
img 147.jpg	1280	1920	12	99	95	81
img 148.jpg	1280	1920	16	99	91	82
img 149.jpg	1280	853	21	99	83	70
img 150.jpg	1280	853	18	99	79	79
img 151.jpg	1280	852	18	99	62	76
img 152.jpg	1280	914	17	99	78	76
img 153.jpg	1280	1280	16	99	72	79
img 154.jpg	1280	720	17	99	72	76
img 155.jpg	1280	853	16	99	81	76
img 156.jpg	1280	853	24	99	75	69

img 157.jpg	1280	960	17	99	73	69
img 158.jpg	1280	868	20	99	76	69
img 159.jpg	1280	853	18	99	85	75
img 160.jpg	1280	853	16	99	84	69
img 161.jpg	1280	856	13	99	63	72
img 162.jpg	1280	853	9	99	71	74
img 163.jpg	1280	853	18	99	76	72
img 164.jpg	1280	853	18	99	81	72
img 165.jpg	1280	720	17	99	79	72
img 166.jpg	1280	853	14	99	76	70
img 167.jpg	1280	853	15	99	83	70
img 168.jpg	1280	853	18	99	77	70
img 169.jpg	1280	853	13	99	82	71
img 170.jpg	1280	853	18	99	94	69
img 171.jpg	1280	1024	15	99	89	72
img 172.jpg	1280	720	17	99	75	71
img 173.jpg	1280	852	13	99	50	60
img 174.jpg	1280	853	18	99	94	71
img 175.jpg	1280	853	8	99	79	71
img 176.jpg	1280	853	14	99	74	70
img 177.jpg	1280	853	18	99	74	70
img 178.jpg	1280	853	14	99	98	70
img 179.jpg	1280	853	13	99	82	71
img 180.jpg	1280	854	14	99	79	71
img 181.jpg	1280	720	13	99	61	70
img 182.jpg	1280	1920	16	99	74	85
img 183.jpg	1280	854	14	99	83	70

img 184.jpg	1280	853	14	99	78	71
img 185.jpg	1280	853	12	99	72	70
img 186.jpg	1280	853	18	99	72	69
img 187.jpg	1280	853	16	99	82	70
img 188.jpg	1280	852	12	99	90	71
img 189.jpg	1280	853	18	99	100	68
img 190.jpg	1280	853	18	99	78	73
img 191.jpg	1280	853	16	99	66	71
img 192.jpg	1280	853	12	99	73	72
img 193.jpg	1280	853	24	99	70	72
img 194.jpg	1280	900	14	99	71	73
img 195.jpg	1280	720	19	99	63	71
img 196.jpg	1280	720	19	99	77	71
img 197.jpg	1280	853	6	99	63	71
img 198.jpg	1280	853	16	99	75	73
img 199.jpg	1280	960	12	99	73	73
img 200.jpg	1280	853	14	99	73	72
img 201.jpg	1920	1280	10	99	96	86
img 202.jpg	1920	1280	7	99	87	81
img 203.jpg	1920	1143	6	99	89	76
img 204.jpg	1920	1280	11	99	93	82
img 205.jpg	1920	1280	10	99	93	80
img 206.jpg	1920	1281	12	99	87	84
img 207.jpg	1920	1272	11	99	27	80
img 208.jpg	1920	1115	15	99	87	75
img 209.jpg	1920	1080	18	99	88	76
img 210.jpg	1920	1280	19	99	90	74

img 211.jpg	1920	1280	16	99	63	78
img 212.jpg	1920	1282	16	99	76	84
img 213.jpg	1920	1280	3	99	92	87
img 214.jpg	1920	1280	10	99	85	86
img 215.jpg	1920	1280	14	99	90	87
img 216.jpg	1920	1280	18	99	79	76
img 217.jpg	1920	1280	11	99	81	78
img 218.jpg	1920	1278	20	99	88	74
img 219.jpg	1920	1277	20	99	94	77
img 220.jpg	1920	1280	15	99	77	77
img 221.jpg	1920	1280	10	99	62	81
img 222.jpg	1920	1280	15	99	95	79
img 223.jpg	1920	1281	20	99	40	79
img 224.jpg	1920	1080	12	99	22	86
img 225.jpg	1920	1278	18	99	79	70
img 226.jpg	1920	817	19	99	72	69
img 227.jpg	1920	1156	10	99	60	72
img 228.jpg	1920	1278	4	99	86	79
img 229.jpg	1920	1280	9	99	43	79
img 230.jpg	1920	1434	17	99	73	78
img 231.jpg	1920	1280	17	99	81	73
img 232.jpg	1920	1280	20	99	83	77
img 233.jpg	1920	1080	5	99	74	71
img 234.jpg	1920	1280	9	99	85	74
img 235.jpg	1920	1280	21	99	76	86
img 236.jpg	1920	1280	16	99	61	72
img 237.jpg	1920	1280	22	99	72	72

img 238.jpg	1920	1280	19	99	85	75
img 239.jpg	1920	1272	15	99	89	76
img 240.jpg	1920	1280	13	99	89	77
img 241.jpg	1920	1280	14	99	78	76
img 242.jpg	1920	1185	17	99	77	76
img 243.jpg	1920	1080	20	99	83	77
img 244.jpg	1920	1277	15	99	79	78
img 245.jpg	1920	1278	11	99	94	76
img 246.jpg	1920	1281	18	99	63	76
img 247.jpg	1920	1048	15	99	73	72
img 248.jpg	1920	1283	18	99	79	81
img 249.jpg	1920	1280	17	99	73	78
img 250.jpg	1920	1280	14	99	85	80
img 251.jpg	1920	1280	13	99	77	82
img 252.jpg	1920	1302	17	99	78	78
img 253.jpg	1920	1307	14	99	80	80
img 254.jpg	1920	1081	13	99	77	70
img 255.jpg	1920	1280	13	99	81	73
img 256.jpg	1920	1170	17	99	77	70
img 257.jpg	1920	1278	18	99	87	74
img 258.jpg	1920	1491	17	99	82	77
img 259.jpg	1920	1280	16	99	83	72
img 260.jpg	1920	1272	8	99	69	80
img 261.jpg	1920	1280	12	99	81	84
img 262.jpg	1920	1280	10	99	54	79
img 263.jpg	1920	1280	14	99	73	80
img 264.jpg	1920	1277	17	99	88	75

img 265.jpg	1920	1283	16	99	75	79
img 266.jpg	1920	960	18	99	68	72
img 267.jpg	1920	1281	19	99	77	75
img 268.jpg	1920	1282	14	99	82	75
img 269.jpg	1920	1274	6	99	81	78
img 270.jpg	1920	1130	8	99	81	77
img 271.jpg	1920	1280	13	99	63	73
img 272.jpg	1920	1280	16	99	72	75
img 273.jpg	1920	1280	18	99	68	79
img 274.jpg	1920	1280	16	99	75	83
img 275.jpg	1920	1280	16	99	64	77
img 276.jpg	1920	1277	13	99	81	80
img 277.jpg	1920	1258	16	99	84	78
img 278.jpg	1920	1280	18	99	72	74
img 279.jpg	1920	1280	15	99	80	79
img 280.jpg	1920	1081	16	99	81	77
img 281.jpg	1920	1015	15	99	66	72
img 282.jpg	1920	1280	16	99	83	78
img 283.jpg	1920	1278	13	99	78	73
img 284.jpg	1920	1280	14	99	82	80
img 285.jpg	1920	1280	12	99	69	73
img 286.jpg	1920	1080	18	99	65	72
img 287.jpg	1920	1280	16	99	72	73
img 288.jpg	1920	1280	14	99	76	77
img 289.jpg	1920	1280	10	99	70	77
img 290.jpg	1920	1280	18	99	80	84
img 291.jpg	1920	1277	16	99	86	78

img 292.jpg	1920	1280	16	99	78	70
img 293.jpg	1920	1080	17	99	71	71
img 294.jpg	1920	1280	17	99	73	71
img 295.jpg	1920	1280	13	99	72	71
img 296.jpg	1920	1280	17	99	82	69
img 297.jpg	1920	1280	18	99	71	72
img 298.jpg	1920	1075	19	99	69	66
img 299.jpg	1920	1280	17	99	75	71
img 300.jpg	1920	1283	16	99	61	69

Anexo 6: Comparación de algoritmo criptográficos

Tabla 12 Comparación de algoritmo criptográficos

Criterios	AES	SHA	PHASH	RC6	RSA
Velocidad de encriptación	1	2	1	1	3
Integridad de datos	2	1	1	4	1
Longitud de llaves	3	2	2	3	1
Tamaño de bloques	2	1	1	2	5
Seguridad de claves	3	1	2	3	1

Calificación	
Muy Bueno	1
bueno	2
Regular	3
Malo	4
Muy malo	5

Anexo 7: Comparación Metodologías de desarrollo

Tabla 13 *Comparación Metodologías de desarrollo*

Criterios	SCRUM	XP	Kanban
Nivel de interacción con el cliente	1	2	4
Enfocado al desarrollo del software	3	1	4
Documentación Mínima	1	1	1
Definición de los requerimientos	1	1	1
Definición de Roles	1	1	5
Incorporación de nuevas tareas durante procesos	3	3	1

Calificación	
Muy Bueno	1
bueno	2
Regular	3
Malo	4
Muy malo	5

Anexo 8: Metodología de Desarrollo

1. Introducción

Esta sección describe la adopción del marco de trabajo scrum para la implementación y desarrollo de la investigación 'Sistema web para la encriptación de imágenes basados en los algoritmos AES, SHA y PHASH'.

La metodología Scrum se caracteriza dar entregas parciales al producto final donde el mismo cliente define la prioridad de los requerimientos. Este marco de trabajo se enfoca a proyectos de corto alcance donde se requieren entregables en corto tiempo.

Propósito de este documento

Brindar información sobre el desarrollo del software 'Sistema web móvil para la encriptación de imágenes basados en los algoritmos AES, SHA y PHASH''.

Alcance

El documento especifica el plan de trabajo para el desarrollo un Sistema web móvil para la encriptación de imágenes basados en los algoritmos AES, SHA y PHASH', el cual se proyectó en desarrollo 2 meses.

2. Descripción General de la Metodología

Personas y roles del proyecto

Tabla 14 *Personas y roles del proyecto*

ID	Persona	Contacto	Rol
GRM	**Ramirez Melgarejo Gabriel Gonzalo**	gonzaloramirezmelgarejo98@gmail.com	Product Owner Scrum Master Scrum Team

Historias de Usuarios

- **HISTORIA H1: Análisis y Diseño de base de datos**

Tabla 15 *H1: Análisis y Diseño de base de datos*

Nombre	Análisis y diseño de la base de datos
ID	H02
Descripción	La base de datos se desarrollará en MongoDB se le creare un diccionario de datos con el objetivo de documentar las características de las tablas
Usuario	Programador
Importancia	Muy Alta
	• Conexión a la VDD • Modelo físico de la BDD • Modelo lógico de la BDD • Diccionario de Datos

- **HISTORIA H2: Interfaz del sistema amigable**

Tabla 16 *H2: Interfaz del sistema amigable*

Nombre	Interfaz del sistema amigable
ID	H02
Descripción	Se diseña la interfaz para poder visualizar las imágenes guardadas, reportes y los procedimientos que se le aplicara a una imagen
Usuario	Programador
Importancia	Muy Alta
	• Diseño de interfaz gráfica. • Estructurar las vistas a utilizar.

- **HISTORIA H3: Guardar imágenes**

Tabla 17 *H3: Guardar imágenes*

Nombre	Guardar imágenes
ID	H03
Descripción	registrar imágenes en los formatos jpg, png, gif, etc.
Usuario	Administrador
Importancia	Muy alta
	• El sistema debe permitir guardar imágenes. • El sistema solo debe permitir guardar imágenes en formatos multimedia.

- **HISTORIA H4: Encriptar una imagen**

Tabla 18 *H4: Encriptar una imagen*

Nombre	Encriptar una imagen
ID	H04
Descripción	encriptar una imagen.
Usuario	Administrador
Importancia	Muy Alta
	• El sistema debe permitir encriptar una imagen.

- **HISTORIA H5: Desencriptar una imagen**

Tabla 19 *H5: Desencriptar una imagen*

Nombre	Desencriptar una imagen
ID	H04
Descripción	desencriptar una imagen.
Usuario	Administrador
Importancia	Muy Alta
	• El sistema debe permitir desencriptar una imagen.

- **HISTORIA H6: Eliminar una imagen**

Tabla 20 *H6: Eliminar una imagen*

Nombre	Eliminar una imagen
ID	H06
Descripción	eliminar una imagen.
Usuario	Administrador
Importancia	Alta
	• El sistema debe permitir eliminar una imagen

- **HISTORIA H7: Reportes de las imágenes**

Tabla 21 *H7: Reportes de las imágenes*

Nombre	Reportes de las imágenes
ID	H07
Descripción	Se debe visualizar reporte con el tiempo de encriptación, porcentaje de similitud, porcentaje de uso de RAM, y porcentaje del CPU usado.
Usuario	Administrador
Importancia	Muy Alta
	• El sistema debe permitir generar reportes con respecto al tiempo encriptación, RAM, CPU y similitud de una imagen. • Se debe exportar los reportes en Exel.

Matriz de Impacto

Prioridad	Cantidad
Muy alta	4
Alta	3
Media	2
Baja	0
Muy baja	0

Pila del Producto (Product Backlog)

Tabla 22 *Product Backlog*

Id	Prioridad	Descripción	Tiempo Estimado
H01	Muy alta	Análisis y Diseño de base de datos	5 días
H02	Muy alta	Interfaz del sistema amigable	5 días
H03	Muy alta	Guardar imágenes	3 días
H04	Muy alta	Encriptar una imagen	5 días
H05	Muy alta	Desencriptar una imagen	5 días
H06	Alta	Eliminar una imagen	3 días
H07	Muy alta	Reportes de las imágenes	4 días

Sprint

Tabla 23 *Sprint01*

	HU	Actividad	Tareas	Tiempo	Responsable
SPRINT 01	H01	**RF01:** Análisis y Diseño de base de datos	1. Análisis del negocio 2. Modelo físico de la BDD. 3. Modelo lógico de la BDD. 4. Implementación de la base de datos 5. Diccionario de Datos.	5 días	GRM
	H02	**RF02:** Desarrollar la Interfaz del sistema amigable fácil de usar	1. Análisis, elaboración de páginas y componente para el sistema web.	5 días	GRM
	H03	**RF03:** Guardar imágenes	1. Ingresar o buscar imagen a guardar en el sistema. 2. Validar imágenes guardadas en formatos multimedia.	3 días	GRM
	H04	**RF04:** Encriptar una imagen	1. Desarrollar api que encripte una imagen usando los algoritmos AES, SHA y PHASH.	5 días	GRM
	H05	**RF05:** Desencriptar una imagen	1. Desarrollar api que desencripte una imagen usando los algoritmos AES, SHA y PHASH.	5 días	GRM
	H06	**RF06:** Eliminar una imagen	1. Desarrollar un api que elimine una imagen especifica.	3 días	GRM
	H07	**RF07:** Reportes de las imágenes	1. Visualizar reportes en una vista del sistema web 2. Exportar los reportes a Exel.	4 días	GRM

Ejecución del SPRINT 01

Como unos de los entregables para el primer sprint el usuario podrá interactuar, editar, insertar, eliminar y visualizar las imágenes para la encriptación correcta de imágenes en el sistema web.

RF02: Interfaz del sistema amigable

el sistema debe contar con una página principal donde se visualizan las imágenes registradas se debe poder encriptar y desencriptar en la misma página, además es necesario otra vista para visualizar reportes. En este requerimiento se especifican las páginas y secciones (componentes) que tendrá el sistema web.

Diseño de la interfaz grafica

Se puede observar el diseño y estructura de la interfaz gráfica del sistema web.

Figura 9. Vista principal del sistema.

Código de la interfaz grafica

```
<script>
  import SideBar from './_components/sidebar.svelte';
  import ImageDetail from './_components/imagedetail.svelte';
  import LoadingBar from './_components/LoadingBar.svelte'

  let img;
  const imgselected = (e)=>{
    img = e.detail;
    console.log(img);
  }
</script>

<style>···
</style>

<svelte:head>···
</svelte:head>

<LoadingBar/>

<div class="content">
  <div class="row m-0 h-100">
    <div class="col-md-3 p-0 bg-white h-100 border-right">
      <SideBar on:imgselected={imgselected}/>
    </div>
    <div class="col-md p-0 h-100 bg-white imagen-view">
      <ImageDetail img={img}/>
    </div>
  </div>
</div>
```

Figura 10. Código de la vista principal del sistema.

RF03: Guardar imágenes

el sistema web debe permitir guardar imágenes, se deben visualizar las imágenes en la pantalla principal. En este requerimiento so se permitirá guardar imágenes en formatos multimedia (JPJ y PNG).

Diseño de la interfaz grafica

Se puede observar el diseño y estructura de la interfaz gráfica para guardar imágenes en el sistema web.

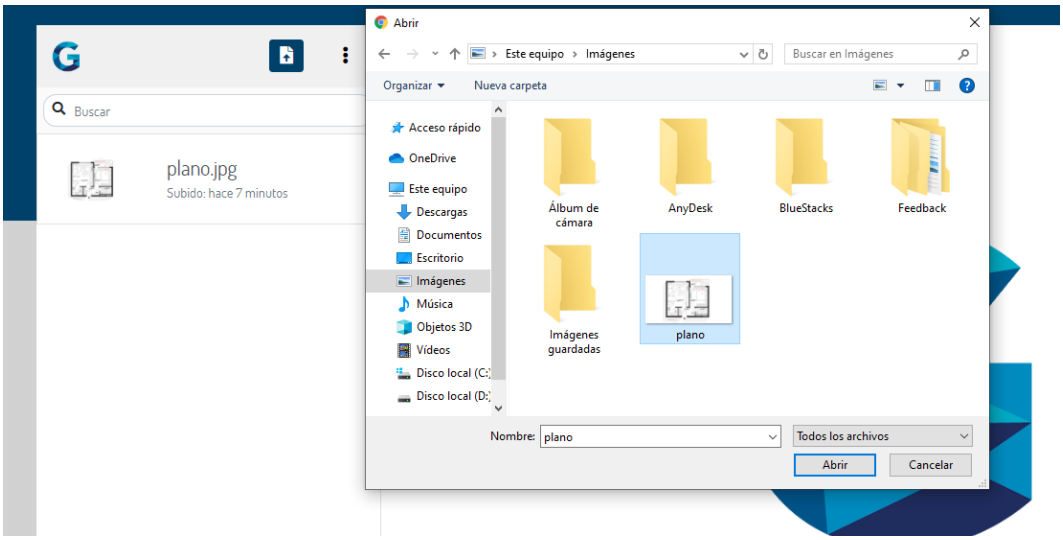


Figura 11. Vista para guardar imágenes.

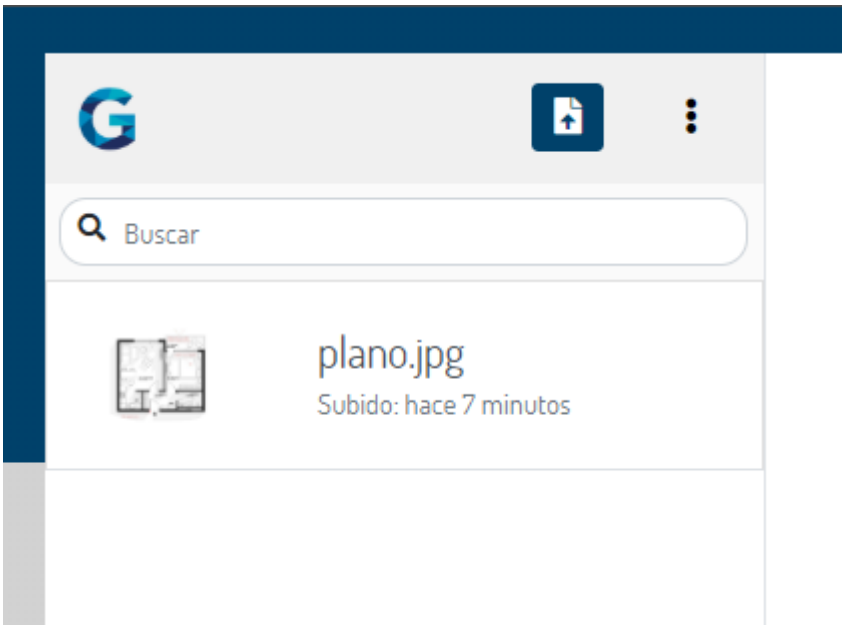


Figura 12. Vista para visualizar imágenes guardadas.

Código de la interfaz grafica

```
<nav class="navbar  navbar-expand-lg">
  <a class="navbar-brand" href="."> ···
  </a>
  <ul class="navbar-nav ml-auto">
    <li class="nav-item mr-4 d-flex align-items-center">
      <div>
        <label for="file-input" class="btn btn-file m-0 text-white">
          <i class="fas fa-file-upload icon"></i>
        </label>
        <input id="file-input" type="file" name="image" on:change={uploadimg} multiple/>
      </div>
    </li>
    <li class="nav-item dropdown p-1"> ···
    </li>
  </ul>
</nav>
```

Figura 13. Código HTML para guardar imágenes.

```
const uploadimg = async () =>{
  try {
    var files = document.getElementById("file-input").files[0];
    var data = new FormData()
    data.append('image', files)
    const response = await api.images.createImage(data)
    .then(function(data) {
      if(data.succesful){
        location.reload();
      }
    })
  } catch (e) {
    let error = e;
    console.log(error);
  }
}
```

Figura 14. Código JavaScript para guardar imágenes.

Código de la Apis utilizadas

Se muestra el código de las Apis para guardar imágenes.

```
const multer   = require('multer');
const path = require('path');
const { v4: uuidv4 } = require('uuid');

const storage = multer.diskStorage({
  destination:path.join(__dirname,'../../public/uploads'),
  filename: (req, file, cb)=>{
    cb(null, uuidv4() + path.extname(file.originalname).toLocaleLowerCase());
  }
});

exports.upload = multer({
  storage,
  dest: path.join(__dirname,'../../public/uploads'),
  fileFilter:(req, file, cb)=>{
    const filetypes = /jpeg|jpg|png|gif/;
    const mimetype = filetypes.test(file.mimetype);
    const extname = filetypes.test(path.extname(file.originalname));
    if(mimetype && extname ){
      return cb(null,true);
    }
    return cb("Error formato no valido");
  }
}).single('image')
```

Figura 15. Middleware guardar imágenes.

```
exports.upload = async (req, res, next) => {
  let data;
  try {
    const image = new Image();
    image.title = req.body.title;
    image.filename = req.file.filename;
    image.path = '/uploads/' + req.file.filename;
    image.originalname = req.file.originalname;
    image.mimetype = req.file.mimetype;
    image.size = req.file.size;
    image.hash = null;
    image.phash = null;
    await image.save();
    data = {msg:"Imagen upload",succesful:true};
  } catch (error) {
    data = {msg:"ERROR, Imagen not upload",error,succesful:false}
  }
  res.json(data)
}
```

Figura 16. Api para guardar imágenes.

RF04: Encriptar una imagen

el sistema web debe permitir encriptar imágenes, se debe seleccionar y visualizar la imagen a encriptar.

Diseño de la interfaz grafica

Se puede observar el diseño y estructura de la interfaz gráfica para encriptar imágenes en el sistema web.

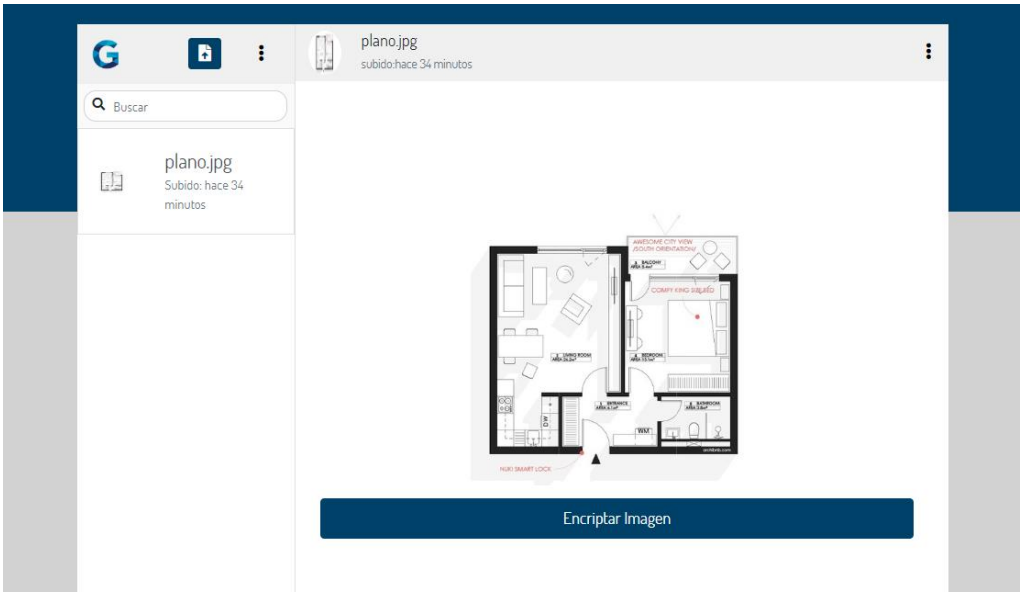


Figura 17. Vista para encriptar imágenes.

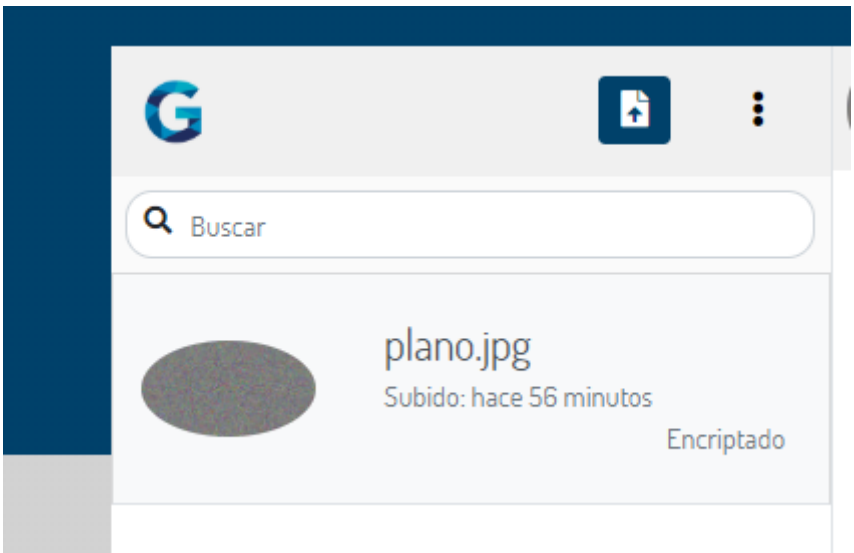


Figura 18. Vista de imágenes encriptadas.

Código de la interfaz grafica

```
<div class="row m-0 p-0 h-100 overflow-auto">
  <div class="col-md-12 h-100 d-flex align-items-center ">
    <div class="">
      <div class="row  m-0">
        <div class="col-md  p-0">
          <div class="d-flex justify-content-center align-items-center">
            <img class="" src="{host+img.path}" alt="img" width="80%">
          </div>
        </div>
      </div>
      <div class="row m-0">
        <div class="col-md p-3">
          {#if img.encrypted}
            <button type="button" class="btn bg-blue btn-lg btn-block text-white" on:click={decrypt}>Desencriptar Imagen</button>
            {:else}
            <button type="button" class="btn bg-blue btn-lg btn-block text-white" on:click={encrypt}>Encriptar Imagen</button>
          {/if}
        </div>
      </div>
    </div>
  </div>
</div>
```

Figura 19. Código HTML para encriptar imágenes.

```
const encrypt = async () =>{
  try {
    const response = await api.images.ecnryptImage(img._id)
    .then(function(data) {
      if(data.succesful){
        location.reload();
      }
    })
  } catch (e) {
    let error = e;
    console.log(error);
  }
}
```

Figura 20. Código JavaScript para encriptar imágenes.

Código de la Apis utilizadas

Se muestra el código de las Apis para encriptar imágenes.

```
exports.findId = async (req, res, next) => {
  let data;
  try {
    const {id} = req.params;
    const image = await Image.findById(id);
    data = {msg:"Imagen found",data:image,succesful:true};
    // console.log(images);
  } catch (error) {
    data = {msg:"ERROR, Imagen not found",error,succesful:false};
  }
  res.json(data)
}
```

Figura 21. Api para buscar imágenes por id.

```
exports.encrypt = async (req, res, next) => {
  const {id} = req.params;
  const img = await Image.findById(id);
  let imgpath  = path.resolve(`./src/public/${img.path}`);
  let data;
  jimp.read(imgpath)
  .then(async image => {
    var imgInfo = {width:image.bitmap.width,height:image.bitmap.height,data:[]}
    img.width = imgInfo.width;
    img.height = imgInfo.height;
    img.encrypted = true;
    await img.save();
    let start = microprofiler.start();
    image.scan(0, 0, imgInfo.width, imgInfo.height,async function(x, y, idx) {
      imgInfo.data[idx + 0] = encrytedpixel(this.bitmap.data[idx + 0]);
      imgInfo.data[idx + 1] = encrytedpixel(this.bitmap.data[idx + 1]);
      imgInfo.data[idx + 2] = encrytedpixel(this.bitmap.data[idx + 2]);
      imgInfo.data[idx + 3] = encrytedpixel(this.bitmap.data[idx + 3]);
      this.bitmap.data[idx + 0] = getrandompixel();
      this.bitmap.data[idx + 1] = getrandompixel();
      this.bitmap.data[idx + 2] = getrandompixel();
      this.bitmap.data[idx + 3] = getrandompixel();
      if (x == image.bitmap.width - 1 && y == image.bitmap.height - 1) {
        var file = fs.createWriteStream(`./src/public/encryptimg/${img._id}.txt`);
        file.on('error', function(err) {console.log(err);});
        file.write(imgInfo.data.join('|'));
        file.end();
        image.write(imgpath);
        let encryptime = microprofiler.measureFrom(start);
        const reportenc = new ReportEncrypt();
        reportenc.id_image = img._id;
        reportenc.time_encrypt = Math.round(parseInt(encryptime) / 1e+6  );
        var cpu = osu.cpu
        cpu.usage()
          .then(cpuPercentage => {···
        data = {msg:"Imagen Encrypted",encryptimg:img,succesful:true}
        res.json(data);
      }
    });
  })
  .catch(err => {
    console.log(err);
    data = {msg:"Imagen no Encrypted",err,succesful:false}
    res.json(data)
  });
  // res.json(data)
}
```

Figura 22. Api para encriptar imágenes.

```
15  const encrytedpixel = (value)=>{
16    var key = [ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16 ];
17    var iv = [ 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34,35, 36 ];
18    var text = multiple16(value);
19    var textBytes = aesjs.utils.utf8.toBytes(text);
20    var aesCbc = new aesjs.ModeOfOperation.cbc(key, iv);
21    var encryptedBytes = aesCbc.encrypt(textBytes);
22    var encryptedHex = aesjs.utils.hex.fromBytes(encryptedBytes);
23    return encryptedHex;
24  }
```

Figura 23. Código para encriptar pixeles.

```
231  const multiple16 = (value)=>{
232    let number = String(value) + 'a';
233    while ((number.length % 16) !== 0 ) {
234      number = number + 0;
235      // let random = String(Math.round(Math.random() * (99999999999999999  - 100000000) + 100000000))
236    }
237    return number;
238  }
```

Figura 24. Código para convertir valor en múltiplos de 16.

RF05: Desencriptar una imagen

el sistema web debe permitir desencriptar imágenes, se debe seleccionar y visualizar la imagen a desencriptar.

Diseño de la interfaz grafica

Se puede observar el diseño y estructura de la interfaz gráfica para desencriptar imágenes en el sistema web.

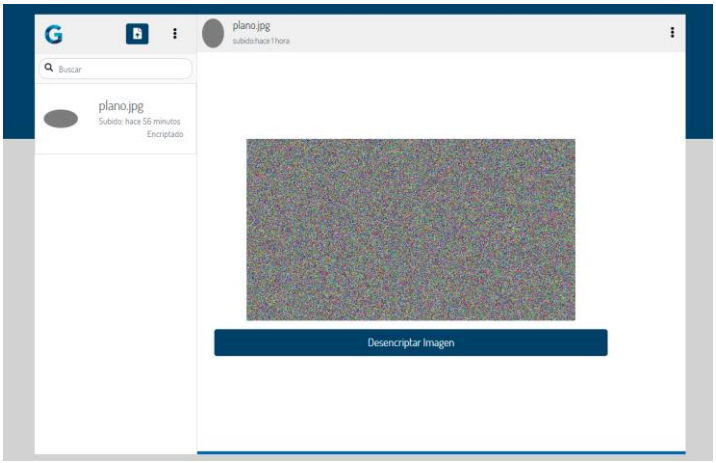


Figura 25. Vista para desencriptar imágenes.

Código de la interfaz grafica

```
<div class="row m-0 p-0 h-100 overflow-auto">
  <div class="col-md-12 h-100 d-flex align-items-center ">
    <div class="">
      <div class="row  m-0">
        <div class="col-md  p-0">
          <div class="d-flex justify-content-center align-items-center">
            <img class="" src="{host+img.path}" alt="img" width="80%">
          </div>
        </div>
      </div>
      <div class="row m-0">
        <div class="col-md p-3">
          {#if img.encrypted}
            <button type="button" class="btn bg-blue btn-lg btn-block text-white" on:click={decrypt}>Desencriptar Imagen</button>
          {:else}
            <button type="button" class="btn bg-blue btn-lg btn-block text-white" on:click={encrypt}>Encriptar Imagen</button>
          {/if}
        </div>
      </div>
    </div>
  </div>
</div>
```

Figura 26. Código HTML para desencriptar imágenes.

```
const decrypt = async () =>{
  try {
    const response = await api.images.decryptImage(img._id)
    .then(function(data) {
      if(data.succesful){
        location.reload();
      }
    })
  } catch (e) {
    let error = e;
    console.log(error);
  }
}
```

Figura 27. Código JavaScript para desencriptar imágenes.

Código de la Apis utilizadas

Se muestra el código de las Apis para desencriptar imágenes.

```
exports.decrypt = async (req, res, next) =>{
  const {id} = req.params;
  const img = await Image.findById(id);
  let data;
  let imgpath  = path.resolve(`./src/public/encryptimg/${img._id}.txt`);
  fs.readFile(imgpath, function(err, data) {
    try {
      var array = data.toString().split("|");
      let start = microprofiler.start();
      for(i in array) {
        array[i] = getvalue(decrytedpixel(array[i]));
      }
      let buffer = Buffer.from(array);
      unlink(imgpath);
      new jimp({ data: buffer, width: img.width, height: img.height }, async (err, image) => {
        try {
          img.encrypted = false;
          await img.save();
          image.write(path.resolve(`./src/public/uploads/${img.filename}`))
          let decryptime = microprofiler.measureFrom(start);
          const phash1 = image.pHash();
          const phash0 = img.phash;
          const compare = jimp.compareHashes(phash0, phash1);
          const reportdec = new ReportDecrypt();
          reportdec.id_image = img._id;
          reportdec.time_decrypt = Math.round(parseInt(decryptime) / 1e+6  );
          reportdec.integrity=null;
          if (compare == 0){
            reportdec.integrity = 99;
          }
          var cpu = osu.cpu
          cpu.usage()
            .then(cpuPercentage => {···
          data = {msg:"imagen decrypt",succesful:true};
          res.json(data);
        } catch (error) {···
        }
      });
    } catch (error) {···
    }
  });
}
```

Figura 28. Api para desencriptar imágenes.

```
const decrytedpixel = (value)=>{
  var key = [ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16 ];
  var iv = [ 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34,35, 36 ];
  var encryptedBytes = aesjs.utils.hex.toBytes(value);
  var aesCbc = new aesjs.ModeOfOperation.cbc(key, iv);
  var decryptedBytes = aesCbc.decrypt(encryptedBytes);
  var decryptedText = aesjs.utils.utf8.fromBytes(decryptedBytes);
  return decryptedText;
}
```

Figura 29. Código para desencriptar imágenes.

RF06: Eliminar una imagen

el sistema web debe permitir eliminar imágenes, se debe seleccionar y visualizar la imagen a eliminar.

Diseño de la interfaz grafica

Se puede observar el diseño y estructura de la interfaz gráfica para eliminar imágenes en el sistema web.

Figura 30. Vista para eliminar imágenes.

Código de la interfaz grafica

```
<div class="row m-0">
  <div class="col-md-12 bg-nav p-0">
    <nav class="navbar  navbar-expand-lg p-0">
      <div class="navbar-brand p-2">
        <div class="row">
          <div class="col d-flex justify-content-center p-0">
            <img src="{host+img.path}" class="rounded-circle" width="40" heigh="40" alt="img" loading="lazy">
          </div>
          <div class="col p-0">
            <h5 class="mb-0">{img.originalname}</h5>
            <span class="text-muted">subido:{format(img.createdAt,'es_ES')}</span>
          </div>
        </div>
      </div>
      <ul class="navbar-nav ml-auto">
        <li class="nav-item dropdown p-1">
          <button class="dropdown-toggle menu" type="button" id="dropdownMenuButton" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false" >
            <i class="fas fa-ellipsis-v icon "></i>
          </button>
          <div class="shadow-sm dropdown-menu  dropdown-menu-right" aria-labelledby="dropdownMenuButton">
            <!-- <a class="dropdown-item" href=".">Desactivar</a> -->
            <a class="dropdown-item" href="." on:click={deletedimg}>Eliminar</a>
          </div>
        </li>
      </ul>
    </nav>
  </div>
</div>
```

Figura 31. Código HTML para eliminar imágenes.

```
<script>
  import { format, register } from 'timeago.js';
  import * as api from 'api';

  export let img;
  let host = process.env.API_BASE_URL;
  const deletedimg = async() =>{
    try {
      const response = await api.images.deleteImage(img._id)
      .then(function(data) {
        if(data.succesful){
          location.reload();
        }
      })
    } catch (e) {
      let error = e;
      console.log(error);
    }
  }
</script>
```

Figura 32. Código JavaScript para eliminar imágenes.

Código de la Apis utilizadas

Se muestra el código de las Apis para eliminar imágenes.

```
87   exports.deleted = async (req, res, next) => {
88     let data;
89     try {
90       const {id} = req.params;
91       const image = await Image.findByIdAndDelete(id);
92       if (fs.existsSync(path.resolve(`./src/public/${image.path}`))) {
93         unlink(path.resolve(`./src/public/${image.path}`));
94         console.log('The path',image.path);
95       }
96       if (fs.existsSync(path.resolve(`./src/public/encrytimg/${image._id}.txt`))) {
97         unlink(path.resolve(`./src/public/encrytimg/${image._id}.txt`));
98         console.log('The path txt',image._id);
99       }
100      data = {msg:"Deleted imagen",succesful:true};
101    } catch (error) {
102      data = {msg:"ERROR,Imagen not deleted",succesful:false};
103    }
104    res.json(data)
105  }
106
```

Figura 33. Api para eliminar imágenes.

RF07: Reportes de las imágenes

el sistema web debe permitir visualizar reportes de las imágenes guardadas.

Diseño de la interfaz grafica

Se puede observar el diseño y estructura de la interfaz gráfica de los reportes en el sistema web.

Código de la Apis utilizadas

Se muestra el código de las Apis de los reportes.

```
exports.reportImagen = async (req, res, next) => {
  let data;
  try {
    let report = [];
    const images = await Image.find({},'_id originalname',async (err, result) => {
      for(var index in result){
        const RptEncrypt = await ReportEncrypt.find({id_image:result[index]._id},'time_encrypt porcent_cpu porcent_memory').sort({$natural:-1}).limit(1);;
        const RptDecrypt = await ReportDecrypt.find({id_image:result[index]._id},'time_decrypt integrity porcent_cpu porcent_memory:').sort({$natural:-1}).limit(1);
        let repo = {img:result[index],rptencrypt:RptEncrypt[0],rptdecrypt:RptDecrypt[0]};
        report.push(repo);
      }
      data = {msg:"Listed images",data:report,succesful:true};
      res.json(data);
    })
  } catch (error) {
    data = {msg:"ERROR, Unlisted images",error,succesful:false};
    // console.log(data);
    res.json(data);
  }
}
```

Figura 34. Api para ver reportes de las imágenes.

Anexo 9: Base de datos

```
const cryptojs = require("crypto-js");
const fs = require('fs-extra');
const path = require('path');
const jimp = require('jimp');

const {Schema, model} = require('mongoose');

const imageSchema = new Schema({
  filename:{type: String},
  path:{type: String},
  originalname:{type: String},
  mimetype:{type:String},
  size:{type:Number},
  width:{type:Number,default:0},
  height:{type:Number,default:0},
  hash:{type:String},
  phash:{type:String},
  encrypted:{type:Boolean, default:false},
  encryptedaes:{type:Boolean, default:false},
  img_state:{type:Boolean, default:true}
},{
  timestamps:true
});

imageSchema.pre('save',function(next){
  const image = this;
  if(!image.isModified('hash')){
    return next();
  }

  fs.readFile(path.resolve('./src/public/'+image.path), 'base64', function (err,data) {···
  });

})

module.exports = model('Image',imageSchema);
```

Figura 35. Modelo lógico – tabla image.

```
const {Schema, model} = require('mongoose');

const reportencrypted = new Schema({
  id_image:{type: Schema.Types.ObjectId, ref:'Image'},
  time_encrypt:{type: Number},
  porcent_cpu:{type: Number},
  porcent_memory:{type: Number},
  memomry_use:{type: Number}
},{
  timestamps:true
});

module.exports = model('Report_AesEncrypt',reportencrypted);
```

Figura 36. Modelo lógico – tabla Report_AesEncrypt.

```
const {Schema, model} = require('mongoose');
const Image = require('./Imagen');
const cryptojs = require("crypto-js");
const fs = require('fs-extra');
const path = require('path');
const jimp = require('jimp');

const reportdecrypted = new Schema({
  id_image:{type: Schema.Types.ObjectId, ref:'Image'},
  time_decrypt:{type: Number},
  porcent_cpu:{type: Number},
  porcent_memory:{type: Number},
  memomry_use:{type: Number},
  // path:{type: String},
  // phash:{type:String},
  integrity:{type:Number}
},{
  timestamps:true
});



module.exports = model('Report_AesDecrypt',reportdecrypted);
```

Figura 37. Modelo lógico – tabla Report_AesDecrypt.

```
const {Schema, model} = require('mongoose');

const reportencrypted = new Schema({
  id_image:{type: Schema.Types.ObjectId, ref:'images'},
  time_encrypt:{type: Number},
  porcent_cpu:{type: Number},
  porcent_memory:{type: Number},
  memomry_use:{type: Number}
},{
  timestamps:true
});

module.exports = model('Report_Encrypt',reportencrypted);
```

Figura 38. Modelo lógico – tabla Report_Encrypt.

```
const {Schema, model} = require('mongoose');
const Image = require('./Imagen');
const cryptojs = require("crypto-js");
const fs = require('fs-extra');
const path = require('path');
const jimp = require('jimp');

const reportdecrypted = new Schema({
  id_image:{type: Schema.Types.ObjectId, ref:'Image'},
  time_decrypt:{type: Number},
  porcent_cpu:{type: Number},
  porcent_memory:{type: Number},
  memomry_use:{type: Number},
  // path:{type: String},
  // phash:{type:String},
  integrity:{type:Number}
},{
  timestamps:true
});


module.exports = model('Report_Decrypt',reportdecrypted);
```

Figura 39. Modelo lógico – tabla Report_Decrypt.

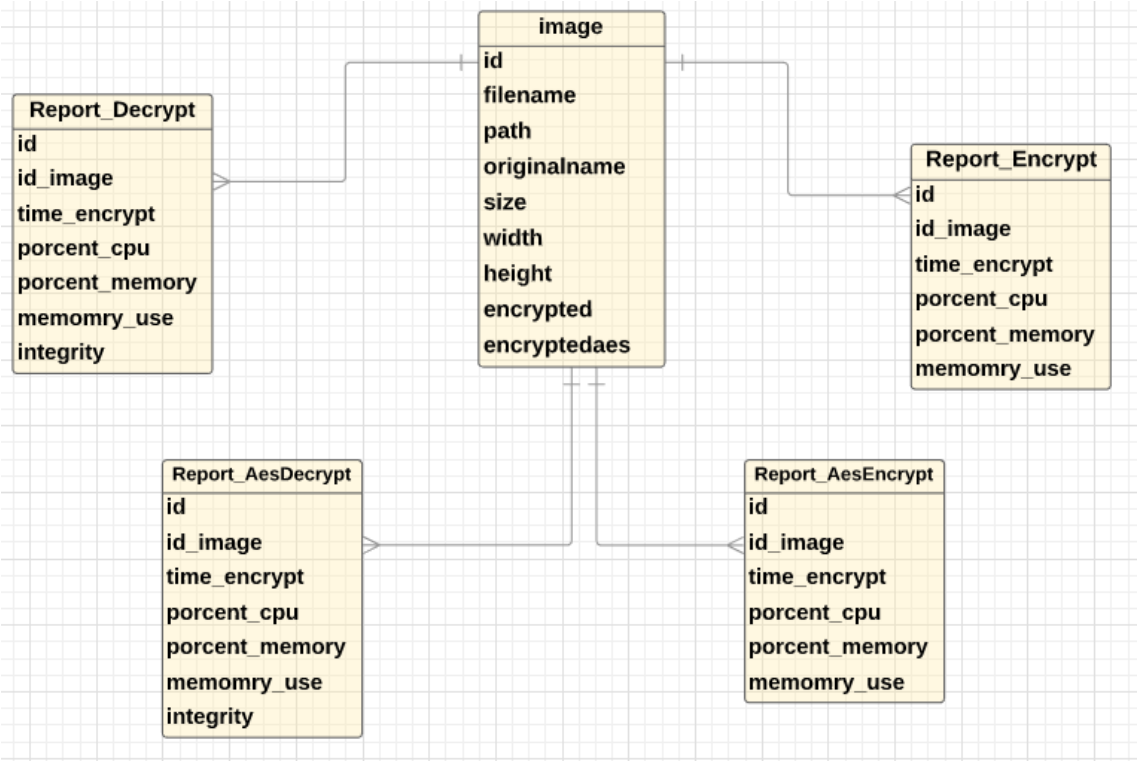


Figura 40. Modelo físico – Diseño de BDD.

Anexo 10: Algoritmo Propuesto

Pseudocódigo Algoritmo GRM (encriptar)

```
Inicio
    Leer imagen;
    Leer info; //informacion de hardware
    sha_img = SHA-256(imagen.data); //buffer de la imagen
    phash_img = PHASH (imagen.data); //buffer de la imagen
    leer inicioencrypt = hora.now(); //obtener la hora actual
    //recorrer imagen para obtener pixels
    Para (y =0; y<imagen.height; y++) //altura de la imagen
        Para (x =0; x<imagen.width; x++) //ancho de la imagen
            Idx = (imagen.width * y + x) << 2 //posición de inicio en el búfer de mapa de bits
            //obtener el valor del pixel en RGBA
            imagen.data[idx+0] = AesEncrypt(imagen.data[idx+0], sha_img); //valor de red
            imagen.data[idx+1] = AesEncrypt (imagen.data[idx+1], sha_img); //valor de green
            imagen.data[idx+2] = AesEncrypt (imagen.data[idx+2], sha_img); //valor de blue
            imagen.data[idx+3] = AesEncrypt (imagen.data[idx+3], sha_img); //valor de Alpha
            Sí (x = (imagen.widht – 1) && y = (imagen.height – 1)) //fin del recorrido de la
            imagen
                imgencrypt = imagen.data; //imagen encriptado
                guardar “imgencrypt.txt”; //guardar imagen encriptada en un archivo txt
                finencrypt = hora.now() - inicioencrypt ; //tiempo de demora del algoritmo
                cpu.usage = cpuPercentage; //porcentaje de cpu usado
                mem.usage = (info.usedMemMb * 100) //porcentaje de memoria usado
            FinSi
        FinPara
    FinPara
Fin
```

Proceso AesEncrypt (val, key) //algoritmo AES encritpar

```
Leer t = val; //(texto a cifrar) //array
Leer _ke = key; //se declara la clave
Leer rounds = _Ke.length - 1; //tamaño de la clave menos 1
Leer a = [0, 0, 0, 0]; //variable temporal array
//Etapa Inicial
//addroundkey
Para (var i = 0; i < 4; i++) {
        t[i] = t[i] ^_Ke[0][i]; //^ realiza una operación XOR a nivel binario
FinPara
//Etapa Rondas 9
// subbytes, Shiftrows, mixcolumns, addroundkey
Para (I = 1; I < rounds; i++)
        Para (var i = 0; i < 4; i++) {
            a[i] = (T1[(t[ i         ] >> 24) & 0xff] ^
            T2[(t[(i + 1) % 4] >> 16) & 0xff] ^
            T3[(t[(i + 2) % 4] >>  8) & 0xff] ^
            T4[ t[(i + 3) % 4]        & 0xff] ^
            _Ke[r][i]);
            //% obtiene el residuo de una división
            //>> realiza un desplazamiento de bits a la derecha
            //& 0xff realiza una operación AND a nivel binario con 0xff
            //^ realiza una operación XOR a nivel binario
        FinPara
t = a.copyarrays(); // copiar array a la variable t
FinPara
//Etapa Final
// subbytes, Shiftrows, mixcolumns, addroundkey
Leer result = []; //array
Leer tt;
Para (leer i = 0; i < 4; i++)
  tt = _Ke[rounds][i];
  result[4 * i    ] = (S[(t[ i         ] >> 24) & 0xff] ^ (tt >> 24)) & 0xff;
  result[4 * i + 1] = (S[(t[(i + 1) % 4] >> 16) & 0xff] ^ (tt >> 16)) & 0xff;
  result[4 * i + 2] = (S[(t[(i + 2) % 4] >>  8) & 0xff] ^ (tt >>  8)) & 0xff;
```

```
result[4 * i + 3] = (S[ t[(i + 3) % 4]        & 0xff] ^  tt       ) & 0xff;
 //se realiza un reemplazo con su entrada en una tabla de búsqueda S-box
 //% obtiene el residuo de una división
 //>> realiza un desplazamiento de bits a la derecha
 //& 0xff realiza una operación AND a nivel binario con 0xff
 //^ realiza una operación XOR a nivel binario

FINPARA
Retornar result;
```

FinProceso

```
// S-box
S = [0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5, 0x30, 0x01, 0x67, 0x2b,
0xfe, 0xd7, 0xab, 0x76, 0xca, 0x82, 0xc9, 0x7d, 0xfa, 0x59, 0x47, 0xf0, 0xad,
0xd4, 0xa2, 0xaf, 0x9c, 0xa4, 0x72, 0xc0, 0xb7, 0xfd, 0x93, 0x26, 0x36, 0x3f,
0xf7, 0xcc, 0x34, 0xa5, 0xe5, 0xf1, 0x71, 0xd8, 0x31, 0x15, 0x04, 0xc7, 0x23,
0xc3, 0x18, 0x96, 0x05, 0x9a, 0x07, 0x12, 0x80, 0xe2, 0xeb, 0x27, 0xb2, 0x75,
0x09, 0x83, 0x2c, 0x1a, 0x1b, 0x6e, 0x5a, 0xa0, 0x52, 0x3b, 0xd6, 0xb3, 0x29,
0xe3, 0x2f, 0x84, 0x53, 0xd1, 0x00, 0xed, 0x20, 0xfc, 0xb1, 0x5b, 0x6a, 0xcb,
0xbe, 0x39, 0x4a, 0x4c, 0x58, 0xcf, 0xd0, 0xef, 0xaa, 0xfb, 0x43, 0x4d, 0x33,
0x85, 0x45, 0xf9, 0x02, 0x7f, 0x50, 0x3c, 0x9f, 0xa8, 0x51, 0xa3, 0x40, 0x8f,
0x92, 0x9d, 0x38, 0xf5, 0xbc, 0xb6, 0xda, 0x21, 0x10, 0xff, 0xf3, 0xd2, 0xcd,
0x0c, 0x13, 0xec, 0x5f, 0x97, 0x44, 0x17, 0xc4, 0xa7, 0x7e, 0x3d, 0x64, 0x5d,
0x19, 0x73, 0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90, 0x88, 0x46, 0xee, 0xb8,
0x14, 0xde, 0x5e, 0x0b, 0xdb, 0xe0, 0x32, 0x3a, 0x0a, 0x49, 0x06, 0x24, 0x5c,
0xc2, 0xd3, 0xac, 0x62, 0x91, 0x95, 0xe4, 0x79, 0xe7, 0xc8, 0x37, 0x6d, 0x8d,
0xd5, 0x4e, 0xa9, 0x6c, 0x56, 0xf4, 0xea, 0x65, 0x7a, 0xae, 0x08, 0xba, 0x78,
0x25, 0x2e, 0x1c, 0xa6, 0xb4, 0xc6, 0xe8, 0xdd, 0x74, 0x1f, 0x4b, 0xbd, 0x8b,
0x8a, 0x70, 0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6, 0x0e, 0x61, 0x35, 0x57, 0xb9,
0x86, 0xc1, 0x1d, 0x9e, 0xe1, 0xf8, 0x98, 0x11, 0x69, 0xd9, 0x8e, 0x94, 0x9b,
0x1e, 0x87, 0xe9, 0xce, 0x55, 0x28, 0xdf, 0x8c, 0xa1, 0x89, 0x0d, 0xbf, 0xe6,
0x42, 0x68, 0x41, 0x99, 0x2d, 0x0f, 0xb0, 0x54, 0xbb, 0x16];
```

//Inverse S-box (S is for Substitution)

Si =[0x52, 0x09, 0x6a, 0xd5, 0x30, 0x36, 0xa5, 0x38, 0xbf, 0x40, 0xa3, 0x9e,
0x81, 0xf3, 0xd7, 0xfb, 0x7c, 0xe3, 0x39, 0x82, 0x9b, 0x2f, 0xff, 0x87, 0x34,
0x8e, 0x43, 0x44, 0xc4, 0xde, 0xe9, 0xcb, 0x54, 0x7b, 0x94, 0x32, 0xa6, 0xc2,
0x23, 0x3d, 0xee, 0x4c, 0x95, 0x0b, 0x42, 0xfa, 0xc3, 0x4e, 0x08, 0x2e, 0xa1,
0x66, 0x28, 0xd9, 0x24, 0xb2, 0x76, 0x5b, 0xa2, 0x49, 0x6d, 0x8b, 0xd1, 0x25,
0x72, 0xf8, 0xf6, 0x64, 0x86, 0x68, 0x98, 0x16, 0xd4, 0xa4, 0x5c, 0xcc, 0x5d,
0x65, 0xb6, 0x92, 0x6c, 0x70, 0x48, 0x50, 0xfd, 0xed, 0xb9, 0xda, 0x5e, 0x15,
0x46, 0x57, 0xa7, 0x8d, 0x9d, 0x84, 0x90, 0xd8, 0xab, 0x00, 0x8c, 0xbc, 0xd3,
0x0a, 0xf7, 0xe4, 0x58, 0x05, 0xb8, 0xb3, 0x45, 0x06, 0xd0, 0x2c, 0x1e, 0x8f,
0xca, 0x3f, 0x0f, 0x02, 0xc1, 0xaf, 0xbd, 0x03, 0x01, 0x13, 0x8a, 0x6b, 0x3a,
0x91, 0x11, 0x41, 0x4f, 0x67, 0xdc, 0xea, 0x97, 0xf2, 0xcf, 0xce, 0xf0, 0xb4,
0xe6, 0x73, 0x96, 0xac, 0x74, 0x22, 0xe7, 0xad, 0x35, 0x85, 0xe2, 0xf9, 0x37,
0xe8, 0x1c, 0x75, 0xdf, 0x6e, 0x47, 0xf1, 0x1a, 0x71, 0x1d, 0x29, 0xc5, 0x89,
0x6f, 0xb7, 0x62, 0x0e, 0xaa, 0x18, 0xbe, 0x1b, 0xfc, 0x56, 0x3e, 0x4b, 0xc6,
0xd2, 0x79, 0x20, 0x9a, 0xdb, 0xc0, 0xfe, 0x78, 0xcd, 0x5a, 0xf4, 0x1f, 0xdd,
0xa8, 0x33, 0x88, 0x07, 0xc7, 0x31, 0xb1, 0x12, 0x10, 0x59, 0x27, 0x80, 0xec,
0x5f, 0x60, 0x51, 0x7f, 0xa9, 0x19, 0xb5, 0x4a, 0x0d, 0x2d, 0xe5, 0x7a, 0x9f,
0x93, 0xc9, 0x9c, 0xef, 0xa0, 0xe0, 0x3b, 0x4d, 0xae, 0x2a, 0xf5, 0xb0, 0xc8,
0xeb, 0xbb, 0x3c, 0x83, 0x53, 0x99, 0x61, 0x17, 0x2b, 0x04, 0x7e, 0xba, 0x77,
0xd6, 0x26, 0xe1, 0x69, 0x14, 0x63, 0x55, 0x21, 0x0c, 0x7d];

T1 = [0xc66363a5, 0xf87c7c84, 0xee777799, 0xf67b7b8d, 0xfff2f20d,
0xd66b6bbd, 0xde6f6fb1, 0x91c5c554, 0x60303050, 0x02010103, 0xce6767a
0x562b2b7d, 0xe7fefe19, 0xb5d7d762, 0x4dababe6, 0xec76769a, 0x8fcaca45
0x1f82829d, 0x89c9c940, 0xfa7d7d87, 0xeffafa15, 0xb25959eb, 0x8e4747c9,
0xfbf0f00b, 0x41adadec, 0xb3d4d467, 0x5fa2a2fd, 0x45afafea, 0x239c9cbf,
0x53a4a4f7, 0xe4727296, 0x9bc0c05b, 0x75b7b7c2, 0xe1fdfd1c, 0x3d9393ae
0x4c26266a, 0x6c36365a, 0x7e3f3f41, 0xf5f7f702, 0x83cccc4f, 0x6834345c,
0x51a5a5f4, 0xd1e5e534, 0xf9f1f108, 0xe2717193, 0xabd8d873, 0x62313153
0x2a15153f, 0x0804040c, 0x95c7c752, 0x46232365, 0x9dc3c35e,
0x30181828, 0x379696a1, 0x0a05050f, 0x2f9a9ab5, 0x0e070709,
0x24121236, 0x1b80809b, 0xdfe2e23d, 0xcdebeb26, 0x4e272769, 0x7fb2b2c
0xea75759f, 0x1209091b, 0x1d83839e, 0x582c2c74, 0x341a1a2e,
0x361b1b2d, 0xdc6e6eb2, 0xb45a5aee, 0x5ba0a0fb, 0xa45252f6,
0x763b3b4d, 0xb7d6d661, 0x7db3b3ce, 0x5229297b, 0xdde3e33e,
0x5e2f2f71, 0x13848497, 0xa65353f5, 0xb9d1d168, 0x00000000, 0xc1eded2c
0x40202060, 0xe3fcfc1f, 0x79b1b1c8, 0xb65b5bed, 0xd46a6abe, 0x8dcbcb46
0x67bebed9, 0x7239394b, 0x944a4ade, 0x984c4cd4, 0xb05858e8, 0x85cfcf4a
0xbbd0d06b, 0xc5efef2a, 0x4faaaae5, 0xedfbfb16, 0x864343c5, 0x9a4d4dd7,
0x66333355, 0x11858594, 0x8a4545cf, 0xe9f9f910, 0x04020206, 0xfe7f7f81,
0xa05050f0, 0x783c3c44, 0x259f9fba, 0x4ba8a8e3, 0xa25151f3, 0x5da3a3fe,
0x804040c0, 0x058f8f8a, 0x3f9292ad, 0x219d9dbc, 0x70383848, 0xf1f5f504,
0x63bcbcdf, 0x77b6b6c1, 0xafdada75, 0x42212163, 0x20101030, 0xe5ffff1a,

0xfdf3f30e, 0xbfd2d26d, 0x81cdcd4c, 0x180c0c14, 0x26131335, 0xc3ecec2f, 0xbe5f5fe1, 0x359797a2, 0x884444cc, 0x2e171739, 0x93c4c457, 0x55a7a7f2, 0xfc7e7e82, 0x7a3d3d47, 0xc86464ac, 0xba5d5de7, 0x3219192b, 0xe6737395, 0xc06060a0, 0x19818198, 0x9e4f4fd1, 0xa3dcdc7f, 0x44222266, 0x542a2a7e, 0x3b9090ab, 0x0b888883, 0x8c4646ca, 0xc7eeee29, 0x6bb8b8d3, 0x2814143c, 0xa7dede79, 0xbc5e5ee2, 0x160b0b1d, 0xaddbdb76, 0xdbe0e03b, 0x64323256, 0x743a3a4e, 0x140a0a1e, 0x924949db, 0x0c06060a, 0x4824246c, 0xb85c5ce4, 0x9fc2c25d, 0xbdd3d36e, 0x43acacef, 0xc46262a6, 0x399191a8, 0x319595a4, 0xd3e4e437, 0xf279798b, 0xd5e7e732, 0x8bc8c843, 0x6e373759, 0xda6d6db7, 0x018d8d8c, 0xb1d5d564, 0x9c4e4ed2, 0x49a9a9e0, 0xd86c6cb4, 0xac5656fa, 0xf3f4f407, 0xcfeaea25, 0xca6565af, 0xf47a7a8e, 0x47aeaee9, 0x10080818, 0x6fbabad5, 0xf0787888, 0x4a25256f, 0x5c2e2e72, 0x381c1c24, 0x57a6a6f1, 0x73b4b4c7, 0x97c6c651, 0xcbe8e823, 0xa1dddd7c, 0xe874749c, 0x3e1f1f21, 0x964b4bdd, 0x61bdbddc, 0x0d8b8b86, 0x0f8a8a85, 0xe0707090, 0x7c3e3e42, 0x71b5b5c4, 0xcc6666aa, 0x904848d8, 0x06030305, 0xf7f6f601, 0x1c0e0e12, 0xc26161a3, 0x6a35355f, 0xae5757f9, 0x69b9b9d0, 0x17868691, 0x99c1c158, 0x3a1d1d27, 0x279e9eb9, 0xd9e1e138, 0xebf8f813, 0x2b9898b3, 0x22111133, 0xd26969bb, 0xa9d9d970, 0x078e8e89, 0x339494a7, 0x2d9b9bb6, 0x3c1e1e22, 0x15878792, 0xc9e9e920, 0x87cece49, 0xaa5555ff, 0x50282878, 0xa5dfdf7a, 0x038c8c8f, 0x59a1a1f8, 0x09898980, 0x1a0d0d17, 0x65bfbfda, 0xd7e6e631,

0x844242c6, 0xd06868b8, 0x824141c3, 0x299999b0, 0x5a2d2d77, 0x1e0f0f11, 0x7bb0b0cb, 0xa85454fc, 0x6dbbbbd6, 0x2c16163a];

T2 = [0xa5c66363, 0x84f87c7c, 0x99ee7777, 0x8df67b7b, 0x0dfff2f2, 0xbdd66b6b, 0xb1de6f6f, 0x5491c5c5, 0x50603030, 0x03020101, 0xa9ce6767, 0x7d562b2b, 0x19e7fefe, 0x62b5d7d7, 0xe64dabab, 0x9aec7676, 0x458fcaca, 0x9d1f8282, 0x4089c9c9, 0x87fa7d7d, 0x15effafa, 0xebb25959, 0xc98e4747, 0x0bfbf0f0, 0xec41adad, 0x67b3d4d4, 0xfd5fa2a2, 0xea45afaf, 0xbf239c9c, 0xf753a4a4, 0x96e47272, 0x5b9bc0c0, 0xc275b7b7, 0x1ce1fdfd, 0xae3d9393, 0x6a4c2626, 0x5a6c3636, 0x417e3f3f, 0x02f5f7f7, 0x4f83cccc, 0x5c683434, 0xf451a5a5, 0x34d1e5e5, 0x08f9f1f1, 0x93e27171, 0x73abd8d8, 0x53623131, 0x3f2a1515, 0x0c080404, 0x5295c7c7, 0x65462323, 0x5e9dc3c3, 0x28301818, 0xa1379696, 0x0f0a0505, 0xb52f9a9a, 0x090e0707, 0x36241212, 0x9b1b8080, 0x3ddfe2e2, 0x26cdebeb, 0x694e2727, 0xcd7fb2b2, 0x9fea7575, 0x1b120909, 0x9e1d8383, 0x74582c2c, 0x2e341a1a, 0x2d361b1b, 0xb2dc6e6e, 0xeeb45a5a, 0xfb5ba0a0, 0xf6a45252, 0x4d763b3b, 0x61b7d6d6, 0xce7db3b3, 0x7b522929, 0x3edde3e3, 0x715e2f2f, 0x97138484, 0xf5a65353, 0x68b9d1d1, 0x00000000, 0x2cc1eded, 0x60402020, 0x1fe3fcfc, 0xc879b1b1, 0xedb65b5b, 0xbed46a6a, 0x468dcbcb, 0xd967bebe, 0x4b723939, 0xde944a4a, 0xd4984c4c, 0xe8b05858, 0x4a85cfcf, 0x6bbbd0d0, 0x2ac5efef, 0xe54faaaa, 0x16edfbfb, 0xc5864343, 0xd79a4d4d, 0x55663333, 0x94118585, 0xcf8a4545, 0x10e9f9f9, 0x06040202, 0x81fe7f7f,

0xf0a05050, 0x44783c3c, 0xba259f9f, 0xe34ba8a8, 0xf3a25151, 0xfe5da3a3,
0xc0804040, 0x8a058f8f, 0xad3f9292, 0xbc219d9d, 0x48703838, 0x04f1f5f5,
0xdf63bcbc, 0xc177b6b6, 0x75afdada, 0x63422121, 0x30201010, 0x1ae5ffff,
0x0efdf3f3, 0x6dbfd2d2, 0x4c81cdcd, 0x14180c0c, 0x35261313, 0x2fc3ecec,
0xe1be5f5f, 0xa2359797, 0xcc884444, 0x392e1717, 0x5793c4c4, 0xf255a7a7,
0x82fc7e7e, 0x477a3d3d, 0xacc86464, 0xe7ba5d5d, 0x2b321919,
0x95e67373, 0xa0c06060, 0x98198181, 0xd19e4f4f, 0x7fa3dcdc, 0x66442222,
0x7e542a2a, 0xab3b9090, 0x830b8888, 0xca8c4646, 0x29c7eeee,
0xd36bb8b8, 0x3c281414, 0x79a7dede, 0xe2bc5e5e, 0x1d160b0b,
0x76addbdb, 0x3bdbe0e0, 0x56643232, 0x4e743a3a, 0x1e140a0a,
0xdb924949, 0x0a0c0606, 0x6c482424, 0xe4b85c5c, 0x5d9fc2c2,
0x6ebdd3d3, 0xef43acac, 0xa6c46262, 0xa8399191, 0xa4319595,
0x37d3e4e4, 0x8bf27979, 0x32d5e7e7, 0x438bc8c8, 0x596e3737,
0xb7da6d6d, 0x8c018d8d, 0x64b1d5d5, 0xd29c4e4e, 0xe049a9a9,
0xb4d86c6c, 0xfaac5656, 0x07f3f4f4, 0x25cfeaea, 0xafca6565, 0x8ef47a7a,
0xe947aeae, 0x18100808, 0xd56fbaba, 0x88f07878, 0x6f4a2525, 0x725c2e2e,
0x24381c1c, 0xf157a6a6, 0xc773b4b4, 0x5197c6c6, 0x23cbe8e8, 0x7ca1dddd
0x9ce87474, 0x213e1f1f, 0xdd964b4b, 0xdc61bdbd, 0x860d8b8b, 0x850f8a8a,
0x90e07070, 0x427c3e3e, 0xc471b5b5, 0xaacc6666, 0xd8904848,
0x05060303, 0x01f7f6f6, 0x121c0e0e, 0xa3c26161, 0x5f6a3535, 0xf9ae5757,
0xd069b9b9, 0x91178686, 0x5899c1c1, 0x273a1d1d, 0xb9279e9e,
0x38d9e1e1, 0x13ebf8f8, 0xb32b9898, 0x33221111, 0xbbd26969,
0x70a9d9d9, 0x89078e8e, 0xa7339494, 0xb62d9b9b, 0x223c1e1e,
0x92158787, 0x20c9e9e9, 0x4987cece, 0xffaa5555, 0x78502828, 0x7aa5dfdf,
0x8f038c8c, 0xf859a1a1, 0x80098989, 0x171a0d0d, 0xda65bfbf, 0x31d7e6e6,
0xc6844242, 0xb8d06868, 0xc3824141, 0xb0299999, 0x775a2d2d, 0x111e0f0f,
0xcb7bb0b0, 0xfca85454, 0xd66dbbbb, 0x3a2c1616];

T3 = [0x63a5c663, 0x7c84f87c, 0x7799ee77, 0x7b8df67b, 0xf20dfff2,
0x6bbdd66b, 0x6fb1de6f, 0xc55491c5, 0x30506030, 0x01030201, 0x67a9ce67,
0x2b7d562b, 0xfe19e7fe, 0xd762b5d7, 0xabe64dab, 0x769aec76, 0xca458fca,
0x829d1f82, 0xc94089c9, 0x7d87fa7d, 0xfa15effa, 0x59ebb259, 0x47c98e47,
0xf00bfbf0, 0xadec41ad, 0xd467b3d4, 0xa2fd5fa2, 0xafea45af, 0x9cbf239c,
0xa4f753a4, 0x7296e472, 0xc05b9bc0, 0xb7c275b7, 0xfd1ce1fd, 0x93ae3d93,
0x266a4c26, 0x365a6c36, 0x3f417e3f, 0xf702f5f7, 0xcc4f83cc, 0x345c6834,
0xa5f451a5, 0xe534d1e5, 0xf108f9f1, 0x7193e271, 0xd873abd8, 0x31536231,
0x153f2a15, 0x040c0804, 0xc75295c7, 0x23654623, 0xc35e9dc3,
0x18283018, 0x96a13796, 0x050f0a05, 0x9ab52f9a, 0x07090e07,
0x12362412, 0x809b1b80, 0xe23ddfe2, 0xeb26cdeb, 0x27694e27, 0xb2cd7fb2,
0x759fea75, 0x091b1209, 0x839e1d83, 0x2c74582c, 0x1a2e341a,
0x1b2d361b, 0x6eb2dc6e, 0x5aeeb45a, 0xa0fb5ba0, 0x52f6a452,
0x3b4d763b, 0xd661b7d6, 0xb3ce7db3, 0x297b5229, 0xe33edde3,
0x2f715e2f, 0x84971384, 0x53f5a653, 0xd168b9d1, 0x00000000, 0xed2cc1ed,
0x20604020, 0xfc1fe3fc, 0xb1c879b1, 0x5bedb65b, 0x6abed46a, 0xcb468dcb,
0xbed967be, 0x394b7239, 0x4ade944a, 0x4cd4984c, 0x58e8b058, 0xcf4a85cf,

0xd06bbbd0, 0xef2ac5ef, 0xaae54faa, 0xfb16edfb, 0x43c58643, 0x4dd79a4d,
0x33556633, 0x85941185, 0x45cf8a45, 0xf910e9f9, 0x02060402, 0x7f81fe7f,
0x50f0a050, 0x3c44783c, 0x9fba259f, 0xa8e34ba8, 0x51f3a251, 0xa3fe5da3,
0x40c08040, 0x8f8a058f, 0x92ad3f92, 0x9dbc219d, 0x38487038, 0xf504f1f5,
0xbcdf63bc, 0xb6c177b6, 0xda75afda, 0x21634221, 0x10302010, 0xff1ae5ff,
0xf30efdf3, 0xd26dbfd2, 0xcd4c81cd, 0x0c14180c, 0x13352613, 0xec2fc3ec,
0x5fe1be5f, 0x97a23597, 0x44cc8844, 0x17392e17, 0xc45793c4, 0xa7f255a7,
0x7e82fc7e, 0x3d477a3d, 0x64acc864, 0x5de7ba5d, 0x192b3219,
0x7395e673, 0x60a0c060, 0x81981981, 0x4fd19e4f, 0xdc7fa3dc, 0x22664422,
0x2a7e542a, 0x90ab3b90, 0x88830b88, 0x46ca8c46, 0xee29c7ee,
0xb8d36bb8, 0x143c2814, 0xde79a7de, 0x5ee2bc5e, 0x0b1d160b,
0xdb76addb, 0xe03bdbe0, 0x32566432, 0x3a4e743a, 0x0a1e140a,
0x49db9249, 0x060a0c06, 0x246c4824, 0x5ce4b85c, 0xc25d9fc2,
0xd36ebdd3, 0xacef43ac, 0x62a6c462, 0x91a83991, 0x95a43195,
0xe437d3e4, 0x798bf279, 0xe732d5e7, 0xc8438bc8, 0x37596e37,
0x6db7da6d, 0x8d8c018d, 0xd564b1d5, 0x4ed29c4e, 0xa9e049a9,
0x6cb4d86c, 0x56faac56, 0xf407f3f4, 0xea25cfea, 0x65afca65, 0x7a8ef47a,
0xaee947ae, 0x08181008, 0xbad56fba, 0x7888f078, 0x256f4a25, 0x2e725c2e,
0x1c24381c, 0xa6f157a6, 0xb4c773b4, 0xc65197c6, 0xe823cbe8, 0xdd7ca1dd,
0x749ce874, 0x1f213e1f, 0x4bdd964b, 0xbddc61bd, 0x8b860d8b, 0x8a850f8a,
0x7090e070, 0x3e427c3e, 0xb5c471b5, 0x66aacc66, 0x48d89048,
0x03050603, 0xf601f7f6, 0x0e121c0e, 0x61a3c261, 0x355f6a35, 0x57f9ae57,
0xb9d069b9, 0x86911786, 0xc15899c1, 0x1d273a1d, 0x9eb9279e,
0xe138d9e1, 0xf813ebf8, 0x98b32b98, 0x11332211, 0x69bbd269,
0xd970a9d9, 0x8e89078e, 0x94a73394, 0x9bb62d9b, 0x1e223c1e,
0x87921587, 0xe920c9e9, 0xce4987ce, 0x55ffaa55, 0x28785028, 0xdf7aa5df,
0x8c8f038c, 0xa1f859a1, 0x89800989, 0x0d171a0d, 0xbfda65bf, 0xe631d7e6,
0x42c68442, 0x68b8d068, 0x41c38241, 0x99b02999, 0x2d775a2d, 0x0f111e0f,
0xb0cb7bb0, 0x54fca854, 0xbbd66dbb, 0x163a2c16];

T4 = [0x6363a5c6, 0x7c7c84f8, 0x777799ee, 0x7b7b8df6, 0xf2f20dff,
0x6b6bbdd6, 0x6f6fb1de, 0xc5c55491, 0x30305060, 0x01010302, 0x6767a9ce,
0x2b2b7d56, 0xfefe19e7, 0xd7d762b5, 0xababe64d, 0x76769aec, 0xcaca458f,
0x82829d1f, 0xc9c94089, 0x7d7d87fa, 0xfafa15ef, 0x5959ebb2, 0x4747c98e,
0xf0f00bfb, 0xadadec41, 0xd4d467b3, 0xa2a2fd5f, 0xafafea45, 0x9c9cbf23,
0xa4a4f753, 0x727296e4, 0xc0c05b9b, 0xb7b7c275, 0xfdfd1ce1, 0x9393ae3d,
0x26266a4c, 0x36365a6c, 0x3f3f417e, 0xf7f702f5, 0xcccc4f83, 0x34345c68,
0xa5a5f451, 0xe5e534d1, 0xf1f108f9, 0x717193e2, 0xd8d873ab, 0x31315362,
0x15153f2a, 0x04040c08, 0xc7c75295, 0x23236546, 0xc3c35e9d,
0x18182830, 0x9696a137, 0x05050f0a, 0x9a9ab52f, 0x0707090e,
0x12123624, 0x80809b1b, 0xe2e23ddf, 0xebeb26cd, 0x2727694e, 0xb2b2cd7f,
0x75759fea, 0x09091b12, 0x83839e1d, 0x2c2c7458, 0x1a1a2e34,
0x1b1b2d36, 0x6e6eb2dc, 0x5a5aeeb4, 0xa0a0fb5b, 0x5252f6a4,
0x3b3b4d76, 0xd6d661b7, 0xb3b3ce7d, 0x29297b52, 0xe3e33edd,
0x2f2f715e, 0x84849713, 0x5353f5a6, 0xd1d168b9, 0x00000000, 0xeded2cc1,

0x20206040, 0xfcfc1fe3, 0xb1b1c879, 0x5b5bedb6, 0x6a6abed4, 0xcbcb468d,
0xbebed967, 0x39394b72, 0x4a4ade94, 0x4c4cd498, 0x5858e8b0, 0xcfcf4a85,
0xd0d06bbb, 0xefef2ac5, 0xaaaae54f, 0xfbfb16ed, 0x4343c586, 0x4d4dd79a,
0x33335566, 0x85859411, 0x4545cf8a, 0xf9f910e9, 0x02020604, 0x7f7f81fe,
0x5050f0a0, 0x3c3c4478, 0x9f9fba25, 0xa8a8e34b, 0x5151f3a2, 0xa3a3fe5d,
0x4040c080, 0x8f8f8a05, 0x9292ad3f, 0x9d9dbc21, 0x38384870, 0xf5f504f1,
0xbcbcdf63, 0xb6b6c177, 0xdada75af, 0x21216342, 0x10103020, 0xffff1ae5,
0xf3f30efd, 0xd2d26dbf, 0xcdcd4c81, 0x0c0c1418, 0x13133526, 0xecec2fc3,
0x5f5fe1be, 0x9797a235, 0x4444cc88, 0x1717392e, 0xc4c45793, 0xa7a7f255,
0x7e7e82fc, 0x3d3d477a, 0x6464acc8, 0x5d5de7ba, 0x19192b32,
0x737395e6, 0x6060a0c0, 0x81819819, 0x4f4fd19e, 0xdcdc7fa3, 0x22226644,
0x2a2a7e54, 0x9090ab3b, 0x8888830b, 0x4646ca8c, 0xeeee29c7,
0xb8b8d36b, 0x14143c28, 0xdede79a7, 0x5e5ee2bc, 0x0b0b1d16,
0xdbdb76ad, 0xe0e03bdb, 0x32325664, 0x3a3a4e74, 0x0a0a1e14,
0x4949db92, 0x06060a0c, 0x24246c48, 0x5c5ce4b8, 0xc2c25d9f,
0xd3d36ebd, 0xacacef43, 0x6262a6c4, 0x9191a839, 0x9595a431,
0xe4e437d3, 0x79798bf2, 0xe7e732d5, 0xc8c8438b, 0x3737596e,
0x6d6db7da, 0x8d8d8c01, 0xd5d564b1, 0x4e4ed29c, 0xa9a9e049,
0x6c6cb4d8, 0x5656faac, 0xf4f407f3, 0xeaea25cf, 0x6565afca, 0x7a7a8ef4,
0xaeaee947, 0x08081810, 0xbabad56f, 0x787888f0, 0x25256f4a, 0x2e2e725c,
0x1c1c2438, 0xa6a6f157, 0xb4b4c773, 0xc6c65197, 0xe8e823cb, 0xdddd7ca1,
0x74749ce8, 0x1f1f213e, 0x4b4bdd96, 0xbdbddc61, 0x8b8b860d, 0x8a8a850f,
0x707090e0, 0x3e3e427c, 0xb5b5c471, 0x6666aacc, 0x4848d890,
0x03030506, 0xf6f601f7, 0x0e0e121c, 0x6161a3c2, 0x35355f6a, 0x5757f9ae,
0xb9b9d069, 0x86869117, 0xc1c15899, 0x1d1d273a, 0x9e9eb927,
0xe1e138d9, 0xf8f813eb, 0x9898b32b, 0x11113322, 0x6969bbd2,
0xd9d970a9, 0x8e8e8907, 0x9494a733, 0x9b9bb62d, 0x1e1e223c,
0x87879215, 0xe9e920c9, 0xcece4987, 0x5555ffaa, 0x28287850, 0xdfdf7aa5,
0x8c8c8f03, 0xa1a1f859, 0x89898009, 0x0d0d171a, 0xbfbfda65, 0xe6e631d7,
0x4242c684, 0x6868b8d0, 0x4141c382, 0x9999b029, 0x2d2d775a, 0x0f0f111e,
0xb0b0cb7b, 0x5454fca8, 0xbbbbd66d, 0x16163a2c];

Proceso multiple16(value)

Leer number = value + ‘a’;

Mientras ((number.tamaño % 16) ! == 0)

number = number + 0;

FinMientres

Return number;

Fin

Proceso SHA-256 (imagen.data)

INICIO

Leer m= imagen.data;

Leer W = [];

//Variables

Definir hash [8]; // Valor de hash

Definir w [63];

Definir a, b, c, d, e, f, g, h; //Variables de estado.

Definir k [63]; // Vector para la constante K

Definir s0, s1;

Definir S0, S1, maj, ch; //Variables auxiliares para la actualización de variables de estado

Definir temp1, temp2;

```
Definer k ={0x428A2F98, 0x71374491, 0xB5C0FBCF, 0xE9B5DBA5,
0x3956C25B, 0x59F111F1, 0x923F82A4, 0xAB1C5ED5, 0xD807AA98,
0x12835B01, 0x243185BE, 0x550C7DC3, 0x72BE5D74, 0x80DEB1FE,
0x9BDC06A7, 0xC19BF174,0xE49B69C1, 0xEFBE4786, 0x0FC19DC6,
0x240CA1CC, 0x2DE92C6F, 0x4A7484AA, 0x5CB0A9DC, 0x76F988DA,
0x983E5152, 0xA831C66D, 0xB00327C8, 0xBF597FC7, 0xC6E00BF3,
0xD5A79147, 0x06CA6351, 0x14292967, 0x27B70A85, 0x2E1B2138,
0x4D2C6DFC, 0x53380D13, 0x650A7354, 0x766A0ABB, 0x81C2C92E,
0x92722C85, 0xA2BFE8A1, 0xA81A664B, 0xC24B8B70, 0xC76C51A3,
0xD192E819, 0xD6990624, 0xF40E3585, 0x106AA070, 0x19A4C116,
0x1E376C08, 0x2748774C, 0x34B0BCB5, 0x391C0CB3, 0x4ED8AA4A,
0x5B9CCA4F, 0x682E6FF3, 0x748F82EE, 0x78A5636F, 0x84C87814,
0x8CC70208, 0x90BEFFFA, 0xA4506CEB, 0xBEF9A3F7, 0xC67178F2}
```

// Inicialización del vector hash.

```
Definir hash [0] =0x6A09E667, hash [1] =0xBB67AE85, hash [2]
=0x3C6EF372, hash [3] =0xA54FF53A, hash [4] =0x510E527F, hash [5]
=0x9B05688C, hash [6] =0x1F83D9AB, hash [7] =0x5BE0CD19;
```

Para (i = 0; i < 64; i++)

// Computation - función de compresión:

SI (i < 16)

W[i] = M[offset + i] | 0;

SI NO

//se procesas el mensaje en fragmentos sucesivos

Leer gamma0x = W [i - 15];

Leer gamma0 = ((gamma0x << 25) | (gamma0x >>> 7)) ^ ((gamma0x << 14) | (gamma0x >>> 18)) ^ (gamma0x >>> 3);

Leer gamma1x = W [i - 2];

Leer gamma1 = ((gamma1x << 15) | (gamma1x >>> 17)) ^ ((gamma1x << 13) | (gamma1x >>> 19)) ^ (gamma1x >>> 10);

W[i] = gamma0 + W[i - 7] + gamma1 + W[i - 16];

FIN SI

//operaciones

Leer ch = (e & f) ^ (~e & g);

Leer maj = (a & b) ^ (a & c) ^ (b & c);

Leer sigma0 = ((a << 30) | (a >>> 2)) ^ ((a << 19) | (a >>> 13)) ^ ((a << 10) | (a >>> 22));

Leer sigma1 = ((e << 26) | (e >>> 6)) ^ ((e << 21) | (e >>> 11)) ^ ((e << 7) | (e >>> 25));

Leer t1 = h + sigma1 + ch + K[i] + W[i];

Leer t2 = sigma0 + maj;

h = g;

g = f;

f = e;

e = (d + t1) | 0;

d = c;

c = b;

b = a;

a = (t1 + t2) | 0; //se agregar el valor comprimido al valor actual

FINPARA

// Intermediate hash value se agregar el valor comprimido al valor actual

H[0] = (H[0] + a) | 0;

H[1] = (H[1] + b) | 0;

H[2] = (H[2] + c) | 0;

H[3] = (H[3] + d) | 0;

H[4] = (H[4] + e) | 0;

H[5] = (H[5] + f) | 0;

H[6] = (H[6] + g) | 0;

H[7] = (H[7] + h) | 0;

//valor hash final (big-endian): H0+H1+H2+H3+H4+H5+H6+H7

retornar hash = H,

FIN

Proceso PHASH (imagen)

```
Leer size = 32;

Leer smallerSize = 8;

Leer img = imagen;

Leer vals = []; //valores de los pizeles

//Calculo de los pixeles de la imagen

PARA (x = 0; x < img.width; x++)

        vals[x] = [];

        PARA (y = 0; y < img.height; y++)

                vals[x][y] = intToRGBA(img.getPixelColor(x, y)).b;

        FINPAR

FINPARA

//-------------------------------------------------------

//Calculo de la DCT 32*32 - comprension de imagenes  (transformada de coseno discreta)

Leer dctVals = applyDCT(vals, size); // Reducir la DCT.

//-------------------------------------------------------

//Calcule el valor promedio usando frecuencia 8x8

LEER total = 0;

 PARA (var _x = 0; _x < smallerSize; _x++) {

   PARA (var _y = 0; _y <smallerSize; _y++) {

     total += dctVals[_x][_y];

   FINPARA

FINPARA

 Leer avg = total / (this.smallerSize * this.smallerSize);

//-------------------------------------------------------

// Reducir aún más la DCT. Establecer hash en 0 o 1 dependiendo de si cada uno de los 64 valores está por encima o por debajo del valor medio.

Leer hash = '';

  PARA (_x2 = 0; _x2 <smallerSize; _x2++) {

    PARA (_y2 = 0; _y2 <smallerSize; _y2++) {

      hash += dctVals[_x2][_y2] > avg ? '1' : '0';
```

```
            FINPARA
          FINPARA
        //-----------------------------------------------------------
        //hash 01001001000001100101100010010111
        return hash;
FinProceso

Proceso applyDCT (f, size)
    leer N = size; //tamaño
    leer F = [];
      PARA      (u = 0; u < N; u++)
        F[u] = [];
        PARA  (v = 0; v < N; v++)
            Leer sum = 0;
            //recorrer imagen
            PARA      (i = 0; i < N; i++)
               PARA (j = 0; j < N; j++)
                     sum += coseno((2 * i + 1) / (2.0 * N) * u * PI) * coseno ((2 * j + 1) /
                     (2.0 * N) * v * PI) * f[i][j]; //pi 3.14159
               FINPARA
            FINPARA
            //-----------------------------------------------------
            sum *= c[u] * c[v] / 4; //acumulador
            F[u][v] = sum;
        FINPARA
      FINPARA
      return F;
FinProceso
```

Pseudocódigo Algoritmo GRM (desencriptar)

INICIO

```
Leer imagen = buscar imagen(id);
Leer info; //informacion de hardware
//recorrer imagen para obtener pixels
leer archivo (imagen. path);
leer arc_img = archivo.data.convertarray(); //obtener datos del archivo txt
leer iniciodecrypt = hora.now(); //obtener la hora actual
leer sha = imagen.sha;
HagaPara (re en arc_img)
    arc_img[re] = getvalue(AesDecryted(arc_img[re]));
FinHagaPara
Leer New_Imagen = convert_imagen(arc_img);
Leer Fin_decrypt = hora.now() - iniciodecrypt;  //tiempo de demora del algoritmo
Leer cpu.usage = cpuPercentage; //porcentaje de cpu usado
leer mem.usage = (info.usedMemMb * 100) //porcentaje de memoria usado
leer phash = New_Imagen.phash(); //algoritmo phash
leer img_state = (phash == imagen.phash)? true : false; //comparison de imagines
```

FIN

Proceso AesDecryted (val, key) //algoritmo aes desencryptar

```
Leer t = val; //texto a cifrar en array
Leer _Kd = key; //se declara la clave
Leer rounds = _Kd.length - 1; //tamaño de la clave menos 1
Leer a = [0, 0, 0, 0]; //variable temporal array
//Etapa Inicial
//invaddroundKey
Para (var i = 0; i < 4; i++) {
        t[i] = t[i] ^_Kd [0][i]; //se realiza una operación xor a nivel binario
FinPara
//Etapa Rondas 9
// subbytes, Shiftrows, mixcolumns, addroundkey
Para (I = 1; I < rounds; i++)
        Para (var i = 0; i < 4; i++) {
                a[i] = (T1[(t[ i         ] >> 24) & 0xff] ^
                T2[(t[(i + 3) % 4] >> 16) & 0xff] ^
                T3[(t[(i + 2) % 4] >>  8) & 0xff] ^
                T4[ t[(i + 1) % 4]        & 0xff] ^
                _Kd [r][i])];
                //% obtiene el residuo de una división
                //>> realiza un desplazamiento de bits a la derecha
                //& 0xff realiza una operación AND a nivel binario con 0xff
                //^ realiza una operación XOR a nivel binario
        FinPara
        t = a.slice(); // copiar array a la variable t
FinPara
//Etapa Final
// subbytes, Shiftrows, mixcolumns, addroundkey
Leer result = []; //array
Leer tt;
Para (leer i = 0; i < 4; i++)
  tt = _Kd[rounds][i];
  result[4 * i    ] = (Si[(t[ i         ] >> 24) & 0xff] ^ (tt >> 24)) & 0xff;
```

```
        result[4 * i + 1] = (Si[(t[(i + 3) % 4] >> 16) & 0xff] ^ (tt >> 16)) & 0xff;
        result[4 * i + 2] = (Si[(t[(i + 2) % 4] >>  8) & 0xff] ^ (tt >>  8)) & 0xff;
        result[4 * i + 3] = (Si[ t[(i + 1) % 4]        & 0xff] ^  tt  ) & 0xff;
        //se realiza un reemplazo con su entrada en una tabla de búsqueda S-boxinverse
        //% obtiene el residuo de una división
        //>> realiza un desplazamiento de bits a la derecha
        //& 0xff realiza una operación AND a nivel binario con 0xff
        //^ realiza una operación XOR a nivel binario
      FINPARA
      Retornar result;
FinProceso

Proceso getvalue(value)
      Leer index = value.buscar('a');
       let val = parseInt(value.substring(0,index));
      retornar val;
FinProceso
```