UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

Facultad de Ingeniería, Arquitectura y Diseño

Maestría y Doctorado en Ciencias e Ingeniería

DESARROLLO DE UN SISTEMA CRIPTOGRÁFICO MULTIPLATAFORMA USANDO EL MAPA CAÓTICO DE HÉNON Y EL PROTOCOLO MQTT PARA TRANSMISIÓN SEGURA DE IMÁGENES SOBRE INTERNET

TESIS

Que para cubrir parcialmente los requisitos necesarios para obtener el grado de

MAESTRO EN INGENIERÍA

presenta:

Alejandro Vargas Villavicencio

Director de tesis

Dr. Everardo Inzunza González

Ensenada, Baja California, México, octubre de 2021

UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

Facultad de Ingeniería, Arquitectura y Diseño Maestría y
Doctorado en Ciencias e Ingeniería

DESARROLLO DE UN SISTEMA CRIPTOGRÁFICO MULTIPLATAFORMA USANDO EL MAPA CAÓTICO DE HÉNON Y EL PROTOCOLO MQTT PARA TRANSMISIÓN SEGURA DE IMÁGENES SOBRE INTERNET

TESIS

Que para obtener el título de

MAESTRO EN INGENIERÍA

Presenta:

ALEJANDRO VARGAS VILLAVICENCIO

y aprobada por el siguiente comité:

Dr. Everardo Inzunza González
Director de tesis

Dr. Enrique Efrén García Guerrero
Co-Director de tesis

Dr. Oscar Roberto López Bonilla
Miembro del comité

M.C. Sergio Omar Infante Prieto
Miembro del comité

Dr. Jesús Everardo Olguín Tiznado
Miembro del comité

RESUMEN de la tesis de **Alejandro Vargas Villavicencio**, presentada como requisito para obtener el grado de MAESTRO EN INGENIERÍA, del programa de Maestría y Doctorado en Ciencias e Ingeniería de la UABC. Ensenada, Baja California, México, junio 2021.

DESARROLLO DE UN SISTEMA CRIPTOGRÁFICO MULTIPLATAFORMA USANDO EL MAPA CAÓTICO DE HÉNON Y EL PROTOCOLO MQTT PARA TRANSMISIÓN SEGURA DE IMÁGENES SOBRE INTERNET

Resumen aprobado por:

Dr. Everardo Inzunza González
Director de Tesis

Dr. Enrique Efrén García Guerrero
Co-Director de Tesis

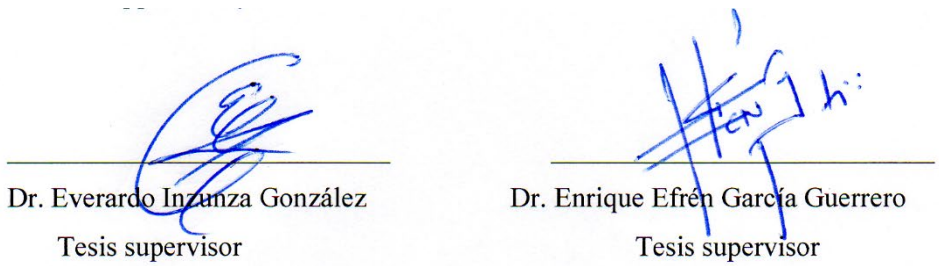
En este trabajo de tesis de maestría, se presenta el desarrollo e implementación de un prototipo encriptador-desencriptador caótico de imágenes digitales basado en Raspberry Pi con comunicación WiFi empleando el protocolo de comunicación MQTT (Message Queue Telemetry Transport). El sistema criptográfico propuesto presenta varias ventajas de innovación tecnológica para aplicaciones en comunicaciones seguras: i) Permite la elección de clave de cifrado automática o manual, ii) el sistema criptográfico propuesto ejecuta el proceso de encriptado-desencriptado en el sistema embebido, iii) el usuario final interactúa con el dispositivo encriptador-desencriptador a través de una interfaz gráfica amigable que permite el envío de imágenes digitales en tiempo real sobre Internet y iv) el sistema propuesto permite evaluar la seguridad de los criptogramas mediante ataques diferenciales NPCR (Number of Changing Pixel Rate) y UACI (Unified Averaged Changed Intensity) e histogramas estadísticos. Para verificar la viabilidad y seguridad del sistema criptográfico propuesto, se realizan análisis de seguridad como histogramas estadísticos, entropía de la información, correlación de pixeles adyacentes, espacio de claves y ataques diferenciales como NPCR y UAC. Los resultados de análisis de seguridad muestran que el sistema criptográfico propuesto, es robusto contra diferentes tipos de ataques. Finalmente se presenta un análisis comparativo del rendimiento entre el sistema embebido y una computadora personal.

Palabras clave: Encriptamiento de imágenes, Desencriptado, WiFi, MQTT, Raspberry Pi, Caos.

ABSTRACT of the thesis of **Alejandro Vargas Villavicencio**, presented as a requirement to obtain the degree of MASTER IN ENGINEERING, of the program of MSc and PhD in Sciences and Engineering of the UABC. Ensenada, Baja California, México, June 2021.

DEVELOPMENT OF A MULTIPLATFORM CRYPTOSYSTEM BY USING THE HENON CHAOTIC MAP AND THE MQTT PROTOCOL FOR SECURE TRANSMISSION OF IMAGES OVER THE INTERNET

Abstract approved by:



Dr. Everardo Inzunza González
Tesis supervisor

Dr. Enrique Efrén García Guerrero
Tesis supervisor

In this master's thesis, the development and implementation of a chaotic encryption-decryption prototype of digital images based on Raspberry Pi with WiFi communication through the MQTT (Message Queue Telemetry Transport) communication protocol. The implemented cryptosystem offers several advantages of technological innovation for secure communication applications. i) It allows to choose automatic or manual encryption key, ii) the proposed cryptosystem executes the encryption-decryption process on the embedded system, iii) the graphic user interface interacts with the cryptosystem through a friendly interface that allows sending digital images in real-time over the Internet, iv) the proposed cryptosystem allows to evaluate the security of cryptograms using differential attacks such as NPCR (number of changing pixel rate) and UACI (unified averaged changed intensity)and statistical histograms. Security analysis such as information entropy, correlation of adjacent pixels, and key space is performed to verify the viability and security of the proposed cryptosystem. The security analysis results show that the proposed cryptosystem is robust against different types of attacks, such as statistical histograms, information entropy, key space, correlation of adjacent pixels, and differential attacks such as NPCR and UAC. Finally, a comparative analysis of the performance between the embedded system and a personal computer is presented.

Keywords: Image encryption, Decryption, WiFi, MQTT, Raspberry Pi, Chaos.

Dedicatoria a mi familia:

Gracias a mis padres por darme la oportunidad de superarme día a día, por apoyarme y aconsejarme siempre en los momentos de incertidumbre, por dar todo lo que tienen para que yo triunfe en esta vida.

Gracias a mis hermanos por siempre estar ahí cuando los necesito, por su apoyo y críticas constructivas que me ayudan a ser mejor persona.

Gracias padre, madre y hermanos.

Agradecimientos

Al profesor **Dr. Everardo Inzunza González** por su excelente trabajo como director de tesis el gran apoyo durante el programa de maestría, que estuvo siempre ahí para ayudar y auxiliar en todos los obstáculos que se aparecieron por el camino.

Al profesor **Dr. Enrique Efrén García Guerrero** por su excelente trabajo y su apoyo durante el programa de maestría como mi codirector de tesis.

Al profesor **Dr. Oscar Roberto López Bonilla** quien me ayudo y me enseño como hablar en público sin miedo y corregir los errores de dicción como las muletillas, por su gran conocimiento en el área de electrónica y su apoyo en mi carrera profesional.

Al **Dr. Jesús Everardo Olguín Tiznado** y **M.C. Sergio Omar Infante Prieto** por aceptar ser miembros de mi comité de tesis y sus valiosos comentarios y sugerencias que ayudaron a clarificar el manuscrito de tesis.

A mi **familia** y **amigos** por todos los buenos y malos momentos que hacen la vida.

A **Marlha Doarair Ramírez Magallanes** una excelente e importante mujer para mí, la cual me ha acompañado desde antes de mis estudios de posgrado, quien me ha apoyado incondicionalmente y me inspira para seguir siendo mejor persona tanto en lo personal como profesionalmente.

Al **Consejo Nacional de Ciencia y Tecnología** por permitirme ser partícipe de su beca de manutención durante mis estudios de maestría.

A la **Universidad Autónoma de Baja California** por haberme aceptado y depositado su confianza en mí para realizar una maestría en sus instalaciones.

A la **Facultad de Ingeniería, Arquitectura y Diseño**, por aceptarme como estudiante de posgrado y por su apoyo durante mis estudios de maestría en ingeniería.

Tabla de contenido

Lista de Figuras

Lista de Tablas

Capítulo I

1. Introducción

En la actualidad existe una necesidad de tener confidencialidad y secrecía en cualquier entorno de lo profesional, académico e intelectual. De esta forma las telecomunicaciones, gobiernos, corporaciones e individuos difícilmente podrían funcionar, y es que, los sistemas criptográficos convencionales en las redes de telecomunicaciones recientes permanecen basados preferentemente en el encriptamiento con llave pública, los cuales, en la actualidad dejan de ser incondicionalmente seguros. Estos sistemas son potencialmente inseguros debido a los adelantos tecnológicos en capacidad computacional, dando como resultado que dichos sistemas sean vulnerables independientemente de la complejidad del algoritmo de encriptación empleado. Con el constante desarrollo del Internet de las cosas, ya se encuentran enlazados más de 50,000 millones de dispositivos con tecnología de comunicación digital en sistemas embebidos [1], [2] La tecnología está en constante evolución y con el progreso en las capacidades operacionales de cómputo, siguen emergiendo amenazas y vulnerabilidades que comprometen la estabilidad de la información de todos los dispositivos de telecomunicaciones, de la misma forma que la telefonía IP [3]. Bajo esta visión, la estabilidad criptográfica forma parte integral e indivisible de dichos procesos evolutivos, por lo que, se requiere el desarrollo constante de nuevos procedimientos y técnicas de protección de información [4], [5]. Cada vez se procesan mayores cantidades de información por lo cual, con el paso del tiempo es necesario aumentar los niveles de estabilidad y rapidez de ejecución de los sistemas criptográficos implementados preferentemente en sistemas embebidos.

En este trabajo se implementa el uso de un sistema embebido para explorar la solidez del sistema criptográfico desarrollado, debido a que los sistemas actuales demandan una mayor rapidez de procesamiento y menor consumo de energía, entre otros. Se diseña un sistema encriptador-desencriptador basado en el modelo caótico de Hénon y en el protocolo MQTT para transmisión segura de imágenes digitales. El objetivo de este trabajo es presentar una alternativa viable de seguridad para el encriptado-desencriptado de información basado en un sistema embebido de última generación que incremente los niveles de seguridad en aplicaciones del Internet de las cosas (IoT). Para este fin, se desarrolla un generador de series pseudo-aleatorias de *bits* para garantizar la eficiencia en

consumo de energía y tiempo de procesamiento. Se estudia y evalúa la aleatoriedad que presentan los modelos caóticos empleados y se estima su potencialidad a partir de un análisis de seguridad, tiempo de procesamiento y consumo de energía. Adicionalmente, se desarrolla e implementa un algoritmo de lógica operacional simple y eficiente que da soporte a la implementación y que permite alcanzar buenos niveles de estabilidad en los procesos de encriptado y desencriptado de información confidencial, tal como señales de audio e imágenes.

1.1 Antecedentes

En la literatura se reportan algunos métodos que proponen soluciones al problema de la precisión numérica con aritmética de punto flotante, el propósito es el de disminuir la degradación digital del caos [6] con base en tres vertientes principales:

i) Implementación de modelos caóticos cuyas dinámicas son muy complejas, o de múltiples enrollamientos [7]–[9].

ii) Empleo de más de un modelo caótico en cascada [4], [10].

iii) Incremento en la precisión numérica utilizando supercomputadoras o sistemas embebidos basados en FPGA (Field Programmable Gate Array) o SoC (System on a Chip) [11], [12].

En todos los casos, la implementación depende directamente de los sistemas integrales de desarrollo, de los sistemas de compilación, o de los lenguajes exclusivos de programación que requiere el dispositivo o sistema digital empleado. Tal es el caso, por ejemplo, de los sistemas embebidos basados en microcontrolador, en donde se requiere de compiladores de programas de acuerdo al fabricante, o de lenguajes de descripción de *hardware* complejos como VHDL (Very High Speed Integrated Circuits) [13] para el diseño en sistemas embebidos basados en FPGA (matriz de puertas programables).

Recientemente se reportan algunos trabajos que emplean cómputo paralelo para el encriptado de las imágenes digitales, ver por ejemplo [14]–[21]. Los cuales mejoran los tiempos de procesamiento de la información.

1.2 Planteamiento del problema

La transmisión de información por medios privados de manera segura cada día es más cuestionable y más aún por canales de comunicación públicos, una forma de proteger dicha información es mediante el uso de la criptografía.

En la actualidad el mundo es testigo de la creciente demanda de servicios en telecomunicaciones. Esta demanda se soporta por un rápido incremento en la transmisión y capacidad de conmutación en redes de todo tipo. Los requerimientos más grandes en esas redes son la privacidad y la seguridad. En nuestros días se emplea software convencional para codificación, el cual hasta el momento ha mostrado cierta eficacia para codificar información. No obstante, los continuos avances tecnológicos en el desarrollo de computadoras personales cada vez de mayores capacidades amenazan la seguridad de estas técnicas de encriptado. Una alternativa para abordar y proponer una solución a este problema, es el empleo de un proceso de encriptamiento adicional en los niveles físicos, utilizando para este fin, portadoras caóticas generadas por sistemas embebidos de alto rendimiento computacional, tal como Raspberry Pi 2. En años recientes, se ha probado tanto teórica como experimental que utilizando la teoría de caos se incrementa significativamente los niveles de seguridad en las comunicaciones.

Muchos de los esquemas de encriptado de imágenes digitales propuestos están diseñados empleando mapeos caóticos con arquitectura de permutación-difusión. Si bien la mayoría de estos esquemas reportan buenas propiedades estadísticas, su velocidad de ejecución es lenta debido a la dependencia inherente de los datos de los esquemas propuestos. Algunos de estos esquemas están diseñados utilizando sistemas caóticos complejos que requieren recursos computacionales significativos para obtener el flujo de claves para el encriptado de la información, por lo que, se requiere de computadoras personales de alta capacidad, las cuales consumen mayor energía y requieren más espacio. Por su parte, existen problemas abiertos de investigación científica en el ámbito de la seguridad de información confidencial con dispositivos embebidos utilizados en aplicaciones de Internet de las Cosas e Industria 4.0, particularmente en telefonía IP, transmisión de imágenes digitales y video en streaming, entre otros.

1.3 Propuesta de solución

Se propone el desarrollo de un sistema criptográfico implementado en un sistema embebido de alto rendimiento computacional, operando como cifrador o descifrador de imágenes digitales empleando el modelo caótico de Hénon y el protocolo MQTT, diseñado para aplicaciones de Internet de las cosas. Dentro de las ventajas que proporcionan los sistemas embebidos están sus dimensiones físicas, bajo consumo de energía, peso ligero, confiabilidad y flexibilidad de integrarse a múltiples aplicaciones, entre otras.

El sistema propuesto debe cumplir con el requerimiento básico de todo sistema criptográfico basado en caos [21], con la prueba estadística NIST SP 800-22 ver Tabla 5 diseñada para módulos criptográficos y con otros ataques bien conocidos en la literatura (diferenciales NPCR, UACI, correlación de pixeles adyacentes, entropía de la información, espacio de claves, calidad del encriptado, entre otros) comúnmente empleados para cuantificar la robustez de los sistemas criptográficos.

1.4 Objetivos

Objetivo general:

El objetivo principal de este trabajo, es diseñar un prototipo encriptador-desencriptador de imágenes digitales para transmisión segura sobre Internet.

Objetivos específicos:

1.- Implementar el protocolo MQTT en el sistema embebido Raspberry Pi 2.

2.- Estudiar algoritmos de encriptado caótico eficientes para implementarse en sistemas embebidos de baja potencia.

3.- Implementar algoritmos de encriptado caótico en nuevas plataformas.

4.- Evaluar el nivel de seguridad de la información encriptada.

5.- Evaluar el rendimiento del prototipo final.

6.- Desarrollar una interface amigable para el usuario en Python.

Capítulo II

2. Marco Teórico

2.1 Historia de la criptografía

La necesidad de poder transmitir un mensaje preservando la confidencialidad, motivó técnicas para ocultar información ya hace miles de años; frente a las diferentes formas de transmitir información, los gobiernos y los ejércitos empleaban distintas técnicas de comunicación vulnerables a ser descubiertos los mensajes ocultos. Los primeros métodos de codificación que se emplearon para asegurar la transmisión de mensajes fueron por medio de la estenografía, término que nace de los vocablos griegos steganos y graphein, que significan respectivamente "encubierto" y "redactar" [22], que entendemos actualmente como encubrimiento u ocultación de información. De los relatos más antiguos de los que se tiene registro y en los que se emplea el encubrimiento de información, es el de Heródoto. En Las Historias narra que Demarato, envió un mensaje a los espartanos para advertir de los peligros que representaba el plan de invasión del monarca persa Jerjes; la técnica que empleo Demarato fue la de ocultar con cera entablillas, donde estaba inscrito el mensaje. Otro relato del que se tiene memoria es el de Histaiaeo, que envía información a Aristágoras de Mileto, lo hace afeitando la cabeza de su mensajero; ahí redactó el mensaje, cuando creció el pelo de su sirviente el mensaje quedó escondido. Otra forma de ocultación la describió Plinio el Viejo: con la "leche" de la planta Thithymallus se podía hacer tinta invisible.

"Más importante que el oro, diamantes y cualquier otro mineral, podría ser el razonamiento, pues podría ser la clave para generar una poderosa arma para triunfar una guerra e incluso para mantener en incognito los continuos avances tecnológicos y los desarrollos científicos" [23].

"Pero ¿cómo podían los individuos asegurar que los mensajes enviados en caso de ser interceptados, solo tengan la posibilidad de usarlo y comprenderlo entre las partes involucradas y no sus adversarios?

Plasmando el razonamiento en una representación simbólica, comúnmente redactada y después transformándola, por medio de una clave secreta, en una representación

equivalente (criptograma) e ininteligible para esos que la desconozcan y que únicamente esos conocedores de la clave y el procedimiento para eso, podrían regresarla a su forma original" [24]. Al arte de utilizar estas transformaciones hasta llegar a una representación *"enigmática"* por medio de una clave secreta, se le llama actualmente criptografía.

Una referencia antigua de aplicación criptográfica para enviar información con fines militares, es la de Julio César, quien usó técnicas estenográficas, criptográficas y de codificación para garantizar la confidencialidad de sus mensajes. El historiador Suetonio menciona lo cual se ha denominado "la cifra del César", al procedimiento de sustitución o movimiento de +3 sobre las letras del alfabeto. Otro método empleado por Julio Cesar [25], ha sido el de codificación, redactaba los mensajes en griego, lenguaje desconocido por sus enemigos, de esta forma, en caso de ser interceptados los mensajes, resultaban indescifrables, por ignorar el lenguaje, "el código".

El francés Henri Poincaré (1854-1912), es considerado como uno de los mejores matemáticos y fundador de la teoría del caos [26]. Por otra parte, desde hace unas dos décadas que da inicio el estudio de los sistemas caóticos para la encriptación de la información, R. Matthews en 1989, propuso por primera vez implementar un algoritmo caótico para emplearlo en la criptografía [27].

2.1.1 Orígenes de la criptografía

Desde tiempos remotos se habla de la existencia de técnicas criptográficas, la mayor parte de culturas antiguas las han utilizado de una manera u otra. El reemplazo de símbolos, la manera más elemental de criptografía, se pueden encontrar en escrituras de las grandes culturas mesopotámica y egipcia. El ejemplo más antiguo conocido de criptografía se localizó en la tumba de un noble egipcio denominado Khnumhotep II, que vivió hace alrededor de unos 3,900 años.

El cifrado de mensajes se ha venido llevando a cabo desde hace más de 4,000 años. Etimológicamente criptografía proviene del griego: krypto, «oculto», y graphos, «escribir»; Lo que actualmente entendemos como, "escritura" y "esconde".

A lo largo de los años se han venido desarrollando diferentes dispositivos y creando nuevas técnicas de encriptación. La Figura 1 ilustra algunas de las técnicas que fueron empleadas en épocas más antiguas al igual que los dispositivos implementados [28]–[30].

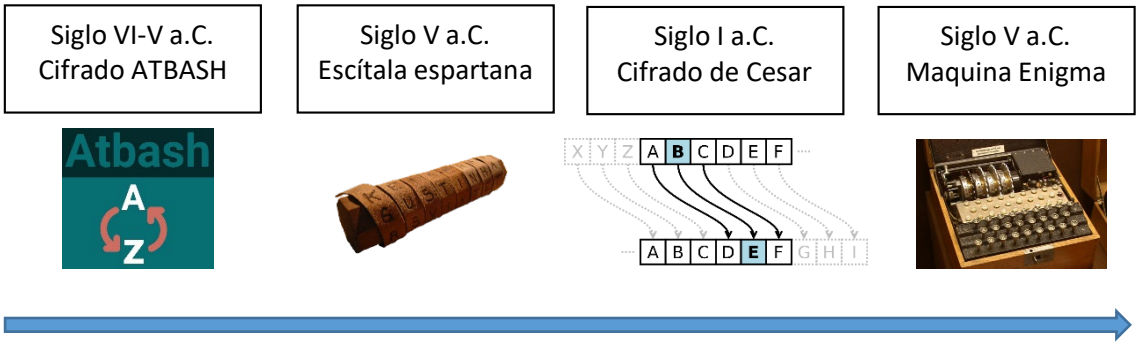


Figura 1: Línea del tiempo de dispositivos criptográficos.

2.1.2 Sustitución mono alfabética cifrado ATBASH

La Figura 2 muestra el cifrado Atbash, es uno de los métodos más comunes de cifrado y uno de los más antiguos, fue utilizado entre el siglo VI y V a.C., este pertenece a la criptografía clásica y se le conoce como cifrado por sustitución [29].

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A

Figura 2: Alfabeto representativo del cifrado Atbash.

Cifrado:

Las letras superiores se sustituyen por las inferiores para cifrar un escrito. El proceso que se lleva a cabo es invertir el abecedario, intercambiando la letra A por la Z, la B por la Y, la C por X y así sucesivamente. Si se tiene como palabra de entrada ATBASH el resultado encriptado es ZGYZHS.

Descifrado:

El proceso de descifrado se hace de la misma manera que al cifrarlo, debemos sustituir las letras del abecedario invertido por las del abecedario original.

2.1.3 Escítala espartana

En la antigua Grecia, los espartanos empleaban un procedimiento para trasmitir información confidencial. La alusión a este procedimiento está en el tomo III de las Vidas Paralelas de Plutarco. Los espartanos implementaron en el siglo V a. C., la Escítala (ver Figura 3), "*el primer sistema de criptografía por transposición, ocultar el sentido de un escrito alterando el orden de las letras que lo componen"* [30].

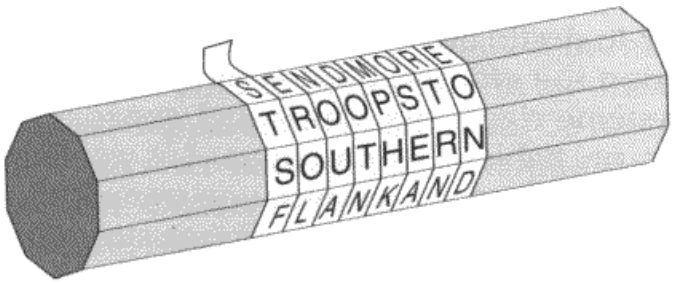


Figura 3: Representación de una vara Escítala, tomado de [31].

Cifrado:

Los militares escribían sus mensajes sobre una tira de cuero o papiro que enrollaban a una vara de cierto grosor y tamaño, después simplemente lo desenrollaban y se creaba un mensaje totalmente oculto. Una vez el mensaje oculto, era enviado al receptor quien contaba con una vara idéntica para descifrar el mensaje, si no se sabía la medida del diámetro de la escítala era realmente difícil descifrar el mensaje.

Descifrado:

Para el descifrado del mensaje bastaba con tener conocimiento del diámetro de la escítala y reorganizar la cinta para obtener el mensaje legible, en caso contrario de no contar con el diámetro exacto el mensaje seguiría siendo ilegible por parte del receptor o intruso que haya logrado interceptar el mensaje.

2.1.4 Cifrado de Cesar

Los romanos en el siglo I a. C. crearon su propio sistema de encriptación llamado cifrado de Cesar (ver Figura 4) también conocido como encriptado por desplazamiento, éste consistía en reemplazar cada letra del mensaje por otra que corresponda al

desplazamiento de +3 posiciones hacia la derecha a partir de la letra origen en el alfabeto [30].

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

Figura 4: Alfabeto representativo del cifrado de Cesar.

La Figura 4 muestra un ejemplo del método encriptador empleado por Julio Cesar, en la parte superior se encuentra el abecedario español, en la parte inferior el abecedario español recorrido tres posiciones a la derecha.

Cifrado:

El método consiste en cambiar cada letra del mensaje original por la que le corresponde al abecedario desplazado. El cifrado de César originalmente emplea un movimiento de +3 posiciones: una ’A’ en el escrito original se sustituye por una ’D’ en el mensaje escondo. Si tenemos como ingreso la palabra CESAR el resultado cifrado va a ser FHVDU.

Descifrado:

Para descifrar el mensaje es necesario realizar lo contrario al cifrado, si tenemos como ingreso la palabra FHVDU basta con recorrer cada letra -3 posiciones a la izquierda, teniendo como resultado la palabra descifrada CESAR.

2.1.5 Máquina Enigma

La máquina enigma creada por Arthur Scherbius fue patentada en 1918 y fue clave durante la segunda guerra mundial, fue un dispositivo que se implementó para encriptar información y fue tan efectiva que hizo que cambiara el curso de la guerra, sus mensajes eran indescifrables.

La máquina enigma es considerada un dispositivo electromecánico (ver Figura 5), el mecanismo estaba construido básicamente por un teclado a base de interruptores eléctricos el cual se asemeja al de las máquinas de escribir, contaba con un engranaje

mecánico y un panel de luces que contenía las letras del abecedario. Los nazis creyeron que enigma era indescifrable y lo fue por un tiempo, hasta que el matemático Alan Turing quien fue contratado por los servicios secretos británicos descifró las claves de Enigma y contribuyó a dar fin a la guerra. Para el año 1941 la armada logra interceptar un barco alemán con equipos y códigos de cifrado, así como el submarino U-110 donde encontraron una maquina enigma, un libro de códigos y un manual de operaciones [32], [33].

Figura 5: Maquina enigma usada por Alemania nazi durante la segunda guerra mundial, tomado de [31].

Cifrado:

Para encriptar en la maquina enigma es necesario primero configurar los rotores, por ejemplo, el rotor No. 1 en la hendidura 7, el No. 2 en la 4 y el No. 3 en la 6, de esta manera la hendidura 1 corresponde a la letra X, la hendidura 2 corresponde a la letra J y la hendidura 3 corresponde a la letra A. Al presionar las teclas los rotores giran, por lo que, si tenemos como ingreso la palabra ABCABC el resultado encriptado va a ser XHTLOA, de esta manera el análisis de frecuencia es inútil en la práctica debido a que aparentan ser letras enviadas al azar.

Descifrado:

Para lograr descifrar el mensaje, el funcionamiento se invierte. El receptor debe poner la máquina enigma en la configuración inicial e introducir el mensaje recibido.

2.1.6 Criptografía clásica

El término de criptografía inicialmente se manejaba como esteganografía, que trata el estudio y la aplicación de distintas técnicas que permiten esconder un mensaje a simple vista. Los procedimientos iniciales de cifrado se basan en la sustitución y transposición de letras y números, pero que han quedado rebasados frente a la tecnología presente. Dichos procedimientos son simétricos, es decir, que se utiliza la misma clave para cifrar que para descifrar. Algunos de los métodos de criptografía clásica son: Cifrado ATBASH (siglo IV a. C.), escítala espartana (siglo V a. C.), tablero de Polibio (siglo II a. C.), cifrado de Cesar (siglo I a. C.), Disco de Alberti (1467), maquina enigma (1918), entre otros. [34].

La Figura 6 muestra los métodos de criptografía más empleados y que corresponden:

Sustitución monoalfabeto: Trata de suplir cada letra del mensaje por otra del mismo abecedario. La correspondencia de aquellas letras va a ser la misma para todo el mensaje. El procedimiento de Julio Cesar, es un cifrado por sustitución.

Sustitución polialfabeto: La sustitución dependerá del lugar que ocupa cada carácter en el texto, puede ser tanto periódica como no periódica. La máquina enigma es periódica y progresiva y corresponde a esta clase de sustitución.

Permutación: Este cambia de lugar los caracteres en el texto. La clave se encuentra en el número de columnas o de bloques y los cambios que hay en los bloques. Otra forma de cifrar es modificar filas por columnas.

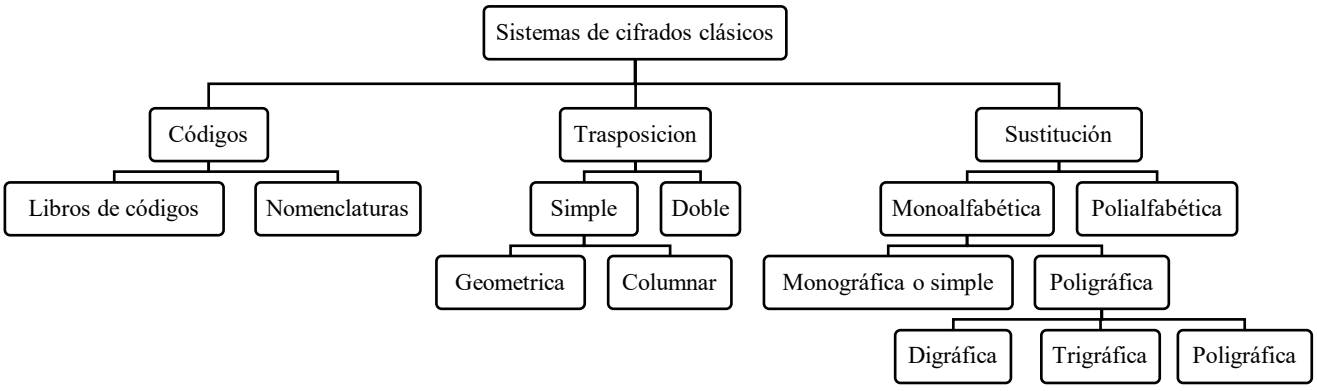


Figura 6: Diagrama de sistemas de cifrados clásicos, tomado de [34].

2.1.7 Criptografía moderna

Se considera la criptografía moderna a partir de que el ingeniero y matemático Claude E. Shannon presenta en 1949 un trabajo sobre la teoría de la información donde profundiza los aspectos teóricos de la criptografía [35], adicionalmente coadyuva el nacimiento de la computación donde la información emigra a una representación por medio de *bits* y no de caracteres. Adicionalmente se introduce el manejo de clave pública y clave privada en criptografía.

2.2 Tipos de encriptados

IBM (International Business Machines) desarrolló el algoritmo de cifrado DES (Data Encryption Standard), dos años después, se transformaría en un FIPS 46-3 (Federal Information Processing Standard) empleado en todo el mundo. En el año 2001, DES se ve amenazado por AES (Advanced Encryption Standard) que después de cinco años de revisión, se ha convertido en un estándar.

Whitfield Diffie y Martin Hellman criptógrafos estadounidenses publican en 1976 el artículo "*New Directions in Cryptography*" [36], que permite sentar las bases de la criptografía asimétrica, es decir, se requiere de una clave pública y una clave privada.

La criptografía asimétrica hoy es esencial para lograr con mayor seguridad transacciones llevadas a cabo por medio del uso de Internet, como en las páginas que utilizan el protocolo HTTPS (Hypertext Transfer Protocol Secure) en los navegadores web o para cifrar mensajes utilizando PGP (Pretty Good Privacy) que corresponde a un criptosistema hibrido que logra combinar tanto criptografía simétrica como criptografía asimétrica.

2.2.1 Criptografía simétrica o de clave privada

La Figura 7, muestra un esquema de cifrado simétrico y que ha sido ampliamente utilizada. En el esquema se representa el uso de clave privada y que corresponde a una sola clave, la cual se emplea tanto para cifrar y descifrar el mensaje permitiendo una comunicación segura entre emisor y receptor. Por su parte, dicha clave deberá ser compartida entre ambos interlocutores por un canal seguro [37], [38].

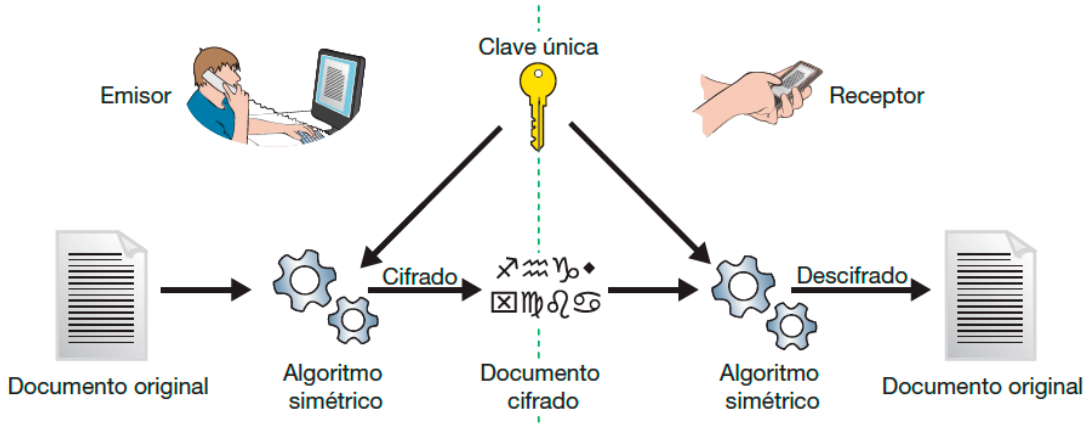


Figura 7: Funcionamiento de clave simétrica, tomado de [39].

Los cifrados simétricos se dividen en dos tipos:

Cifradores de flujo: Cada carácter del texto sin formato se cifra uno por uno la operación que empleamos es un XOR, el cual sirve para realizar la combinación de cada digito que típicamente es un *bit* con la clave de cifrado.

Cifradores de bloque: Toman un grupo de *bits* normalmente de 64 *bits* y lo cifran como si fuera uno solo. Se ingresan bloques de texto plano y se producen bloques de texto cifrado.

2.2.2 Criptografía asimétrica o de clave publica

La Figura 8 muestra un esquema del método de encriptado asimétrico o de clave pública. Diffie y Hellman presentan las bases de la criptografía asimétrica, sin embargo, su consolidación llega con el método de RSA (Rivest Shamir y Ademan).

La criptografía asimétrica o de clave pública está basada en dos claves diferentes, una se emplea para encriptar el mensaje que es pública y no hay riesgo al compartirla por cualquier medio, aunque esta sea interceptada por algún tercero. Ésta llave solo se emplea para encriptar la información, por otra parte, la otra llave es privada y solo la tendrá el propietario para ser usada en el descifrado o desencriptado del mensaje anteriormente cifrado.

De esta manera el emisor encripta el archivo con la clave pública del receptor y lo

manda por un medio inseguro como lo puede ser el Internet, el documento esta total mente protegido en su viaje hasta llegar al receptor quien cuenta con la clave privada para obtener el mensaje que ha sido encriptado [37], [38].

Figura 8: Funcionamiento de clave asimétrica, tomado de [39].

2.3 Teoría del caos

La teoría del caos es una rama de las matemáticas además de otras ciencias, estudia distintos sistemas dinámicos que pueden ser estables, inestables y caóticos, algo en lo que se destaca este tipo de sistemas es en su gran sensibilidad a pequeños cambios en sus condiciones iniciales.

Un sistema estable tiende con el paso del tiempo a un punto u orbita (atractor), un sistema inestable se escapa de los tractores y un sistema caótico muestra los dos comportamientos anteriores [40].

Algunos ejemplos de los fenómenos que se pueden estudiar con los sistemas caóticos son: control de la población, epidemias, el vuelo de las aves, movimiento de bancos de peces, insectos migratorios, predicción del tiempo, entre otros.

Edward Norton Lorenz fue pionero en la investigación y desarrollo de la teoría del caos que surge en la segunda mitad del siglo XX. En el año 1963 trabajo en un sistema de ecuaciones que esperaban le ayudaran a comprender las dinámicas atmosféricas y lograr predecir los cambios climatológicos, sus ecuaciones diferenciales fueron implementadas en computadoras de la época para tratar de observar el comportamiento y llevar a cabo un análisis más detallado [41].

Lorenz encontró que una pequeña variación en las condiciones iniciales (alrededor de 3 o 5 decimales de diferencia) llevaba a un resultado totalmente distinto en las predicciones del modelo. De tal manera que una pequeña perturbación o alteración en las condiciones iniciales de su sistema de ecuaciones puede tener un gran efecto sobre el resultado final. De aquí surge la frase conocida como "Efecto mariposa", que implica que, "*el aleteo de las alas de una mariposa puede provocar un Tsunami en el otro lado del mundo*" [42].

2.3.1 Atractor de Lorenz

La Figura 9 muestra el atractor de Lorenz introducido en 1963 y está representado matemáticamente por el sistema de ecuaciones (1) [43].

Ecuación que define el mapeo de Lorenz:

$$\begin{aligned} \frac{dx}{dt} &= a(y - x) \\ \frac{dy}{dt} &= x(b - z) - y \\ \frac{dz}{dt} &= xy - cz \end{aligned} \tag{1}$$

donde con los valores de los parámetros a=10, b=28 y c=8/3, el sistema muestra un comportamiento caótico.

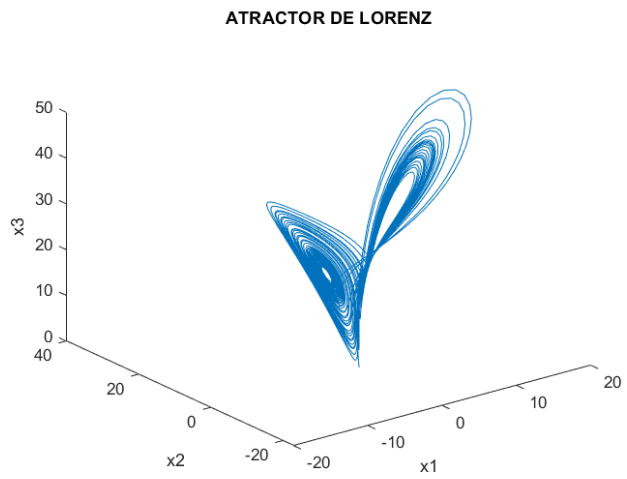


Figura 9: Atractor de Lorenz en 3D.

2.3.2 Circuito de Chua

El atractor de Chua representando en la Figura 10 surge del estudio de un circuito electrónico que muestra un comportamiento caótico continuo, esto se puede interpretar como un oscilador no periódico, que produce una forma de onda que nunca se repite, fue desarrollado en 1983 por Leon O. Chua y se representa matemáticamente por el sistema de ecuaciones (2) (3) [44].

Ecuación que define el circuito de Chua:

$$\begin{aligned} \frac{dx}{dt} &= \alpha[y - x - f(x)] \\ RC_2\frac{dy}{dt} &= x - y + Rz \\ \frac{dz}{dt} &= -\beta_y \end{aligned} \tag{2}$$

$$(x_1) = bx_1 + \frac{1}{2}(a-b)(|x_1+1| - |x_1 f - 1|) \tag{3}$$

donde con los valores de los parámetros alfa=10, beta=14.87, a=-1.27 y b=-0.68, el sistema muestra un comportamiento caótico.

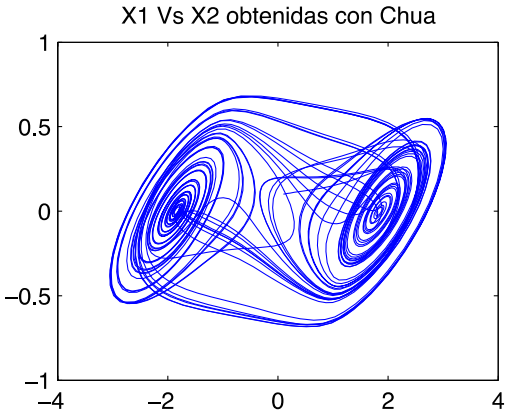


Figura 10: Atractor de Chua 2D.

2.3.3 Atractor de Rössler

El atractor de Rössler representado en la Figura 11 desarrollado por Otto E. Rössler en 1976, es un sistema de tres ecuaciones diferenciales ordinarias no lineales [45]. Matemáticamente el sistema diferencial se representa por el sistema de ecuaciones (4).

Ecuación que define el atractor de Rössler:

$$\begin{aligned} x1 &= -x2 - x3, \\ x2 &= x1 + ax_{2,} \\ x3 &= b + x3(x1 - c) \end{aligned} \tag{4}$$

Donde con los valores de los parámetros a=0.398, b=2 y c=4 y las condiciones iniciales son x1(0)=0, x2(0)=0.1 y x3(0)=0 permiten el régimen caótico.

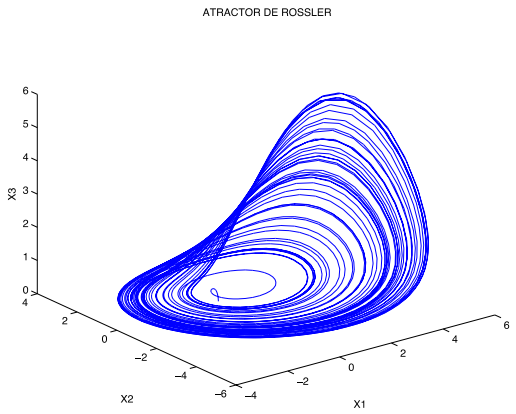


Figura 11: Atractor de Rössler en 3D.

2.3.4 Mapeo de Hénon

El mapeo de Hénon se observa en la Figura 12, es un sistema dinámico que es discreto en el tiempo [46]. Las ecuaciones que definen al sistema está dado por (5).

Ecuación que define el mapeo de Hénon:

$$\begin{aligned} x_n + 1 &= 1 - \alpha x_n^2 + y_n \\ y_n + 1 &= \beta x_n \end{aligned} \tag{5}$$

donde con los valores de los parámetros $\alpha = 1.4$ y $\beta = 0.3$ generan el comportamiento caótico.

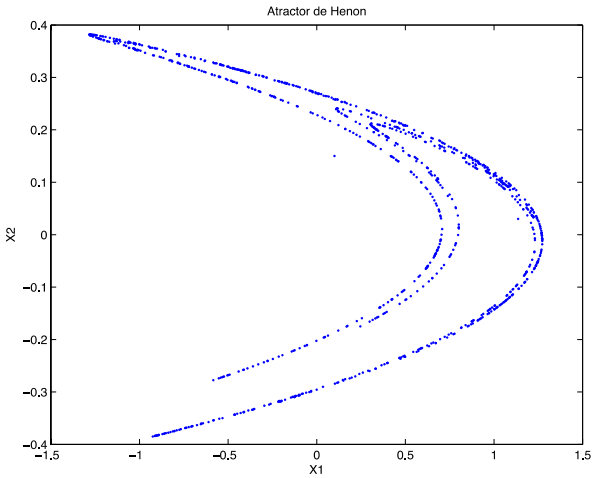


Figura 12: Mapeo de Hénon (x,y).

2.4 Sistemas embebidos

Un sistema embebido o empotrado se trata de un sistema que procesa datos y están diseñados para una función específica, su estructura se basa en el uso de microprocesadores o microcontroladores, se implementan para un fin sistematizado que no requiere muchas tareas, son sistemas que cumplen con tareas específicas y están optimizados en cuanto a su tamaño, costo, consumo de energía, confiabilidad y desempeño. Los sistemas embebidos los podemos clasificar en dos tipos [47]:

i) Sistemas embebidos pequeños: Éstos sistemas pequeños o simples se basan en microcontroladores que incluyen un único chip procesador, memoria, interfaces, conversores y timers.

ii) Sistemas embebidos grandes: Éstos sistemas se basan en microcontroladores al igual que los pequeños, pero contienen distintos tipos de sensores, como pueden ser giroscopio, sensor de humedad, wifi, sensor de temperatura, barómetro, etc.

Algunos ejemplos de sistemas embebidos son: taxímetro, teléfono celular, reproductor de audio, sistema de control de acceso, entre muchos otros.

Por lo general los sistemas embebidos pueden ser programados con lenguaje ensamblador, lenguaje C o C++ si la velocidad de procesamiento no es crítica, se puede usar el lenguaje JAVA que es un programa orientado a objetos.

Muchos de los sistemas embebidos que se emplean deben cumplir con la restricción de operar en tiempo real. Se debe tener en cuenta que la respuesta del sistema debe ser en un lapso de tiempo determinado, la información recibida puede ser correcta, sin embargo, si llega desfasada en tiempo, puede ya no ser relevante y comprometer el sistema por completo dado al retraso de respuesta del dispositivo embebido [48].

Algunos sistemas embebidos operan bajo un sistema operativo como es el caso de la Raspberry Pi.

2.4.1 FPGA

La Figura 13 muestra un FPGA que son un arreglo matricial de bloques lógicos programables desarrolladas en 1984 por Bernard Von Der Schmitt y Ross Freeman. La lógica programable que utilizan puede ejecutar operaciones sencillas como una compuerta lógica hasta complejos sistemas como lo hace un chip.

Los FPGA tienden a ser utilizados en aplicaciones similares a los de ASIC (Application Specific Integrated Circuit), aunque son más lentas, tienen un mayor consumo de energía y no pueden contener sistemas tan complejos, la ventaja es que son reprogramables y sus costos de desarrollo y adquisición son menores. La interfaz JTAG (Joint Test Action Group) es el método de programación de prueba y descarga de datos que implementan los dispositivos FPGA, los códigos de programación se pueden crear con HDL, pero el más utilizado es el lenguaje de descripción de hardware como VHDL [49].

Figura 13: Estructura física de un FPGA, tomada de https://www.mouser.mx/new/xilinx/xilinx-sp701-eval-kit/.

2.4.2 Microcontrolador ESP32

La Figura 14 muestra el sistema ESP32 creado por Espressif Systems, es un sistema embebido de bajo consumo, bajo costo, contiene Wi-Fi, Bluetooth de 2,4 GHz y está diseñado con el TSMC (Taiwan Semiconductor Manufacturing Company) de 40 nm de potencia ultra baja. Esta creado para conseguir la mejor potencia y rendimiento de RF (Radio Frequency), demostrando robustez, confiabilidad y versatilidad en distintas variedades de aplicaciones y escenarios. El ESP32 está pensado para aplicaciones móviles, dispositivos electrónicos portátiles y para el Internet de las cosas (IoT). Cuenta con las propiedades de vanguardia de los chips de baja potencia, incluida la sincronización de reloj, diversos métodos de potencia y escalado dinámico de potencia. En un escenario de aplicación de concentrador de sensores de IoT de baja potencia, por ejemplo, el ESP32 se activa periódicamente y solo una vez que se detecta una condición específica. El periodo de trabajo bajo se usa para reducir la proporción de energía que gasta el chip. La salida del amplificador de potencia además es ajustable, contribuyendo de esta forma a un equilibrio óptimo entre el rango de comunicación, la rapidez de datos y el consumo de energía [50].

Sus características principales son:

- Procesador principal: Tensilica Xtensa LX6 de 32 bits.
- Wi-Fi: 802.11 b / g / n / e / i (802.11n @ 2.4 GHz hasta 150 Mbit / s).
- Bluetooth: v4.2 BR / EDR y Bluetooth Low Energy (BLE).
- Frecuencia de Clock: Programable, hasta 240MHz.
- Rendimiento: hasta 600DMIPS.
- ROM: 448KB, para arranque y funciones básicas.
- SRAM: 520KiB, para datos e instrucciones.

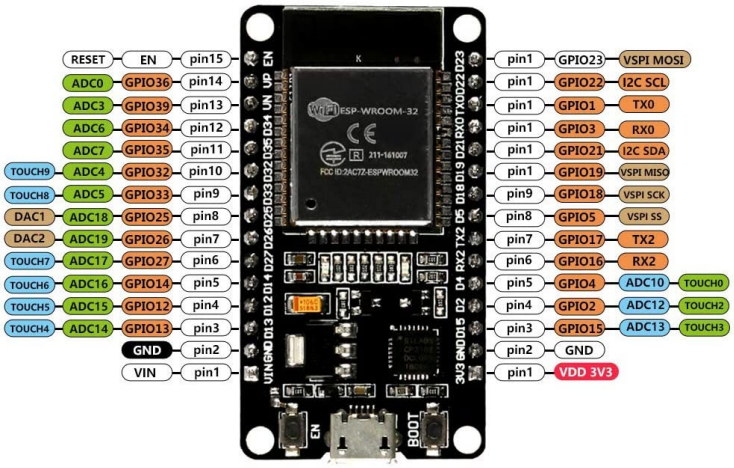


Figura 14: Descripción de terminales del ESP32 y representación física, tomado de https://m.media-amazon.com/images/I/71TbFu6PnoL._AC_SL1001_.jpg.

2.4.3 Raspberry Pi 2

La Figura 15 muestra una Raspberry Pi 2 conocida como ordenadores de placas reducidas u ordenadores de placa simple, fueron desarrollados en reino unido por Raspberry Pi Foundation, son en comparación a otros sistemas embebidos de bajo costo.

Figura 15: Tarjeta Raspberry Pi 2 modelo B (Pi, 2016), tomado de https://www.raspberrypi.org/products/raspberry-pi-2-model-b/.

La Tabla 1 presenta las características de la Raspberry Pi, sus dimensiones de placa son 8.5 × 5.3 cm, cuenta con un chip integrado Broadcom BCM2836 con un procesador ARM11, un procesador gráfico VideoCore IV, 1 GB de memoria RAM, además cuenta con una salida de video HDMI, una salida de audio a través de un minijack 3.5 mm, conexión Ethernet, 4 puertos USB y con puerto GPIO de 40 pines.

Tabla 1: Características de Raspberry Pi 2 Modelo B [51].

Recurso	*Característica*
SoC	Broadcom BCM2836
CPU	ARM11 ARMv7 ARM Cortex A7, 4 núcleos @ 900 MHz.
GPU	Broadcom VideoCore IV 250 MHz. OpenGL 2.0
RAM	1 GB l PDDR2 SDRAM 450 MHz
USB 2.0	4
Salidas de video	HDMI 1.4 @ 1920x1200 pixeles
Almacenamiento	microSD
Bus de expansión GPIO	40 pines

La Raspberry Pi muestra su estructura interna en la Figura 16. Fue creada con la intención de lograr la enseñanza de sistemas informáticos en las escuelas, pero resulto ser más popular de lo que se esperaba debido a que se comenzó a implementar en el sector de robótica. Su manejo no es complicado, se le puede instalar el sistema operativo Windows y la interacción con el usuario se hace como cualquier computadora, cuenta con su propia versión de sistema operativo llamada Raspberry Pi Os, ésta versión cuenta con una tarjeta SD y los nuevos modelos cuentan con una tarjeta MicroSD [52].

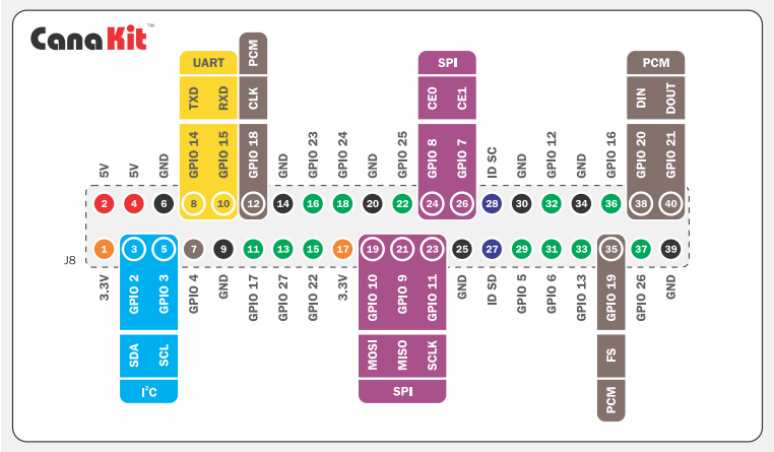


Figura 16: Estructura interna tarjeta Raspberry Pi 2 modelo B (Pi, 2016).

2.5 Protocolos de comunicación

Un protocolo de comunicación es un conjunto de reglas o normas que los sistemas (teléfonos celulares, computadoras, etc.) y programas están obligadas a seguir para lograr una comunicación exitosa de datos entre los ordenadores. Los protocolos de comunicación se pueden implementar tanto en hardware, software o en una combinación de ambos. Se trata de reglas que definen la sincronización de la comunicación, semántica, sintaxis, así como también los métodos que se emplean para la recuperación de posibles errores.

Algunos ejemplos que se implementan en la comunicación analógica como digitales podemos mencionar el protocolo DNP (Distributed Network Protocol), IP (Internet Protocol), TCP (Transmission Control Protocol), algunos otros asociados a Internet son POP (Post Office Protocol), SMTP (Simple Mail Transfer Protocol) y HTTP, todos ellos permiten crear conexiones de comunicación entrelazadas con los diferentes ordenadores [53].

La Tabla 2 muestra algunos protocolos de red más conocidos según las capas del modelo OSI (Open Systems Interconnection) [54], [55].

Tabla 2 Protocolos de redes por capas y niveles.

Capas	*Niveles*	*Protocolos*	*Categoría*
Capa 1	Físico	USB, Ethernet, DSL, Etherloop, Infrared, Frame Relay, SDH, SONET	Transporte de datos
Capa 2	Enlace de datos	DCAP, FDDI, HDLC, LAPD, PPP, STP, VTP VLAN, MPLS	Transporte de datos
Capa 3	Red	ARP, BGP, ICMP, IPv4, IPv6, IPX, OSPF, RARP	Transporte de datos
Capa 4	Transporte	IL, SPX, SCTP, TCP, UDP, iSCSI, DCCP	Aplicación

Capa 5	Sesión	NFS, SMB, RPC, SDP, SMB, SMPP	Aplicación
Capa 6	Presentación	TLS, SSL, XDR, MIME	Aplicación
Capa 7	Aplicación	DHCP, DNS, HTTP, HTTPS, POP3, SMTP, Telnet	Aplicación

Algunos protocolos de comunicación IoT de capa de aplicación son: MQTT, CoAP, SNMP, API REST/HTTP, Websockets, entre otros.

2.5.1 Protocolo HTTP

El protocolo de comunicación HTTP (Hypertext Transfer Protocol) fue desarrollado por Word Wide Web Consortium y la Internet Engineering Task Force a principios del año 1990, su colaboración termino en 1999. Es un protocolo que ha ido cambiando con el tiempo y se transmite sobre el protocolo TCP y también por el protocolo encriptado TLS (Transport Layer Security). Puede ser usado tanto para transmisión de imágenes, videos, datos.

Este protocolo permite realizar un intercambio de datos y recursos, como lo son los documentos HTML. Este protocolo realiza intercambios de datos en la web y tiene una estructura de cliente-servidor, es decir, el cliente es el que inicia la petición de datos y el navegador web es quien responde a lo solicitado. Una página web completa es el resultado de distintos sub-documentos como se muestra en la Figura 17, por ejemplo: Imágenes, texto, videos, scripts, etc.

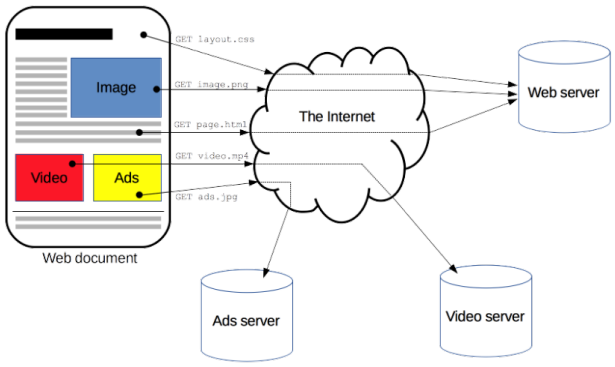


Figura 17: Estructura de funcionamiento HTTP, tomado de https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview.

La Figura 18 muestra la estructura estándar RESTful http, su uso es principalmente para aplicaciones web, aunque, algunos también la emplean para aplicaciones de Internet de las cosas. Las aplicaciones de IoT que emplean HTTP generalmente aprovechan REST, que es un estilo arquitectónico que define un conjunto de restricciones y propiedades basadas en HTTP. Los servicios web que se ajustan al estilo arquitectónico REST, o servicios web RESTful, proporcionan interoperabilidad entre sistemas informáticos en Internet [56].

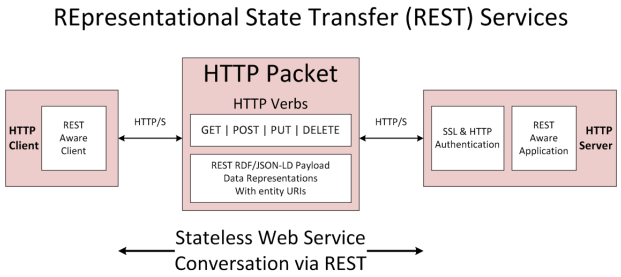


Figura 18: Estructura de paquete estándar RESTful HTTP, tomado de [57].

2.5.2 Protocolo CoAP

El protocolo de comunicación CoAP (Constrained Application Protocol) fue lanzado por IETF en 2013, es un protocolo creado para implementarse en dispositivos de IoT de baja capacidad especialmente para la comunicación M2M (Machine to Machine). El modelo REST (Representational State Transfer) de HTTP emplea cabeceras reducidas, agregando multicast, soporte UDP (User Datagram Protocol) y mecanismos de seguridad.

Las características del protocolo CoAP son:

- Transferencia de documentos cliente/servidor REST
- Fácil de traducir a HTTP para la integración web
- Topología uno a uno con conexiones directas
- Metadatos para diferenciar clases de documentos
- UDP
- Seguridad a través de DTLS

La Figura 19 muestran la estructura de funcionamiento del protocolo CoAP, es otro protocolo de capa de sesión diseñado por el grupo de trabajo IETF Constrained RESTful Environment (Core) para proporcionar una interfaz RESTful (HTTP) liviana. Representational State Transfer (REST) es la interfaz estándar entre el cliente HTTP y los servidores. Sin embargo, para aplicaciones livianas como IoT, REST podría resultar en una sobrecarga y un consumo de energía significativos. CoAP está diseñado para permitir que los sensores de baja potencia utilicen servicios RESTful mientras cumplen con sus limitaciones de potencia [58].

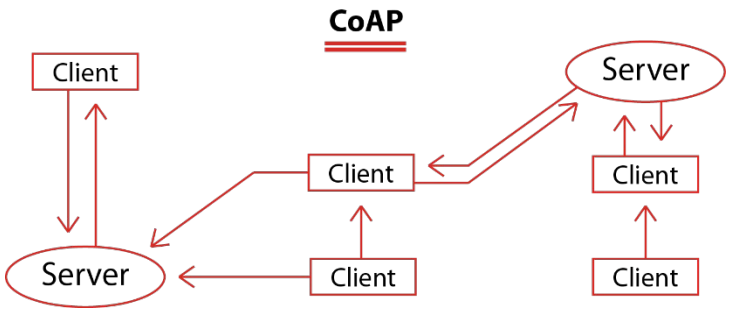


Figura 19: Estructura de funcionamiento estándar CoAP, tomado de https://www.pickdata.net/news/MQTT-vs-coap-best-iot-protocol.

2.5.3 Protocolo MQTT

El protocolo MQTT (Transporte de Telemetría de Message Queue Server) fue desarrollado e inventado por el IBM a finales de los 90. La Figura 20 muestra la estructura de funcionamiento estándar del protocolo MQTT. Es un protocolo de mensajería basado en publicación/suscripción, que está diseñado para dispositivos restringidos y redes de bajo ancho de banda, alta latencia y poco confiables. Los principios de diseño del

protocolo son minimizar los anchos de banda de la red y los requisitos de recursos del dispositivo, al mismo tiempo que intentan garantizar la confiabilidad y cierto grado de garantía de entrega.

Estos principios hacen que el protocolo sea ideal para las comunicaciones M2M en dispositivos de Internet de las cosas, aplicación en la industria 4.0, y para las aplicaciones móviles donde la energía de la batería y el ancho de banda son muy importantes. MQTT funciona sobre el protocolo TCP / IP. Utiliza el puerto 1883 que es asignado por la Autoridad de Números Asignados de Internet (IANA). Para usar MQTT sobre SSL, se usa el puerto 8883 [59].

Algunos de los corredores locales de MQTT son Mosquitto, Mosca, HiveMQ, entre otros. Los corredores públicos son iot.eclipse.org, test.mosquitto.org, broker.hivemq.com, www.cloudMQTT.com, MQTT.dioty.co, etc.

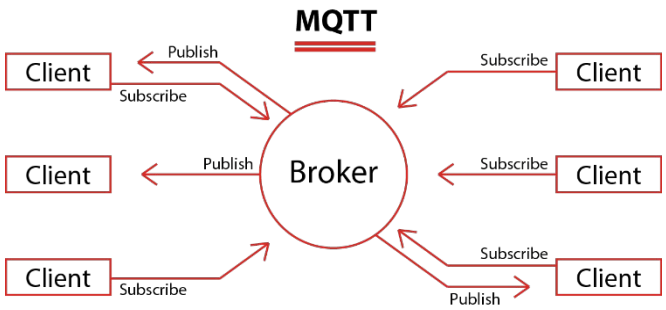


Figura 20: Estructura de funcionamiento estándar MQTT, tomado de https://www.pickdata.net/news/MQTT-vs-coap-best-iot-protocol.

Entre las ventajas del protocolo MQTT, permite el sistema de comunicación M2M, funciona con el patrón pub/sub, es escalable, desacoplamiento entre clientes, sencillez, ligereza, adecuado para realizar aplicaciones en IoT donde normalmente se emplean dispositivos con deficiencia en la potencia, menor necesidad de recursos es igual a un menor consumo de energía, requiere un ancho de banda mínimo, cuenta con tres niveles de calidad del servicio (QoS), aporta robustez y fiabilidad.

Hoy en día el protocolo MQTT ha sido reconocido como uno de los mejores para aplicaciones IoT tanto en lo comercial como para realizar prácticas e investigación.

2.5.4 Comparación de protocolos HTTP vs CoAP vs MQTT

En esta sección se muestran los resultados observados en la literatura sobre la evaluación y eficiencia de consumo de energía de cada protocolo, la topología implementada para la evaluación es de punto a punto ya que es la forma más simple de comunicación entre dos dispositivos, el dispositivo que se emplea es una placa NodeMCU ESP-12E, la red que se implemento es local los servidores son Mosquitto, Copper y Json, el mensaje transmitido es "Hey, I'm Paul!" y se envía una vez con un intervalo de 5 segundos, se realizó la prueba diez veces con cada protocolo. Los resultados obtenidos se muestran en la Tabla 3:

Tabla 3: Resultados de mediciones energéticas *[57]*.

	HTTP	*CoAP*	*MQTT*
No.	Current	Current	Current
1	63.539	58.651	83.089
2	63.539	68.426	68.426
3	64.5161	63.539	63.53
4	58.651	58.651	68.426
5	63.539	58.651	73.314
6	68.426	58.651	73.314
7	53.763	63.539	68.426
8	58.651	63.539	68.426
9	58.651	63.539	68.426
10	73.314	58.651	73.134
AVERAGE	62.65891	61.5837	70.87

2.6 Resumen

Hemos analizado distintos protocolos para implementación en IoT, uno de los protocolos más interesantes es el MQTT debido a que es un protocolo de red liviano, opera como pub/sub, y resalta lo sencillo que es para los desarrolladores de IoT la comunicación con este protocolo.

El protocolo HTTP espera a que el servidor responda y esto ha generado problemas al escalar los proyectos, actualmente hay en el mundo del IoT un gran número de dispositivos conectados a redes no confiables y de alta latencia. Esto ha hecho que la comunicación síncrona sea un problema, las comunicaciones asíncronas son más adecuadas para realizar aplicaciones en IoT, el protocolo HTTP es un protocolo 1-1, el cliente debe hacer una solicitud y el servidor es quien se encarga de responder, tratar de transmitir el mensaje a todos los dispositivos conectados de la red es lo que habitualmente pasa en las aplicaciones IoT, por lo que, resultaría más costoso y difícil de implementar.

Para el sistema Arduino, CoAP es una alternativa interesante a HTTP, de lo contrario MQTT sería el indicado, aunque el consumo de energía del protocolo CoAP sea ligeramente menor, las ventajas del protocolo MQTT aporta mayores beneficios a nuestro proyecto en particular [60].

Si bien es cierto el protocolo de comunicación dependerá bastante de las herramientas con las que se cuenten y los dispositivos embebidos que se tengan en su momento, también dependerá bastante del tipo de proyecto o trabajo se quiera llevar acabo. Ambos protocolos tienen pros y contras, elegir el más adecuado depende de su aplicación.

Realice experimentos, cree prototipos e implemente dispositivos de prueba en redes.

Ambos protocolos, y todo el resto de los "protocolos de IoT" tienen mucho potencial y poder en el mundo de IoT junto a la implementación de la industria 4.0. MQTT es una excelente opción para redes de múltiples dispositivos y CoAP es ideal para conexiones punto a punto cuando los recursos son limitados. La Tabla 4 presenta un comparativo de las principales características de los protocolos.

Tabla 4: Comparación entre protocolos HTTP, CoAP, MQTT [61].

Criteria	*HTTP*	*CoAP*	*MQTT*
Architecture	Client/Server	Client/Server or	Client/Broker
Abstraction	Request/Response	Client/Broker	Publish/Subscribe
Header Size	Undefined	4 Byte	2 Byte
Message Size	Large and Undefined (Depends on the web server or the programming technology)	Small and Undefined (Normally small to fit in single IP datagram)	Small and Undefined (Up to 256 MB maximum size)
Semantics/Methods	Get, Post, Head, Put, Patch, Options, Connect, Delete	Get, Post, Put, Delete	Connect, Disconnect, Publish, Subscribe, Unsubscribe, Close
Quality of Services (QoS)/Reliability	Limited (Via Transport Protocol – TCP)	Confirmable Message or Non-confirmable Message	QoS 0 – At most once QoS 1 – At least once QoS 2 –Exactly once
Transport Protocol	TCP	UDP, TCP	TCP (MQTT-SN can use UDP)
Security	TLS/SSL	DTLS/IPSEC	TLS/SSL
Default	80/443 (TSL/SSL)	5683 (UDP)/5684 (DTLS)	1883/8883 (TLS/SSL)

Capítulo III

3. Análisis de seguridad

Los parámetros de análisis de seguridad o el criptoanálisis son los procesos que se realizan para intentar descifrar un texto claro o en todo caso la clave con la que fue encriptada, un cifrado es computacionalmente seguro si se cumple con uno o ambos criterios siguientes:

- El costo que se realizara para descifrar el mensaje es más elevado que el valor de la misma información cifrada.

- El tiempo que se requiere de procesamiento para descifrar la información excede la vida útil de la misma, quiere decir que para cuando se descifre el mensaje ya no será relevante la información.

Existen diferentes técnicas de criptoanálisis, como, por ejemplo, el análisis de frecuencia que estudia la frecuencia con la que aparecen las letras, es decir, que tanto se usa cada letra en un texto cifrado. Existen otros métodos como el estudio de los histogramas del criptograma, espacio de claves, resistencia a ataques diferenciales, análisis de coeficientes de correlación, análisis de sensibilidad a las condiciones iniciales, correlación de pixeles adyacentes y entropía de la información, entre otros [62]–[65].

Para el caso del UACI y NPCR, lo ideal es que los valores sean del 34 % y 100% respectivamente, es decir, que no exista ningún otro pixel con el mismo valor y en la misma posición para dos imágenes que han sido cifradas, el indicador UACI nos ayuda a evaluar que tan diferente son dos imágenes cifradas entre sí.

La prueba estadística NIST SP 800-22: "Conjunto de pruebas estadísticas para generadores de números aleatorios y pseudoaleatorios para aplicaciones criptográficas" La prueba NIST nos ayuda a enumerar algunas pruebas estadísticas para distinguir buenos generadores de bits aleatorios de los malos ver Tabla 5.

Tabla 5: Comparación de los resultados proporcionales según las 16 pruebas estadísticas del NIST [66].

NIST SP 800-22 Statistical Test	*Hénon*	
	X_{n+1}	Y_{n+1}
Frequency	0.99	0.99
Block Frequency	1.00	1.00
Cumulative sums-Forward	0.99	1.00
Cumulative sums-Reverse	0.99	1.00
Runs	0.90	0.95
Long Runs of Ones	0.94	0.92
Rank	1.00	0.98
Spectral DFT	0.99	0.99
Non-overlapping Templates	0.99	0.99
Overlapping Templates	0.96	0.97
Universal	0.97	1.00
Approximate Entropy	0.96	0.96
Random Excursions	0.98	0.98
Random Excursions Variant	0.98	0.98
Linear Complexity	0.98	0.99
Serial (2m ∇Ψ)	0.98	0.96
Average	0.97	0.97

3.1 Histograma estadístico

Un histograma representa gráficamente una variable a modo de barras, donde el área de las barras puede relacionarse a la frecuencia de los datos representados. Se emplean para obtener información de cómo están distribuidos los datos respecto a las características cuantitativas. También es costumbre que el eje vertical se represente la frecuencia y el eje horizontal los valores que tienen las variables.

Para el caso de los procesos de encriptado, para que un histograma muestre un alto índice de seguridad en la información encriptada, deberá presentar un comportamiento más o menos uniforme a un nivel de frecuencia. A manera de ejemplo, en la Figura 21 se presenta una imagen digital y su correspondiente imagen encriptada, así como los histogramas correspondientes [62]–[64].

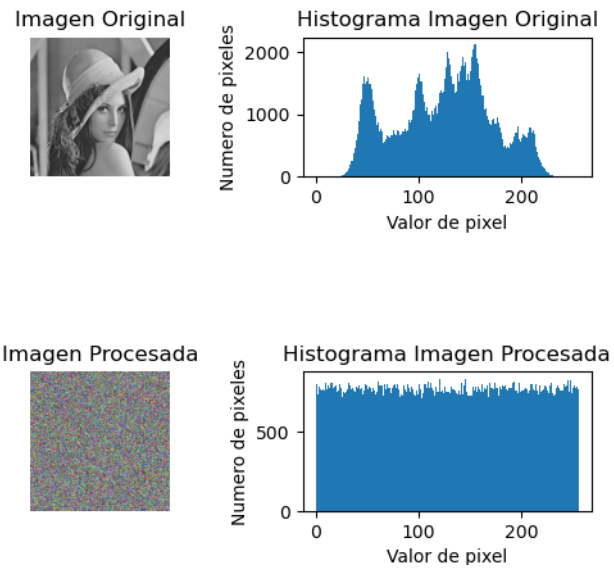


Figura 21: Histograma de seguridad de la imagen y criptograma de Lena.

Se puede observar en el histograma de la imagen original la distribución de los diferentes valores en frecuencia, en contrapartida, para la imagen encriptada se observa que la distribución de frecuencias es más uniforme y lo que no permite ver vestigios de la información original.

3.2 Análisis de sensibilidad a las condiciones iniciales

Una de las características que presentan los sistemas caóticos es la sensibilidad al más mínimo cambio reflejado a las condiciones iniciales, es decir, un pequeño cambio en una de sus condiciones iniciales genera una respuesta caótica única. Esto presenta una ventaja en los sistemas criptográficos basados en caos, ya que, para encriptar y desencriptar información es necesario tener las mismas condiciones iniciales de lo contrario obtendremos distintas dinámicas caóticas, lo que significa, que no se podrá recuperar la información original. Ésta sensibilidad a las condiciones iniciales, tienen un límite que depende de la resolución del sistema digital en el que se aplique [67].

3.3 Correlación de pixeles adyacentes

El análisis de correlación de pixeles adyacentes se realiza en las direcciones horizontal, vertical y diagonal para las imágenes encriptadas buscando que los resultados

tiendan a cero. Los resultados experimentales nos muestran la robustez, eficiencia y el buen nivel de seguridad del algoritmo, en la Figura 27, Figura 28, Figura 29 y Figura 30 se muestran los resultados obtenidos del análisis de correlación de pixeles.

Los métodos tradicionales de criptografía no son los apropiados para imágenes debido a su vulnerabilidad a los ataques estadísticos, debido a la fuerte correlación que existe entre pixeles adyacentes y el análisis a los histogramas a colores (RGB), lo que puede ayudar al intruso a identificarlos dentro de la imagen, es por eso que se propone un algoritmo para cifrar imágenes empleando atractores caóticos [68].

Dos pixeles solo serán adyacentes si, y solo si, tienen en similitud una de sus fronteras, o por lo menos una de sus esquinas. En la Figura 22 se muestra los pixeles adyacentes.

(a) (b)

Figura 22: Ejemplo de dos pixeles adyacentes. (a) Pixeles adyacentes por frontera, (b) pixeles adyacentes por esquina.

3.4 Entropía de la información

La entropía de la información o entropía de Shannon [69] mide la incertidumbre de una fuente de información, nos muestra la aleatoriedad de los datos, puede ser empleado para evaluar la seguridad de la información encriptada por ejemplo una imagen [70].

El cálculo de la entropía se lleva a cabo a partir de la ecuación (6) [65]:

$$H(s) = \sum_{n=0}^{2^N-1} P(S_i) * Log_2(\frac{1}{P(S_i)})bits \quad (6)$$

donde $P(S_i)$ nos representa la probabilidad del símbolo S_i, N es el número de bits que representa la unidad básica de la fuente, 2^N son todas las combinaciones de la unidad básica, para una fuente aleatoria se esperaría una entropía de H(s) = N, si tomamos una imagen con pixeles aleatorios en escala de grises de 8 bits, su entropía seria H(s) = 8.

Cuando el resultado H(s) de un criptograma sea menor de 8 bits, debemos tomar medidas debido a que nuestro modelo cifrador tiene cierto grado de predictibilidad, por lo que la seguridad del sistema se puede ver potencialmente comprometida [71].

Capítulo IV

4. Desarrollo del sistema Encriptador-Desencriptador propuesto

En este capítulo se presenta la estructura, funcionamiento, fiabilidad del prototipo encriptador-desencriptador desarrollado en este trabajo de tesis basado en caos y operando con un protocolo de comunicación para IoT. El prototipo interactúa con el usuario a partir de una interfaz gráfica y un script creados en Python, el script se encuentra en la Raspberry Pi 2 donde se ejecuta el programa y se encarga de almacenar y desencriptar las imágenes enviadas, la GUI creada permite elegir la imagen a encriptar, como también usar condiciones iniciales preestablecidas o ingresarlas manualmente, adicionalmente, lleva a cabo pruebas de seguridad como ataques diferenciales NPCR/UACI, o despliega los histogramas tanto de la imagen original como de la encriptada, almacena la imagen que ha sido encriptada con el nombre que el usuario lo establezca y la envía empleado el protocolo MQTT sobre Wifi.

4.1 Sistema embebido propuesto para el encriptado

El sistema embebido propuesto es empleado tanto como encriptador o desencriptador de imágenes y está conformado por dos Raspberry Pi 2, uno que actúa como emisor quien es el que se encarga de seleccionar, encriptar y enviar la imagen y el otro actúa como receptor que se encarga de recibir, desencriptar y almacenar la imagen.

Al inicio el sistema embebido propuesto fue empleado la ESP32-CAM, sin embargo, el primer problema a resolver fue el almacenamiento de las imágenes adquiridas, debido a que no cuenta con memoria interna para almacenamiento, lo cual se solucionó incluyendo una memoria MicroSD, el segundo problema que se encuentra es la complejidad y el exceso de código que se necesita programar para guardar la imagen y poderla procesar para su correcto encriptamiento, además de realizar las conexiones al

cliente MQTT y Wifi. Se requieren distintas librerías y es necesario cargar una gran cantidad de código lo cual está en oposición con el objetivo de este trabajo de tesis, el cual indica que: "*realizar un sistema que contenga un código eficiente y reducido, que trabaje en estado óptimo disminuyendo el consumo de energía y reservando los recursos del sistema para el procesamiento de la imagen.*"

De estas desventajas en la ESP2-CAM, se implementa la Raspberry Pi 2 por las distintas características que aporta, tales como: ser un ordenador de placa reducida, con fuente de alimentación por cable micro USB V8, cuenta con puertos USB para conexión de mouse, teclado, HDMI, modulo Wifi y la interface es un sistema operativo siendo gráficamente más amigable con los usuarios. Si bien es cierto el costo es más elevado y que varía acorde al modelo, el implementar el proyecto de este trabajo de tesis en la Raspberry Pi 2, reduce significativamente el código a solo 268 líneas de código en su fase final, optimizando los recursos del sistema embebido. La Figura 23 muestra el esquema a bloques de la solución propuesta, el sistema Raspberry Pi 2 adquiere la imagen a encriptar como la clave simétrica, se genera la señal caótica, se realiza el encriptado y la imagen es enviada por el servidor MQTT, el receptor adquiere la imagen a desencriptar, como la clave simétrica, se genera la señal caótica, se realiza el desencriptado y obtiene la imagen original recuperada.

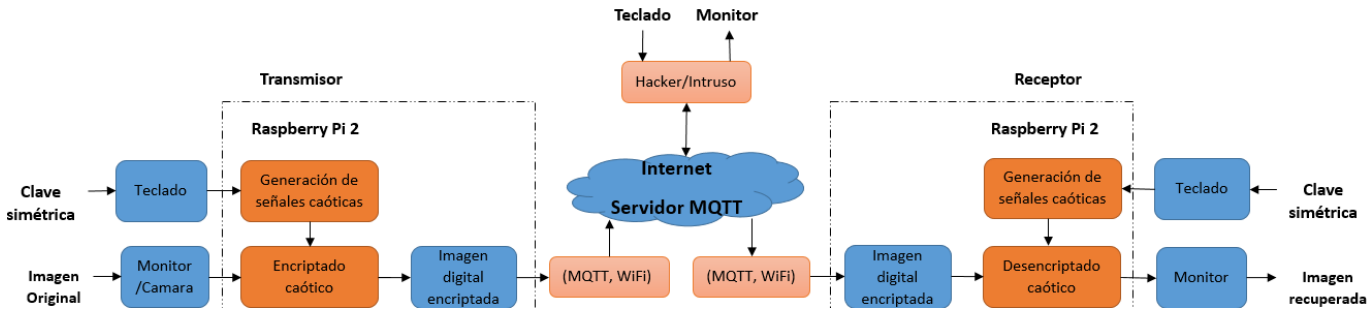


Figura 23: Esquema a bloques de la solución propuesta

4.2 Algoritmo propuesto para el encriptado

La Tabla 6 muestra algunos resultados encontrados en la literatura de los mapeos Hénon, Ikeda, Kaplan-Yorke, Logístico 2D y Rössler [31]. Preferentemente tomando en cuenta los tiempos de encriptado y desencriptado, el algoritmo propuesto en este trabajo de tesis se basa en el mapeo de Hénon (5). La Figura 24 muestra detalladamente la manera

en que funciona el algoritmo propuesto, está compuesto por 6 procesos importantes, selección de imagen, ingreso de claves, encriptado/desencriptado, análisis de seguridad, almacenar y enviar.

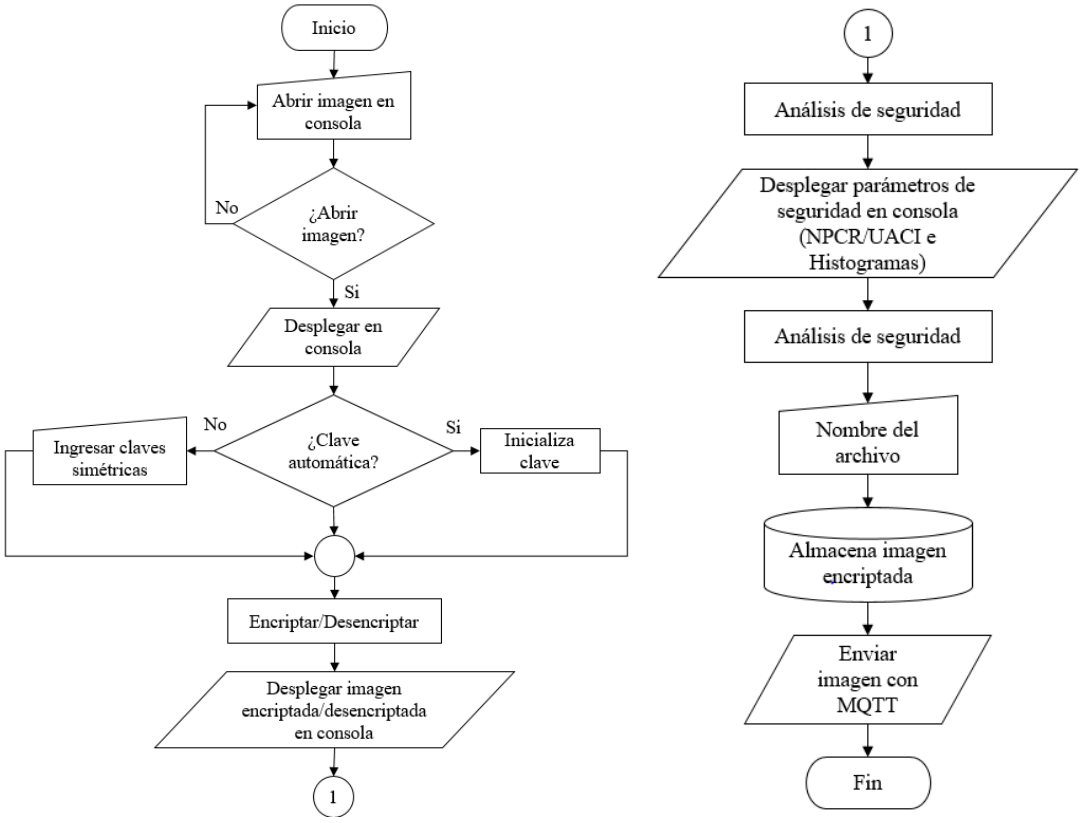


Figura 24: Diagrama de Flujo del algoritmo propuesto "Encriptador-Desencriptador"

Tabla 6: Comparación de resultados de encriptación para cada mapeo utilizado [31].

Parámetro	*Hénon*	*Ikeda*	*Kaplan-Yorke*	*Logístico 2D*	*Rössler*
Clave 1	51258	74815	48512	85631	85631
Clave 2	24301	25301	98621	01425	01425
Tiempo de encriptado	19.1145s	35.2947s	19.1046s	22.6513s	26.4605s
Tiempo de desencriptado	19.1052s	35.1632s	19.1389s	22.6927s	26.4483s
Histograma resistente	Si	Si	Si	Si	Si
FFT de espectro continuo	Si	Si	Si	Si	Si

Mensaje recuperado	Si	Si	Si	Si	Si
Análisis de sensibilidad	Si	Si	Si	Si	Si
Espacio de claves	2^{83}	2^{66}	2^{49}	2^{99}	2^{149}
Entropía del mensaje de voz	2.8417	2.8692	1.9697	2.2894	2.5901
Entropía del criptograma	7.9761	7.9749	7.8327	7.9714	7.9736

4.3 Software usado para el desarrollo del sistema propuesto

Con fines de validación, operatividad y análisis del algoritmo propuesto se implementa inicialmente en MATLAB, sin embargo, la implementación final en el sistema embebido, se hace con el lenguaje Python y se desarrolla en el entorno Spyder 4.

Las características generales de Spyder 4 son:

- Es multilenguaje
- Consola interactiva, cuenta con la integración de figuras Matplotlib
- Visor de documentos
- Explorador de variables
- Explorador de archivos

La instalación de Spyder 4 en el sistema embebido Raspberry Pi 2, se lleva a cabo con el siguiente comando:

```
sudo pip install Spyder 4
```

Para el caso de su desinstalación se emplea el siguiente comando:

```
sudo pip uninstall Spyder 4
```

4.4 Protocolo de comunicación propuesto

Se han analizado varios protocolos de comunicaciones, tales como: HTTP, CoAP y MQTT. El protocolo MQTT es fácil de usar, nos entrega buenos resultados con el tiempo de respuesta, el rendimiento, el menor uso de la batería y el ancho de banda están en primer lugar para dar solución a problemas diversos, presenta una gran confiabilidad en caso de conectividad intermitente y es el más adecuado para el desarrollo de IoT. Adicionalmente, MQTT es liviano y tiene un bajo consumo de batería.

Por su parte, el protocolo CoAP presenta menor confiabilidad en sus mensajes, sin embargo, presenta una mejor eficiencia en el consumo de la batería. MQTT y CoAP son útiles como protocolos de IoT, pero presentan diferencias fundamentales [57].

MQTT es un protocolo de comunicación que permite el intercambio de mensajes entre varios clientes a través de un intermediario central. Desacopla al productor y al consumidor al permitir que los clientes publiquen y que el corredor decida dónde enrutar y copiar los mensajes. Si bien MQTT tiene cierto soporte para la persistencia, funciona mejor como bus de comunicaciones para datos en tiempo real.

CoAP es principalmente un protocolo uno a uno para transferir información de estado entre el cliente y el servidor. Si bien tiene soporte para la observación de recursos, CoAP se adapta mejor a un modelo de transferencia de estado, no puramente basado en eventos.

Los clientes MQTT realizan una conexión TCP (Transmission Control Protocol) saliente de larga duración con un intermediario. Por lo general, esto no presenta ningún problema para los dispositivos detrás de NAT (Network Address Translation). Por su parte, los clientes y servidores CoAP envían y reciben paquetes UDP (User Datagram Protocol). En entornos NAT, se pueden utilizar túneles o reenvío de puertos para permitir CoAP, o los dispositivos pueden iniciar primero una conexión a la cabecera como en LWM2M (Lightweight M2M).

MQTT no ofrece soporte para etiquetar mensajes con tipos u otros metadatos para ayudar a los clientes a comprenderlos. Los mensajes MQTT se pueden usar para cualquier propósito, pero todos los clientes deben conocer los formatos de los mensajes por adelantado para permitir la comunicación. CoAP, por el contrario, proporciona soporte

incorporado para la negociación y el descubrimiento de contenido, lo que permite que los dispositivos se prueben entre sí para encontrar formas de intercambiar datos. Ambos protocolos tienen pros y contras, elegir el más adecuado depende de su aplicación, en este trabajo de tesis, se implementa el protocolo MQTT [72].

Capítulo V

5. Resultados experimentales

En este capítulo se presentan los resultados de los prototipos implementados empleando MATLAB y Python, la interfaz gráfica de usuario (GUI) desarrollada en MATLAB permite encriptar tanto imágenes como audio, la GUI creada en Python permite encriptar imágenes y enviarlas por Internet empleando el protocolo de comunicación MQTT a través de una conexión web. Los prototipos desarrollados se implementan en el sistema embebido Raspberry Pi 2 en donde se ejecutan los programas desarrollarlos, con apoyo de la interfaz gráfica implementada.

Para llevar a cabo el proceso de encriptado se emplean las imágenes Cameraman, Lena, Mandril, Lena RGB, Loto y Paisaje Figura 25 y Figura 26.

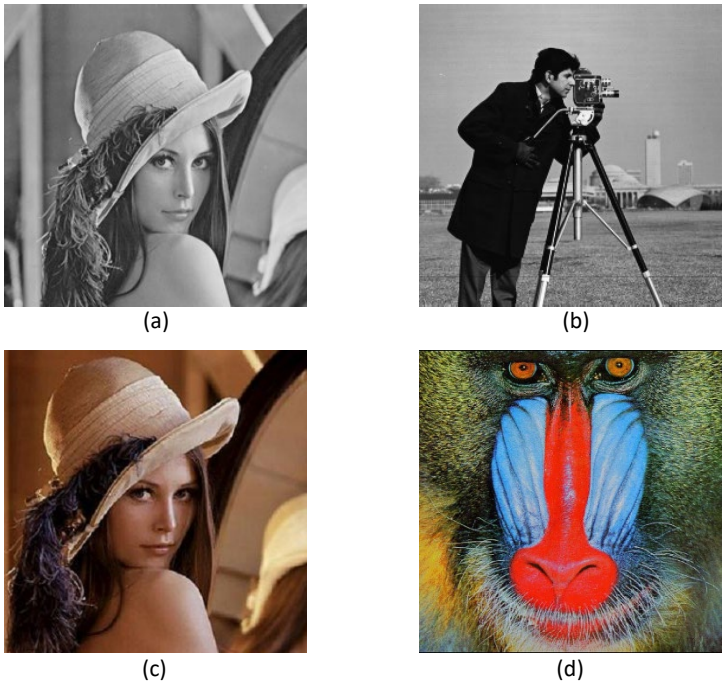
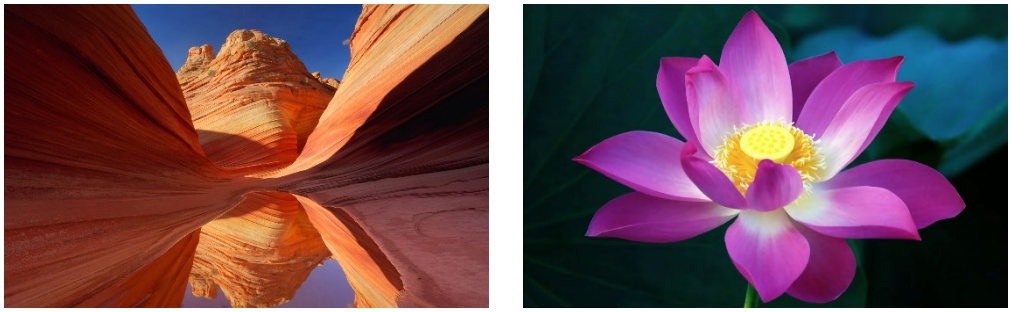


Figura 25: Imágenes usadas para realización de pruebas. (a) Imagen 1 Lena 256 X 256 pixeles en escala de grises, (b) imagen 2 Cameraman 256 X 256 pixeles en escala de grises, (c) imagen 3 Lena 320 X 320 en formato RGB, (d) Imagen 4 Mandril 700 X 700 en formato RGB.



(a) (b)

Figura 26: Imágenes grandes implementadas para realización de pruebas. (a) Imagen 1 Paisaje 1920*1200 en formato RGB, (b) imagen 2 Loto 2560*1600 en formato RGB.

La Tabla 7 presenta los resultados del análisis de seguridad: entropía de la información, correlación de pixeles adyacentes, ataques diferenciales NPCR y UACI, de os criptogramas obtenidos con el método propuesto en este trabajo de tesis.

Tabla 7: Resultados de las pruebas de seguridad Entropía, NPCR, UACI.

Nombre de Imagen Encriptada	*Entropía*	*NPCR (%)*	*UACI (%)*	*Tamaño del Criptograma en pixeles*
Cameraman	7.9991	99.6063	33.4992	256 X 256 (8 bits)
Lena	7.9991	99.6063	33.5001	256 X 256 (8bits)
Lena RGB	7.9994	99.5990	33.4415	320 X 320 (32bits)
Mandril	7.9999	99.6056	33.4762	700 X 700 (32bits)
Paisaje	8	99.6079	33.4437	1920 X 1200 (24bits)
Loto	8	99.6057	33.4302	2560 X 1600 (24bits)

La Figura 27, Figura 28, Figura 29 y Figura 30 presentan el análisis de correlación de pixeles. En todos los casos, las gráficas del lado izquierdo muestran la correlación de pixeles adyacentes de las imágenes originales y las del lado derecho muestran los resultados para las imágenes encriptadas respectivamente. Para estos últimos casos se observa que los puntos están distribuidos de manera errática y se distribuyen por todo el plano, lo que indica que no existe una correlación de pixeles adyacentes.

La Tabla 8 presenta los valores obtenidos de la correlación de pixeles adyacentes de los criptogramas obtenidos con el método propuesto en este trabajo de tesis. de las imágenes originales Lena, Cameraman, Lena RGB y Mandril que se muestran en la Figura 25, los resultados obtenidos son muy próximos a cero, lo que indica que no existe correlación de pixeles en los criptogramas obtenidos.

Tabla 8: Coeficientes de correlación de los criptogramas.

Imagen	*Horizontal*	*Vertical*	*Diagonal*
Lena	0.0478	-0.0214	-0.0189
Cameraman	-0.0017	0.0039	-0.0387
Lena RBG	-0.0172	-0.0051	-0.0096
Mandril	0.0060	-0.0343	0.0166

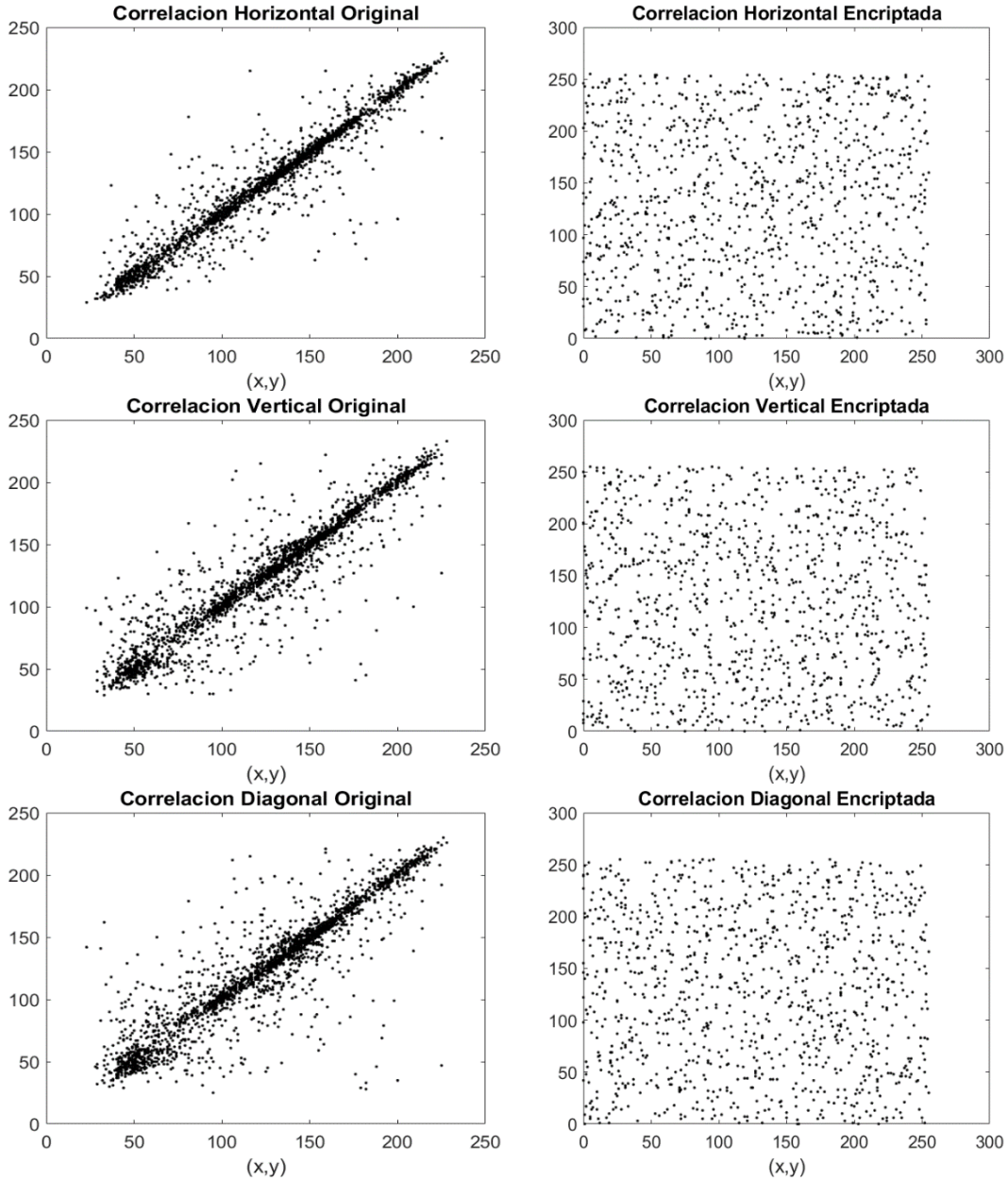


Figura 27: Correlación de pixeles adyacentes del criptograma Lena usando el algoritmo propuesto.

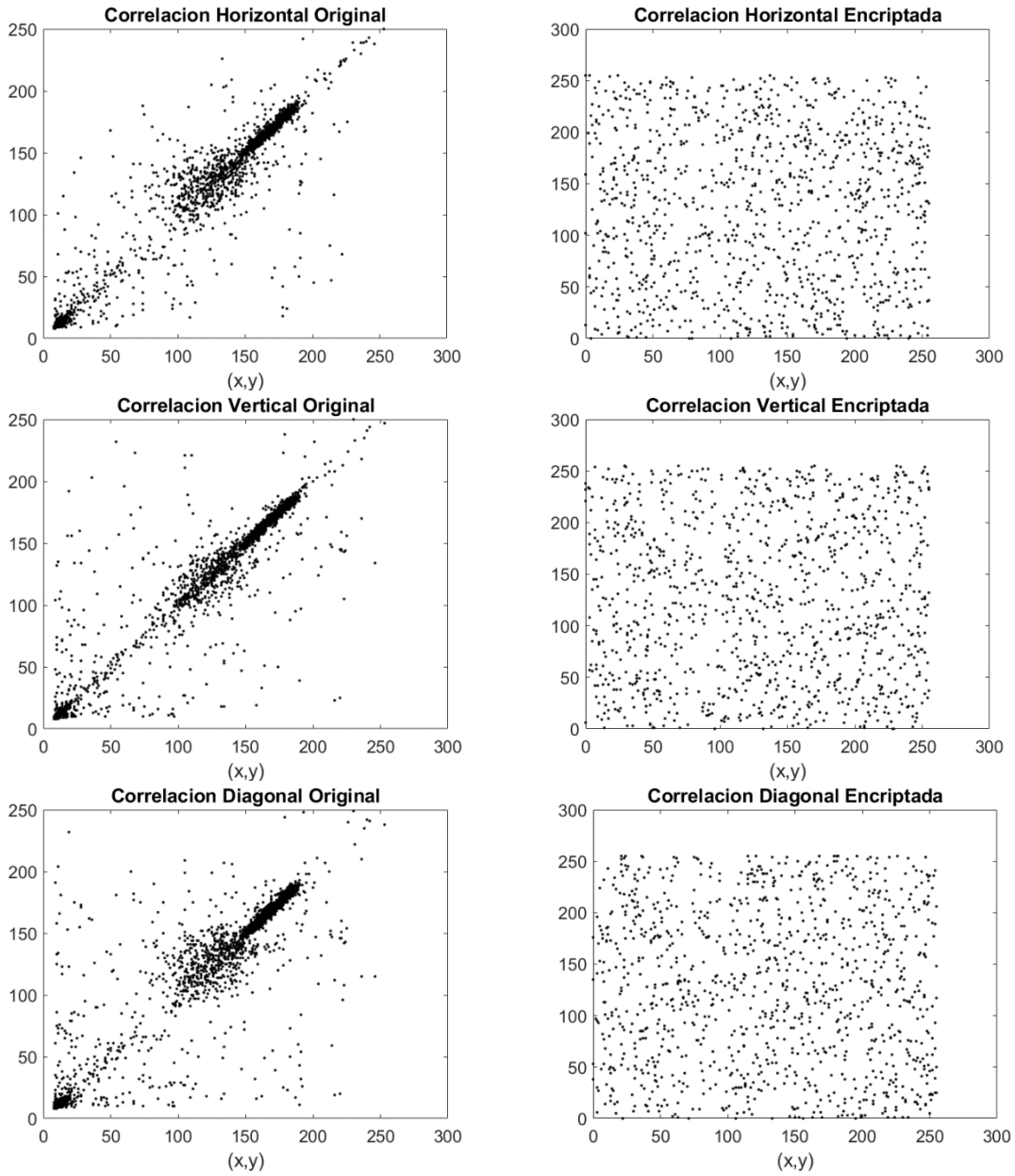


Figura 28: Correlación de pixeles adyacentes del criptograma Cameraman usando el algoritmo propuesto.

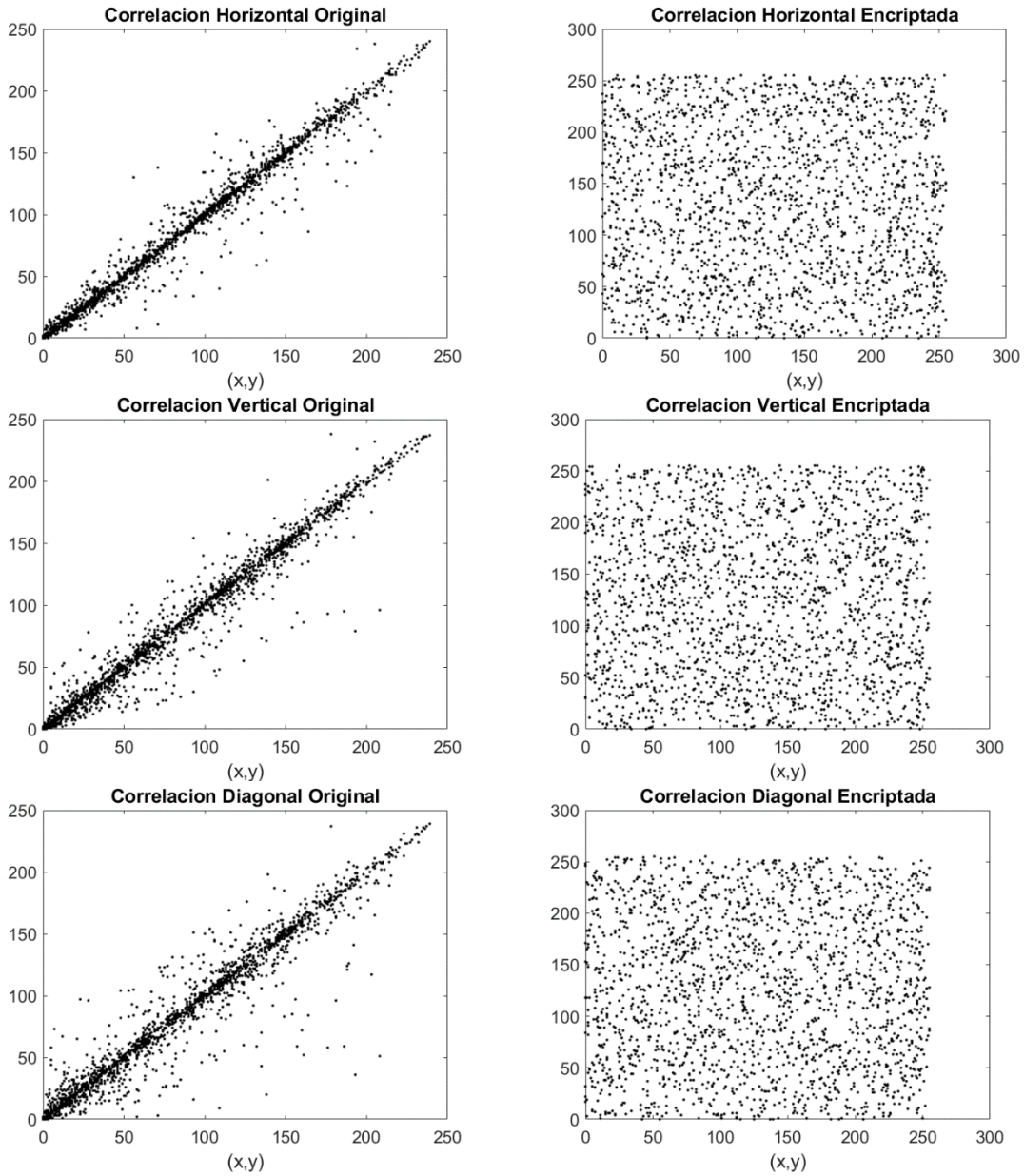


Figura 29: Correlación de pixeles adyacentes del criptograma Lena RGB usando el algoritmo propuesto.

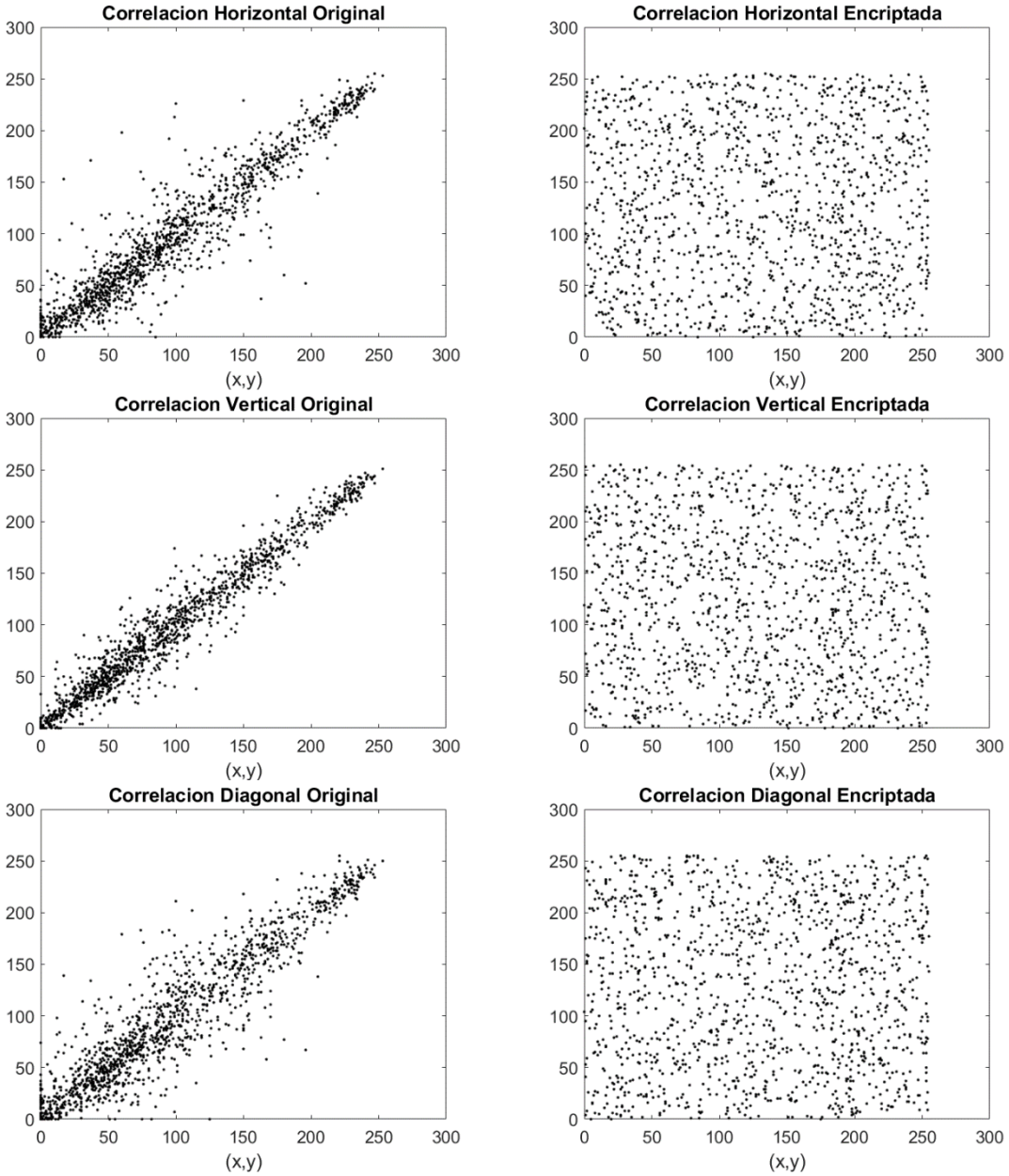


Figura 30: Correlación de pixeles adyacentes del criptograma usando el algoritmo propuesto.

Se observa que la diferencia en la distribución de los pixeles de la imagen original contra la imagen encriptada, es debido a la pseudo-aleatoriedad, de tal forma que evita que haya una correlación de pixeles adyacentes, por lo tanto, se verifica que el algoritmo encriptador propuesto es robusto y fiable.

5.1 Encriptado-Desencriptado usando MATLAB

En este capítulo se presentan los resultados obtenidos con la GUI que se visualizan en la Figura 31. Esta GUI fue desarrollada usando MATLAB y está diseñada para encriptar imágenes y audio aplicando el mapeo de Hénon y el modelo de Lorenz. Esta sección está dividida en dos partes, en la primera se aborda el encriptado de imágenes y en la segunda el encriptado de audio.

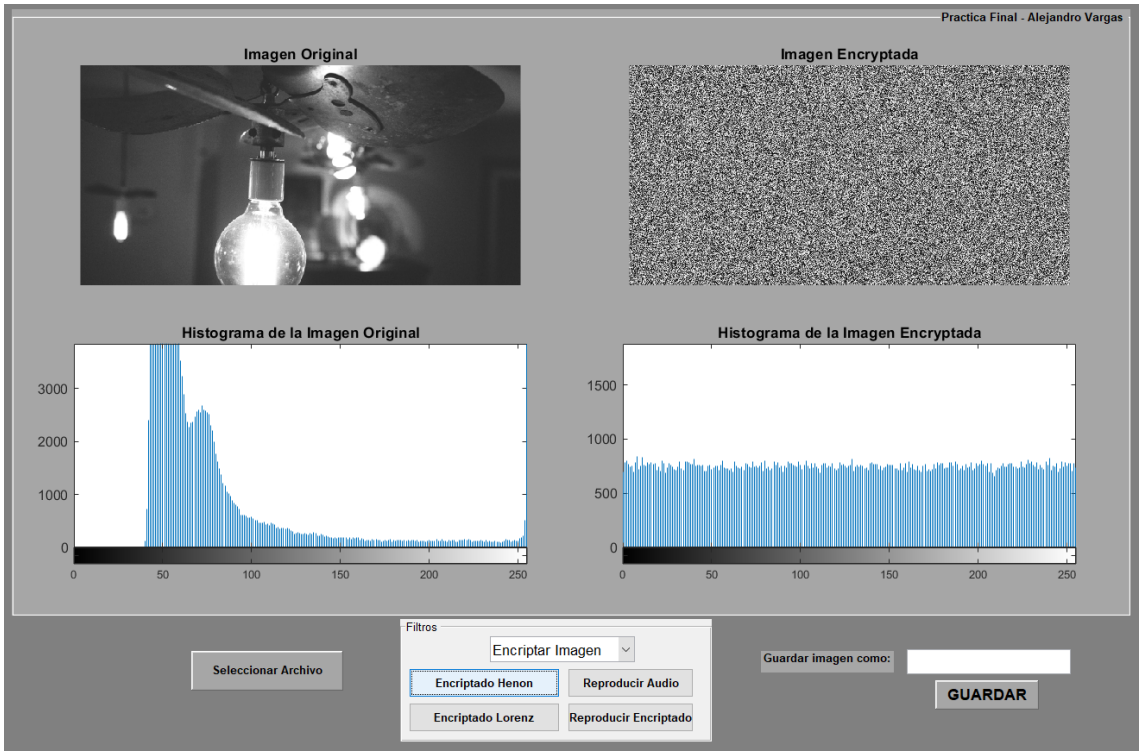


Figura 31: Interfaz gráfica del programa realizado en MATLAB para encriptar imágenes y audio.

5.1.1 Ejemplo de encriptado de imagen

Se realizan pruebas de encriptado de imágenes con el atractor de Lorenz con el programa previamente diseñado en MATLAB, los resultados que se obtienen no son los deseados, el sistema carece de seguridad y permite observar vestigios de información, por otra parte, el tiempo que tarda en encriptar las imágenes es lento y consume demasiada memoria RAM como consecuencia arroja el error Out of memory (sin memoria).

Al implementar el mapeo de Hénon a nuestro sistema, se obtienen excelentes resultados y el tiempo de procesamiento y encriptado es muy superior al de Lorenz.

La Figura 32 presenta las imágenes que fueron empleadas para llevar a cabo el proceso de encriptado desarrollado en este trabajo de tesis y que a partir de la GUI implementada se pueden observar resultados en tiempo real.

(a) (b) (c)

Figura 32: Imágenes en escala de grises implementadas para obtener los criptogramas en MATLAB. (a) Imagen 1 Cameraman 255 X 255, (b) imagen 2 Snake 235 X 350, (c) imagen 3 Lena 255 X 255.

Las Figura 33, Figura 34 y Figura 35 muestra la interfaz desarrollada y presenta los resultados obtenidos del encriptado de la imágenes originales utilizando el modelo de Lorenz. En todos los casos, dentro de la interfaz el recuadro superior izquierdo despliega las imágenes originales, el recuadro superior derecho despliega los criptogramas obtenidos que en estos casos se utilizó el modelo de Lorenz. Se puede observar claramente vestigios de la información original en los criptogramas obtenidos. Adicionalmente, en la GUI se despliegan en el recuadro inferior izquierdo el histograma de la imagen original y el recuadro inferior derecho despliega el histograma del criptograma respectivo.

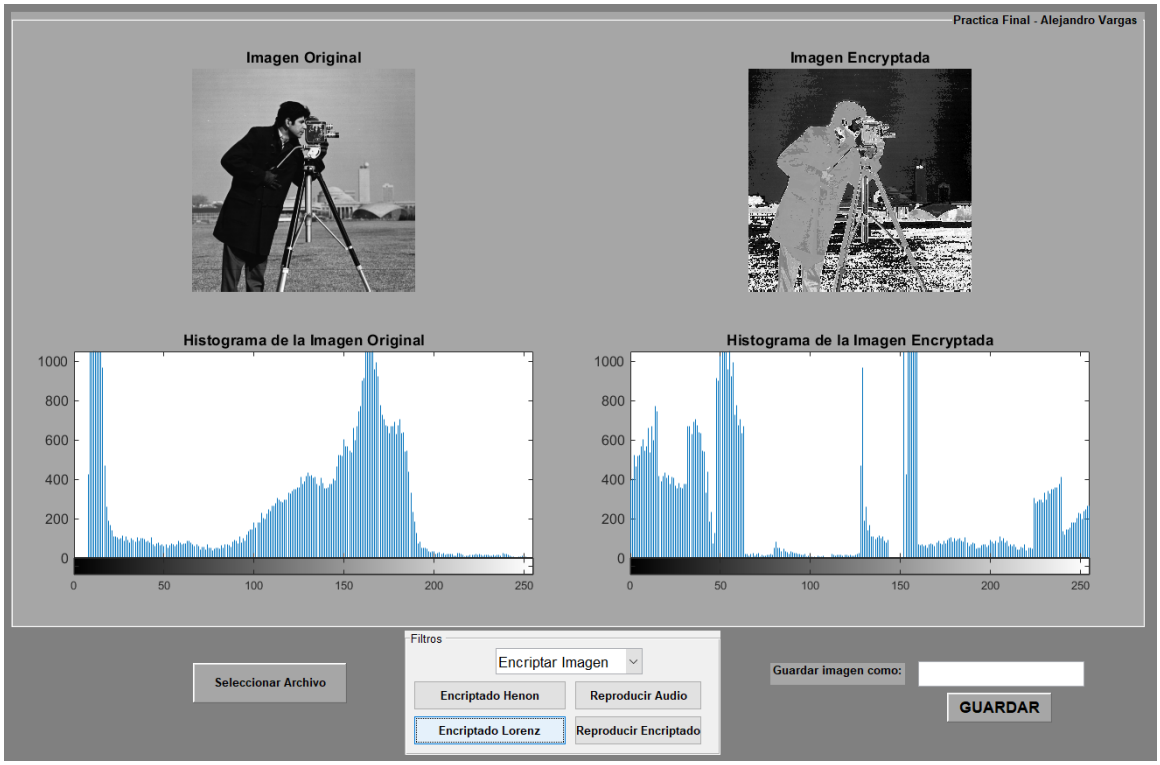


Figura 33: Imagen Cameraman encriptado con Lorenz, criptograma e histogramas de seguridad.

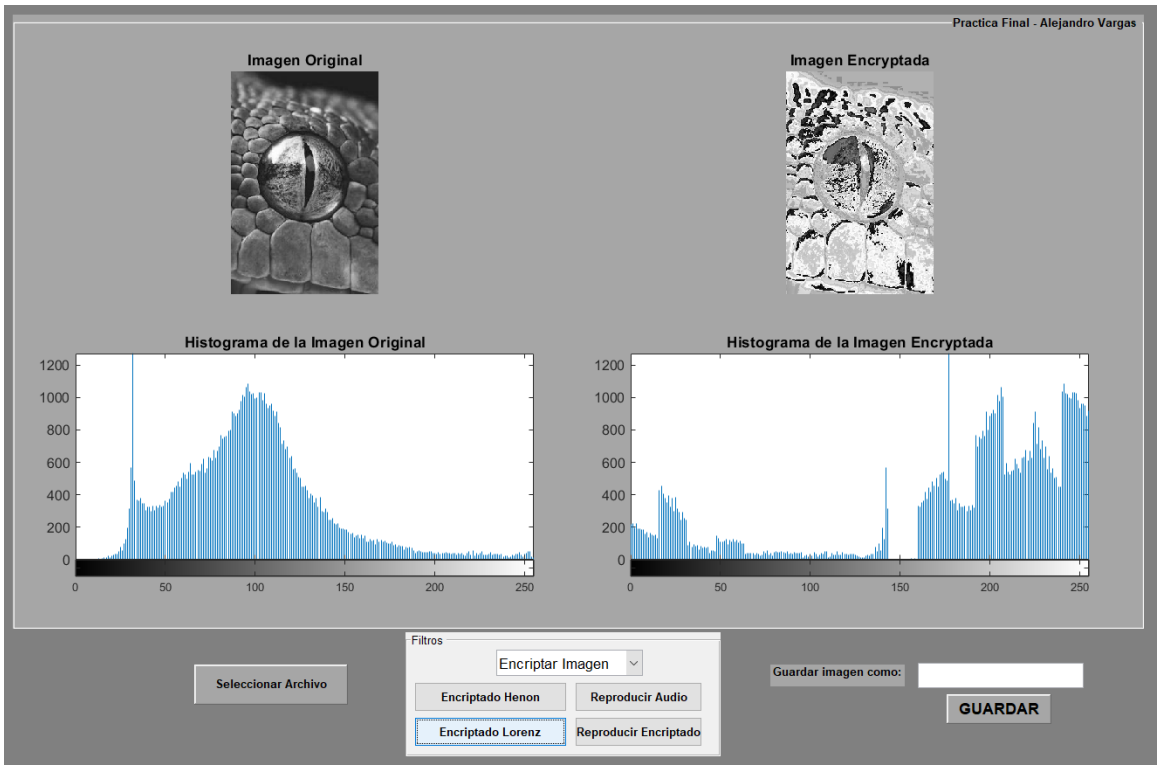


Figura 34: Imagen Snake encriptado con Lorenz, criptograma e histogramas de seguridad.

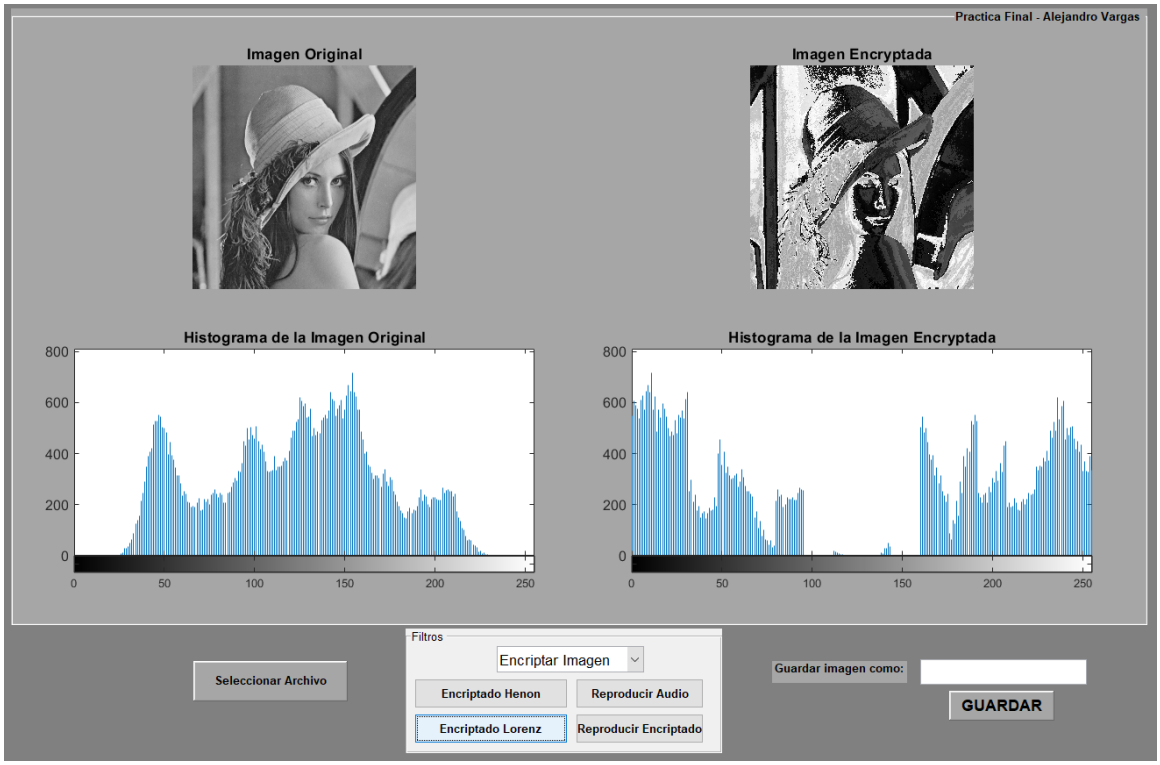


Figura 35: Imagen Lena encriptado con Lorenz, criptograma e histogramas de seguridad.

De manera análoga en las Figura 36, Figura 37, Figura 38 se presentan los resultados obtenidos empleando para el proceso de encriptado el mapeo de Hénon.

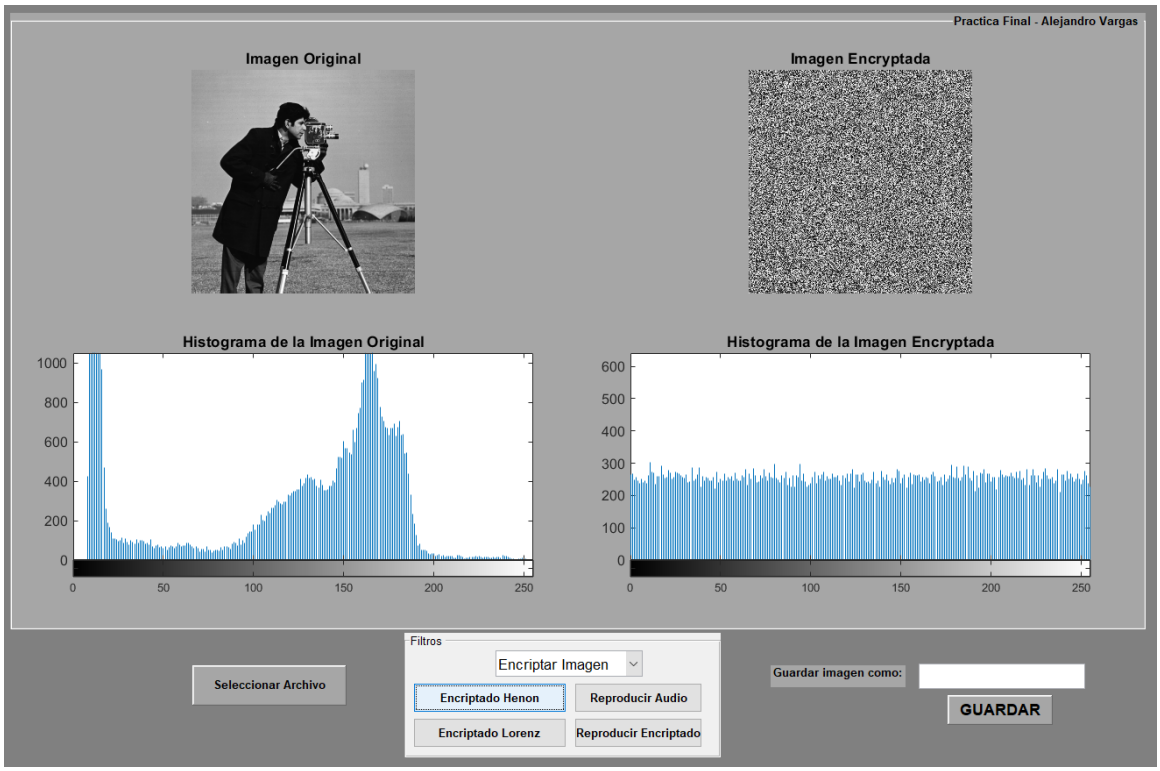


Figura 36: Imagen Cameraman encriptado con Hénon, criptograma e histogramas de seguridad.

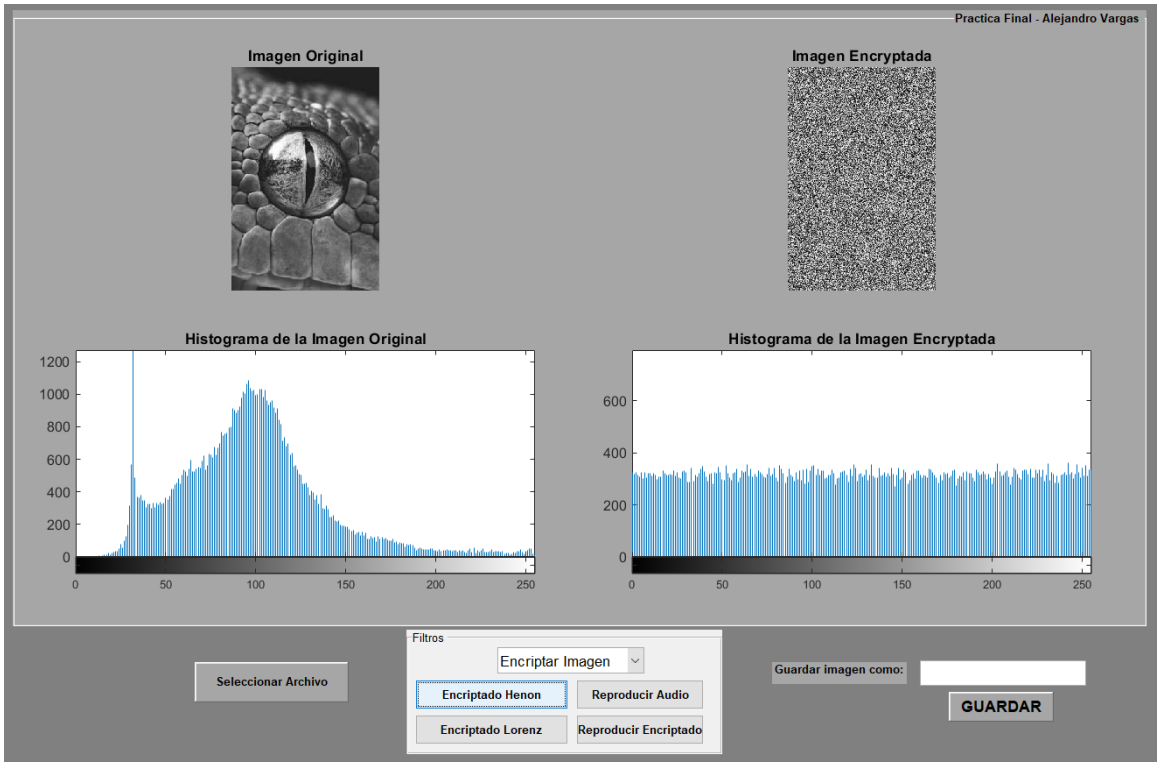


Figura 37: Imagen Snake encriptado con Hénon, criptograma e histogramas de seguridad.

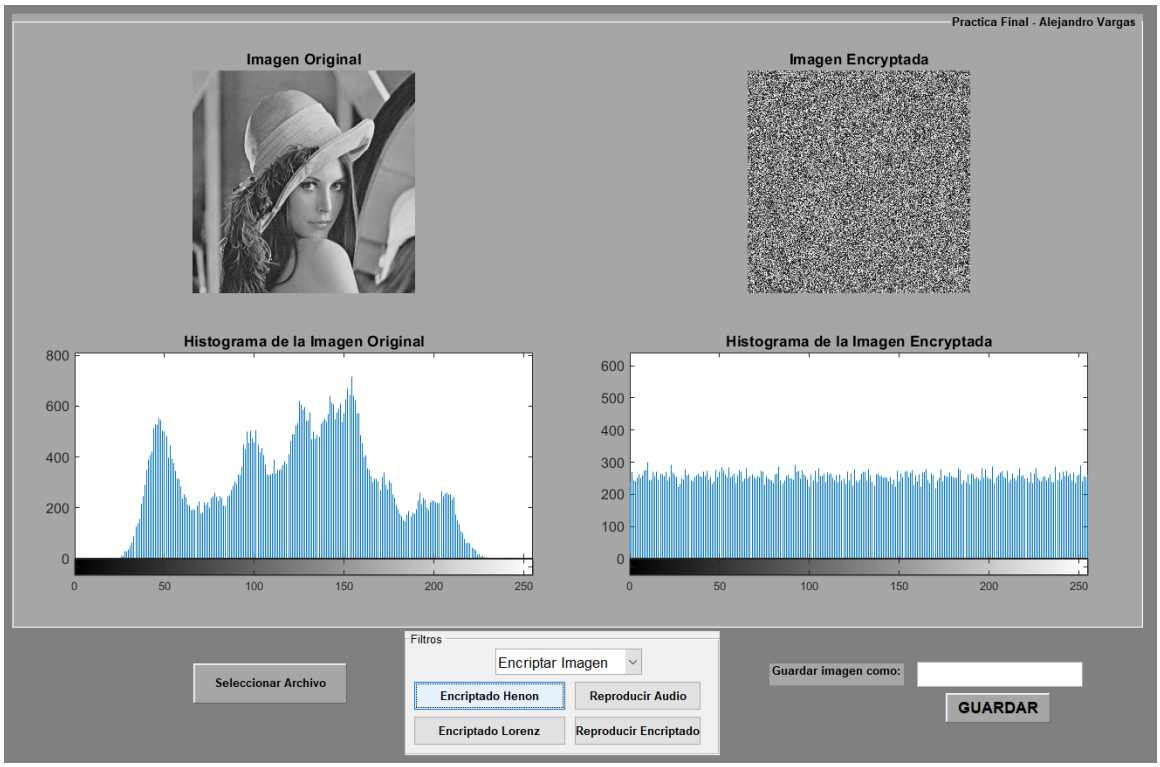


Figura 38: Imagen Lena encriptado con Hénon, criptograma e histogramas de seguridad.

Empleando el mapeo de Hénon, se puede observar en todos los casos mejores resultados en los criptogramas obtenidos, ya que no presentan vestigios de información de las imágenes originales, como se hace evidente en el comportamiento uniforme de los histogramas correspondientes.

La Tabla 9 presenta un comparativo de los tiempos de ejecución que desarrolla el algoritmo propuesto, en el proceso de encriptado bajo el modelo de Lorenz y el mapeo de Hénon, Claramente se observa que los tiempos de procesamiento empleando el modelo de Lorenz es mayor al tiempo de ejecución bajo el mapeo de Hénon.

Tabla 9: Tabla de tiempos de encriptado de imágenes MATLAB.

Algoritmo	*Imagen*	*Tiempo de encriptado en segundos*	*Tamaño imagen (KB)*
Lorenz	Cameraman	1.16	37.8
Hénon	Cameraman	0.056	37.8
Lorenz	Snake	1.93	13.9
Hénon	Snake	0.062	13.9
Lorenz	Lena	1.25	55.8
Hénon	Lena	0.050	55.8

5.1.2 Ejemplo de encriptado de señal de audio

De manera análoga que, para el caso del encriptado de imágenes descrito en la sección anterior, se encripta audio con el sistema desarrollado en este trabajo de tesis bajo el modelo de Lorenz y el mapeo de Hénon. Los resultados de encriptado obtenidos para ambos casos cumplen con lo esperado, es decir, el audio encriptado no presenta ningún vestigio de la información original. Sin embargo, nuevamente el tiempo de procesamiento del encriptado con el mapeo de Hénon es superior al obtenido con el modelo de Lorenz.

La Figura 39, Figura 40 y Figura 41, presenta los resultados que se obtienen al

encriptar con el atractor de Lorenz, la Figura 42, Figura 43, Figura 44, presentan los resultados que se obtienen al encriptar con Hénon, en el recuadro superior izquierdo se despliega el audio original, el recuadro superior derecho despliega el audio encriptado. En este caso los recuadros inferiores presentan el espectro de Fourier el cual permite evaluar la frecuencia y magnitud de los audios. El recuadro inferior izquierdo despliega la frecuencia y magnitud del audio original y el recuadro inferior derecho despliega la frecuencia del audio encriptado que en este caso es más uniforme evitando observar vestigios de la información original. Resultados similares se obtiene con ambos modelos caóticos.

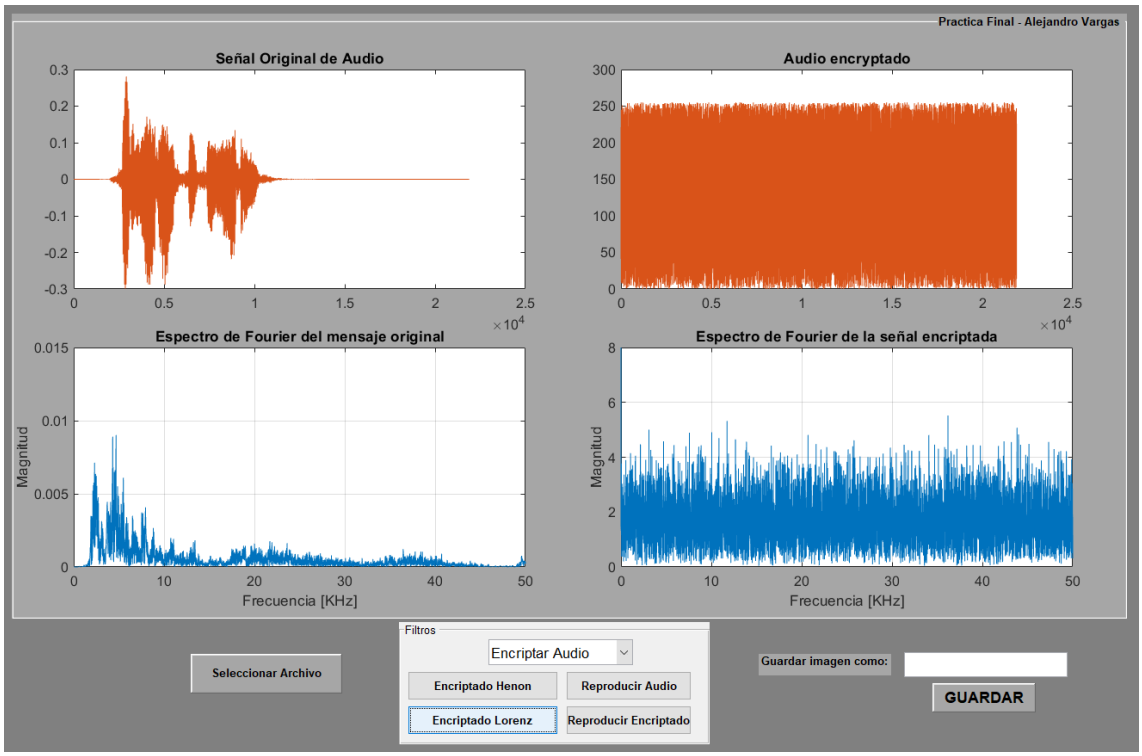


Figura 39: Audio 1 encriptado con Lorenz audio, criptograma e histogramas de seguridad.

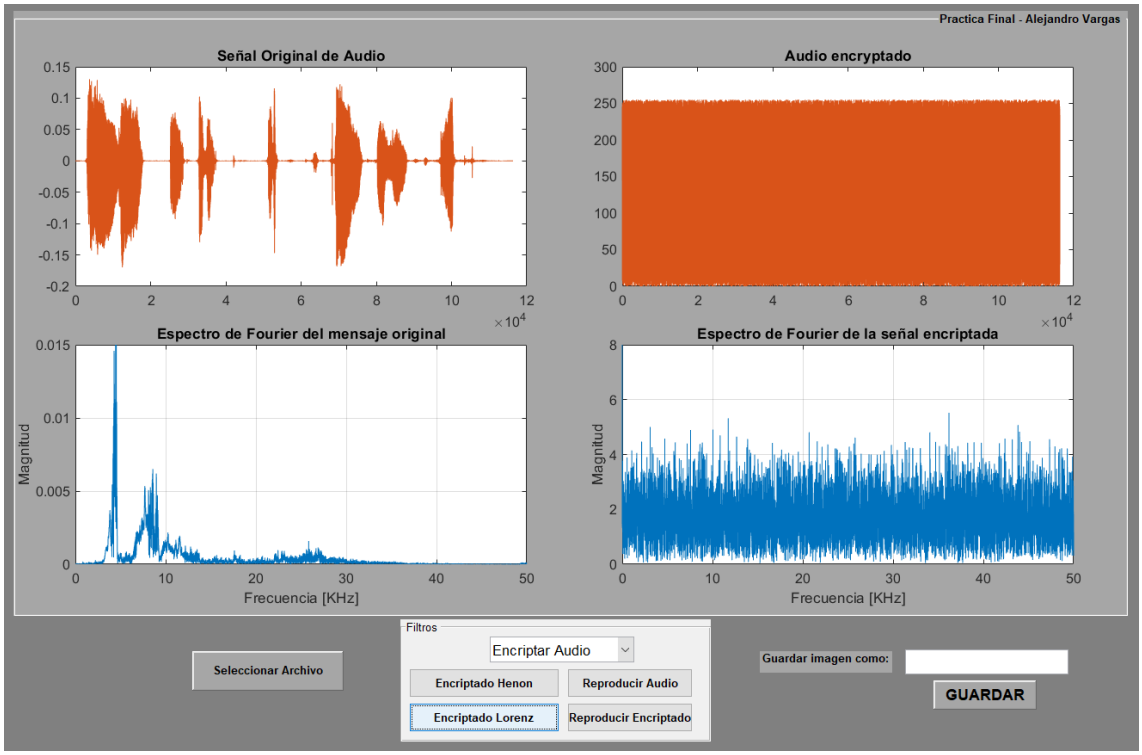


Figura 40: Audio 2 encriptado con Lorenz audio, criptograma e histogramas de seguridad.

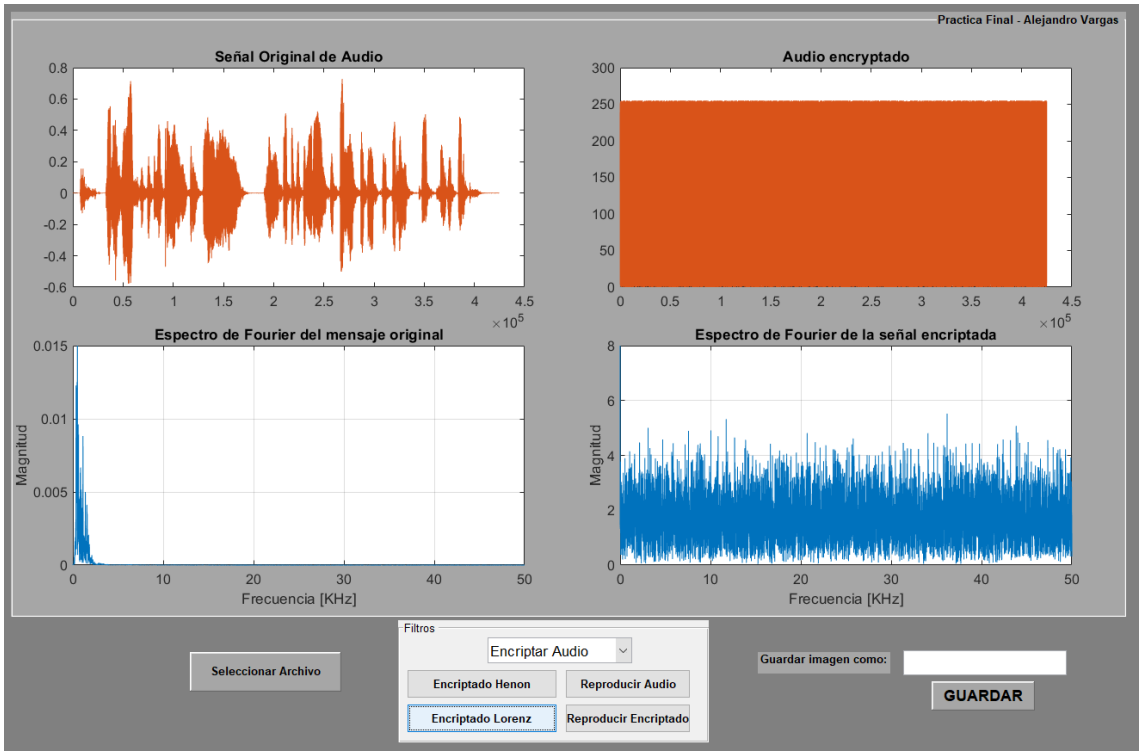


Figura 41: Audio 3 encriptado con Lorenz audio, criptograma e histogramas de seguridad.

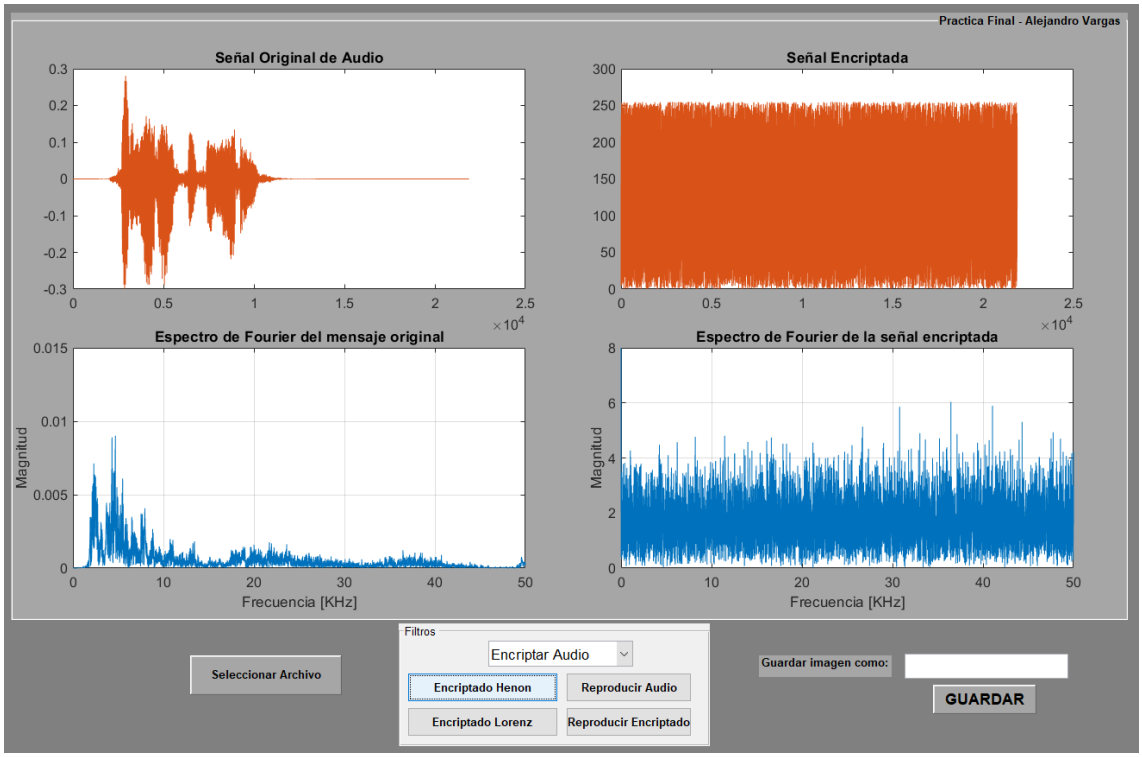


Figura 42: Audio 1 encriptado con Hénon audio, criptograma e histogramas de seguridad.

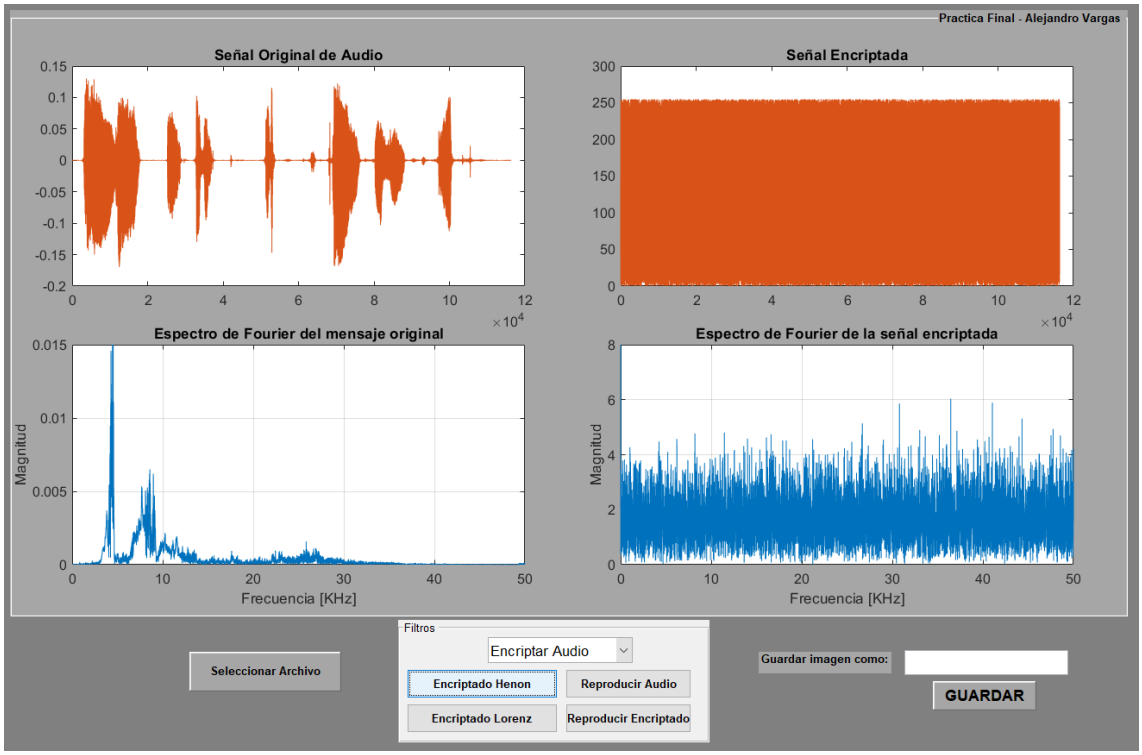


Figura 43: Audio 2 encriptado con Hénon audio, criptograma e histogramas de seguridad

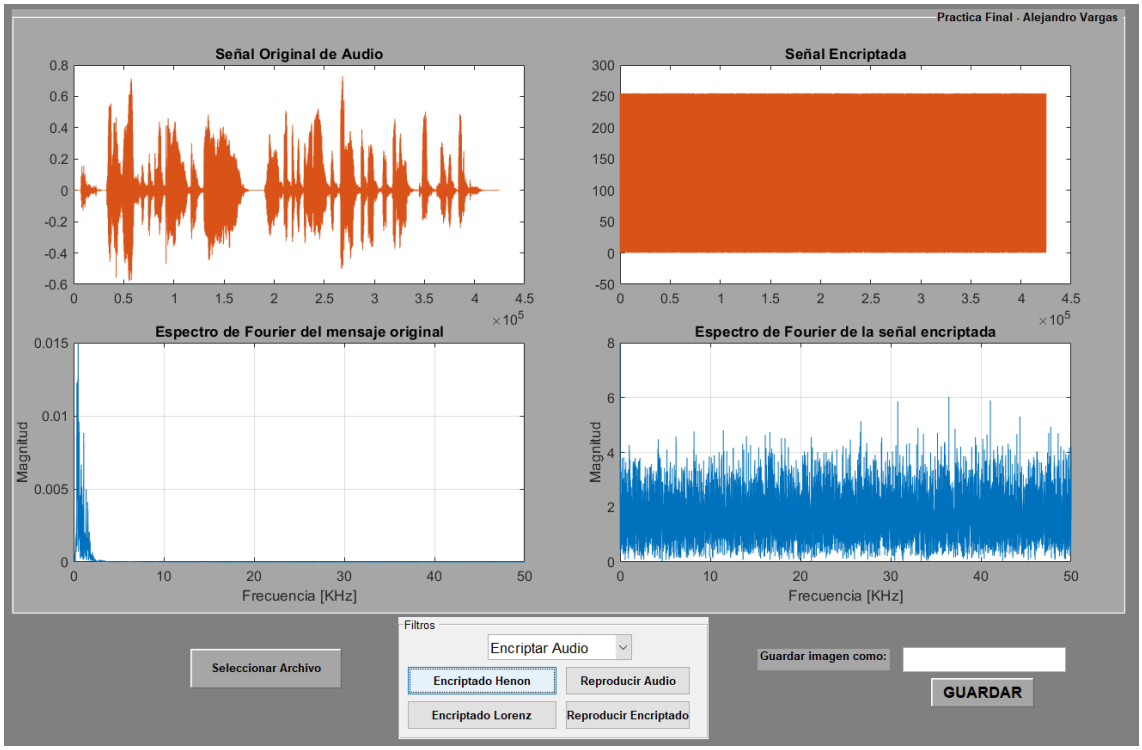


Figura 44: Audio 3 encriptado con Hénon audio, criptograma e histogramas de seguridad.

Se llevaron a cabo pruebas para tres audios diferentes con la finalidad de contrastar la efectividad y rapidez del algoritmo propuesto. La

Tabla 10 presenta los tiempos de procesamiento obtenidos en el encriptado bajo el modelo de Lorenz y bajo el mapeo de Hénon, se puede observar que como para el caso de la imágenes que el tiempo es mayor para Lorenz que para Hénon.

Tabla 10: Tabla de tiempos de encriptado de audios MATLAB.

Algoritmo	*Audio*	*Tiempo de encriptado en segundos*	*Tamaño del audio (KB)*
Lorenz	Audio 1	0.211712	85.6
Hénon	Audio 1	0.010642	85.6
Lorenz	Audio 2	0.615554	454
Hénon	Audio 2	0.043062	454
Lorenz	Audio 3	1.727890	1658.88
Hénon	Audio 3	0.165582	1658.88

5.2 Encriptado-Desencriptado imagen usando Python en CPU

La Figura 45 muestra la carátula de la interfaz gráfica desarrollada en Python para el encriptado de imágenes y ser enviadas a través de Internet bajo el protocolo de comunicación MQTT.

Figura 45: Interfaz gráfica del programa realizado en Spyder 4, para encriptar y enviar imágenes por Internet empleando el protocolo MQTT.

Ésta interfaz gráfica proporciona una interacción amigable para los usuarios, facilitando el manejo de las imágenes a encriptar y los criptogramas obtenidos, proporcionando las siguientes opciones:

1.- Para elegir la imagen deseada se selecciona el botón “Elegir imagen”, esto le permite al usuario navegar entre las distintas carpetas que tenga en su dispositivo para seleccionar la imagen que se requiera encriptar y enviar a su destino.

2.- Para modificar las condiciones iniciales y parámetros respectivos al modelo caótico se debe seleccionar ícono etiquetado como “Manual”, en caso contrario debe seleccionarse el ícono con la etiqueta de “Automático, lo que significa que existen datos previamente preestablecidos.

3.- Para el caso de encriptar una imagen, se elige el botón “Encriptar/Desencriptar”, cuando se requiera el proceso inverso, es decir, desencriptar un criptograma se elige la misma casilla.

4.- El botón de “Guardar” permite almacenar la imagen encriptada/desencriptada según corresponda con el nombre elegido por el usuario. Este paso en necesario antes de ejecutar

el análisis de seguridad y/o enviar la imagen.

5.- El botón “NPCR/UACI”, permite obtener los datos estadísticos de la imagen encriptada.

6.- El envío de las imágenes por Internet empleando el protocolo MQTT se ejecuta con el botón “Enviar”.

7.- El botón “Histogramas”, permite la visualización de los histogramas de la imagen, según correspondan.

Cabe mencionar que el programa creado es un complemento del dispositivo embebido, este puede ser usado en cualquier computador para su uso normal.

5.2.1 Transmisión de imágenes encriptadas

En esta sección se presenta la operatividad de todo el sistema de encriptado/desencriptado desarrollado en este trabajo de tesis. La interacción entre el usuario transmisor y el usuario receptor se lleva acabo atreves de la interfaz gráfica, cuando el trasmisor envía una imagen encriptada (criptograma) es recibida por el receptor en cuyo dispositivo un script programado en Python se encarga de recibirla y almacenarla, para ser desencriptada y visualizarla en el momento que se requiera.

La Figura 46 muestra la interfaz gráfica para el caso de un usuario transmisor. Despliega al mismo tiempo la imagen seleccionada (original) y la imagen encriptada (criptograma) a ser enviada al receptor. Para llevar a cabo esta operación primeramente se ejecuta el programa “GUI_Encriptar”, una vez dentro de la interfaz se selecciona la imagen con el botón “Elegir imagen”, para el caso que se presenta se despliega la imagen Lena_RGB. Para ejecutar el proceso de encriptado, es necesario proporcionar las condiciones iniciales las cuales se pueden ingresar manualmente o seleccionar las preestablecidas. En el caso que se presenta se muestra la selección de ingreso “Manual”, lo cual habilita las casillas correspondientes para ingresar los datos respectivos. Éstas condiciones, deberán ser las mismas que ingrese en su momento el usuario receptor para recuperar (desencriptar) la imagen original. Una vez ingresados los datos de las condiciones iniciales se ejecuta el encriptado de la imagen seleccionada a partir del botón

“Encriptar/Desencriptar”, para el caso que se presenta se despliega en el criptograma que se muestra.

Figura 46: GUI encriptación manual.

Una vez obtenido el criptograma se le asigna un nombre, por ejemplo Lena_RBG_Cryp y al pulsar el botón “Guardar” se despliega un mensaje como el que se muestra en la Figura 47.

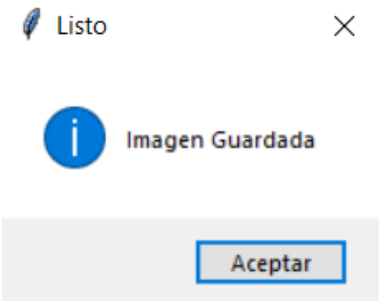


Figura 47: Imagen guardada correctamente.

Una vez que la imagen ha sido guardada está en condiciones para enviarse, para lo cual, al presionar el botón “Enviar” se despliega un mensaje de que la imagen ha sido enviada como el que se muestra en la Figura 48.

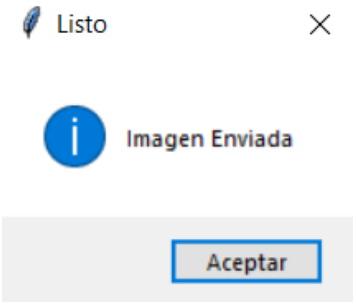


Figura 48: Imagen enviada con éxito.

La Figura 49 muestra a manera de ejemplo y para el caso de un usuario receptor el tipo de mensajes que aparece en la consola de comandos, quien registra los pasos que se van ejecutando y despliega información relevante el usuario, tal como: Tamaño de imagen seleccionada, el tipo de proceso que se ejecuta encriptado/desencriptado, el tiempo que el sistema tarda en encriptar/desencriptar el cual depende de cada dispositivo ya sea sistema embebido o computador personal. Cuando se envía el criptograma, el sistema se conecta automáticamente al servidor MQTT mediante Internet e introduce las credenciales para lograr una conexión exitosa, se despliega también el tiempo que tarda en ser enviada la imagen.

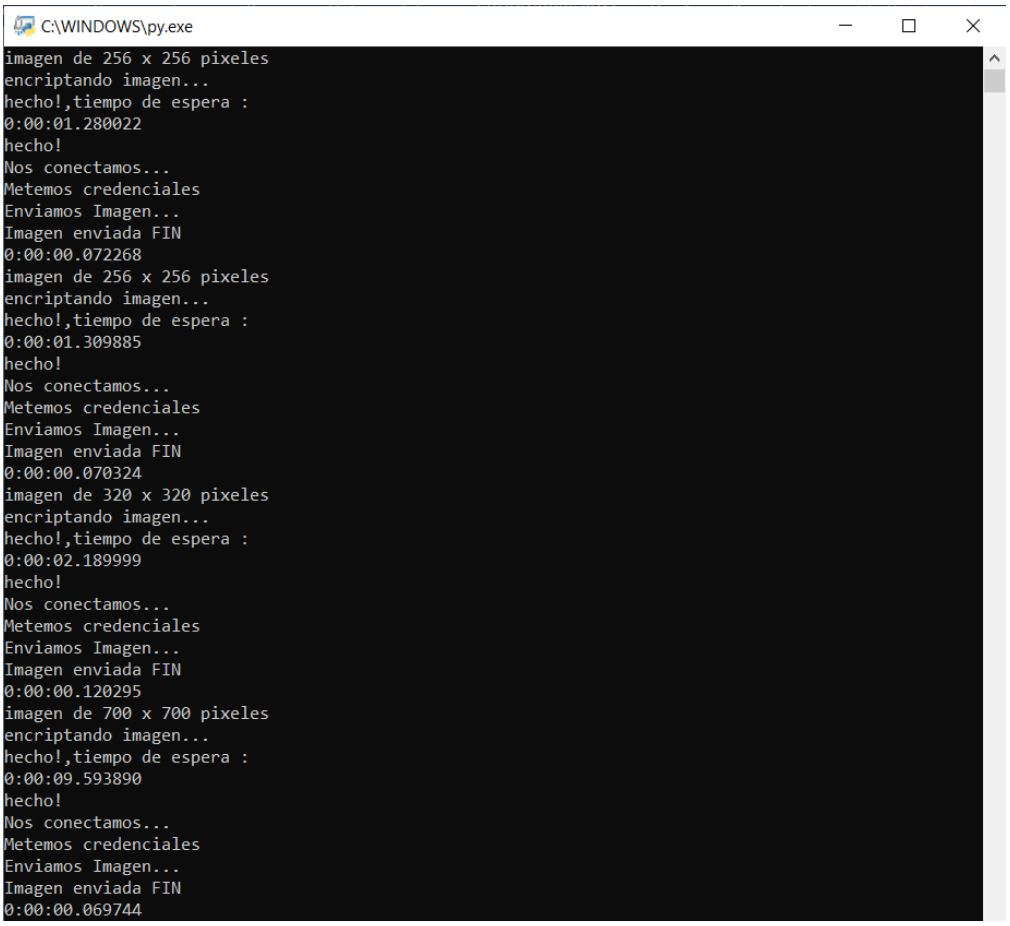


Figura 49: Registro de actividades por consola emisora.

5.2.2 Realizando pruebas de seguridad a imágenes digitales

La interfaz desarrollada (ver Figura 45) permite evaluar el nivel de seguridad ante ataques diferenciales. Para llevar a cabo este proceso, se deben obtener dos criptogramas de la misma imagen original con una variación de entre 1×10^{-10} a 1×10^{-15} en uno de los valores de las condiciones iniciales. Acorde a la configuración desarrollada en este trabajo las imágenes se deben guardar con el nombre "1" y "2" respectivamente, de lo contrario despliega un error tal como el que se muestra en la Figura 50.

Figura 50: Mensaje de error "se requieren dos imágenes para las pruebas NPCR/UACI".

Una vez que las imágenes han sido apropiadamente guardadas se selecciona el botón NPCR/UACI mostrando en pantalla los resultados respectivos de esta prueba. Es conocido que los valores deseados para NPCR es de 100% y para UACI es del 34%, estos valores se despliegan en pantalla como los que se muestra en la Figura 51. Particularmente se observa que los resultados obtenidos son muy cercanos a los valores deseados, validando la fiabilidad y robustez del sistema criptográfico desarrollado en este trabajo de tesis.

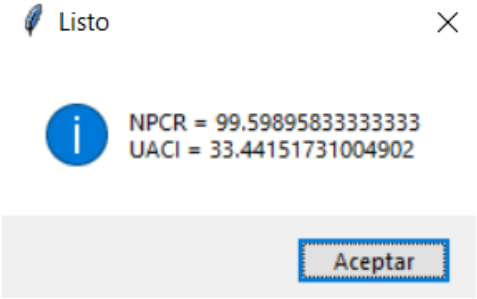


Figura 51: Valores desplegados por ataques diferenciales NPCR/UACI.

Adicionalmente, la interfaz desarrollada permite obtener los histogramas de la imagen original como el de la imagen encriptada. La Figura 52 muestra los histogramas

respectivos y en sus niveles RGB. Se puede apreciar que para el caso de la imagen encriptada el histograma presenta una distribución tipo “ruido”, lo que en significa que no permite ver vestigios de la información original. Esto valida que el sistema criptográfico desarrollado es robusto y proporciona un buen nivel de seguridad. Estos resultados se obtienen al seleccionar el botón “Histograma” de la interfaz.

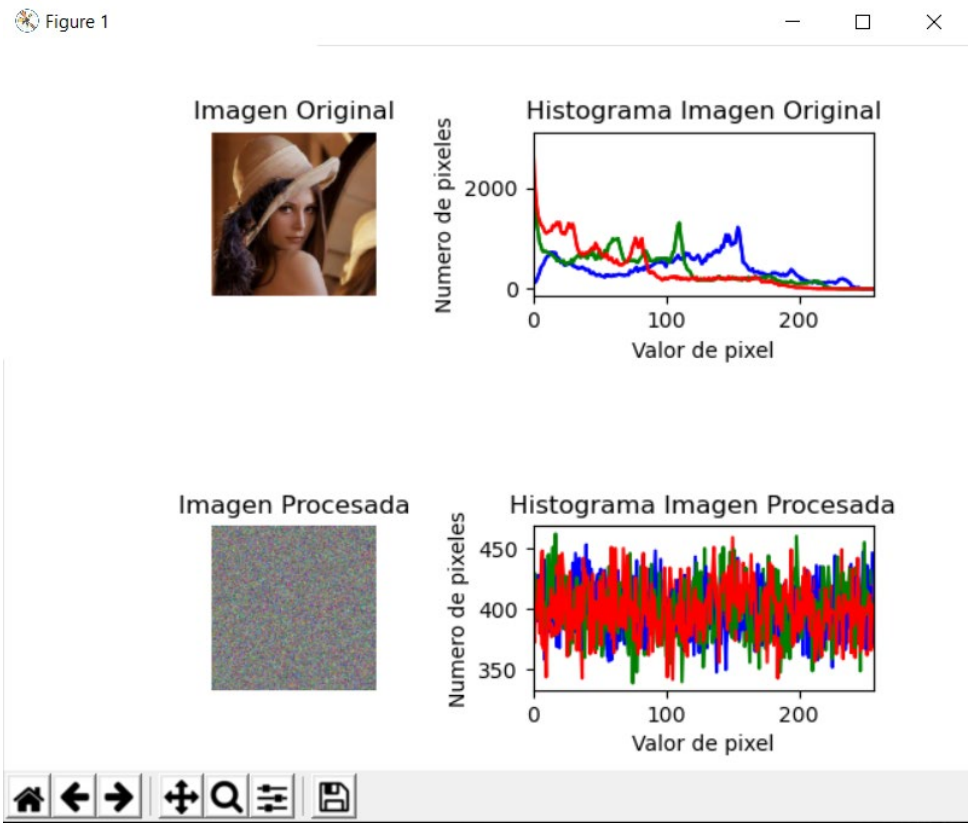


Figura 52: Histogramas de imagen sometida a encriptación.

5.2.3 Recepción de imágenes digitales

Para el caso del usuario receptor, es necesario un script que ejecute el proceso de recepción de imágenes. Al ejecutarlo se subscribe automáticamente al tópico previamente establecido, el programa se ejecuta en la consola de comandos y está a la espera de recepción de datos. La Figura 53 muestra el tipo de información que se despliega en la consola de comandos: RUN indica la conexión con el servidor MQTT subscrito al tópico deseado, al recibir la información se reciben los datos de la imagen encriptada, ésta se despliega y se guarda con el nombre “Recibido_fecha_1” cuya fecha corresponde a la fecha recepción. La Figura 54 muestra la imagen encriptada recibida junto a la imagen recibida ya desencriptada. En la consola de comandos se indica las dimensiones de la imagen así como el tiempo que tardo en realizar dicha operación (ver Figura 53). La

imagen recuperada se almacena con el nombre “Rec_Des1” (ver Figura 54) y se despliega para su visualización. La Figura 55 muestra a manera de ejemplo, el criptograma recibido y la imagen original recuperada para el caso de Lena_RGB.

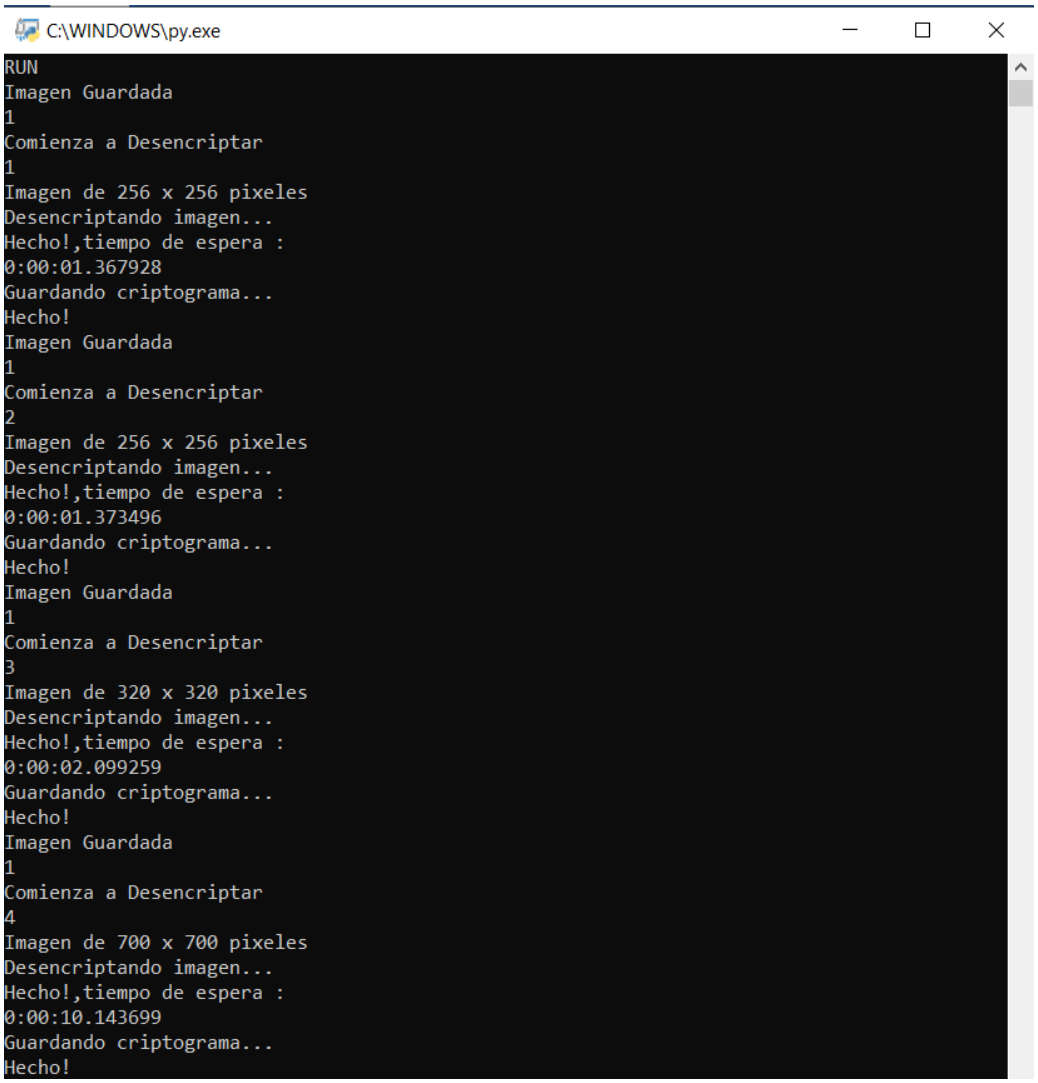


Figura 53 Registro de actividades por consola receptora.

Figura 54: Imágenes recibidas y guardadas por el script receptor.

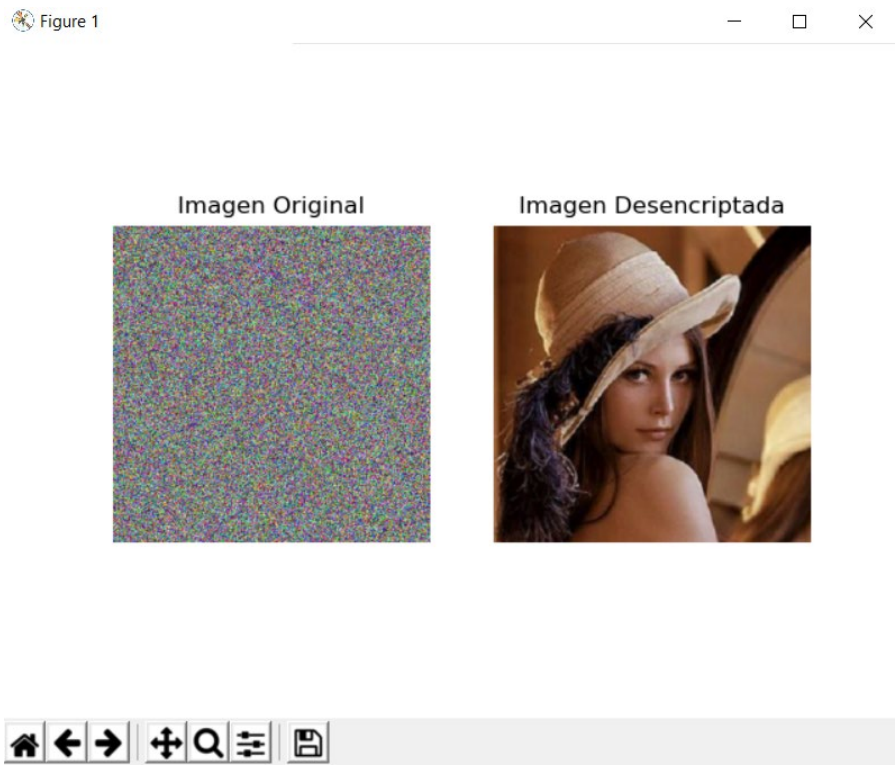


Figura 55: Imagen original e imagen descifrada desplegadas para su visualización.

La Figura 56 muestra a manera de ejemplo datos crudos que pudiera recibir la consola de Windows sin ningún tratamiento o procesamiento. El script permite procesar los datos recibidos y dar forma en consecuencia al criptograma enviado.

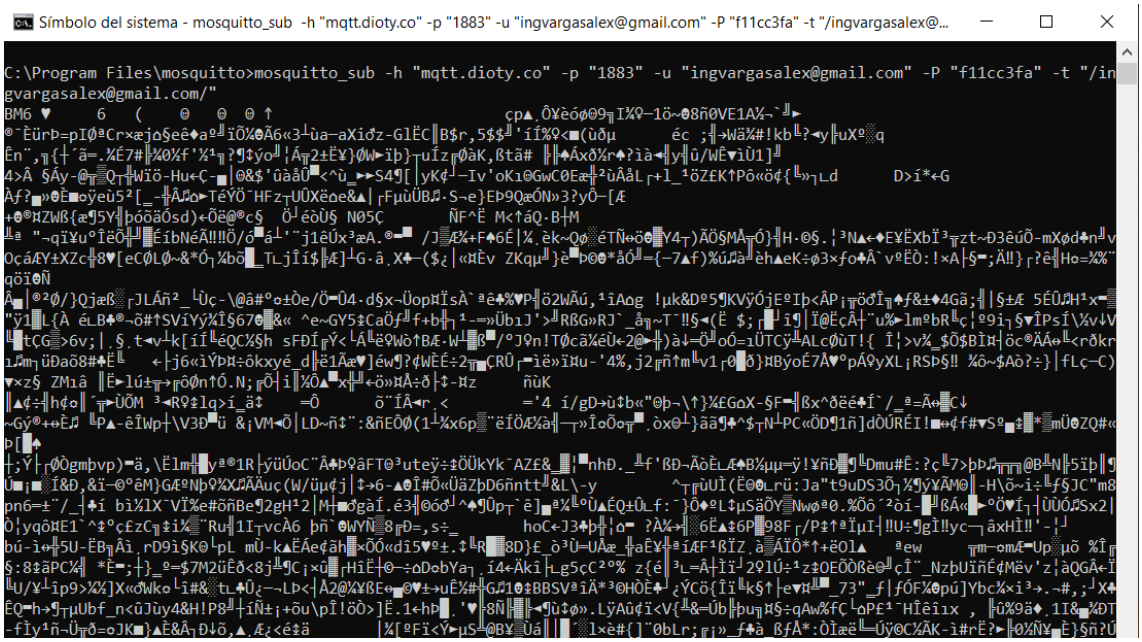


Figura 56: Suscripción al tópico receptor de imagen desde Windows 10.

La Figura 57 muestra la suscripción con el software Spyder 4 al tópico que se encarga de recibir la información respectiva. Se puede observar que los datos recibidos son caracteres de código hexadecimal, esta información recibida son los datos crudos que son enviados por la interfaz del usuario transmisor, la interfaz convierte la imagen a *bytes* y la envía en formato de texto, por su parte el script en el entorno del usuario receptor se encarga de recibir la cadena de texto y convertirla nuevamente a imagen. Por lo que, sin el script un intruso solo vería un texto sin posibilidad de observar la imagen enviada.

```
In [1]: runfile('C:/Users/admin/Documents/Maestria/Tercer Semestre/IoT/Python recibir Imagenes/
receive_Image.py', wdir='C:/Users/admin/Documents/Maestria/Tercer Semestre/IoT/Python recibir Imagenes')
Guardado
b'BM6\x00\x03\x00\x00\x00\x00\x006\x00\x00\x00(\x00\x00\x00\x00\x01\x00\x00\x00\x01\x00\x00\x00\x01\x00\x18\x00\x00\
x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x87p\x1e\xf7\xe2\xbe
\x8a\xa2\x9b\x019\xbbI\xf3\x0c\xc41\x94~\x028\xa40VE1A\xab\xaa`\xbc\x10\n\xa9\xee\xd4\x81r\xe8=pI\x9d\xa6Cr\x9e\
x91j\x7f\x15e\x88\x04a\xa7\xbc\x8b\xe5\xac\x02\xc76\xae3\xc1\x97a\xc4aX\x07i\x0bz\xf0Gl\xd3C\xbaB$r,5$$
\xbc\'\xa1\xd6%\x0c<\xfe(\x97\xd0\xe6\t\x82c ;\xb9\x1aW\x84\xf3#!kb\xc8?\x11y\xccuX\xa7\xb0q\n\x99|
\xbd\x9d\xca\xad\xe6>\xd1\x8b\x06\xb0\x19.\xb5\x02\xec:M\xcb\x16A\xb2,\xf5\xac^U\xc7$-
\x89lJ\xda\x1d6\xee\x8e\xf8h\xff\xda\xd2\x88\xe5\xc5ZM\xcdty$
\xd80\xf1\x1f1\x97\xb8\x14\xdf\x17\xd4\xe0\x85\xc7U\xd3\x0b\xef=\xa7\xefULR\xf7U\xee[\r\xd2n\xf9,
\xbb{\xc5\xef\xc6\xcd.\xf3\x907#\xcc\xf30\xabf\'\xab\xfb\xbb?\x14\x12\xeco\xbc\xdd\xb5\xcb2\xf1\xd3\xbe}
\x9dW\x10\x8b\xe7}\xc2u\xd6z\xc9\x9d\x85K,\xe1t\xc6#\xff\xcc\xcc\x06\xb5x\xd0\xabr\x06?
\x8d\x85\x11\xb9y\xb9\x96/W\xd2\x1f\x8d\xeb1]\xbc\xff\n4>\xb6\x00\x15\xb5y-@\xcb\xb1Q\xc2\xceW\x8b\x94-
Hu\x1b\x80-\xdc\xb3\x01&$
\'\x96\x85\x86\xea\xdf\x1e\x08<^\x97\xf2\x10\x10S4N\x08\xf4[\xb3yK\xbd\xd9\xc4Iv\'oK\xd5\x01GwC0E\x91\xce\xfd\x9
7\xb6\x86L\xda+l_\xfb\x94Z\x9cK\x18P\x93\xae\x94\xbd{\xc8\xaf\xbf\x1cd\tD>\xa1*\x1bG\n\xb7\x9f?
\xdc\xaf\x02\xd4\xfe\x0f\x98e\x975\xfd[\xf2-\xce\xb6\x0e\x7f\x10T\x82\xed\x99\xeeHFz\xc2U\xeaX\x89\x7fe&\x1e|
\xdaF\xe6\x97\x9aB\x0e\xfaS\xaae}E\xe89Q\x91\xe0N\xaf3?y\xe2\xc4[\x92\n+
\x02\xa9\xcfZW\xe1{\x91\x145Y\xb9\xe7\xa2\xe4\x84\xe0sd)\x1b\xe5\x89@\xa9c\xf5
\x99\xd9\x82\x95\xeb\xf5\xffN05\x80\t\xa5F^\xd3\xffM<\x18\xa0Q\xfaB\xc5M\nH\x19\xd6\x00\xb5\xc0q\x8e\x8c9E*\xd58
jX\xd5\xa9x\xc1\x00w\xf6\xb6\x1b.\xd1\x1e\xda\xdf\x08\xc7u\x9aP\xa9\r\xca\xa6
"\xaaq\x8b\xbeu\xf8\xd7\x89\xe5\xce\xbc\xb2\x90\xa1bN\x82\xc7\x13\x13\x99/
```

Figura 57: Suscripción al tópico receptor de imagen con Windows 10 con el Software Spyder 4.

La operación de transmisión, encriptado y desencriptado se ejecuta en tiempo real, los resultados obtenidos se reportan en la Tabla 11.

Tabla 11: Tabla de tiempos de encriptado y desencriptado de imágenes en CPU

Imagen enviada	*Tiempo de envió de mensaje en segundos*	*Tiempo de encriptado en segundos*	*Tiempo de desencriptado en segundos*	*Tamaño imagen (KB)*	*Tamaño del Criptograma en pixeles*
Cameraman	0.0722	1.2800	1.3679	38	256 X 256 (8 bits)
Lena	0.0703	1.3098	1.3734	55.8	256 X 256 (8bits)
Lena RGB	0.1202	2.1899	2.099	176	320 X 320 (32bits)
Mandril	0.069	9.5938	10.1436	1157.12	700 X 700 (32bits)

5.3 Tiempos de procesamiento en sistema embebido

En esta sección presentamos el funcionamiento general del sistema propuesto y los tiempos de operación del mismo. El sistema consta de dos sistemas embebidos Raspberry Pi 2, un operando como el usuario transmisor y otro ubicado en forma remota como el usuario receptor. La Figura 58 muestra la consola de comandos para el caso del usuario transmisor en la que se van registrando los datos y el procedimiento que se lleva a cabo. El usuario transmisor a través de la interfaz (ver Figura 45) selecciona la imagen respectiva, ejecuta el proceso de encriptado, la almacena y la envía.

La Figura 59 corresponde a la consola de comandos del usuario receptor, que corresponde a otro sistema embebido remoto Raspberry Pi 2. Éste requiere el script previamente programado quien se encarga de subscribirse al tópico, realizar la recepción de datos, ejecutar el proceso de desencriptado, guardar la imagen (ver Figura 60) y desplegar los archivos respectivos. La operación de transmisión y recepción se ejecuta en tiempo real, los resultados obtenidos se reportan en la Tabla 12.

La Figura 25 muestra las imágenes empleadas para realizar las pruebas de operación del sistema, éstas son de las más empleadas en la literatura: Cameraman, Lena, Mandril,

Lena RGB.

Tabla 12: Tabla de tiempos de encriptado y desencriptado de imágenes en Raspberry Pi.

Imagen enviada	*Tiempo de envió de mensaje en segundos*	*Tiempo de encriptado en segundos*	*Tiempo de desencriptado en segundos*	*Tamaño imagen (KB)*	*Tamaño del Criptograma*
Cameraman	0.1328	23.4914	21.6180	38	256*256 pixeles (8 bits)
Lena	0.08538	23.9362	23.4174	55.8	256*256 pixeles (8bits)
Lena RGB	0.0774	37.0623	37.5057	176	320*320 pixeles (32bits)
Mandril	0.123089	172.0219	177.2115	1157.12	700*700 pixeles (32bits)

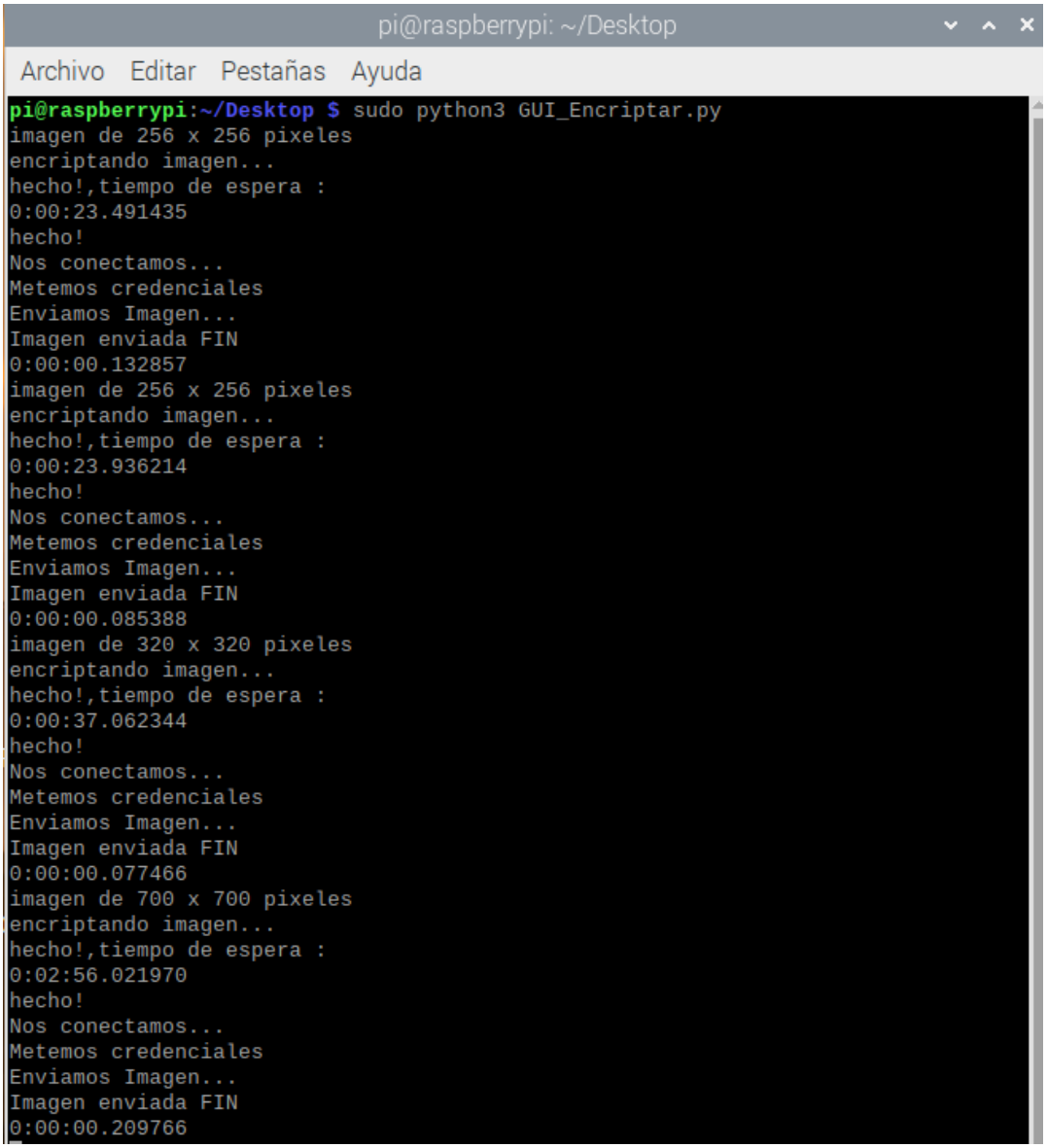


Figura 58: Registro de actividades por consola emisora en Raspberry Pi.

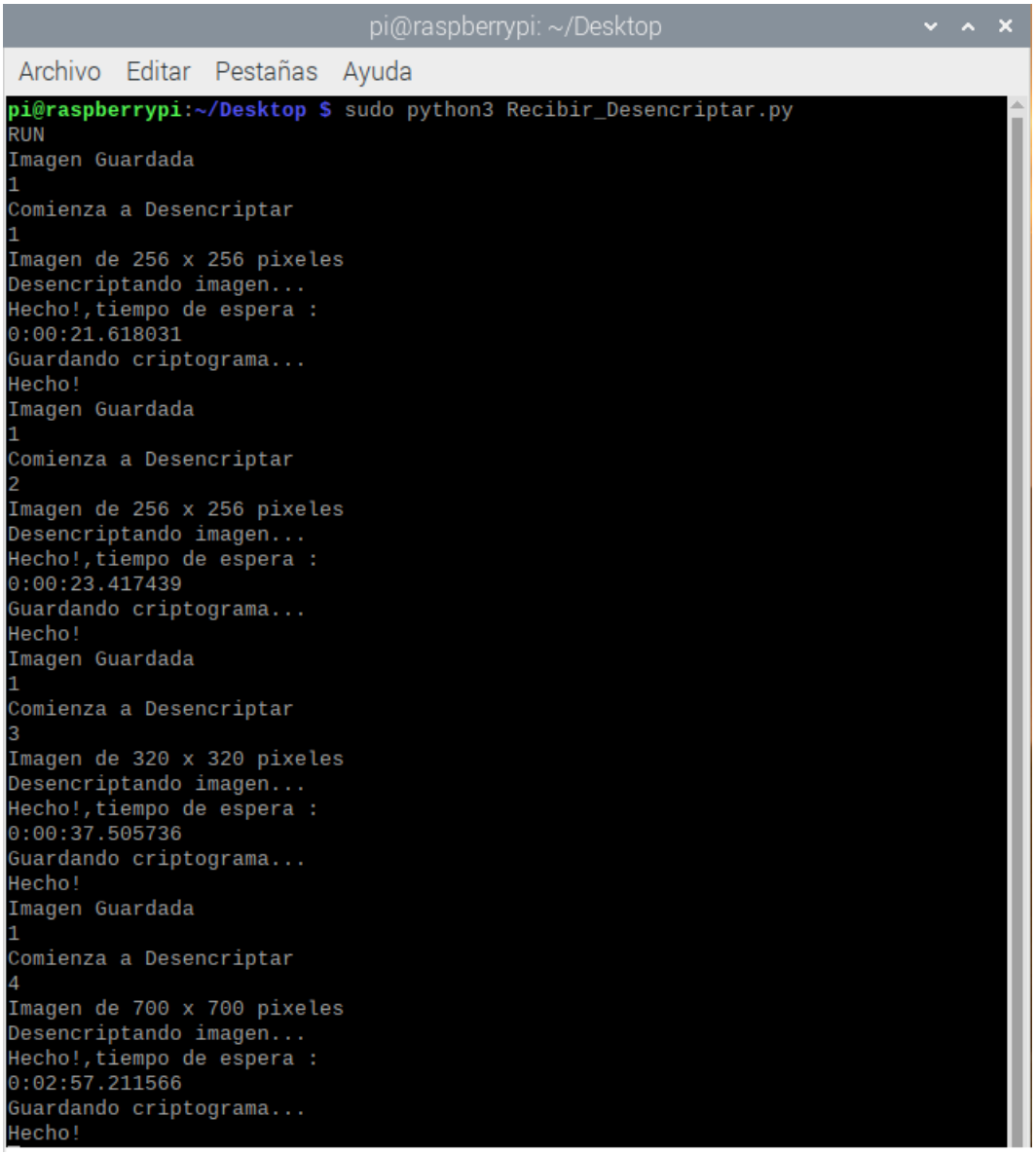


Figura 59: Consola de resultados del script desarrollado en Python para la recepción y descifrado de imágenes.

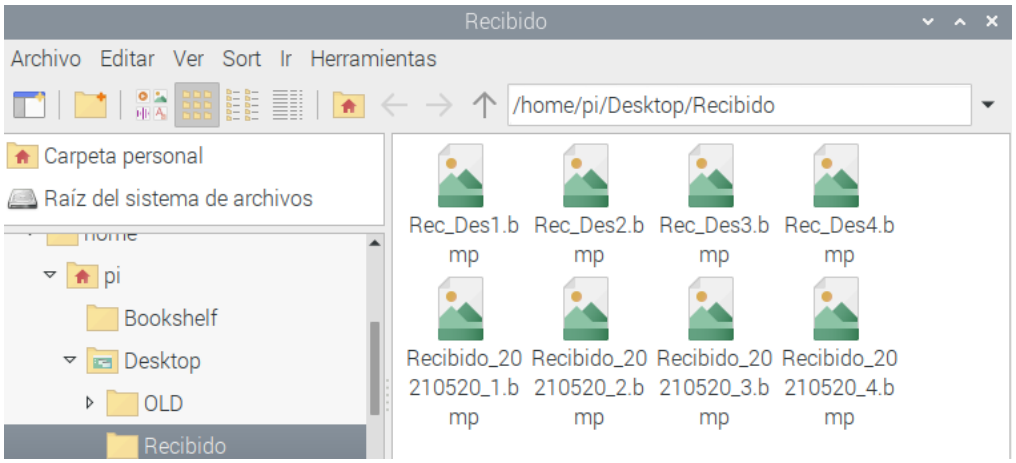


Figura 60: Imágenes recibidas y desencriptadas guardadas automáticamente por el script de Python.

La Figura 61 muestra los resultados experimentales en el entorno del usuario receptor, las imágenes fueron desplegadas mediante el script y se muestran los criptogramas recibidos y la correspondiente imagen recuperada.

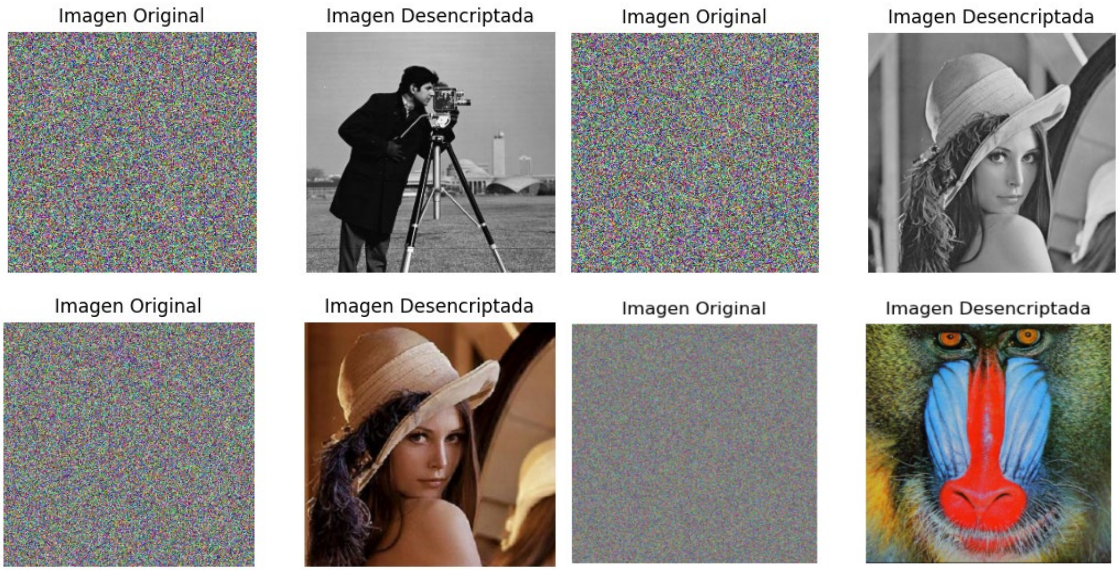


Figura 61: Imágenes desplegadas al usuario mediante el script receptor de imágenes.

5.4 Pruebas en hardware de computadora personal

En este trabajo de tesis se lleva acabo también un proceso de encriptado de imágenes de mayor tamaño con la finalidad de validar el sistema criptográfico desarrollado. Esto se ejecuta en dos distintas CPU personales, cuyas características se presentan en la Tabla 13 y Tabla 14.

Tabla 13: Características del computador HP para el encriptado de imágenes grandes.

Recurso	*Característica*
Modelo	HP ZBook 15 G4
CPU	Intel(R) Core(TM) i7-7820HQ CPU @ 2.90GHz
GPU	NVIDIA Quadro M2200
RAM	16 GB DDR4
Resolución	1920 x 1080 píxeles
Almacenamiento	Disco duro 256 GB estado sólido
Sistema Operativo	Windows 10 Pro 64 bits

Tabla 14: Características del computador MacBook Pro para el encriptado de imágenes grandes.

Recurso	*Característica*
Modelo	MacBook Pro Retina 13", principios de 2015
CPU	Intel(R) Core(TM) i5 Dual Core CPU de 2.90GHz
GPU	Intel Iris Graphics 6100 1536 MB
RAM	8 GB @ 1867 MHz DDR3
Resolución	2560 x 1600 píxeles
Almacenamiento	Disco duro 500 GB estado sólido
Sistema Operativo	macOS High Sierra Versión 10.13.6

La Tabla 15 reporta los tiempos de encriptado y desencriptado que se obtienen al utilizar las imágenes de la Figura 26 Se observa mayores tiempos de ejecución, sin embargo el sistema en su conjunto opera correctamente tal como se muestra en las respectivas consolas de comandos (ver Figura 62 y ver Figura 63),) para el usuario transmisor y receptor respectivamente. La Figura 64 muestra las imágenes recibidas y las recuperadas por el usuario receptor.

Tabla 15: Tabla de tiempos de encriptado y desencriptado de imágenes grandes.

Imagen enviada	*Tiempo de envió de mensaje en segundos*	*Tiempo de encriptado en segundos*	*Tiempo de desencriptado en segundos*	*Tamaño imagen (MB)*	*Tamaño del Criptograma en pixeles*
Paisaje	0.0898	50.2795	44.0603	1.91	1600 X 2560 (24 bits)
Loto	0.0899	80.3107	78.2574	3.4	1200 X 1920 (24bits)

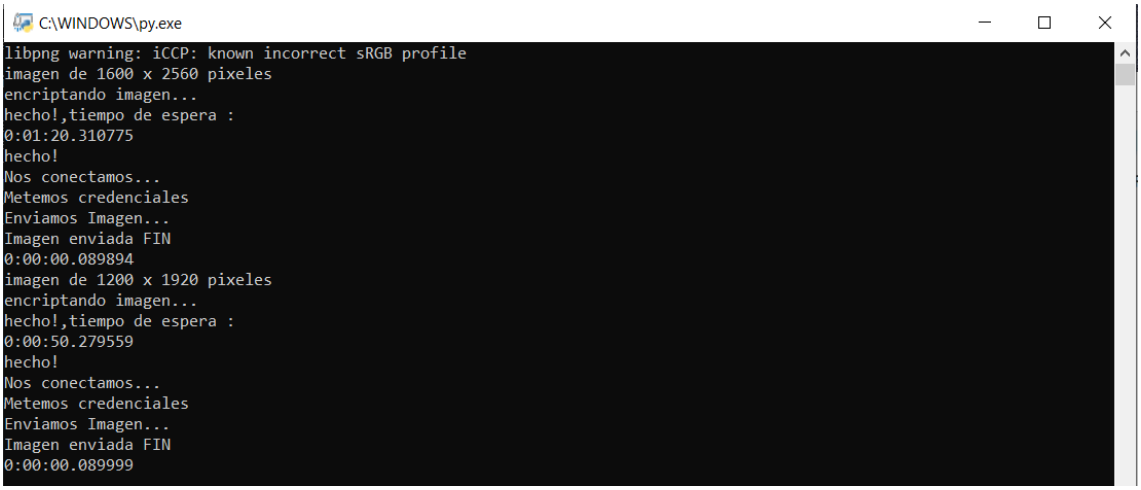


Figura 62: Consola de resultados del script creado en Python para él envió y cifrado de imágenes.

Figura 63: Consola de resultados del script creado en Python para la recepción y descifrado de imágenes.

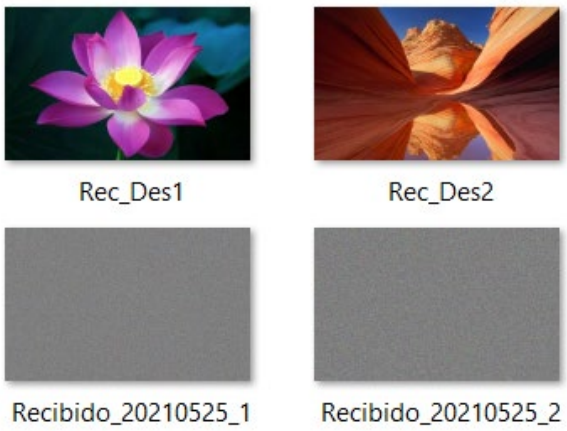


Figura 64: Imágenes recibidas y desencriptadas guardadas automáticamente.

5.5 Comparación de tiempos de procesamiento en Raspberry Pi versus laptop HP y MAC

La Tabla 16 presenta una comparación del tiempo que tarda en generarse una señal caótica de un millón de bits en cada uno de los dispositivos explorados en este trabajo de tesis. Tal es el caso de la Raspberry Pi 2 cuyo hardware fue descrito en la Tabla 1, CPU HP ZBook 15 G4 descrito en Tabla 13 y Laptop MAC con CPU 2.4 GHz Intel Core i5, descrito en Tabla 14.

Tabla 16: Tabla de tiempos de generación de caos

Número de experimento	*Raspberry Pi 2 ARM 900 MHz Raspbian [segundos]*	*CPU 2.90 GHz Intel Core i7 Windows 10 [segundos]*	*CPU 2.9 GHz Intel Core i5 MAC OS X 10.13.6 [segundos]*
1	0.9642	0.0500	0.1046
2	0.8997	0.0599	0.1069
3	0.8967	0.0599	0.1053
4	0.9090	0.0600	0.1045
5	0.8954	0.0600	0.1071
6	0.9496	0.0599	0.1047
7	0.9754	0.0599	0.1089
8	0.9149	0.0600	0.1056
9	0.9394	0.0599	0.1047
10	0.8754	0.0599	0.1058
Promedio	**0.9219**	**0.0589**	**0.1058**

Como podemos observar en la Tabla 16, los resultados son distintos y varían dependiendo de los recursos de cada dispositivo, entre mejores características de hardware tenga el CPU o sistema embebido los tiempos mejoran. Sin embargo, los resultados criptográficos obtenidos, así como las pruebas de seguridad realizadas, son igual de efectivas con cualquiera de los tres hardware empleados, la gran diferencia es el costo/beneficio según corresponda, por ejemplo, un sistema embebido tiene un costo de 80.00 Dlls contra un CPU de laptop de 1,000.00 Dlls o más.

Capítulo VI

6. Conclusiones y resumen

En este trabajo de tesis se propone un sistema criptográfico multiplataforma, usando el mapa caótico de Hénon y el protocolo MQTT para transmisión segura de imágenes sobre internet, implementado en Raspberry Pi 2 modelo B, como sistema embebido con comunicación vía WiFi, para transmisión segura de imágenes digitales sobre Internet.

Se cumple satisfactoriamente con el objetivo general logrando diseñar una interface encriptador-desencriptador de imágenes digitales para transmisión segura sobre internet, al mismo tiempo cumpliendo con los objetivos específicos, implementando el protocolo MQTT, estudiar e implementar un algoritmo caótico eficiente en un sistema embebido, se realizan pruebas de seguridad exitosamente garantizando la robustez del sistema.

El potencial del sistema se logra a partir de los distintos elementos que lo componen:

El mapeo de Hénon fue el que se propuso como algoritmo para el encriptado debido a los grandes resultados vistos a lo largo del desarrollo del sistema. Programación realizada en Python, el cual es multiplataforma en hardware y software. Además de su forma tan rápida y amigable de trabajar facilita a la hora de realizar las operaciones minimizando líneas de código.

El protocolo de comunicación que se decidió utilizar es el protocolo MQTT, debido a las ventajas que nos proporciona como la conexión en esquema M2M, bajo consumo de energía, bajo ancho de banda y la capacidad de trabajar con múltiple suscripción permitiendo ser escalable el sistema.

Los resultados de análisis de seguridad, mostraron que el sistema criptográfico propuesto, es robusto contra diferentes tipos de ataques, tal como: histogramas estadísticos, entropía de información, espacio de clave, correlación de pixeles adyacentes y ataques diferenciales como NPCR y UAC.

6.1 Trabajos a futuro

Como trabajo futuro se propone:

1. Mejorar la interfaz gráfica para que sea más atractiva visualmente para el usuario y agregar un ajuste de resolución para distintos monitores.

2. Se propone utilizar otros tipos de datos para cifrar como lo pueden ser video, texto, audio, etc.

3. Agregar una cámara para la toma de fotografías en tiempo real y mandarlas encriptadas por el mismo medio.

4. Proponer nuevos algoritmos criptográficos debido a la vida finita de los mismos, para siempre mantenernos a la vanguardia y manteniendo la seguridad.

5. Emplear la llave asimétrica para evitar el riesgo de robo de claves.

6. Trabajar los algoritmos en otros hardware de sistemas embebidos, tal como: FPGA, ASIC, SoC, GPUs, JetSon nano, etc.

Bibliografía

[1] ITU Newsroom, “ITU standards to integrate Internet of Things in Smart Cities.”, 2015. https://www.itu.int/net/pressoffice/press_releases/2015/22.aspx

[2] K. Ashton, “That ‘Internet of Things’ Thing.”, *(2009).*, [En línea]. Disponible en: http://www.rfidjournal.com/articles/view?4986

[3] B. Sanou, *Global Cybersecurity Index (GCI).* Geneva, Switzeland: International Telecommunications Union., 2017. [En línea]. Disponible en: https://www.itu.int/dms_pub/itu-d/opb/str/D-STR-GCI.01-2017-PDF-E.pdf

[4] E. Inzunza-Gonzalez y C. Cruz-Hernandez, “Double Hyperchaotic Encryption for Security in Biometric Systems”, p. 14, 2013.

[5] H. G. C. Ferreira y R. T. de Sousa Junior, “Security analysis of a proposed Internet of things middleware”, *Clust. Comput.*, vol. 20, núm. 1, pp. 651–660, mar. 2017, doi: 10.1007/s10586-017-0729-3.

[6] S. Li, G. Chen, y X. Mou, “ON THE DYNAMICAL DEGRADATION OF DIGITAL PIECEWISE LINEAR CHAOTIC MAPS”, *Int. J. Bifurc. Chaos*, vol. 15, núm. 10, pp. 3119–3151, oct. 2005, doi: 10.1142/S0218127405014052.

[7] Y. Deng, H. Hu, N. Xiong, W. Xiong, y L. Liu, “A general hybrid model for chaos robust synchronization and degradation reduction”, *Inf. Sci.*, vol. 305, pp. 146–164, jun. 2015, doi: 10.1016/j.ins.2015.01.028.

[8] Y. Xu, B. Bao, y Q. Xu, “Grid-scroll hyperchaotic system based on microcontroller digital hardware implementation”, *Wuli XuebaoActa Phys. Sin.*, vol. 59, pp. 5959–5965, sep. 2010, doi: 10.7498/aps.59.5959.

[9] E. Tlelo-Cuautle, J. J. Rangel-Magdaleno, A. D. Pano-Azucena, P. J. Obeso-Rodelo, y J. C. Nunez-Perez, “FPGA realization of multi-scroll chaotic oscillators”, *Commun. Nonlinear Sci. Numer. Simul.*, vol. 27, núm. 1–3, pp. 66–80, oct. 2015, doi: 10.1016/j.cnsns.2015.03.003.

[10] G. Heidari-Bateni y C. D. McGillem, “A chaotic direct-sequence spread-spectrum communication system”, *IEEE Trans. Commun.*, vol. 42, núm. 2/3/4, pp. 1524–1527, feb. 1994, doi: 10.1109/TCOMM.1994.582834.

[11] S. Sadoudi, C. Tanougast, M. S. Azzaz, y A. Dandache, “Design and FPGA implementation of a wireless hyperchaotic communication system for secure real-time image transmission”, *EURASIP J. Image Video Process.*, vol. 2013, núm. 1, p. 43, dic. 2013, doi: 10.1186/1687-5281-2013-43.

[12] E. Tlelo-Cuautle, V. H. Carbajal-Gomez, P. J. Obeso-Rodelo, J. J. Rangel-Magdaleno, y J. C. Núñez-Pérez, “FPGA realization of a chaotic communication system applied to image processing”, *Nonlinear Dyn.*, vol. 82, núm. 4, pp. 1879–1892, dic. 2015, doi: 10.1007/s11071-015-2284-x.

[13] “IEEE Std 1076-2008 (Revision of IEEE Std 1076-2002) IEEE Standard VHDL Language Reference Manual”, p. 640.

[14] Y. Luo, R. Zhou, J. Liu, Y. Cao, y X. Ding, “A parallel image encryption algorithm based on the piecewise linear chaotic map and hyper-chaotic map”, *Nonlinear Dyn.*, vol. 93, núm. 3, pp. 1165–1181, ago. 2018, doi: 10.1007/s11071-018-4251-9.

[15] W.-K. Lee, R. C.-W. Phan, W.-S. Yap, y B.-M. Goi, “SPRING: a novel parallel chaos-based image encryption scheme”, *Nonlinear Dyn.*, vol. 92, núm. 2, pp. 575–593, abr. 2018, doi: 10.1007/s11071-018-4076-6.

[16] H.-M. Yuan, Y. Liu, T. Lin, T. Hu, y L.-H. Gong, “A new parallel image cryptosystem based on 5D hyper-chaotic system”, *Signal Process. Image Commun.*, vol. 52, pp. 87–96, mar. 2017, doi: 10.1016/j.image.2017.01.002.

[17] D. Burak, “Parallelization of an Encryption Algorithm Based on a Spatiotemporal Chaotic System and a Chaotic Neural Network”, *Int. Conf. Comput. Sci. ICCS 2015*, vol. 51, pp. 2888–2892, ene. 2015, doi: 10.1016/j.procs.2015.05.453.

[18] O. Mirzaei, M. Yaghoobi, y H. Irani, “A new image encryption method: parallel sub-image encryption with hyper chaos”, *Nonlinear Dyn.*, vol. 67, núm. 1, pp. 557–566, ene. 2012, doi: 10.1007/s11071-011-0006-6.
[19] Q. Zhou, K. Wong, X. Liao, T. Xiang, y. Hu, “Parallel image encryption algorithm based on discretized chaotic map”, *Chaos Solitons Fractals*, vol. 38, núm. 4, pp. 1081–1092, nov. 2008, doi: 10.1016/j.chaos.2007.01.034.
[20] G. M. Amdahl, “Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities”. AFIPS Conference Proceedings, 1967. [En línea]. Disponible en: https://www3.cs.stonybrook.edu/~rezaul/Spring-2012/CSE613/reading/Amdahl-1967.pdf
[21] G. Alvarez y S. Li, “SOME BASIC CRYPTOGRAPHIC REQUIREMENTS FOR CHAOS-BASED CRYPTOSYSTEMS”, *Int. J. Bifurc. Chaos*, vol. 16, núm. 08, pp. 2129–2151, ago. 2006, doi: 10.1142/S0218127406015970.
[22] Andrea Sgarro, *CÓDIGOS SECRETOS*, 1st Edition. Ediciones Piramide, 1990.
[23] M. J. López Barrientos, “Criptografía”. Universidad Nacional Autónoma de México, 2009.
[24] B. Schneier, “Applied Cryptography”. Second Edition. John Wiley & Sons, 1996.
[25] *LA GUERRA DE LAS GALIAS*, Ediciones Orbis, S. A. l Iberia, S. A., 1982. [En línea]. Disponible en: http://www.webs.hesperides.es/Cesar_files/Julio%20Cesar%20-%20La%20guerra%20de%20las%20Galias.pdf
[26] E. Ott, *Chaos in Dynamical Systems*. Estados Unidos de America: CAMBRIDGE UNIVERSITY PRESS, 1993.
[27] R. Matthews, “ON THE DERIVATION OF A ‘CHAOTIC’ ENCRYPTION ALGORITHM”, *Cryptologia*, vol. 13, núm. 1, pp. 29–42, ene. 1989, doi: 10.1080/0161-118991863745.
[28] R. A. Mollin, *An Introduction to Cryptography*, 2nd Edition. Boca Raton London New York: Chapman and Hall/CRC, 2006. [En línea]. Disponible en: https://mrajacse.files.wordpress.com/2012/01/an-introduction-to-cryptography.pdf
[29] A. Sgarro, *CÓDIGOS SECRETOS*, 1st Edition. Ediciones Piramide, 1990. [En línea]. Disponible en: http://www.librosmaravillosos.com/codigossecretos/pdf/Codigos%20secretos%20-%20Andrea%20Sgarro.pdf
[30] S. Fernández, “LA CRIPTOGRAFÍA CLÁSICA”. Revista sigma, 2004.
[31] F. Zamora Arellano, “DISEÑO DE UN PROTOTIPO ENCRIPTADOR-DESENCRIPTADOR HIPERCAÓTICO DE AUDIO CON COMUNICACIÓN WIFI”, *Univ. Auton. BAJA Calif.*, p. 66, 2013.
[32] D. Kahn, *Seizing the Enigma: The Race to Break the German U-Boats Codes, 1939-1943*. Houghton Mifflin Harcourt, 1991.
[33] F. H. Hinsley y A. Stripp, *Codebreakers: The Inside Story of Bletchley Park*. Oxford University Press, 1993.
[34] J. R. Soler Fuensanta, “Una introducción a la Criptografía Clásica.”
[35] C. E. Shannon, “Communication Theory of Secrecy Systems*”, *Bell Syst. Tech. J.*, vol. 28, núm. 4, pp. 656–715, oct. 1949, doi: 10.1002/j.1538-7305.1949.tb00928.x.
[36] W. Diffie y M. E. Hellman, “New Directions in Cryptography”. 1976. [En línea]. Disponible en: https://ee.stanford.edu/~hellman/publications/24.pdf
[37] O. C. Moran Torres, R. A. P. Moreno, y L. B. Quintero Cadena, “‘Criptografía moderna’ (Criptosistemas de clave pública y privada)”. 2003.
[38] E. Inzunza-Gonzalez, C. Cruz-Hernandez, R. M. Lopez-Gutierrez, E. E. Garcia-Guerrero, L. Cardoza-Avendaño, y H. Serrano-Guerrero, “Software to Encrypt Messages Using Public-Key Cryptography”. World Academy of Science, Engineering and Technology 54 2009.
[39] T. MAMANI TTITO, “MODELO DE SISTEMA CRIPTOGRÁFICO DE SEGURIDAD PARA LAS REDES DE COMUNICACIONES EN LA REGIÓN PUNO – 2012”, UNIVERSIDAD NACIONAL DEL ALTIPLANO, Puno - Peru, 2014.
[40] R. Piol, “La Teoría del Caos: ¿Ultima oportunidad para la interpretación del Mercado Inmobiliario?” SOITAVE.
[41] E. N. Lorenz, *La esencia del caos: un campo de conocimiento que se ha convertido en parte*

importante del mundo que nos rodea, 1a. edición. España: Debate, 1995.
[42] J. Gleick y V. Press, *Chaos: Making a New Science*. Viking, 1987.
[43] L. M. Resler, "Edward N Lorenz's 1963 paper, 'Deterministic nonperiodic flow', in Journal of the Atmospheric Sciences, Vol 20, pages 130–141: Its history and relevance to physical geography", *Prog. Phys. Geogr. Earth Environ.*, vol. 40, núm. 1, pp. 175–180, ene. 2016, doi: 10.1177/0309133315623099.
[44] C. Leon O., I. M., K. L., y E. K., "Chaos Synchronization in Chua's Circuit", *EECS Dep. Univ. Calif. Berkeley*, 1992, [En línea]. Disponible en: https://www2.eecs.berkeley.edu/Pubs/TechRpts/1992/2179.html
[45] O. E. Rössler, "An equation for continuous chaos", *Phys. Lett. A*, vol. 57, núm. 5, pp. 397–398, jul. 1976, doi: 10.1016/0375-9601(76)90101-8.
[46] M. Hénon, "A two-dimensional mapping with a strange attractor", *Commun. Math. Phys.*, vol. 50, núm. 1, pp. 69–77, feb. 1976, doi: 10.1007/BF01608556.
[47] J. Valvano, *Introducción a los sistemas de microcomputadoras embebidos: SIMULACIÓN DE MOTOROLA G811 Y G812*, SI Edition. Mexico City, Mexico: Cengage Learning Editores S.A. de C.V., 2004.
[48] J. D. M. Frías, *Sistemas Empotrados en Tiempo Real. Una introducción basada en FreeRTOS y en el microcontrolador ColdFire MCF5282*. José Daniel Muñoz Frías, 2009.
[49] C. Quintáns, J. M. Lago, L. M. Menéndez, y E. Mandado, "Plataforma hardware para el autoaprendizaje de las FPGA y sus aplicaciones", *Actas VII Congr. Tecnol. Apl. Enseñ. Electrónica Madr.*, 2006.
[50] "ESP32 Datasheet, Espressif Systems." mar. 06, 2017.
[51] "Raspberry Pi 2 Model B", 2016. https://www.raspberrypi.org/products/raspberry-pi-2-model-b/
[52] R. Karunamoorthi *et al.*, "Design and development of IoT based home computerization using Raspberry pi", *Mater. Today Proc.*, dic. 2020, doi: 10.1016/j.matpr.2020.10.673.
[53] M. Aboubakar, M. Kellil, y P. Roux, "A review of IoT network management: Current status and perspectives", *J. King Saud Univ. - Comput. Inf. Sci.*, abr. 2021, doi: 10.1016/j.jksuci.2021.03.006.
[54] Jithin, "Protocolos de red comunes", *Interserver*, 2016. https://www.interserver.net/tips/kb/common-network-protocols-ports/
[55] "Protocolos de red", *Rouse, Margaret. TechTarget. (s/a).* https://searchnetworking.techtarget.com/definition/protocol
[56] H. V. Nguyen y L. Lo Iacono, "Chapter 10 - RESTful IoT Authentication Protocols", en *Mobile Security and Privacy*, M. H. Au y K.-K. R. Choo, Eds. Boston: Syngress, 2017, pp. 217–234. doi: 10.1016/B978-0-12-804629-6.00010-9.
[57] M. V. Masdani y D. Darlis, "A comprehensive study on MQTT as a low power protocol for Internet of things application", *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 434, p. 012274, dic. 2018, doi: 10.1088/1757-899x/434/1/012274.
[58] M. R. Nikseresht y M. Mollamotalebi, "Providing a CoAP-based technique to get wireless sensor data via IoT gateway", *Comput. Commun.*, vol. 172, pp. 155–168, abr. 2021, doi: 10.1016/j.comcom.2021.03.026.
[59] C. Patel y N. Doshi, ""A Novel MQTT Security framework In Generic IoT Model"", *Third Int. Conf. Comput. Netw. Commun. CoCoNet19*, vol. 171, pp. 1399–1408, ene. 2020, doi: 10.1016/j.procs.2020.04.150.
[60] Z. Laaroussi, R. Morabito, y T. Taleb, "Service Provisioning in Vehicular Networks Through Edge and Cloud: An Empirical Analysis", en *2018 IEEE Conference on Standards for Communications and Networking (CSCN)*, oct. 2018, pp. 1–6. doi: 10.1109/CSCN.2018.8581855.
[61] R. Morabito, Z. Laaroussi, y J. Jiménez, "Evaluating the Performance of CoAP, MQTT, and HTTP in Vehicular Scenarios".
[62] X. Huang, "Image encryption algorithm using chaotic Chebyshev generator", *Nonlinear*

Dyn., vol. 67, núm. 4, pp. 2411–2417, mar. 2012, doi: 10.1007/s11071-011-0155-7.
[63] C. Fu, B. Lin, Y. Miao, X. Liu, y J. Chen, "A novel chaos-based bit-level permutation scheme for digital image encryption", *Opt. Commun.*, vol. 284, núm. 23, pp. 5415–5423, nov. 2011, doi: 10.1016/j.optcom.2011.08.013.
[64] L. Shubo, J. Sun, y Z. Xu, "An Improved Image Encryption Algorithm based on Chaotic System", *JCP*, vol. 4, pp. 1091–1100, nov. 2009, doi: 10.4304/jcp.4.11.1091-1100.
[65] S. Behnia, A. Akhshani, S. Ahadpour, H. Mahmodi, y A. Akhavan, "A fast chaotic encryption scheme based on piecewise nonlinear chaotic maps", *Phys. Lett. A*, vol. 366, núm. 4, pp. 391–396, jul. 2007, doi: 10.1016/j.physleta.2007.01.081.
[66] E. E. Garc, "Randomness improvement of chaotic maps for image encryption in a wireless communication scheme using PIC-microcontroller via Zigbee channels", p. 13, 2020.
[67] V. Patidar, N. Pareek, y G. Purohit, "A robust and secure chaotic standard map based pseudorandom permutation-substitution scheme for image encryption", *Opt. Commun.*, vol. 284, pp. 4331–4339, sep. 2011, doi: 10.1016/j.optcom.2011.05.028.
[68] I. Rodríguez, E. Barrera, C. Parra, y J. Posada, "Algoritmo de Encriptación de Imágenes Utilizando el Atractor Caótico de Lorenz", *Ingeniería*, vol. 22, p. 396, sep. 2017, doi: 10.14483/23448393.11976.
[69] C. E. Shannon, "A Mathematical Theory of Communication", *SIGMOBILE Mob Comput Commun Rev*, vol. 5, núm. 1, pp. 3–55, ene. 2001, doi: 10.1145/584091.584093.
[70] M. Y. Y. y D. Z. C., "A new image encryption algorithm of inputoutput feedback based on multi-chaotic system." En Applied Mechanics and Materials, Vol. 40, 2011.
[71] S. Behnia, A. Akhshani, H. Mahmodi, y A. Akhavan, "A novel algorithm for image encryption based on mixture of chaotic maps", *Chaos Solitons Fractals*, vol. 35, núm. 2, pp. 408–419, ene. 2008, doi: 10.1016/j.chaos.2006.05.011.
[72] N. Naik, "Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP", en *2017 IEEE International Systems Engineering Symposium (ISSE)*, oct. 2017, pp. 1–7. doi: 10.1109/SysEng.2017.8088251.