UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

FACULTAD DE INGENIERÍA, ARQUITECTURA Y DISEÑO,
CAMPUS ENSENADA

SISTEMA DE TELEMEDICINA SEGURO BASADO EN CRIPTOGRAFÍA CAÓTICA PARA EL MONITOREO Y PREVENCIÓN TEMPRANA DE ENFERMEDADES CARDIACAS

TESIS

que para cubrir parcialmente los requisitos necesarios para obtener el título de

INGENIERO EN COMPUTACIÓN

presenta:

Kevin Becerra Santamaría

Ensenada, Baja California, México, Marzo de 2024.

UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA
FACULTAD DE INGENIERÍA, ARQUITECTURA Y DISEÑO,
CAMPUS ENSENADA

"SISTEMA DE TELEMEDICINA SEGURO BASADO EN CRIPTOGRAFÍA CAÓTICA PARA EL MONITOREO Y PREVENCIÓN TEMPRANA DE ENFERMEDADES CARDIACAS"

TESIS

PARA CUBRIR LOS REQUISITOS NECESARIOS PARA OBTENER EL TÍTULO DE

Ingeniero en Computación

PRESENTA

Kevin Becerra Santamaría
360885

A quien el Comité de Tesis autoriza el trabajo terminal, después de haber efectuado una revisión minuciosa del mismo y de acuerdo con el Art. 19 del R.G.E.P.E.P., las y los señores profesores emiten los siguientes votos aprobatorios mediante rúbrica:

M.C. Elitania Jiménez García
DIRECTOR

Dr. Miguel Ángel Murillo Escobar
CODIRECTOR

M.C. Irma Alejandra Amaya Patrón
SINODAL

M.I. Luz Evelia López Chico
SINODAL

Dra. Rosa Martha López Gutiérrez
SINODAL

"Por la Realización Plena del Ser"

C.c.p.- Archivo
C.c.p.- Minutario

RESUMEN de la tesis de **Kevin Becerra Santamaría**, presentada como requerimiento parcial para obtener el título de INGENIERO en COMPUTACIÓN, del programa de Licenciatura de la Universidad Autónoma de Baja California. Ensenada, Baja California, México. Marzo de 2024.

SISTEMA DE TELEMEDICINA SEGURO BASADO EN CRIPTOGRAFÍA CAÓTICA PARA EL MONITOREO Y PREVENCIÓN TEMPRANA DE ENFERMEDADES CARDIACAS

Resumen aprobado por:

M.C. Elitania Jiménez García
Director de tesis

Dr. Miguel Ángel Murillo Escobar
Codirector de tesis

La presente tesis se centra en la mejora de la seguridad en la transmisión de datos biomédicos en telemedicina a través del desarrollo de un sistema compuesto por un módulo transmisor y un módulo receptor. El módulo transmisor, un microcontrolador de bajo costo, se encarga de recoger y transmitir las señales biomédicas, mientras que el módulo receptor, actuando como servidor web, recibe y procesa estos datos.

Se introduce un algoritmo de cifrado basado en la teoría del caos para proteger la transmisión de señales biomédicas, específicamente las señales de electrocardiograma (ECG). Este algoritmo aprovecha la imprevisibilidad del caos y su sensibilidad a las condiciones iniciales para generar claves de cifrado sólidas y únicas. Dichas claves son implementadas en el microcontrolador del módulo transmisor, garantizando la seguridad en la recopilación y transmisión de las señales biomédicas. Por otro lado, el servidor web, actuando como módulo receptor, utiliza el mismo algoritmo para desencriptar los datos recibidos, asegurando así la privacidad de la información médica transmitida.

Se llevan a cabo análisis exhaustivos de seguridad para evaluar la robustez tanto del algoritmo de cifrado como del sistema de transmisión en su totalidad. Estas pruebas incluyen la resistencia a ataques de fuerza bruta, la evaluación de la uniformidad de los datos cifrados para resistir ataques estadísticos y la sensibilidad a cambios mínimos en las claves de cifrado. Además, se mide la entropía de la información para confirmar el grado de caos en los datos cifrados, demostrando así un alto nivel de seguridad. Estos análisis validan que el sistema propuesto ofrece una transmisión segura de señales biomédicas en el contexto de la telemedicina.

Palabras clave: criptografía caótica, análisis de seguridad, señales biomédicas, sistema embebido, mapa logístico, transmisión segura y teoría del caos.

Abstract of the thesis presented by **Kevin Becerra Santamaría**, as a partial requirement to obtain the title in COMPUTER ENGINEERING, of the program of the Autonomous University of Baja California. Ensenada, Baja California, Mexico. March, 2024.

SECURE TELEMEDICINE SYSTEM BASED ON CHAOTIC CRYPTOGRAPHY FOR MONITORING AND EARLY PREVENTION OF HEART DISEASES

Abstract approved by:

M.C. Elitania Jiménez García
Thesis director

Dr. Miguel Ángel Murillo Escobar
Thesis codirector

This thesis addresses the need to enhance security in the transmission of biomedical data in telemedicine. To achieve this, a transmission system comprising a transmitter module and a receiver module is developed. The transmitter module, consisting of a low-cost microcontroller, is responsible for collecting and transmitting the biomedical signals, while the receiver module, functioning as a web server, receives and processes these data.

Furthermore, a proposed encryption algorithm is detailed, which is based on chaos theory to safeguard the transmission of biomedical signals, specifically electrocardiogram (ECG) signals. This algorithm utilizes the unpredictable nature and sensitivity to initial conditions of chaos to generate unique and robust encryption keys. Implemented in a low-cost microcontroller serving as the transmitter module, it ensures secure collection and transmission of biomedical signals. On the other hand, the web server acting as the receiver module utilizes the same algorithm to decrypt the received data, thereby ensuring the privacy of transmitted medical information.

Various security analyses are conducted to assess the robustness of both the encryption algorithm and the transmission system as a whole. These analyses include resistance tests against brute-force attacks, evaluations of the uniformity of encrypted data to withstand statistical attacks, and tests for sensitivity to minimal changes in encryption keys. Additionally, information entropy is measured to confirm the level of chaos in the encrypted data, indicating high security. All these analyses confirm that the proposed system offers secure transmission of biomedical signals in telemedicine.

Keywords: chaotic cryptography, security analysis, biomedical signals, embedded system, logistic map, secure transmission, and chaos theory.

A mi familia y a quienes han sido parte de mi formación personal.

Agradecimientos

A mi Familia, mis padres Eusebio y Sonia, así como a mis hermanos Christian, Carlos y Nayely, por su amor incondicional, comprensión y aliento constante.

A mis Amigos, por su apoyo emocional y por ser una fuente inagotable de motivación.

A la M.C. Elitania Jiménez García, por su constante apoyo y motivación a lo largo de la carrera y sobre todo, durante este trabajo.

Al Dr. Miguel Ángel Murillo Escobar, por su orientación, apoyo y paciencia a lo largo de este proceso. Así como sus valiosas contribuciones, sugerencias y críticas constructivas que ayudaron a mejorar este trabajo.

A mi comité de tesis, Dra. Rosa Martha López Gutiérrez, M.C. Irma Alejandra Amaya Patrón y M.I. Luz Evelia López Chico, por su pericia y valiosas sugerencias que contribuyeron a este trabajo.

A la Universidad Autónoma de Baja California (UABC), por brindarme los recursos y el ambiente propicio para llevar a cabo esta investigación.

Al Consejo Nacional de Humanidades, Ciencia y Tecnología (CONAHCYT), por el apoyo económico recibido a través del Proyecto de Investigación en Ciencia Básica entre instituciones, "Sincronización de Sistemas Complejos y Algunas Aplicaciones". Ref. 166654 y continuación (A1-S-31628).

Ensenada, B.C., México.
Marzo de 2024

Kevin Becerra Santamaría

Tabla de Contenido

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción

En las últimas décadas, el crecimiento exponencial de las telecomunicaciones ha dejado una profunda huella en la sociedad, transformando prácticamente todos los aspectos de la vida cotidiana, desde el trabajo y la educación hasta las interacciones sociales. Este avance tecnológico ha llevado consigo un aumento constante en la cantidad de información compartida a través de medios de comunicación, con Internet como el epicentro de esta revolución digital. En paralelo, se ha observado un desplazamiento gradual de una variedad de servicios hacia el ámbito cibernético, que abarca desde la educación y las tareas de oficina hasta las transacciones comerciales y, en particular, los servicios de atención médica [1].

La telemedicina, o medicina a distancia, es un concepto que ha estado presente durante años, pero su importancia y aplicación se han intensificado en los últimos años debido a la pandemia de la enfermedad por coronavirus 2019 (COVID-19) [2, 3]. La necesidad de brindar atención médica sin exponer innecesariamente a pacientes y personal médico a riesgos de contagio ha impulsado la adopción acelerada de la telemedicina. Esta revolución en la prestación de servicios médicos permite consultas remotas y el monitoreo de signos vitales a distancia, minimizando el contacto físico.

Sin embargo, este cambio hacia la telemedicina ha llevado consigo la creciente transmisión de información médica a través de Internet, lo que plantea un desafío crítico en términos de seguridad y privacidad. La necesidad de salvaguardar la confidencialidad de los datos biomédicos compartidos en línea se ha convertido en una preocupación, ya que la información sensible podría ser interceptada y mal utilizada por individuos no autorizados, dando lugar a estafas y fraudes [4].

Para abordar este desafío, se ha recurrido a la criptografía, una disciplina ancestral que se ha adaptado y evolucionado en respuesta a las demandas de la era digital. La criptografía implica la aplicación de técnicas y métodos para alterar la estructura de un mensaje mediante un algoritmo de cifrado, de manera que solo las partes autorizadas puedan entenderlo. Esto garantiza que, en caso de intercepción por parte de terceros no autorizados, el mensaje sea ininteligible y protegido.

A lo largo de los años, se han desarrollado métodos de cifrado *convencionales* , como AES (Advanced Encryption Standard) y DES (Data Encryption Standard), que han demostrado ser seguros en la protección de datos. Sin embargo, estos métodos pueden resultar ineficientes cuando se trata de cifrar información en tiempo real, como archivos multimedia [5]. Como respuesta a esta limitación, han surgido enfoques *no convencionales* que aprovechan fenómenos complejos, como el caos, para aumentar la seguridad y la complejidad de los algoritmos criptográficos.

En particular, los métodos de cifrado basados en la teoría del caos se han destacado por su capacidad para explotar propiedades intrínsecas del caos, como la sensibilidad a las condiciones iniciales, la ergodicidad y la impredecibilidad, para aumentar la complejidad y la seguridad de los algoritmos criptográficos [6]. Estos métodos, al combinar el caos con procesos de confusión y difusión, generan algoritmos resistentes a ataques *criptoanalíticos*, lo que significa que es altamente improbable que un intruso pueda recuperar el mensaje original interceptado.

En este contexto, este trabajo de tesis se centra en desarrollar un sistema de transmisión inalámbrica de señales biomédicas a través de Internet, utilizando un algoritmo de cifrado caótico. Este sistema tendrá una interfaz amigable para pacientes y especialistas en un servidor web, permitiendo el procesamiento de datos, generación de criptogramas, almacenamiento en la nube y transmisión segura por Internet. Este algoritmo garantizará que las señales sean incomprensibles para personas no autorizadas, protegiendo así la privacidad de los datos biomédicos de los pacientes en el contexto crítico de la telemedicina actual.

1.1. Motivación

La evolución de las tecnologías, en particular Internet y los continuos avances en dispositivos móviles, ha dotado a la sociedad con herramientas poderosas que simplifican nuestras actividades diarias y acortan las distancias entre las personas. Gracias a plataformas en línea, podemos intercambiar información de manera instantánea sin necesidad de movernos físicamente. No obstante, ningún sistema es infalible y puede presentar vulnerabilidades que comprometan la seguridad de los datos que viajan a través de él. Esto convierte a los canales de comunicación en posibles puntos débiles, especialmente cuando se trata de compartir información confidencial, como los datos médicos, ya que existe el riesgo de filtraciones en el proceso de transmisión [7].

Una filtración se refiere a la exposición no consentida de información que puede tener consecuencias perjudiciales para su propietario. De acuerdo al Departamento de Salud y Servicios Humanos de Estados Unidos (DHHS, por sus siglas en inglés) se entiende como filtración el uso o divulgación no autorizada que pone en riesgo la seguridad y privacidad de la información médica protegida, de forma que represente un riesgo significativo de daño financiero, de reputación o de otro tipo [8].

El filtrado de información se puede dar por múltiples situaciones y en diferentes ámbitos, especialmente en esos en los que se maneja la información de muchas personas como lo son: educativo, gubernamental, negocios y evidentemente médico. Sin embargo, como se muestra en [7], fue 2019 el año donde se han presentado más casos de filtraciones desde 2005, abarcando entre el 60 % y 80 % (ver figura 1.1). Dentro de los casos que se han dado a conocer se han reportado incidentes [9], especialmente en 2019, donde millones de registros médicos fueron robados o expuestos. También se muestra que la mayoría de los datos han sido extraídos de fuentes electrónicas relacionadas a las redes de computadoras o fuentes conectadas a internet, como lo son servidores o directamente del correo electrónico.

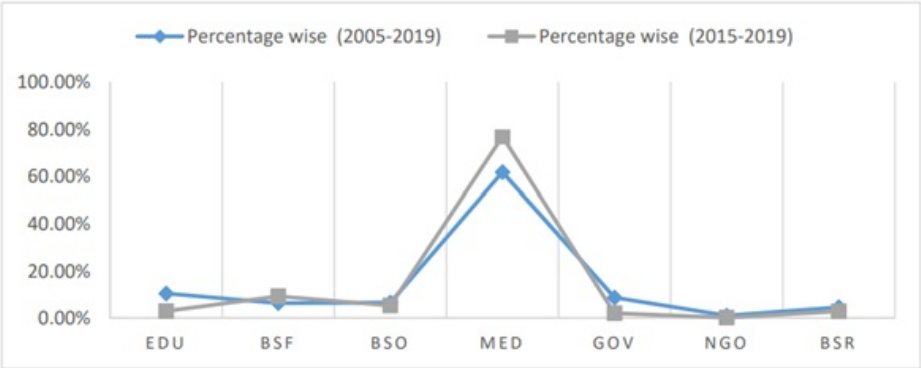


Figura 1.1: Incidentes reportados sobre filtraciones de datos [10].

Considerando lo expuesto anteriormente, es evidente la necesidad continua de mejorar los sistemas de seguridad utilizados para resguardar y compartir la información médica. Esto es crucial para preservar la integridad de los pacientes y brindarles un sentido de confianza y tranquilidad cuando opten por las consultas médicas a distancia. En este contexto, resulta pertinente explorar en mayor profundidad los métodos criptográficos basados en el caos como una solución al desafío planteado. Estos métodos, gracias a sus características distintivas, han demostrado ser eficaces en pruebas de seguridad criptoanalíticas, lo que respalda la necesidad de continuar su desarrollo y perfeccionamiento para fortalecer sus propiedades y aumentar su nivel de seguridad. Además, la implementación de este sistema se llevará a cabo en un entorno de usuario amigable a través de una interfaz web, lo que facilitará su acceso y uso seguro tanto para pacientes como para especialistas médicos.

1.2. Objetivos y alcances

Considerando la necesidad de mejorar la seguridad y privacidad en la transmisión de datos biomédicos en aplicaciones de telemedicina, esta tesis de licenciatura tiene como *objetivo general*:

Diseñar, implementar y evaluar un sistema de encriptación y desencriptación basado en criptografía caótica para la transmisión segura de señales biomédicas.

Para lograr este objetivo general, se plantean los siguientes *objetivos particulares*:

1. Crear un algoritmo de encriptación eficiente basado en la teoría del caos que garantice la seguridad y privacidad de las señales biomédicas transmitidas.
2. Adaptar y programar el algoritmo de encriptación en un microcontrolador de bajo costo.
3. Diseñar y configurar un servidor web remoto que permita el desencriptado seguro de las señales biomédicas recibidas.
4. Evaluar la robustez del sistema mediante pruebas de seguridad criptoanalíticas para garantizar su resistencia a posibles ataques.
5. Evaluar la viabilidad y eficacia del sistema de transmisión, centrándose en su capacidad para conectar a pacientes con especialistas médicos en entornos remotos.

1.3. Organización del manuscrito

El contenido de este trabajo de tesis se distribuye de la siguiente manera:

- **Capítulo 1:** Se presenta una breve introducción, la motivación que llevó a realizar este trabajo de tesis así como los objetivos del mismo.
- **Capítulo 2:** Se expone a la telemedicina en conjunto con una breve reseña histórica, sus distintos campos de aplicación y aspectos importantes relacionados a la seguridad en la misma.
- **Capítulo 3:** Se presenta al caos y sus principales propiedades, así como los mapas caóticos estudiados en este trabajo de tesis de los cuales se seleccionaron los utilizados en el algoritmo de cifrado propuesto. Además, se introduce el concepto de criptografía, sus diferentes clasificaciones y los principales aspectos a considerar en un sistema criptográfico basado en caos.
- **Capítulo 4:** En este capítulo de desarrolla el algoritmo de cifrado propuesto en este trabajo de tesis para el cifrado de señales biomédicas.
- **Capítulo 5:** Se detalla el sistema embebido diseñado para la transmisión de señales biomédicas a través de internet de forma segura. Se presenta el concepto de sistema embebido y el hardware utilizado para implementar el sistema propuesto, además de dos pruebas de transmisión de señales biomédicas. Finalmente, se exponen las diferentes pruebas de seguridad y eficiencia realizadas.

- **Capítulo 6:** Se presentan las conclusiones de este trabajo de tesis, se mencionan las principales contribuciones del trabajo de tesis y algunos puntos a considerar como trabajo futuro.

Capítulo 2

Telemedicina

En este capítulo, se aborda el concepto de telemedicina, se proporciona un breve contexto histórico y se examinan las áreas de aplicación de esta disciplina. Además, se evalúa el nivel de seguridad inherente a los sistemas de telemedicina.

2.1. Introducción

La telemedicina se refiere a la provisión de servicios médicos a distancia, utilizando tecnologías avanzadas como Internet. La Organización Mundial de la Salud (OMS) define la telemedicina como "la entrega de servicios de atención médica, donde la distancia es un factor crítico, a través de profesionales de la salud que utilizan tecnologías de comunicación para el intercambio de información en el diagnóstico, tratamiento, prevención, investigación y educación continua, con el fin de mejorar la salud de las personas y las comunidades" [11, 12].

Los sistemas de telemedicina pueden tener diversas estructuras y complejidades según su aplicación. Sin embargo, todos comparten el objetivo fundamental de conectar a profesionales médicos e instituciones con usuarios ubicados en diferentes áreas geográficas (ver figura 2.1). Estos usuarios pueden ser pacientes que requieren atención médica o personal médico adicional. La elección del canal de transmisión de información varía según el sistema, y puede incluir redes móviles (como 3G y 4G) o Internet.

La telemedicina presenta una serie de beneficios significativos que impactan positivamente a todos los involucrados en el proceso de intercambio de información médica. Estos beneficios abarcan al sistema de atención médica, los profesionales de la salud y los pacientes [14].

Para el sistema de atención médica, la telemedicina contribuye a la eficiencia del servicio de varias maneras. Reduce la necesidad de que los profesionales se desplacen físicamente, lo que a su vez mejora la gestión de la demanda en general. Esto se traduce en una optimización de los recursos y una distribución más equitativa de la atención médica.

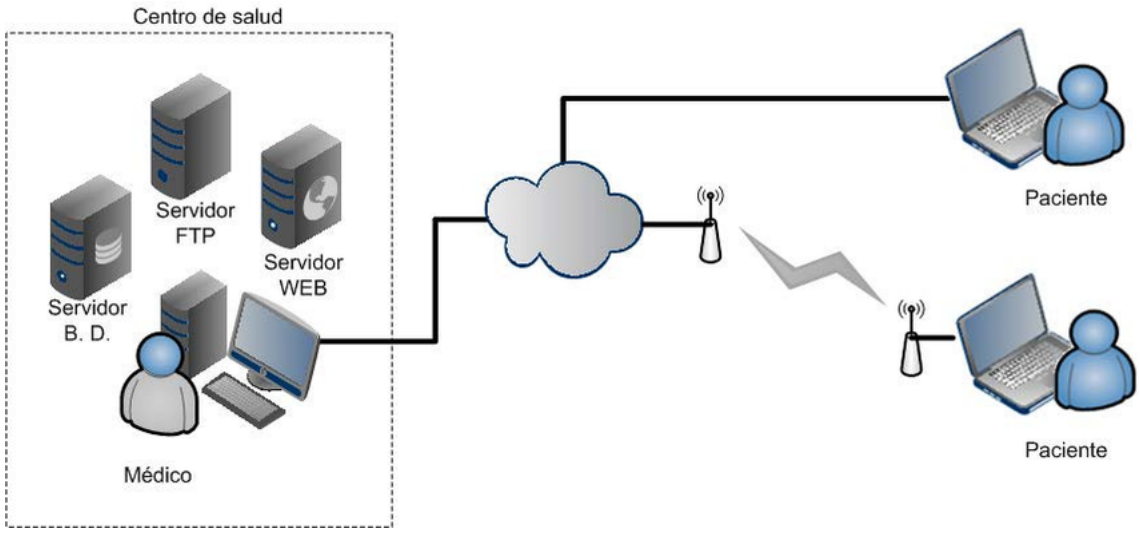


Figura 2.1: Representación básica de una red de telemedicina [13].

Los profesionales de la salud, como los médicos, experimentan una comunicación más fluida entre ellos, lo que facilita las consultas y la colaboración en el diagnóstico y tratamiento de los pacientes. Además, tienen un acceso más rápido y sencillo a la información del paciente y a investigaciones médicas relevantes [15].

Por último, los pacientes obtienen ventajas significativas al utilizar la telemedicina. Evitan la necesidad de desplazarse físicamente a un centro médico cada vez que requieren atención o seguimiento médico. También acceden con mayor facilidad a su historial médico y pueden ser monitoreados en tiempo real mediante dispositivos especializados, lo que permite una atención más proactiva y reactiva en situaciones médicas críticas, mejorando la calidad del servicio médico en general.

2.2. Antecedentes

La evolución de la telemedicina se puede trazar a través de tres etapas históricas distintas: la era de las telecomunicaciones, la era digital y la era de Internet [16]. Aunque no existe un punto de origen exacto en la historia de la telemedicina, varios eventos históricos clave han contribuido al desarrollo de esta disciplina tal como la conocemos hoy en día, algunos de los cuales se mencionan en un estudio previo [17].

Era de las Telecomunicaciones (Hasta la década de los 70): La telemedicina tiene sus raíces en la expansión de las telecomunicaciones al llegar a la década de 1970. Durante este período, se realizaron avances significativos en la transmisión de datos médicos a través de líneas telefónicas y redes de comunicación. Algunos antecedentes importantes de esta época incluyen:

- En 1903, Einthoven, el inventor del electrocardiograma (ECG), llevó a cabo experimentos pioneros para transmitir señales de ECG a través de las líneas telefónicas, y en 1906 logró transmitir datos desde un hospital hasta su propio laboratorio [17].

- En 1950, Gershon-Cohen informó sobre un sistema de diagnóstico de rayos X que se transmitía por radio o líneas telefónicas a través de cortas o largas distancias, lo que podría brindar acceso a áreas rurales a este servicio sin la necesidad de un radiólogo [18].

- El Massachusetts General Hospital estableció en 1967 un servicio de salud ocupacional para empleados del aeropuerto y brindó atención médica y de emergencia a los viajeros. Este acontecimiento podría considerarse en la transición entre la era de las telecomunicaciones y la era digital, ya que involucraba servicios médicos y comunicación, pero antes del advenimiento de la tecnología digital tal como la conocemos hoy en día [19].

Era Digital (Década de los 80): En los años 80, con el advenimiento de la tecnología digital, la telemedicina comenzó a utilizar sistemas de almacenamiento y transmisión de datos más avanzados. Esto permitió una mayor calidad de la información médica transmitida y un acceso más rápido. Sucedieron acontecimientos como:

- En 1986, se llevó a cabo la primera videoconferencia entre médicos en Noruega, marcando un hito en la comunicación médica a distancia [20].

- En 1988, la NASA lanzó el programa *Space Bridge* para colaborar con Armenia y Ufa (en esa época pertenecientes a la Unión Soviética). Las conexiones se realizaron utilizando video unidireccional y voz y fax bidireccionales entre el Centro Médico de Yereván, Armenia, y cuatro hospitales en los Estados Unidos. Este programa se extendió posteriormente a Ufa, donde se utilizó para socorrer a los quemados en un terrible accidente de tren [20].

Era de Internet (Actualidad): La era actual de la telemedicina se ha visto moldeada por la omnipresencia de Internet. La disponibilidad de conexiones de banda ancha y la proliferación de dispositivos móviles han facilitado la comunicación médica remota y el acceso a registros médicos electrónicos en tiempo real. Internet ha revolucionado la telemedicina y ha abierto nuevas posibilidades en la atención médica a distancia, las cuales se muestran a continuación:

- En 1995, la Clínica Mayo estableció una conexión con el Hospital Real de Ammán, en Jordania, permitiendo consultas médicas en vivo a través de Internet con el apoyo de un médico local [20].

- En 2003, la Universidad de Chile inició el Proyecto Argonauta, un proyecto de telemedicina en la Antártica [20].

- En 2007, se puso en marcha el proyecto Tele-Ictus, que conectaba el Hospital General de Vic con el Hospital Vall d'Hebron en España, permitiendo la evaluación y el tratamiento neurológico remoto las 24 horas del día, los 7 días de la semana [21].

Aunque estos hitos históricos marcan el desarrollo de la telemedicina, es importante reconocer que su evolución ha sido un proceso continuo y en constante transformación. Cada era ha aportado avances significativos que han mejorado la atención médica y la accesibilidad de la atención médica en todo el mundo.

2.3. Clasificación de la telemedicina

La Telemedicina se ha dividido en diversas categorías según el tipo de servicio proporcionado o el ámbito de aplicación. Entre estas categorías, se destacan cuatro ramas fundamentales [22]:

1. **Teleconsulta:** La Teleconsulta es una de las prácticas más comunes en Telemedicina, involucrando la búsqueda de información médica y el asesoramiento médico. Puede aplicarse tanto entre pacientes y profesionales de la salud como entre los mismos profesionales. Se puede llevar a cabo de dos formas principales:

    - Modalidad asíncrona: Implica el envío de información clínica, a menudo acompañada de imágenes de apoyo, a través de servicios como el correo electrónico para su posterior evaluación por un especialista. La ventaja clave aquí es la flexibilidad de tiempo para los participantes, ya que no es necesario que estén presentes simultáneamente.
    - Modalidad síncrona: Esta modalidad implica la realización de la consulta en tiempo real mediante tecnologías de la información y las comunicaciones, como las videoconferencias.

2. **Teleducación:** La Teleducación se enfoca en la educación médica a distancia, utilizando tecnologías de la información y las comunicaciones para aumentar las experiencias educativas de los estudiantes, brindando oportunidades de entrenamiento y facilitando la interacción entre estudiantes y especialistas.

3. **Telemonitoreo:** El Telemonitoreo permite la obtención continua y remota de los signos vitales de un paciente. Esto evita que los pacientes tengan que desplazarse al centro de salud para ser monitoreados y se lleva a cabo generalmente desde el hogar del paciente mediante dispositivos terminales que pueden medir y transmitir señales como el electrocardiograma (ECG), niveles de insulina y otras variables fisiológicas.

4. **Telecirugía:** La Telecirugía implica la realización de procedimientos quirúrgicos de forma remota por un especialista, el cirujano, que no se encuentra físicamente cerca del paciente. Estas cirugías suelen realizarse con la ayuda de sistemas robotizados para abordar problemas de accesibilidad o llevar a cabo intervenciones en entornos peligrosos [23].

Además de estas categorías, la Telemedicina se clasifica según su aplicación en diversas especialidades médicas, como Telecardiología, Telerradiología, Tele-bioingeniería y Tele-endoscopía.

2.4. Seguridad en telemedicina

La seguridad en el ámbito de la telemedicina se ha convertido en un tema crítico en la actualidad, ya que, si bien la tecnología ha revolucionado la comunicación global a través de avances como Internet, también ha generado vulnerabilidades que pueden poner en riesgo la privacidad de la información. Es crucial reconocer que junto con estos avances tecnológicos han surgido técnicas que permiten a terceros acceder a datos en la red.

El interés económico se encuentra entre las principales motivaciones de quienes buscan acceder a registros médicos en bases de datos hospitalarias o sistemas de telemedicina. Según un informe en 2017 [24], el precio promedio de un registro médico oscilaba entre 20 y 50 dólares en el mercado negro. Aunque el beneficio financiero suele ser la motivación principal detrás del robo de registros médicos, estos incidentes a menudo se asocian con otras intenciones, como el robo de identidad, la extorsión o incluso motivaciones puramente lúdicas.

La necesidad de salvaguardar la información médica en sistemas de telemedicina es innegable, ya que su acceso no autorizado puede tener consecuencias graves tanto para los pacientes como para los profesionales de la salud. Por lo tanto, abordar adecuadamente la seguridad en la telemedicina es esencial para garantizar la confidencialidad e integridad de los datos médicos en un mundo cada vez más conectado.

En [25] se mencionan siete áreas en las que la seguridad se ve amenazada en un sistema de telemedicina (ver figura 2.2):

1. **Usuario o paciente:** Los usuarios o pacientes, generalmente sin capacitación en ciberseguridad, pueden representar un riesgo al utilizar contraseñas débiles o cometer errores en el manejo de los dispositivos de telemedicina.

2. **Dispositivos de telemedicina:** Los dispositivos de telemedicina, como los smartphones, suelen ejecutar sistemas operativos de propósito general y utilizan aplicaciones móviles externas. Esto los expone a amenazas de seguridad, ya que las aplicaciones pueden tener vulnerabilidades que podrían comprometer la información almacenada.

3. **Red de casa:** La conectividad en el hogar, como Wi-Fi y Bluetooth, se utiliza para la transmisión de datos en los sistemas de telemedicina. Esta red está expuesta a amenazas como la interceptación de datos y ataques “Hombre en el medio”.

4. **Dispositivos de puerta de enlace:** Las puertas de enlace actúan como intermediarios entre el paciente y el sistema de telemedicina y, por lo tanto, enfrentan amenazas similares a las de las redes domésticas.

5. **Internet:** La transmisión de datos médicos a través de Internet es especialmente vulnerable, ya que es un medio de comunicación público y la información médica es privada.

6. **Sistema de telemedicina:** El sistema de telemedicina, que consta de una PC y un software que facilita las consultas remotas, es un punto crítico que atrae diversas amenazas, como malware, accesos no autorizados, ataques "Hombre en el medio" y otros riesgos de seguridad.

7. **Proveedor del servicio de telemedicina:** Los proveedores de servicios de telemedicina establecen conexiones médicas, lo que implica un intercambio de información médica. Esto puede atraer amenazas similares a las mencionadas anteriormente.

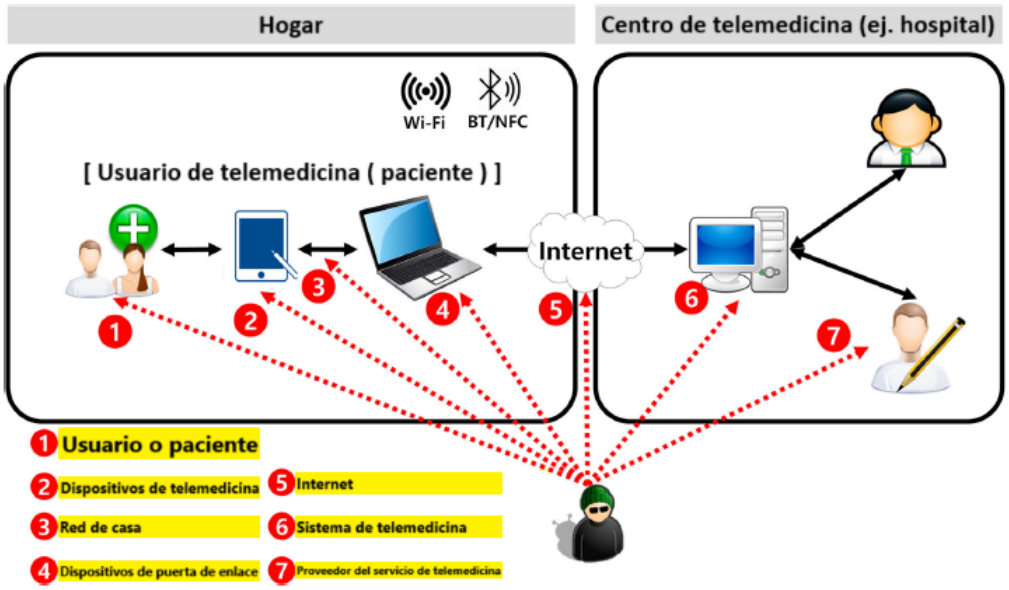


Figura 2.2: Áreas en las que la seguridad se ve amenazada en un sistema de telemedicina [26].

Como se puede evidenciar, en el proceso de transmisión de datos en los sistemas de telemedicina, existen numerosos puntos vulnerables donde la información podría estar en riesgo, lo que, a su vez, comprometería la seguridad del paciente. Por lo tanto, en los últimos años, se ha llevado a cabo una investigación significativa en el ámbito de la seguridad en telemedicina. El propósito de esta investigación es reducir las áreas de vulnerabilidad de la información médica para proporcionar una mayor confidencialidad y seguridad a los pacientes.

2.5. Conclusiones del capítulo

En este capítulo, se exploró el mundo de la telemedicina, desde su definición hasta su evolución histórica en tres etapas clave: la era de las telecomunicaciones, la era digital y la era de Internet. La telemedicina ha transformado la atención médica al permitir la comunicación a distancia entre profesionales de la salud y pacientes, mejorando la eficiencia del sistema y brindando beneficios significativos tanto para el sistema de atención médica como para los pacientes.

Además, se resaltó la importancia de la seguridad en la telemedicina. La creciente conectividad y el intercambio de información médica en línea han generado riesgos de privacidad y seguridad. Se identificaron siete áreas de vulnerabilidad en los sistemas de telemedicina, lo que subraya la necesidad de abordar adecuadamente la seguridad para garantizar la confidencialidad e integridad de los datos médicos y proteger a los pacientes y profesionales de la salud.

Capítulo 3

Caos y criptografía

En este capítulo, se presentan los fundamentos esenciales de la teoría del caos, explorando sus características distintivas y propiedades. Se revisan los precedentes históricos y se analizarán diversos ejemplos de sistemas caóticos, evaluando tanto sus tiempos de ejecución como sus dinámicas caóticas. Estos análisis sirven como base para la selección de los mapas caóticos que desempeñan un papel fundamental en esta investigación.

Además, se introduce el intrigante mundo de la criptografía, desglosando sus pilares fundamentales y su clasificación. Se realiza un breve recorrido histórico para comprender la evolución de la criptografía a lo largo del tiempo y se lleva a cabo un análisis exhaustivo de la seguridad en los sistemas criptográficos, delineando las principales amenazas y desafíos que enfrentan en la era digital actual.

3.1. Caos

A lo largo de la historia, el término *caos* ha sido empleado de forma coloquial para referirse a situaciones que desafían una explicación o comprensión claras. Tradicionalmente, el caos se ha asociado con la noción de desorden y confusión en diversos contextos. No obstante, en la actualidad, el caos ha adquirido una connotación completamente diferente al convertirse en un campo de estudio independiente, gracias al descubrimiento de sus íntimas relaciones con una amplia gama de fenómenos naturales que pueden tener aplicaciones reales.

3.1.1. Antecedentes

Hasta el siglo XIX, la concepción predominante entre los físicos sostenía que los cuerpos mecánicos deberían exhibir un comportamiento regular y predecible. Sin embargo, Henri Poincaré, durante sus investigaciones sobre el *problema de los n cuerpos*, reveló que ciertos sistemas podrían desarrollarse en el tiempo de manera irregular y aperiódica [27]. Además, Poincaré notó que pequeñas alteraciones en las condiciones iniciales de un sistema podrían conducir a diferencias significativas en los resultados finales, lo que tornaba imposible la predicción precisa del sistema en el futuro [28].

Casi setenta años después, en 1963, Edward Lorenz, un meteorólogo y matemático estadounidense, observó un fenómeno similar al de Poincaré, pero en un contexto numérico. Lorenz estaba trabajando con un modelo meteorológico que se basaba en un conjunto de ecuaciones diferenciales no lineales para describir el comportamiento de la atmósfera y predecir el clima. En un intento por reducir el tiempo de procesamiento computacional, Lorenz decidió reducir la precisión decimal de los cálculos de seis a tres decimales, asumiendo que los resultados serían casi idénticos con un margen mínimo de error. Sin embargo, esta simplificación condujo a resultados numéricos significativamente diferentes de los originales. Lorenz notó que estos resultados no exhibían un patrón periódico y, al profundizar en su análisis, descubrió que la trayectoria del sistema se asemejaba a una *figura extraña* [29], que recordaba a las alas de una mariposa.

Este efecto, que se originaba en la extrema sensibilidad del sistema a las condiciones iniciales, fue bautizado por Lorenz como el "efecto mariposa", ilustrando la idea de que el aleteo de una mariposa en un lugar tan lejano como Brasil podría potencialmente desencadenar un tornado en Texas. Estos resultados llevaron a Lorenz a simplificar aún más su modelo original, reduciéndolo de un conjunto de 12 ecuaciones diferenciales no lineales a un sistema de solo 3 grados de libertad, que estaba conformado por las siguientes ecuaciones diferenciales no lineales:

$$\frac{dx}{dt} = \sigma(y - x), \tag{3.1a}$$

$$\frac{dy}{dt} = \rho x - y - xz, \tag{3.1b}$$

$$\frac{dz}{dt} = xy - \beta z \tag{3.1c}$$

donde Edward Lorenz representó el sistema utilizando las variables x_0, y_0, y z_0 para describir sus estados, x_0, y_0, y z_0 como las condiciones iniciales, y σ, ρ, β como los parámetros de control en sus ecuaciones. A medida que observaba el comportamiento del sistema en el plano de fase, notó que se manifestaba el mismo atractor extraño que había descubierto anteriormente, que más tarde se bautizó como el Atractor de Lorenz (ver figura 3.1). Este trabajo pionero de Lorenz marcó el inicio de lo que conocemos hoy como la *teoría del caos*.

El *caos*, desde una perspectiva matemática y física, se caracteriza por manifestar un comportamiento en apariencia caótico e impredecible en sistemas dinámicos no lineales, a pesar de que en un principio su formulación sea determinista [31]. Hoy en día, la teoría del caos desempeña un papel fundamental en diversas disciplinas, como la física, las matemáticas, la medicina e incluso la economía. A lo largo del tiempo, se ha revelado que muchos fenómenos cotidianos pueden ser modelados con precisión mediante ecuaciones que exhiben un comportamiento caótico, como los ciclos de deudas económicas [32] y el equilibrio del cuerpo humano [33].

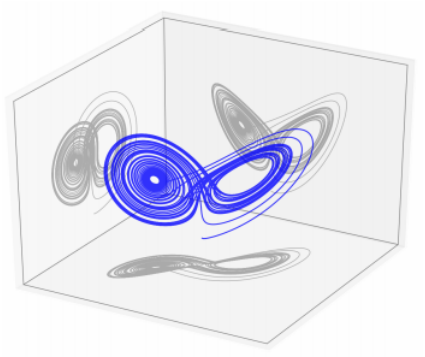


Figura 3.1: Atractor generado por el sistema de Lorenz [30].

3.1.2. Propiedades de los sistemas caóticos

La clasificación de los sistemas dinámicos se basa en su dependencia del tiempo, dividiéndose en dos categorías: sistemas de tiempo continuo, modelados con ecuaciones diferenciales, y sistemas de tiempo discreto, representados mediante ecuaciones en diferencias. Aunque presentan diferencias en su formulación matemática, comparten propiedades fundamentales. La estabilidad, una de las propiedades clave de los sistemas dinámicos, desempeña un papel esencial en su comportamiento a lo largo del tiempo [34]. Esta característica se puede categorizar en tres tipos:

- **Estable:** Se refiere a sistemas cuyas condiciones iniciales cercanas convergen en el espacio de fase hacia un punto fijo o siguen un patrón oscilatorio periódico, conocido como ciclo límite.
- **Inestable:** Los sistemas inestables exhiben soluciones divergentes de manera exponencial cuando las condiciones iniciales son próximas. Esto significa que no generan un atractor en el espacio de fase.
- **Caótico:** Los sistemas caóticos combinan comportamientos estables e inestables. Sus trayectorias no convergen hacia un punto (fuerzas de repulsión), pero están confinadas dentro de un rango de valores por fuerzas de atracción, formando un atractor de *forma extraña*.

Debido a lo anterior, se puede caracterizar un sistema caótico como un sistema dinámico modelado por un conjunto de ecuaciones no lineales en forma de ecuaciones diferenciales o en diferencias. Estas ecuaciones generan trayectorias deterministas de tipo caótico, lo que implica que los estados futuros y pasados del sistema están intrínsecamente vinculados a su estado actual. Además, los sistemas caóticos exhiben las siguientes propiedades [35]:

- **Sensibilidad exponencial a condiciones iniciales y parámetros de control:** Pequeñas variaciones en las condiciones iniciales o parámetros de control conducen a cambios drásticos en la trayectoria del sistema.

- **No linealidad:** Estos sistemas se describen mediante ecuaciones no lineales y no obedecen el principio de superposición, lo que significa que el comportamiento total no es la suma de los comportamientos individuales de sus componentes.
- **Ergodicidad:** Las trayectorias caóticas permanecen confinadas en un espacio llamado atractor extraño a lo largo del tiempo, y este atractor se cubre por completo para cualquier conjunto de condiciones iniciales y parámetros de control.
- **Mezcla de datos:** Un pequeño conjunto de condiciones iniciales abarca la mayoría de las posibles trayectorias caóticas, lo que implica que el sistema exhibe una amplia gama de comportamientos.
- **Exponente de Lyapunov positivo:** La presencia de al menos un exponente de Lyapunov positivo en un sistema de dimensión N indica un comportamiento caótico.
- **Atractor extraño con dimensión fractal:** La representación gráfica de fase de un sistema caótico forma un atractor extraño, cuya dimensión corresponde a un número fraccional, lo que implica una estructura fractal.

3.1.3. Exponente de Lyapunov

El exponente de Lyapunov es una métrica crucial para evaluar la predictibilidad de sistemas dinámicos al medir la tasa de divergencia o convergencia exponencial de trayectorias cercanas en el espacio de fase. Este cálculo se basa en la idea de que pequeñas variaciones en las condiciones iniciales pueden generar estados iniciales casi idénticos, pero con el tiempo, estas trayectorias se alejarán exponencialmente unas de otras, lo que refleja la pérdida de predictibilidad en el sistema [36]. La fórmula para calcular el exponente de Lyapunov se expresa como:

$$\lambda = \frac{1}{T}\ ln\ |\frac{f^n(x_n - \delta_0) - f^n(x_n)}{\delta_0}| \tag{3.2}$$

donde λ es el exponente de Lyapunov, x_0 es una condición inicial, $x'_0 = x_0 + \delta_0$ es otra condición inicial extremadamente cercana y T es el número de iteraciones.

3.1.4. Mapas caóticos estudiados

Durante el desarrollo de este trabajo de tesis, se analizaron diversos mapas caóticos, evaluando sus tiempos de ejecución y sus dinámicas caóticas. Los mapas caóticos estudiados se muestran a continuación:

Mapa caótico iterativo con mapa de modulación de colapso infinito (SLIM)

El mapa está definido como [37]:

$$x_{i+1} = \sin(by_i)\sin\left(\frac{50}{x_i}\right), \tag{3.3a}$$

$$y_{i+1} = a(1 - 2x_{i+1}^2)\sin\left(\frac{50}{y_i}\right) \tag{3.3b}$$

donde x_0 y y_0 son las condiciones iniciales y $a, b \in (0, \infty)$ son los parámetros de control del mapa, cumpliendo que cuando $a \in (0, 4]$, $b = 2\pi$ y $b \in [4, 8]$,$a = 1$ el mapa SLIM presenta un comportamiento hipercaótico (ver figura 3.2).

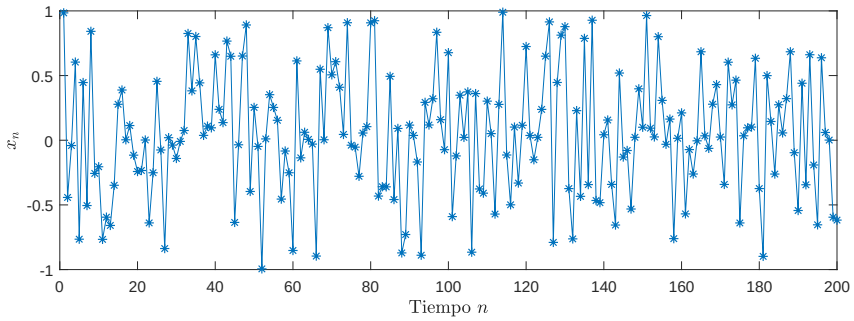


Figura 3.2: Estado x del mapa SLIM con dinámicas caóticas [38].

Mapa Seno-Hénon

Con la finalidad de disminuir la facilidad de predicción de las trayectorias de los mapas Seno y Hénon, en [39] realizaron un acoplamiento entre ambos mapas caóticos, resultando en el mapa Seno-Hénon que se define por el siguiente par de ecuaciones:

$$x_{n+1} = (1 - a\sin^2(x_n) + y_n)\ mod\ 1, \tag{3.4a}$$

$$y_{n+1} = bx_n\ mod\ 1 \tag{3.4b}$$

donde x_0 y y_0 son las condiciones iniciales en tanto que a y b corresponden a los parámetros de control del sistema, siendo que cuando $a \in \mathbb{R}$ y $b \notin [-1, 1]$ el mismo presenta un comportamiento caótico (ver figura 3.3).

Mapa Logístico

El mapa logístico es una representación polinómica de grado 2, introducida por Robert May en 1976. Se basa en una ecuación polinómica que muestra las propiedades de la dinámica caótica. Se encuentra definido por la siguiente ecuación:

$$x_{n+1} = rx_n(1 - x_n) \tag{3.5a}$$

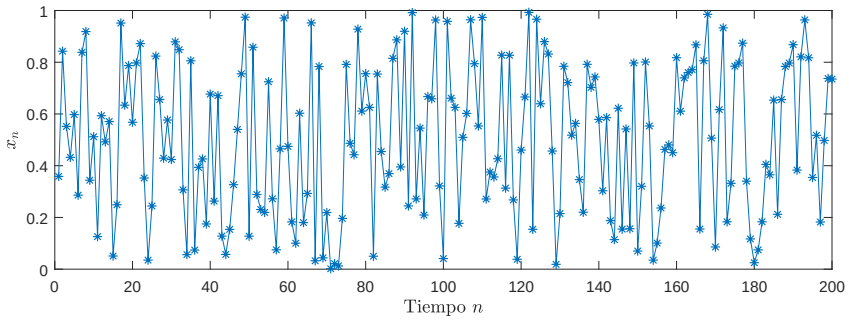


Figura 3.3: Estado x del mapa Seno-Hénon con dinámicas caóticas [40].

donde $x_n \in [0, 1]$ representa el enésimo número caótico, n denota el número de iteración y r es el parámetro de control. El mapa logístico exhibe alternativamente comportamientos regulares y caóticos cuando r pertenece al intervalo [3, 4] (ver figura 3.4).

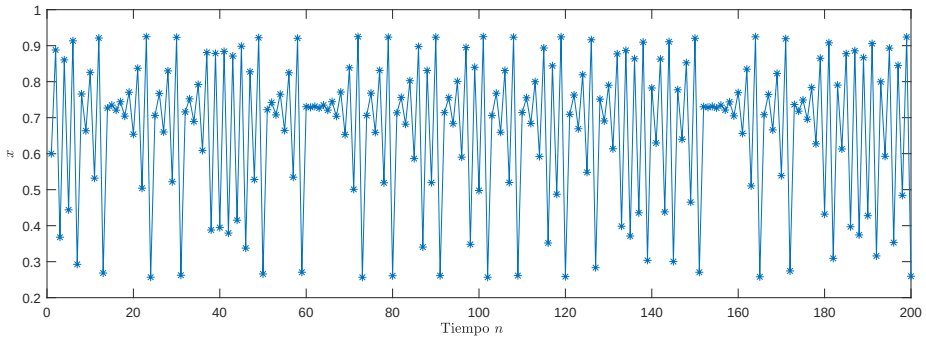


Figura 3.4: Estado x del mapa Logístico con dinámicas caóticas.

3.1.5. Mapas caóticos seleccionados

La elección de los mapas caóticos incorporados en este algoritmo de cifrado se basó en la metodología propuesta en [35]. Inicialmente, se consideró la inclusión de los mapas Seno-Hénon y Slim optimizado debido a sus notables atributos. Estos mapas demostraron un tiempo de procesamiento eficiente, con valores de 5.333 ms y 4.826 ms, respectivamente, y generaron secuencias caóticas con elevados exponentes de Lyapunov; específicamente, 5.37 para SLIM y 2.17 para Seno-Hénon, lo que señaló una fuerte propiedad caótica. En la tabla 3.1 se muestra el máximo exponente de Lyapunov de los mapas caóticos estudiados, comparando los valores provenientes de la literatura y los valores calculados con el método expuesto en [41]

Sin embargo, durante el proceso de selección, surgió una observación crítica. Ambos mapas caóticos empleaban funciones trigonométricas en su definición. El uso de funciones trigonométricas en los mapas caóticos conlleva ciertas dificultades. Al programar

Tabla 3.1: Máximo exponente de Lyapunov de los mapas caóticos estudiados.

Mapa caótico	Máximo exponente de Lyapunov	
	Literatura	Estimado
SLIM	6	5.37
Seno-Hénon	3	2.17
Logístico	0.67	0.68

en dos lenguajes de programación diferentes, uno para el cifrado y otro para el descifrado, se encontró que las funciones trigonométricas generaban variaciones muy pequeñas en los resultados. Estas variaciones pueden tener un impacto significativo cuando se realizan iteraciones extensas en el proceso de cifrado. Las pequeñas diferencias en los resultados pueden dar lugar a resultados divergentes, lo cual es inaceptable en un sistema de cifrado sólido.

Ante los desafíos identificados con los mapas Seno-Hénon y Slim optimizado, se tomó la decisión de optar por un enfoque diferente. El mapa logístico se eligió como una alternativa viable. A diferencia de los mapas caóticos que involucran funciones trigonométricas, el mapa logístico es conocido por su simplicidad y su ausencia de funciones trigonométricas. Esto lo convierte en una elección más favorable en términos de implementación en diferentes lenguajes de programación.

3.2. Criptografía

A lo largo de la historia, la necesidad de mantener la privacidad de la información compartida ha sido fundamental. Ocultar el contenido de un mensaje ha sido una preocupación constante, y aunque existen métodos para mantener la existencia del mensaje en secreto, en ocasiones esto no es suficiente. Es en este contexto que la *criptografía* ha emergido como un procedimiento sofisticado para salvaguardar la confidencialidad de los mensajes.

3.2.1. Definición y antecedentes

La criptografía se define en la actualidad como el estudio de técnicas matemáticas diseñadas para proteger la divulgación de información confidencial a usuarios no autorizados [42]. Su objetivo principal no es ocultar la existencia de un mensaje, sino su significado. De esta manera, incluso si un mensaje cae en manos de un usuario no autorizado, este no puede comprender su contenido, a pesar de poder visualizarlo.

Existen cuatro objetivos fundamentales que busca satisfacer la criptografía [43]:

- *Privacidad*: Garantiza que solo los usuarios autorizados puedan acceder al contenido de la información, manteniéndolo desconocido para *intrusos*. Esto se logra

mediante diversas técnicas, desde protección física hasta algoritmos matemáticos complejos que hacen que la información sea incomprensible.

- *Integridad de la información*: Asegura que los datos no puedan ser manipulados durante su transmisión. Se deben poder detectar distintos tipos de manipulaciones, como la inserción, sustitución o eliminación de datos, realizadas por usuarios no autorizados.
- *Autenticidad*: Confirma que el mensaje recibido fue realmente enviado por la persona que dice ser el remitente, además de que el mensaje coincide con lo que se esperaba recibir (integridad de la información).
- *No rechazo*: Garantiza que un remitente no pueda negar haber enviado un mensaje previamente.

Aunque no se conoce por completo el origen de la criptografía, se encuentra vinculado a los inicios de la comunicación humana, cuando se buscó medios para asegurar la confidencialidad de las comunicaciones. Según autores, el origen de la criptografía se remonta a la antigua Grecia, alrededor del siglo V a.C., cuando los lacedemonios empleaban un dispositivo llamado "escítala" que consistía en un palo en el que se enrollaba una tira de cuero con un mensaje escrito en columnas. Para descifrar el mensaje, el receptor necesitaba un palo idéntico al del emisor, lo que dificultaba la lectura de mensajes interceptados [44]. Este ejemplo temprano destaca la importancia histórica de la criptografía en la protección de información confidencial.

En el siglo I a.C., Julio César introdujo el cifrado Julio César, donde se desplazaba cada letra tres posiciones en el alfabeto [45]. La criptografía clásica incluyó métodos como el cifrado de Alberti y el cifrado Vigenère [46].

La llegada de máquinas mecánicas y eléctricas marcó el inicio de la *criptografía moderna*, con invenciones como la Enigma (ver figura 3.5), Purple, SIGABA y Typex. La computadora revolucionó aún más la criptografía con algoritmos matemáticos avanzados, como el cifrado Lucifer, el cifrado DES y el sistema RSA.

Figura 3.5: Máquina Enigma utilizada por los alemanes para cifrar mensajes [47].

3.2.2. Seguridad en los criptosistemas y cifrado caótico

La seguridad es fundamental en los sistemas criptográficos. Auguste Kerckhoff, en el siglo XIX, estableció principios clave para considerar un sistema seguro. Se destaca que ni el texto claro ni la clave secreta deben ser deducibles a través del análisis del criptograma, y la clave de cifrado se considera privada, mientras que el algoritmo de cifrado es público [48].

Para evaluar la seguridad de los sistemas criptográficos, se han identificado tres categorías de ataques:

1. **Ataque exhaustivo:** Consiste en probar todas las claves posibles para descifrar el criptograma, lo que depende del poder computacional y el espacio de claves utilizados.

2. **Ataque diferencial:** Evalúa la sensibilidad del sistema criptográfico tanto a la clave secreta como al mensaje claro. Incluye el ataque de criptograma conocido, de mensaje claro conocido, de mensaje claro elegido y de criptograma elegido [49].

3. **Ataque estadístico:** Se basa en análisis estadísticos, como histogramas y correlación del mensaje, para aprovechar la uniformidad del criptograma.

Dado el avance del poder computacional, los métodos de cifrado clásico se han vuelto vulnerables. La criptografía no convencional ha ganado relevancia, y los criptosistemas basados en el caos son especialmente investigados. Ofrecen ventajas en ergodicidad, complejidad dinámica y sensibilidad a condiciones iniciales, lo que los convierte en candidatos idóneos para la generación de secuencias pseudoaleatorias en algoritmos de cifrado [50].

Los sistemas criptográficos basados en caos en tiempo discreto son preferibles, ya que son eficientes al generar secuencias para el algoritmo a través de iteraciones. A pesar de esto, no se consideran seguros frente a ataques de criptograma elegido o mensaje claro elegido [51].

Para implementar sistemas criptográficos basados en caos de manera efectiva, se deben considerar las siguientes reglas esenciales [52]:

1. Detalles de implementación del sistema caótico.

2. Facilidad de implementación sin comprometer seguridad, costo y velocidad.

3. Definición precisa de la clave secreta y su rango para evitar regiones no caóticas.

4. Clave con distribución uniforme para no revelar información sobre el mensaje claro.

5. Proceso de generación de claves bien definido.

6. Mínimos cambios en la clave o mensaje deben producir criptogramas completamente diferentes.

7. Análisis para evitar debilidades y ataques.

8. Evaluación de secuencias aleatorias generadas con pruebas estadísticas.

3.3. Conclusiones del capítulo

A lo largo de la historia, la criptografía ha desempeñado un papel esencial en la protección de la privacidad de las comunicaciones humanas. Su evolución ha estado en constante sincronía con los avances tecnológicos de cada época, garantizando la transmisión segura de mensajes. En la actualidad, los métodos criptográficos basados en la teoría del caos destacan como una vanguardia prometedora. Aprovechando las propiedades intrínsecas del caos, como la sensibilidad a las condiciones iniciales y la ergodicidad, estos métodos simplifican los algoritmos y reducen significativamente los tiempos de procesamiento.

En este capítulo, se exploraron tres diferentes mapas caóticos para su aplicación criptográfica. Finalmente, se optó por utilizar el mapa logístico en este trabajo de tesis, en lugar de los mapas SLIM y Seno-Hénon, debido a su capacidad para prescindir de funciones trigonométricas, lo que simplifica la implementación en diferentes lenguajes de programación. Este cambio en la elección del mapa caótico es fundamental para lograr un cifrado y descifrado eficiente y seguro en múltiples ambientes.

Capítulo 4

Sistema de encriptación caótica propuesto

En este capítulo, se aborda el algoritmo de cifrado utilizado durante el desarrollo del sistema de telemedicina. Este algoritmo consiste en utilizar una clave de 32 caracteres hexadecimales (128 bits) para la generación de las condiciones iniciales del mapa logístico.

4.1. Introducción

La telemedicina en la actualidad, juega un papel muy importante dentro del área de la salud y, uno de los componentes fundamentales de esta, es la transmisión de bioseñales seguras. Es por esto que se han registrado algoritmos basados en señales caóticas para el encriptado de señales [53, 54].

En este trabajo de tesis, el algoritmo implementado se basa en [35] realizando modificaciones en los mapas caóticos utilizados.

En la figura 4.1 se encuentra un diagrama de bloques del proceso de encriptación:

1. Se itera el mapa logístico con las condiciones iniciales basadas en la clave secreta de 128 bits y se realiza el cálculo de Z, la cual se obtiene de las secuencias caóticas generadas con el mapa logístico y la señal clara.

2. Se itera otro mapa logístico con base en el valor de Z, la clave secreta y las secuencias caóticas generadas con el primer mapa logístico para continuar con los procesos de confusión y difusión sobre la señal clara.

3. Finalmente agregar el valor de Z al criptograma para que el receptor pueda descifrar correctamente la señal enviada.

En la figura 4.2 se encuentra un diagrama de bloques del proceso de desencriptación, el cual consiste en invertir el proceso de encriptación mostrado:

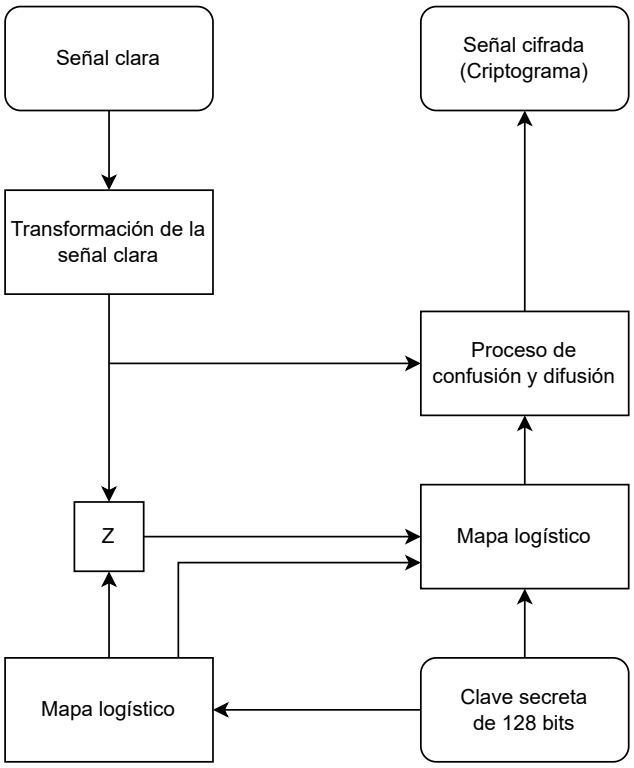


Figura 4.1: Diagrama de bloques del proceso de cifrado propuesto.

1. Se extrae el valor de Z del criptograma para después iterar el mapa logístico utilizando la clave secreta.

2. Con las secuencias caóticas generadas del paso previo y el valor de Z se procede a iterar otro mapa logístico para posteriormente ejecutar el proceso de descifrado sobre el criptograma (procesos de confusión y difusión invertidos).

3. Se realiza el proceso de transformación de la amplitud de manera inversa para obtener la señal clara original.

4.2. Transformación de la amplitud

Para procesar la señal clara, es necesario realizar una transformación para que los valores estén en el rango de 0 a 1. Esto se logra mediante la siguiente fórmula:

$$PT_i = \frac{P_i - (P_{min} - 0.01)}{N_{max} + 0.01}, \quad \text{para } i = 1, 2, 3, \ldots, \ell \tag{4.1}$$

Donde P es la señal clara, P_{min} es el valor mínimo de la señal clara (calculado como $P_{min} = min(P)$), N_{max} es el valor máximo del vector N (calculado como $N = P_i - (P_{min} - 0.01)$), y PT es la señal clara transformada con valores entre 0 y 1.

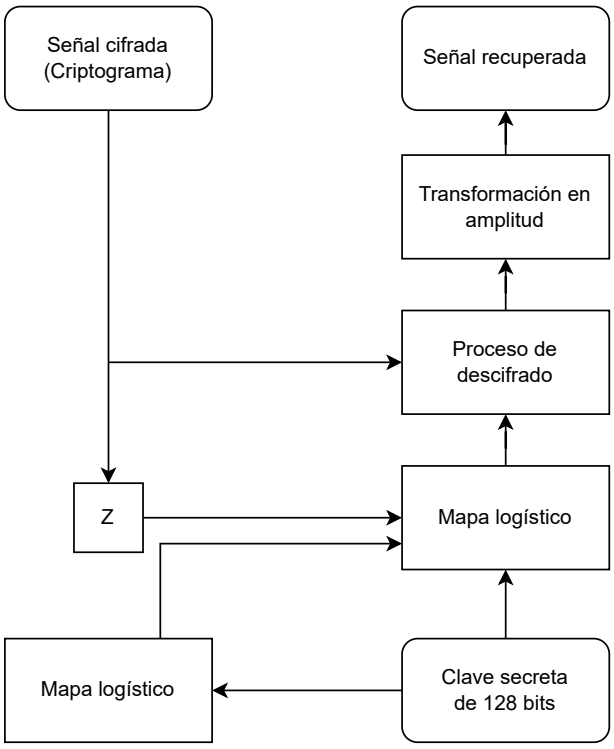


Figura 4.2: Diagrama de bloques del proceso de descifrado propuesto.

4.3. Definición de la clave secreta

La clave secreta K se compone por una secuencia de 128 bits de 32 caracteres hexadecimales de forma que $K \in [0-9, A-F]$. La clave es dividida en cuatro secciones (A,B,C,D) con las cuales se calculan las condiciones iniciales y parámetros de control de los mapas caóticos. Dichos cálculos se pueden observar en la tabla 4.1.

Tabla 4.1: Clave secreta propuesta.

Clave secreta	Parámetro de control	Condición inicial
32 dígitos Hex	$H_1, H_2, \ldots, H_{32}$ donde $H \in [0-9, A-F]$	
Cálculos	$A = \frac{(H_1,H_2,\ldots,H_8)_{10}}{2^{32}+1}$ $B = \frac{(H_9,H_{10},\ldots,H_{16})_{10}}{2^{32}+1}$	$C = \frac{(H_{17},H_{18},\ldots,H_{24})_{10}}{2^{32}+1}$ $D = \frac{(H_{25},H_{26},\ldots,H_{32})_{10}}{2^{32}+1}$
Logístico	$r_1 = 3.999999999999999$	$x_{1_0} = (A+B)\ mod\ 1$
Logístico 2	$r_2 = 3.999999999999999$	$x_{2_0} = (C+D+Z)\ mod\ 1$
Rango	$0 < x_{1_0,2_0} < 1$	
Precisión	10^{-15}	
donde $(a\ mod\ b) = (a-b) \times (a/b)$ con $b \neq 0$		

4.4. Cálculo de Z

El valor de Z se calcula para relacionar la señal clara con el algoritmo de cifrado y aumentar la sensibilidad del criptograma a la señal clara y la clave secreta. Para calcular Z, se itera el primer mapa logístico para obtener una secuencia de datos caóticos. Sin

embargo, la distribución de datos generados por el mapa logístico no es uniforme, lo que puede comprometer la seguridad del cifrado. Para solucionar esto, se optimiza la secuencia de datos eliminando los tres primeros dígitos después del punto decimal de cada dato. Luego, se realiza la suma de todos los elementos de la señal transformada junto con la secuencia de datos caóticos optimizados para obtener el valor de Z. La expresión para calcular Z es:

$$Z = \left\{ Z + [PT_i * x_{1_{I_2+1-i}}] + x^S_{1_{I_2+1-i}} \right\} \ mod\ 1, \quad para\ i = 1, 2, 3, \ldots, I_2 \tag{4.2}$$

Donde PT_i representa el elemento i de la señal transformada, Z es una variable inicializada en cero, y x^S corresponde a la secuencia caótica generada por el mapa logístico.

4.5. Proceso de descifrado

El proceso de descifrado consiste en aplicar en el criptograma el proceso inverso al realizado en el proceso de cifrado, es decir, invertir cada uno de los pasos realizados en este para recuperar la señal clara original. Para ello, se debe hacer uso de la misma clave utilizada en el proceso de cifrado. A continuación se detallan los pasos involucrados:

1. Recuperación de los valores de Z, P_{min} y N_{max}: Del criptograma se extraen los valores de Z, P_{min} y N_{max} que fueron agregados al final del criptograma durante el proceso de cifrado.

2. Generación de vectores de confusión y difusión: Utilizando la clave secreta y el valor de Z recuperado, se generan los vectores de confusión y difusión de manera similar al proceso de cifrado, utilizando los mapas logísticos.

3. Proceso de descifrado: Se realiza el proceso de descifrado de la señal cifrada utilizando los vectores de confusión y difusión. La expresión utilizada es el proceso inverso al de cifrado:

   $$DT_i(CF_i) = E_i - DF_i - DF_{\ell-i+1}, \quad para\ i = 1, 2, 3, \ldots, \ell \tag{4.3}$$

   donde DT_i es la señal recuperada escalada y E_i es la señal cifrada.

4. Proceso de transformación inversa: Se realiza el proceso inverso de transformación utilizando los valores de P_{min} y N_{max} recuperados para obtener la señal clara recuperada D.

4.6. Conclusiones del capítulo

En este capítulo se ha presentado un algoritmo de cifrado basado en caos para la transmisión segura de señales médicas a través de canales inseguros. El algoritmo utiliza mapas caóticos y una clave secreta de 128 bits para aumentar la complejidad de los

procesos de confusión y difusión. Además, se aprovecha la señal médica misma mediante el valor de Z para mejorar la robustez del algoritmo frente a ataques criptoanalíticos. Los resultados obtenidos demuestran que el algoritmo propuesto es capaz de generar secuencias cifradas con características pseudoaleatorias y distribución uniforme, lo que garantiza la confidencialidad de las señales médicas. Se concluye que el algoritmo de cifrado basado en caos es una solución efectiva y segura para proteger la privacidad de las señales médicas durante su transmisión.

Capítulo 5

Diseño e implementación del sistema de telemedicina seguro

En este capítulo, se presenta una introducción a los sistemas embebidos, así como una descripción del hardware utilizado en la estructura del sistema realizado para la implementación del algoritmo de cifrado caótico presentado en conjunto con los resultados obtenidos al transmitir a través del sistema una señal de electrocardiograma (ECG). Finalmente, se presenta el análisis de seguridad realizado para evaluar si es segura la transmisión de las señales biomédicas a través de internet.

5.1. Sistemas embebidos y microcontroladores

El avance de los sistemas embebidos ha impulsado el desarrollo de dispositivos médicos menos invasivos en el campo de la telemedicina. Estos dispositivos permiten la captura y almacenamiento de bioseñales en lugares remotos o incluso en la *nube* [55].

Un *sistema embebido* es un sistema de procesamiento de información integrado en productos para realizar tareas específicas [56]. Estos sistemas son ampliamente utilizados en una variedad de aplicaciones, como seguridad, entretenimiento, electrodomésticos, dispositivos médicos, automóviles, entre otros (ver figura 5.1). Son especialmente adecuados para aplicaciones que requieren un procesamiento rápido y en tiempo real.

Figura 5.1: Aplicaciones de los sistemas embebidos [57].

En cuanto a su estructura, los sistemas embebidos están compuestos por una unidad de control que ejecuta las tareas requeridas por la aplicación del sistema. Esta unidad está formada por un microcontrolador o microprocesador integrado en el sistema embebido. Estos sistemas ofrecen ventajas distintivas en comparación con las computadoras, como un tiempo de ejecución optimizado para una única aplicación, un costo reducido y un tamaño compacto [56].

Por otro lado, un *microcontrolador* se puede definir como un sistema de microcomputadora completo. Está compuesto por un circuito integrado que contiene en su interior un microprocesador, la memoria de datos (memoria RAM), la memoria de programa (memoria ROM/FLASH) y unidades de entrada/salida (ver figura 5.2). Los microcontroladores son ideales para aplicaciones de propósito específico debido a su tamaño compacto, bajo costo y sencillo uso [58].

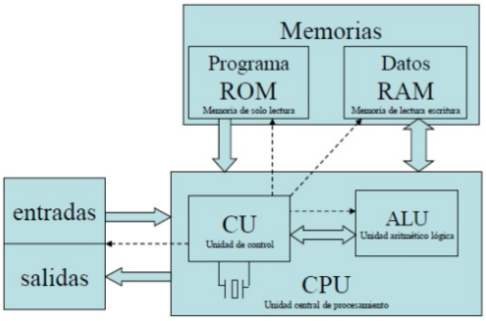


Figura 5.2: Componentes de un microcontrolador [59].

Los microcontroladores tienen diversas aplicaciones en el diseño de dispositivos electrónicos. Aunque se utilizan principalmente en el área de control, también son ampliamente utilizados en el procesamiento de información, especialmente en el campo de la seguridad y la criptografía [60].

5.2. Configuración del sistema

El sistema de telemedicina se encuentra compuesto de dos subsistemas, un bloque transmisor realizado a partir de un sistema embebido y un bloque receptor alojado dentro de un servidor web.

El proceso inicia en el bloque transmisor basado en sistema embebido, donde en primera instancia es almacenada la señal clara que va a ser transmitida mediante una memoria flash externa. Después el módulo solicita al usuario que presione un botón para ejecutar el algoritmo de cifrado propuesto sobre la señal para obtener el criptograma y posteriormente se transmite la señal cifrada a través de una conexión Wi-Fi hasta el servidor web donde se almacena el criptograma.

Por parte de la unidad receptora, se trata de una aplicación web accesible a través de internet, que interpreta el criptograma y tiene una interfaz gráfica que permite ingresar la clave de seguridad para proceder a desencriptar el criptograma. Al desencriptar el criptograma, es posible visualizar el resultado, así como descargarlo para su posterior uso.

5.2.1. Descripción del hardware utilizado

En la implementación del algoritmo de cifrado caótico en el bloque transmisor, se utilizó un ESP32-DevKit (ver figura 5.3), una placa de desarrollo basada en el microcontrolador ESP32 de Espressif System. Este microcontrolador de 32 bits cuenta con conectividad Wi-Fi y Bluetooth, así como una memoria Flash de 4 MB y 520 KB de memoria SRAM. Estas características hacen que el ESP32-DevKit sea una opción ideal para implementaciones IoT (Internet of Things) que requieren almacenamiento de código y comunicación inalámbrica.

Figura 5.3: Placa de desarrollo ESP32-DevKit.

El sistema transmisor se compone de tres elementos principales. En primer lugar, el ESP32-DevKit actúa como el sistema embebido encargado de realizar el proceso de cifrado y el manejo de los datos. En segundo lugar, se utiliza una memoria flash externa para almacenar la señal clara. Por último, se emplea una pantalla LCD de 16x2 para mostrar los mensajes correspondientes en cada etapa del proceso.

Para la lectura de la memoria flash, se utiliza un módulo genérico para lectura de tarjetas de memoria micro SD (ver figura 5.4(a)). Este módulo se comunica a través de una interfaz SPI y es compatible con voltajes de alimentación de 3.3V y 5V, lo que lo hace adecuado para aplicaciones en sistemas embebidos.

La interfaz con el usuario se realiza mediante una pantalla LCD de 16x2 (ver figura 5.4(b)), que se conecta a los pines de propósito general del ESP32-DevKit. Esta pantalla permite mostrar los mensajes y estados del proceso de transmisión.

En la implementación física, los componentes se interconectan de la siguiente manera: la pantalla LCD y el módulo de lectura de memoria micro SD se conectan a los pines de propósito general del ESP32-DevKit. Los pines utilizados para el módulo de lectura de memoria micro SD son los correspondientes a la comunicación SPI del microcontrolador.

(a) (b)

Figura 5.4: Componentes utilizados en el sistema diseñado: a)Módulo para lectura de tarjeta micro SD, b)LCD 16x2.

El diagrama de conexión eléctrica utilizado para el bloque transmisor se muestra en la figura 5.5. Este diagrama ilustra la interconexión de los componentes mencionados anteriormente.

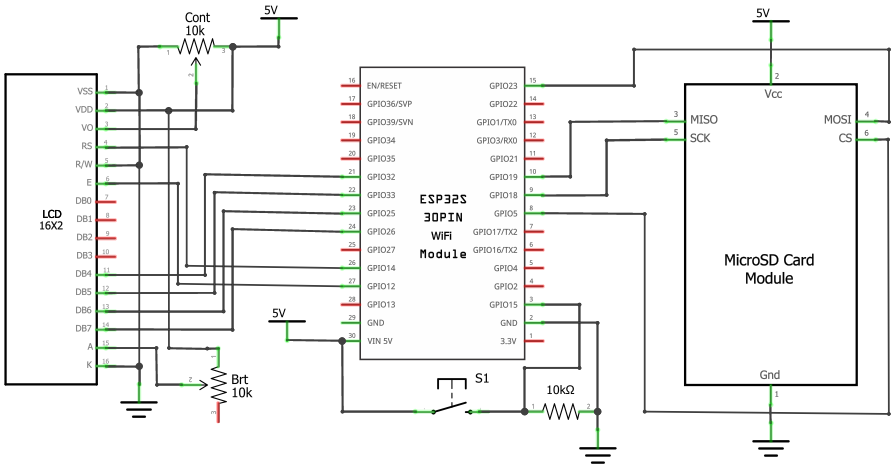


Figura 5.5: Diagrama de conexión eléctrica utilizado para el bloque transmisor [61].

5.2.2. Descripción del servidor web

El objetivo principal de esta tesis fue desarrollar un servidor web que permitiera alojar el sistema de desencriptación. El sistema transmisor se conecta al servidor web a través de una conexión wifi y envía la señal encriptada. El servidor web recibe el archivo y lo almacena para que cualquier persona dentro del grupo de seguridad pueda ingresar, ingresar la clave de encriptado y visualizar y/o descargar la señal clara.

Para lograr esto, se trabajó con dos lenguajes de programación diferentes: C para el sistema transmisor y PHP para el servidor web. C es el lenguaje de programación utilizado en la mayoría de los microcontroladores, mientras que PHP es un lenguaje de programación de código abierto ampliamente utilizado.

Uno de los principales desafíos al desarrollar el servidor web fue la compatibilidad de las operaciones aritméticas en un entorno caótico, es decir, con una gran sensibilidad al cambio incluso en valores pequeños. Esto requirió cambiar el algoritmo propuesto inicialmente para evitar el uso de funciones complejas, como las funciones trigonométricas.

Para la interfaz de usuario, se creó una página web utilizando HTML, CSS y JavaScript. Se utilizó el framework Bootstrap para obtener un diseño funcional y fácil de usar (ver figura 5.6). La página incluye la sección "Home", que muestra el electrocardiograma encriptado y proporciona un medio para realizar la desencriptación utilizando la clave. Al realizar la desencriptación, se muestra el resultado en otra gráfica en la misma página. Si el resultado es incorrecto, se muestra una gráfica encriptada (ruido); si es correcto, se muestra la señal clara. Además, se incluye un visualizador de gráficos para examinar en detalle cada una de las gráficas.

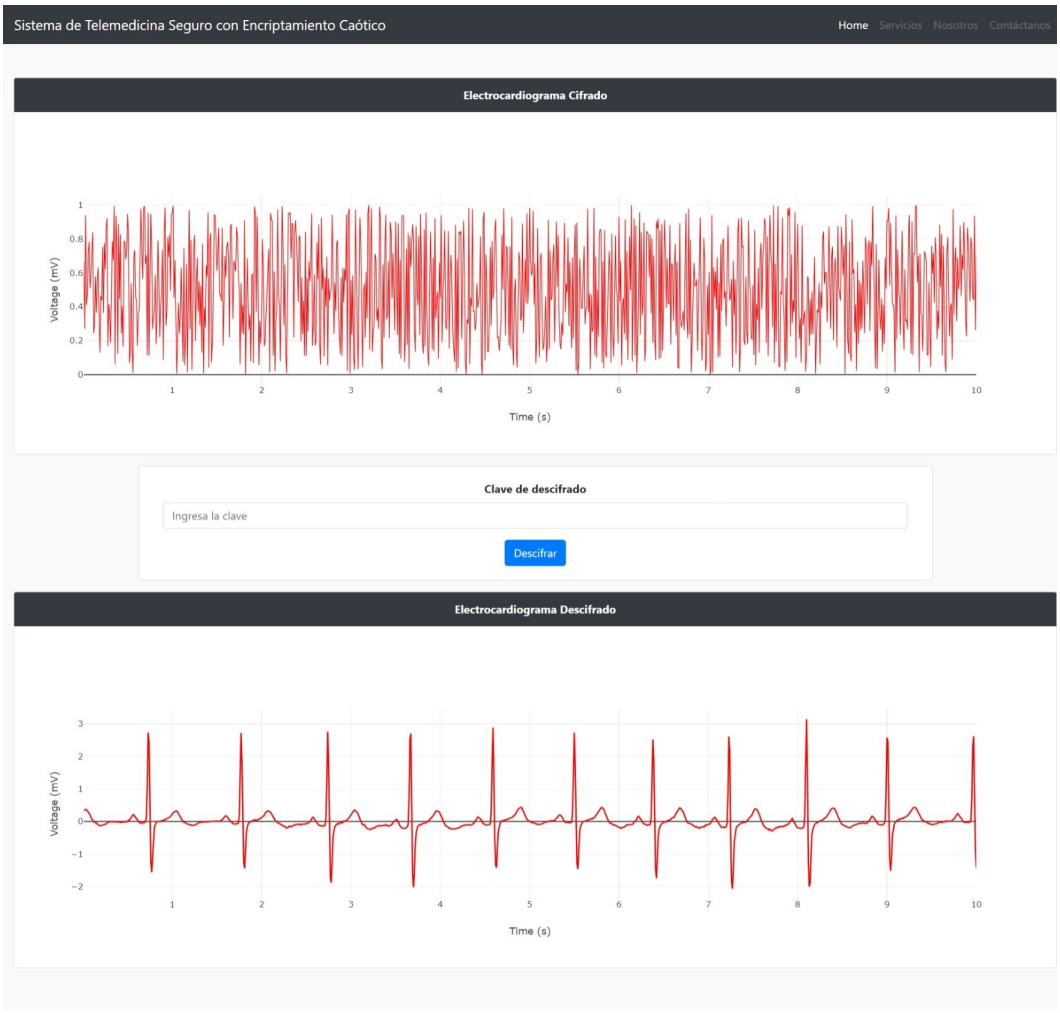


Figura 5.6: Interfaz de usuario de la página web.

5.2.3. Diagrama de bloques del sistema completo

En la figura 5.7 se muestra un diagrama a bloques del sistema completo, donde se pueden observar cada uno de los componentes del sistema diseñado y la comunicación utilizada entre cada uno de ellos como lo es la comunicación SPI entre los módulos para leer las memorias micro SD así como los pines digitales de propósito general del ESP32-DevKit para comunicarse con la pantalla LCD y el push button, además de la fuente de alimentación de cada módulo.

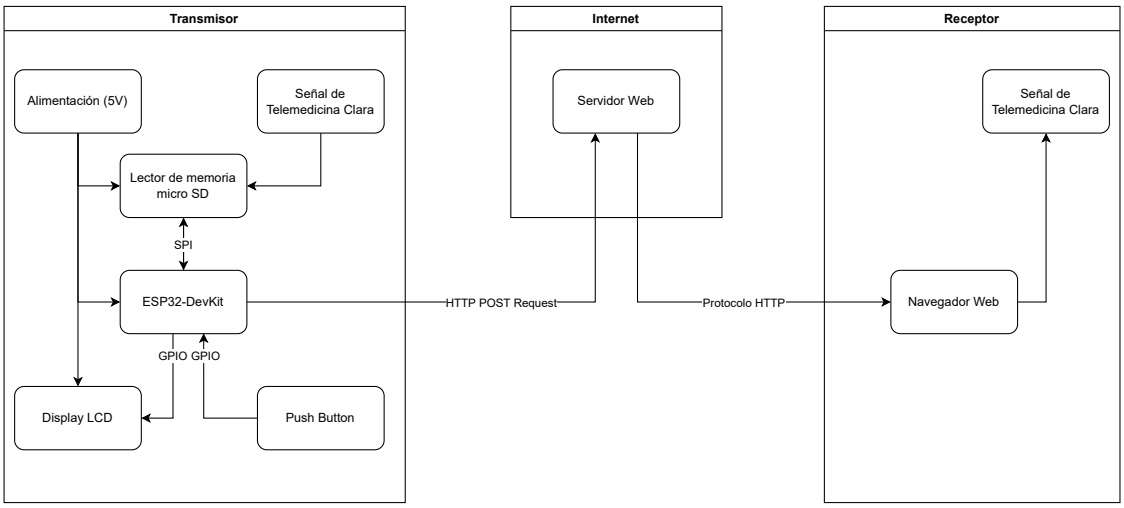


Figura 5.7: Diagrama a bloques del sistema completo.

5.3. Resultados experimentales

En esta sección se presentan los resultados experimentales obtenidos al evaluar el sistema propuesto.

5.3.1. Configuración del sistema

Para la evaluación de los resultados experimentales, se ensambló la electrónica correspondiente para el bloque transmisor del sistema. Se cargó el microcontrolador con su programa en lenguaje C utilizando el software Visual Studio Code. En la figura 5.8 se muestra el entorno de programación en Visual Studio Code con una parte del código del módulo transmisor.

Por otro lado, se realizó la programación correspondiente en lenguaje PHP para el servidor web. El servidor web se ejecutó en un servicio de hosting llamado Webhost000. En la figura 5.9 se puede observar el bloque transmisor y una computadora accediendo a la página web alojada.

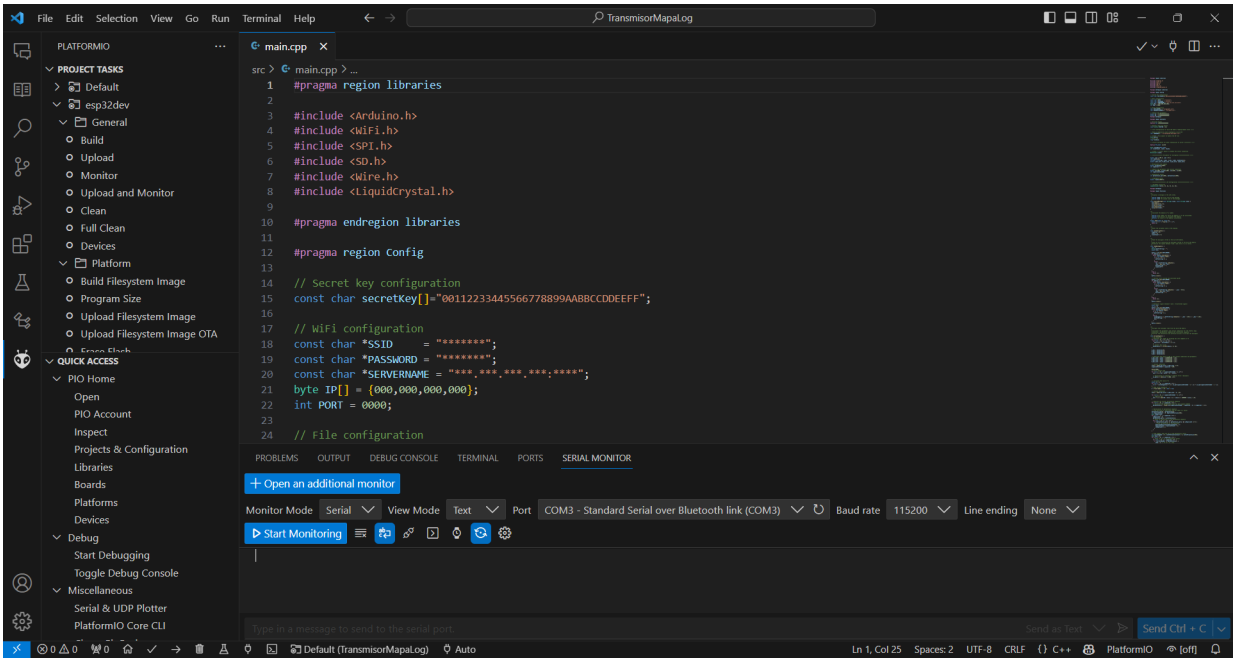
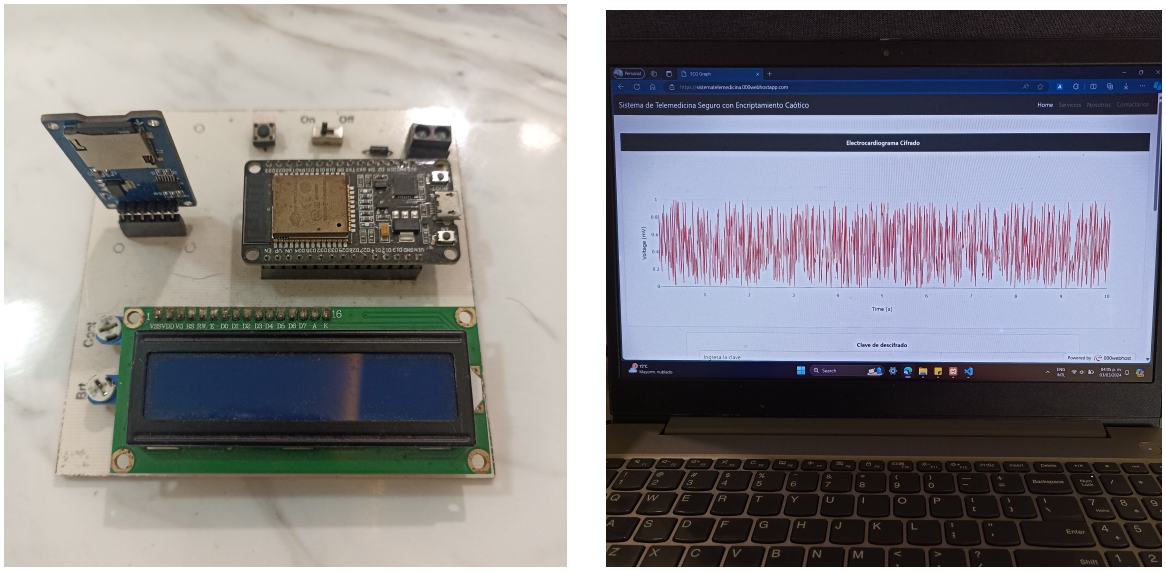


Figura 5.8: Interfaz de programación en software Visual Studio Code.



(a) (b)

Figura 5.9: Partes del sistema descrito: a)Bloque Transmisor, b)Bloque Receptor.

5.3.2. Señal de ECG

Para las pruebas experimentales, se utilizó una señal biomédica de electrocardiograma (ECG) obtenida de la base de datos PhysioBank ATM. La señal clínica tiene una duración de 10 segundos y una frecuencia de muestreo de $F_s = 100$ Hz. La señal utilizada se muestra en la figura 5.10.

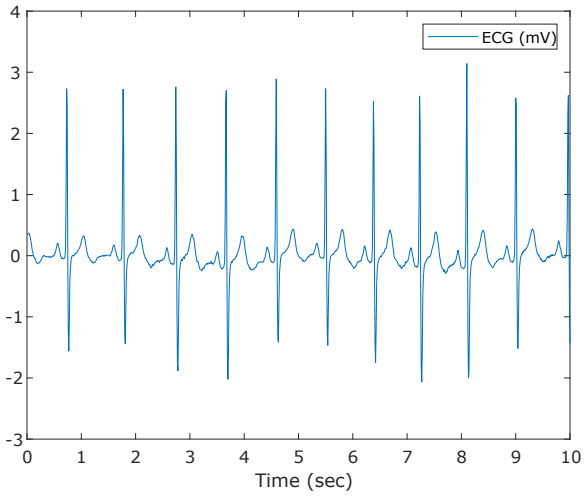
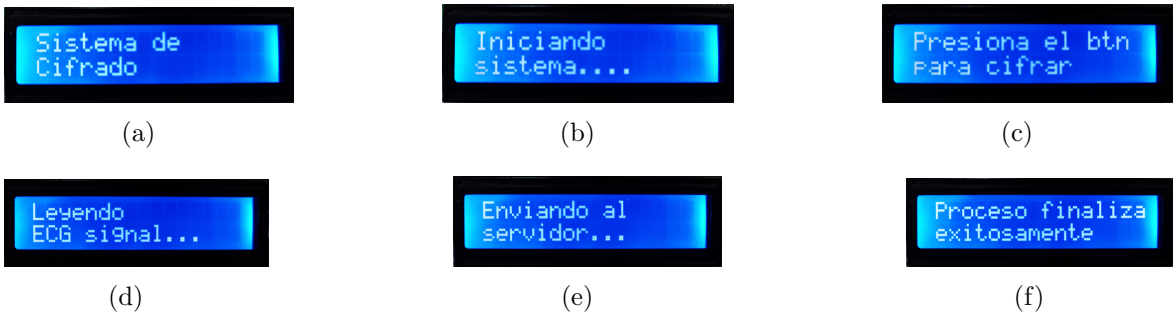


Figura 5.10: Señal ECG a transmitir.

5.3.3. Transmisión de la señal de ECG

El sistema transmisor almacena la señal ECG en una memoria micro-SD, la cifra y la envía al servidor web a través de la conexión a Internet. Se utiliza la clave secreta definida para la encriptación: **00112233445566778899AABBCCDDEEFF**. Durante el proceso, se muestran mensajes en la pantalla LCD del módulo transmisor para indicar el estado del proceso (figura 5.11).



(a) (b) (c)

(d) (e) (f)

Figura 5.11: Mensajes de inicio del módulo transmisor (a-c) y mensajes desplegados durante la ejecución (d-f).

El criptograma generado a partir de la señal ECG puede ser observado en la figura 5.12, el cual fue extraído del servidor web.

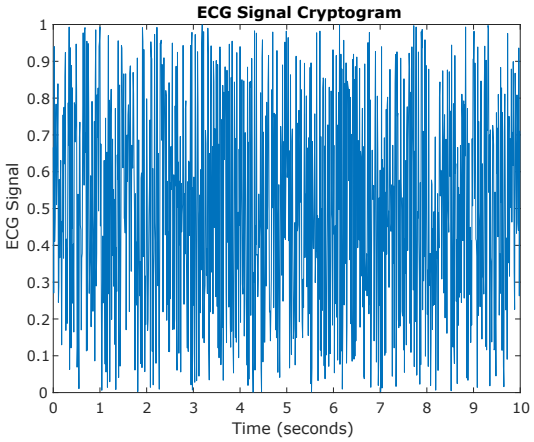


Figura 5.12: Criptograma generado a partir de la Señal ECG.

El sistema receptor procesa el criptograma para obtener la señal recuperada. En la figura 5.13 se muestra la señal recuperada mediante el servidor web y graficada en Matlab.

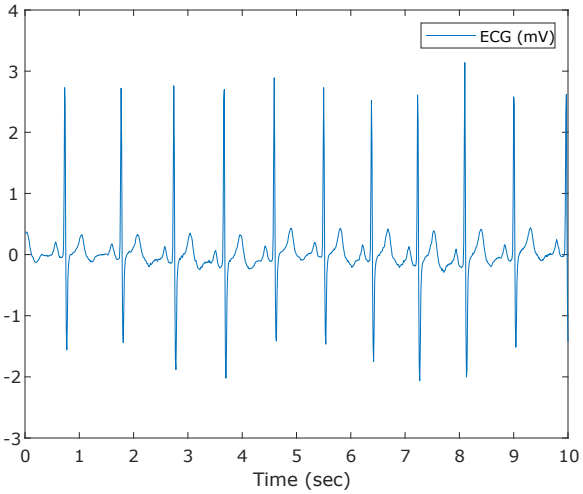


Figura 5.13: Señales recuperadas por el módulo receptor de la Señal ECG.

Para comprobar la integridad de la transmisión, se calcula el error entre la señal recuperada y la señal clara utilizando el software Matlab. Como se puede observar en la figura 5.14, el error es nulo, lo que indica que el cifrado, transmisión, recepción y descifrado de la señal médica se realizó de forma correcta.

5.4. Análisis de seguridad

En la sección anterior, se demostró que las señales biomédicas son cifradas de manera efectiva. Sin embargo, es crucial realizar análisis de seguridad adicionales para garantizar que los criptogramas transmitidos a través de los canales de comunicación

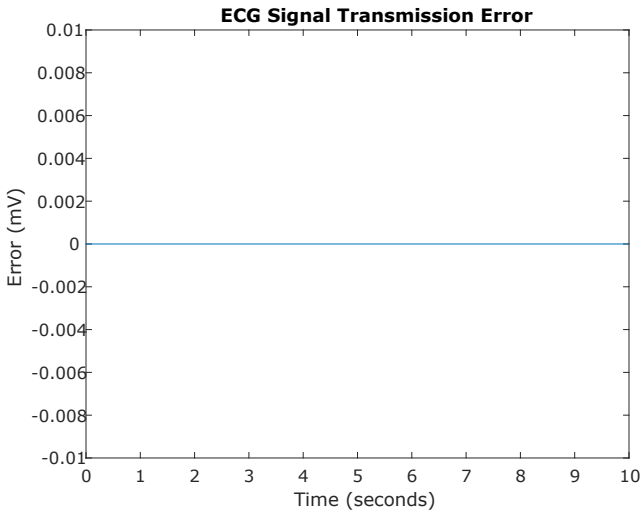


Figura 5.14: Error entre la señal clara y la señal recuperada por el bloque receptor.

sean seguros. Estos análisis deben confirmar que incluso si un individuo no autorizado obtiene los criptogramas, no podrá recuperar las señales claras utilizando métodos criptoanalíticos conocidos.

Con este fin, se han llevado a cabo cinco pruebas en diferentes criptogramas generados por el sistema embebido de cifrado caótico, así como en señales claras recuperadas del servidor web. Estas pruebas incluyen análisis de sensibilidad y análisis estadísticos, abarcando una amplia gama de posibles ataques criptoanalíticos que podrían comprometer la seguridad de las señales biomédicas.

5.4.1. Espacio de claves

Para garantizar la resistencia a un ataque exhaustivo, es fundamental que un algoritmo de cifrado seguro tenga un espacio de claves lo suficientemente amplio. Esto significa que debe haber una gran cantidad de claves posibles antes de encontrar la correcta para descifrar la señal. Según los estándares numéricos, se considera que un sistema criptográfico puede resistir un ataque exhaustivo si su espacio de claves es mayor a 2^{100} [62].

En nuestro caso, como se mencionó en la sección 4.3, la clave secreta utilizada consta de 128 bits, lo que equivale a 32 caracteres hexadecimales. Esto implica que el espacio de claves de nuestro algoritmo de cifrado es de 2^{128}, lo que lo hace altamente resistente a ataques de fuerza bruta.

5.4.2. Histogramas

El análisis de histogramas de la señal clara y el criptograma revela la fortaleza del algoritmo de cifrado frente a ataques estadísticos, destacando la necesidad de que el criptograma exhiba uniformidad en sus datos. En la figura 5.15 se muestra la distribución de frecuencia de los datos de la señal clara de ECG, donde se aprecia una distribución no uniforme. Por otro lado, en la figura 5.16 se observa que el criptograma correspondiente presenta una distribución más uniforme, lo que indica que el algoritmo criptográfico es resistente a ataques basados en histogramas.

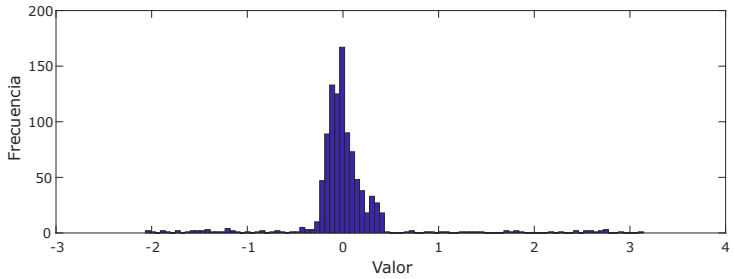


Figura 5.15: Histograma correspondiente a las señal clara transmitida.

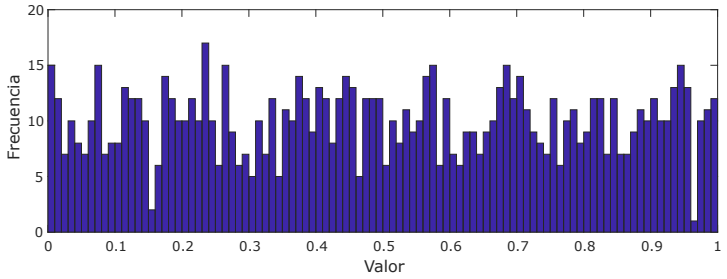


Figura 5.16: Histograma correspondiente al criptograma generado de la señal clara.

5.4.3. Correlación

La correlación es un indicador de la similitud entre el criptograma y la señal clara, revelando cuán diferentes son entre sí. Este análisis es crucial, ya que si el criptograma y la señal clara son similares, un criptoanalista podría descifrar tanto la señal clara como la clave secreta mediante métodos estadísticos. La correlación entre dos secuencias x e y se puede calcular utilizando la siguiente fórmula:

$$Cr = \frac{N \times \sum_{i=0}^{N}(x_i \times y_i) - \sum_{i=0}^{N} x_i \times \sum_{i=0}^{N} y_i}{\sqrt{\left(N \times \sum_{i=0}^{N}(x_i)^2 - \left(\sum_{i=0}^{n} x_i\right)^2\right) \times \left(N \times \sum_{i=0}^{N}(y_i)^2 - \left(\sum_{i=0}^{n} y_i\right)^2\right)}} \tag{5.1}$$

El coeficiente de correlación $Cr \in (-1, 1)$, donde $Cr = 0$ indica una correlación nula.

En nuestro análisis, generamos 50 criptogramas a partir de la señal ECG, cada uno con una clave distinta. Luego, calculamos la correlación de cada criptograma con su respectiva señal clara (ver figura 5.17). El coeficiente de correlación promedio para los criptogramas de ECG fue de $-9.289828296474962 \times 10^{-4}$, lo que indica una correlación casi nula. Por lo tanto, podemos concluir que, en promedio, los criptogramas generados son completamente diferentes a la señal clara.

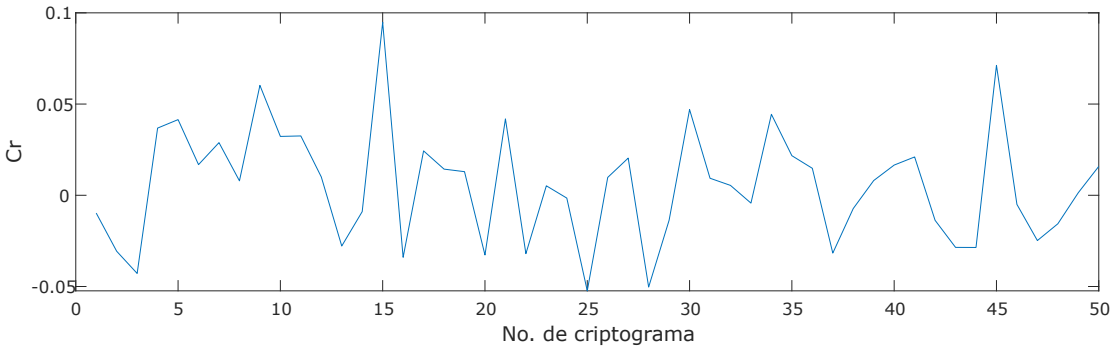


Figura 5.17: Coeficiente de correlación para 50 criptogramas distintos generados a partir de la señal ECG.

5.4.4. Sensibilidad a la clave secreta

Un sistema criptográfico efectivo debe ser altamente sensible a la clave secreta, de manera que un ligero cambio en un solo bit de la clave genere un criptograma completamente distinto a partir de la misma señal médica. Para probar esta sensibilidad, se generan dos criptogramas diferentes de ECG utilizando una CLAVE 1 y una CLAVE 2 (ver tabla 5.1), las cuales difieren en tan solo un bit. Luego, se calcula el coeficiente de correlación entre los criptogramas generados, como se hizo en la sección anterior.

Tabla 5.1: Claves secretas utilizadas para análisis de sensibilidad a la clave.

No. clave	Clave secreta
CLAVE 1	176AC15647C56D1A57EE6F176FEED167
CLAVE 2	176AC15647C56D1A57EE6F176FEED16**8**

Los valores de los coeficientes de correlación entre los criptogramas de ECG ($\mathrm{Cr}_{\mathrm{ECG}}$) generados con la CLAVE 1 y la CLAVE 2 se muestran en la tabla 5.2. Como se puede observar, el valor es cercano a cero, lo que indica una correlación nula entre los criptogramas. Esto demuestra que el algoritmo de cifrado propuesto es altamente sensible a la clave secreta.

Tabla 5.2: Resultados de análisis de correlación para determinar la sensibilidad a la clave secreta en el encriptado.

Prueba	Clave 1 vs. Clave 2
Cr_{ECG}	-0.014691012679597

5.4.5. Entropía de la información

La *entropía* es una medida estadística que nos permite evaluar el nivel de caos o desorden en los datos generados. Un valor alto de entropía indica que los datos son altamente caóticos, mientras que un valor bajo sugiere que los datos tienden a ser repetitivos y predecibles.

La entropía se calcula utilizando la siguiente fórmula:

$$H(m) = \sum_{i=0}^{2^N-1} p(m_i)\ log_2(1/p(m_i)), \tag{5.2}$$

donde N es el número de bits que representa la unidad básica del mensaje m, 2^N son todas las posibles combinaciones de la unidad básica, $p(m_i)$ es la probabilidad de m_i, y $\log_2$ es el logaritmo en base 2. La entropía se expresa en bits, y el valor máximo de entropía es N. Si un mensaje m está cifrado con 2^N posibles valores, la entropía ideal sería $H(m) = N$, lo que indica que el mensaje es completamente aleatorio.

En nuestro caso, estamos analizando los datos de los criptogramas generados. Estos datos están en el rango de $[0, 1]$, por lo que los transformamos a datos de 8 bits, es decir, en el rango de $[0, 255]$. De esta manera, la entropía máxima posible es de 8. Realizamos el análisis de entropía en 50 criptogramas generados a partir de 50 claves secretas diferentes para la señal ECG (ver figura 5.18). El resultado fue una entropía promedio de **7.795226826031191**. Dado que este valor está cerca de 8 (la entropía ideal), podemos concluir que los criptogramas generados tienen un alto nivel de desorden y pueden resistir ataques basados en entropía.

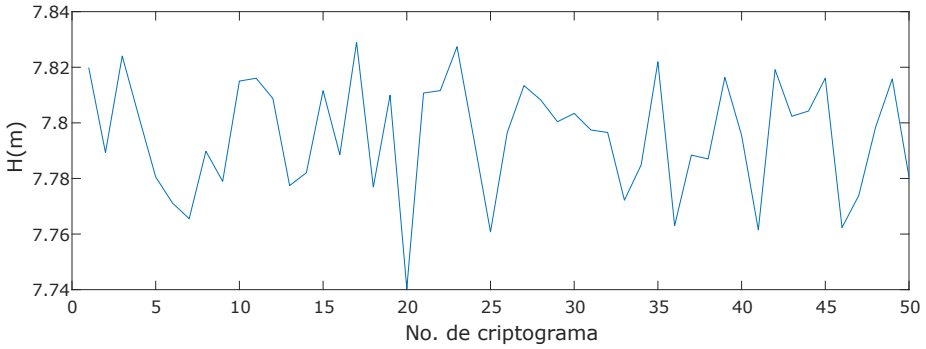


Figura 5.18: Entropía para 50 criptogramas distintos.

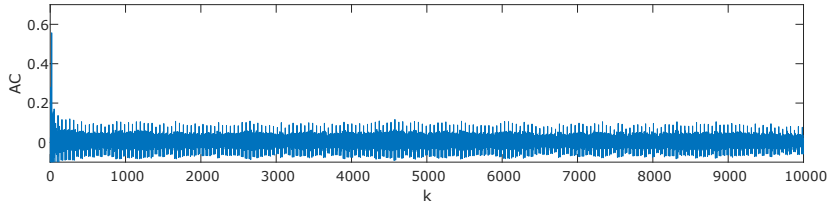
5.4.6. Autocorrelación

La autocorrelación es un cálculo utilizado en el procesamiento de señales para determinar si hay algún patrón o periodicidad en una señal al correlacionarla consigo misma con un desplazamiento de k posiciones. Se calcula utilizando la expresión:

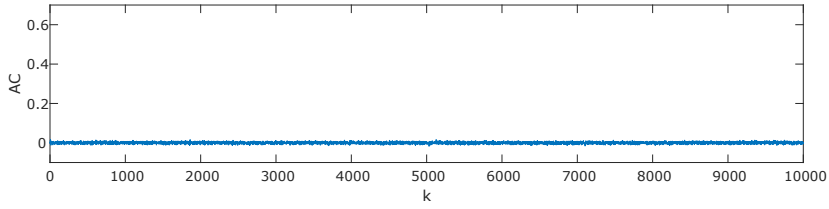
$$AC(k) = \frac{A - D}{T}, \tag{5.3}$$

donde $AC \in [-1, 1]$ es la autocorrelación de la señal desplazada k posiciones, A es el número de elementos que coinciden entre la señal original y la señal desplazada, D es el número de elementos que no coinciden, y T es la longitud de la señal. Los valores altos de autocorrelación indican la presencia de datos idénticos a nivel de bit, mientras que los valores negativos indican la presencia de valores opuestos. Una autocorrelación nula ($AC = 0$) indica un equilibrio entre la cantidad de 1's y 0's en la señal, lo cual es deseable al analizar criptogramas.

Para calcular la autocorrelación de la señal clara y el criptograma, primero se realiza una transformación de los datos a 64 bits y luego se calcula la autocorrelación de cada señal. En la figura 5.19 se muestra la autocorrelación de la señal clara y el criptograma para la señal ECG con un desplazamiento circular a la derecha de hasta $k = 10000$. Se observa que la señal clara presenta algunos patrones con valores positivos y negativos, mientras que el criptograma muestra una autocorrelación cercana a 0, lo que indica que el algoritmo de cifrado genera números pseudoaleatorios de forma uniforme.



(a) Señal ECG clara.



(b) Señal ECG cifrada.

Figura 5.19: Autocorrelación de señales utilizadas.

5.5. Conclusiones del capítulo

En este capítulo se diseñó e implementó un sistema de transmisión segura de señales biomédicas utilizando un microcontrolador ESP32-DevKit, una memoria flash y un servidor web. Se logró cifrar y transmitir la señal biomédica de manera segura, demostrando resistencia a ataques estadísticos y sensibilidad a la clave secreta. Los criptogramas generados presentaron alta entropía y autocorrelación cercana a cero, indicando la generación de números pseudoaleatorios uniformes. Este sistema ofrece una solución efectiva y segura para la transmisión de señales biomédicas en entornos médicos y telemedicina.

Capítulo 6

Conclusiones

6.1. Conclusiones generales

En la tesis presentada, se realizó un estudio exhaustivo sobre el diseño de un sistema de telemedicina seguro basado en criptografía caótica para el monitoreo y prevención temprana de enfermedades cardíacas. Se desarrolló un algoritmo de cifrado eficiente utilizando teoría del caos, que se implementó en un microcontrolador de bajo costo. Además, se configuró un servidor web remoto para el desencriptado seguro de las señales biomédicas recibidas y se realizaron pruebas de seguridad criptoanalíticas para evaluar la robustez del sistema.

Los resultados obtenidos demostraron que el sistema fue capaz de enviar y recibir señales biomédicas de manera exitosa y proporcionó un nivel de seguridad adecuado para utilizar un canal de transmisión inseguro como internet. Esto lo hace potencialmente viable para aplicaciones de telemedicina, permitiendo consultas remotas y monitoreo de signos vitales a distancia. Además, se llevó a cabo un exhaustivo análisis de la seguridad del sistema, evaluando su resistencia a posibles ataques criptoanalíticos. Se realizaron pruebas que confirmaron la robustez del algoritmo de cifrado basado en teoría del caos implementado en el microcontrolador, asegurando la protección de la privacidad de los datos biomédicos de los pacientes y garantizando la confidencialidad de la información transmitida.

Otro aspecto destacado de este trabajo es la transmisión de señales biomédicas entre un servidor web y un microcontrolador. Se implementó de forma exitosa un sistema eficiente y seguro que permitió la encriptación de señales ECG utilizando un microcontrolador de bajo costo. Además, se logró la transmisión de la señal encriptada hacia un servidor web y el desencriptado de estas señales mediante una clave en una interfaz web. Este sistema posibilitó la transmisión segura de datos biomédicos a través de internet, lo cual es fundamental para la implementación de sistemas de telemedicina en entornos clínicos y hospitalarios. Estos hallazgos representan un avance en el campo de la telemedicina y la seguridad de los datos biomédicos.

6.2. Trabajo a futuro

Como trabajo futuro, se propone explorar el uso de señales biomédicas en tiempo real. Esto podría implicar la integración de múltiples bioseñales en redes de área corporal (BAN) para proporcionar un monitoreo más completo y detallado de la salud del paciente. La capacidad de procesar y transmitir múltiples bioseñales en tiempo real podría permitir una detección más temprana de anomalías y una intervención médica más rápida.

Además, se sugiere investigar la aplicación de la criptografía caótica y la modulación en redes multiusuario. Esto podría permitir la transmisión segura de bioseñales de múltiples usuarios simultáneamente, lo que sería especialmente útil en centros médicos o clínicos donde se requiere el monitoreo de varios pacientes al mismo tiempo.

Finalmente, se podría considerar la expansión del sistema para incluir funcionalidades adicionales, como la capacidad de alertar a los profesionales médicos en caso de que se detecten anomalías en las bioseñales. Esto podría mejorar aún más la eficacia del sistema de telemedicina, permitiendo una respuesta más rápida a las emergencias médicas y mejorando los resultados de salud para los pacientes.

Bibliografía

[1] Almazyad I, Rao A, Rozenblit J. A framework for secure data management for medical devices. Simulation Series. 2020;52(1):639-50.

[2] Sinsky CA, Jerzak JT, Hopkins KD. Telemedicine and Team-Based Care: The Perils and the Promise. Mayo Clin Proc. 2021 Feb;96(2):429-37.

[3] Ryu WHA, Kerolus MG, Traynelis VC. Clinicians' user experience of telemedicine in neurosurgery during COVID-19. World neurosurgery. 2021;146:359-67.

[4] Zapateiro De la Hoz M, Acho L, Vidal Y. An experimental realization of a chaos-based secure communication using arduino microcontrollers. The Scientific World Journal. 2015:1-15.

[5] Zhang W, Zhu Z, Yu H. A symmetric image encryption algorithm based on a coupled logistic–bernoulli map and cellular automata diffusion strategy. Entropy. 2019;21(5):504-26.

[6] García-Guerrero EE, Inzunza-González E, López-Bonilla OR, Cárdenas-Valdez JR, Tlelo-Cuautle E. Randomness improvement of chaotic maps for image encryption in a wireless communication scheme using PIC-microcontroller via Zigbee channels. Chaos, Solitons y Fractals. 2020;133:109646-57.

[7] Seh AH, Zarour M, Alenezi M, Sarkar AK, Agrawal A, Kumar R, et al. Healthcare data breaches: Insights and implications. Healthcare. 2020;8(2):133-50.

[8] Wikina SB. What caused the breach? An examination of use of information technology and health data breaches. Perspectives in health information management. 2014;11:1-16.

[9] Hathaliya JJ, Tanwar S. An exhaustive survey on security and privacy issues in Healthcare 4.0. Computer Communications. 2020;153:311-35.

[10] Seh AH, Zarour M, Alenezi M, Sarkar AK, Agrawal A, Kumar R, et al.. Healthcare data breaches: Insights and implications; 2020. Figura 1.1, Representation of Data Breach Incidents; p. 5.

[11] Castillejo JAP. Telemedicina, una herramienta también para el médico de familia. Atención primaria. 2013;45(3):129-32.

[12] Herrera FP, Periche FF. Sistema de Telemedicina UdC: Un nuevo paradigma en la atención médica colombiana para el sur de Bolívar. Informática y Sistemas: Revista de Tecnologías de la Informática y las Comunicaciones. 2017;1(1):4-7.

[13] Guillén Pinto EP, Ramírez López LJ, Estupiñán Cuesta EP. Análisis de seguridad para el manejo de la información médica en telemedicina. Universidad Militar Nueva Granada; 2011. Figura 2.1, Típica Red de Telemedicina; p. 60.

[14] Alvez R. Aplicación de telemedicina para la mejora de los sistemas de emergencias y diagnósticos clínicos. Memoria Investigaciones en Ingeniería. 2011;9:91-7.

[15] Guillén Pinto EP, Ramírez López LJ, Estupiñán Cuesta EP. Análisis de seguridad para el manejo de la información médica en telemedicina. Ciencia e Ingeniería Neogranadina. 2011;21(2):57-89.

[16] Novillo-Ortiz D. Marco de Implementación de un Servicio de Telemedicina. Washington DC: Organización Panamericana de Salud. 2016.

[17] Ramos V. Contributions to the history of Telemedicine of the TICs. In: IEEE Conference on the History of Communications; 2010. p. 1-5.

[18] Zundel KM. Telemedicine: history, applications, and impact on librarianship. Bulletin of the Medical Library Association. 1996;84(1):71-9.

[19] Bashshur RL, Reardon TG, Shannon GW. Telemedicine: a new health care delivery system. Annual review of public health. 2000;21(1):613-37.

[20] Cánovas LPL, Cánovas LBL, Forcelledo AH. Telemedicina, impacto y perspectivas para la sociedad actual. Universidad Médica Pinareña. 2018;14(3):289-303.

[21] Roig F, Saigí F. Dificultades para incorporar la telemedicina en las organizaciones sanitarias: perspectivas analíticas. SciELO Public Health; 2009.

[22] Ibáñez CR, De Cadena ÁZ, Zea AT. Telemedicina: introducción, aplicación y principios de desarrollo. Ces Medicina. 2007;21(1):77-93.

[23] Peñafiel JM, Gil A, Azorín JM, Sabater JM, Pérez C, Morales R. Interfaz avanzada de un sistema de telecirugía. Universidad Miguel Hernández de Elche. 2007:1-7.

[24] Conaty-Buck S. Cybersecurity and healthcare records. American Nurse Today. 2017;12(9):62-4.

[25] Kim Dw, Choi Jy, Han Kh. Risk management-based security evaluation model for telemedicine systems. BMC medical informatics and decision making. 2020;20:1-14.

[26] Kim Dw, Choi Jy, Han Kh. Risk management-based security evaluation model for telemedicine systems. Springer; 2020. Figura 2.2, Seven areas related to telemedicine security threats; p. 4.

[27] Pérez E. Caos en el sistema solar. Miscelánea Matemática. 1997;27:59-69.

[28] Oestreicher C. A history of chaos theory. Dialogues in clinical neuroscience. 2007;9(3):279-89.

[29] Davies B. Exploring chaos: Theory and experiment. CRC Press; 2018.

[30] Santos Burguete C, Simarro Grande JP, Fuertes Marrón D. Física del caos. Agencia Estatal de Meteorología; 2018. Figura 3.1, Atractor de Lorenz; p. 60.

[31] Tachiquín MPR. Teoría del Caos: una visión de su historia y actualidad. Revista del Centro de Investigación de la Universidad la Salle. 2010;9(34):41-7.

[32] Schmitt-Grohé S, Uribe M. Deterministic debt cycles in open economies with flow collateral constraints. Journal of Economic Theory. 2021;192:105195.

[33] West BJ. Fractal physiology and the fractional calculus: a perspective. Frontiers in physiology. 2010;1:1886.

[34] Juan AM, José PG, Omar CS. Estabilidad de Sistemas No-lineales: Sistema de Nivel de Líquidos de Dos Tanques Interconectados. RIEE&C. 2008.

[35] Ceseña Villa A. Seguridad embebida en telemedicina basada en criptografía caótica. 2021.

[36] Dubeibe F. Cálculo del máximo exponente de Lyapunov con Mathematica. Revista Colombiana de Fısica. 2013;45(1):151-5.

[37] Xu Q, Sun K, Cao C, Zhu C. A fast image encryption algorithm based on compressive sensing and hyperchaotic map. Optics and Lasers in Engineering. 2019;121:203-14.

[38] Ceseña Villa A. Seguridad embebida en telemedicina basada en criptografía caótica. Universidad Autónoma de Baja California, Facultad de Ingeniería, Arquitectura y Diseño; 2021. Figura 3.2, Estado x del mapa SLIM con dinámicas caóticas; p. 18.

[39] Wu J, Liao X, Yang B. Image encryption using 2D Hénon-Sine map and DNA approach. Signal processing. 2018;153:11-23.

[40] Ceseña Villa A. Seguridad embebida en telemedicina basada en criptografía caótica. Universidad Autónoma de Baja California, Facultad de Ingeniería, Arquitectura y Diseño; 2021. Figura 3.3, Estado x del mapa Seno-Hénon con dinámicas caóticas; p. 20.

[41] Hrothgar. Chebfun examples collection; 2015. Available from: `https://github.com/chebfun/examples/blob/master/ode-nonlin/LyapunovExponents.m`.

[42] Dodis Y. Exposure-resilient cryptography. Massachusetts Institute of Technology; 2000.

[43] Menezes AJ, Van Oorschot PC, Vanstone SA. Handbook of applied cryptography. CRC press; 2018.

[44] Hermosilla JCH. Breve historia del Espionaje NE revisada y ampliada. A color. Nowtilus; 2022.

[45] García Arnau M. Criptografía clásica.¿ Cómo romper cifrados monoalfabéticos y polialfabéticos? Análisis de frecuencias y método Kasiski. Buran. 2003;(19):95-7.

[46] Nasution SD, Ginting GL, Syahrizal M, Rahim R. Data security using vigenere cipher and goldbach codes algorithm. Int J Eng Res Technol. 2017;6(1):360-3.

[47] United States Government Work. Enigma Cipher Machine During World War II; 2011. Image retrieved from Flickr. https://www.flickr.com/photos/ciagov/5416145081/sizes/o/in/photostream/.

[48] Padrón A, Prieto R, Herrera A, Calva G. Implementación de Protocolos de Comunicación Seguros: Escenarios. In: Congreso de Instrumentación. vol. 29; 2014. p. 1-9.

[49] Katz J, Lindell Y. Introduction to modern cryptography. 3rd ed. Boca Raton, FL: CRC Press; 2020.

[50] Li H, Deng L, Gu Z. A robust image encryption algorithm based on a 32-bit chaotic system. IEEE Access. 2020;8:30127-51.

[51] Munir N, Khan M, Jamal SS, Hazzazi MM, Hussain I. Cryptanalysis of hybrid secure image encryption based on Julia set fractals and three-dimensional Lorenz chaotic map. Mathematics and Computers in Simulation. 2021;190:826-36.

[52] Pushpalatha G, Ramesh S. WITHDRAWN: Chaotic based encryption algorithms for speech signal and cryptographic requirements: A brief survey. Elsevier; 2021.

[53] Wang J, Han K, Fan S, Zhang Y, Tan H, Jeon G, et al. A logistic mapping-based encryption scheme for wireless body area networks. Future Generation Computer Systems. 2020;110:57-67.

[54] Pandey A, Singh B, Saini BS, Sood N. A novel fused coupled chaotic map based confidential data embedding-then-encryption of electrocardiogram signal. Biocybernetics and Biomedical Engineering. 2019;39(2):282-300.

[55] Michel-Macarty J, Murillo-Escobar M, López-Gutiérrez RM, Cruz-Hernández C, Cardoza-Avendaño L. Multiuser communication scheme based on binary phase-shift keying and chaos for telemedicine. Computer methods and programs in biomedicine. 2018;162:165-75.

[56] Marwedel P. Embedded system design: embedded systems foundations of cyber-physical systems, and the internet of things. Springer Nature; 2021.

[57] Ceseña Villa A. Seguridad embebida en telemedicina basada en criptografía caótica. Universidad Autónoma de Baja California, Facultad de Ingeniería, Arquitectura y Diseño; 2021. Figura 5.1, Aplicaciones de los sistemas embebidos; p. 39.

[58] Rossano V. Electrónica & microcontroladores PIC. USERSHOP; 2009.

[59] Medina G V. Microcontroladores ver2.0; 2014. Figura 5.2, Diagrama en bloques de un procesador; p. 10. Available from: `https://es.slideshare.net/vicomg/microcontroladores-ver20`.

[60] Janakiraman S, Thenmozhi K, Rayappan JBB, Amirtharajan R. Lightweight chaotic image encryption algorithm for real-time embedded system: Implementation and analysis on 32-bit microcontroller. Microprocessors and Microsystems. 2018;56:1-12.

[61] Ceseña Villa A. Seguridad embebida en telemedicina basada en criptografía caótica. Universidad Autónoma de Baja California, Facultad de Ingeniería, Arquitectura y Diseño; 2021. Figura 5.5, Diagrama de conexión eléctrica utilizado para los bloques transmisor y receptor; p. 43.

[62] Zhu C. A novel image encryption scheme based on improved hyperchaotic sequences. Optics communications. 2012;285(1):29-37.

Apéndice A

Programa para módulo transmisor

En este apéndice se presenta el código en lenguaje C con el que se programó la placa ESP32 correspondiente al módulo transmisor para el cifrado de la señal médica y la transmisión del criptograma al servidor web a través de la conexión Wi-Fi.

```
1  #pragma region libraries
2
3  #include <Arduino.h>
4  #include <WiFi.h>
5  #include <SPI.h>
6  #include <SD.h>
7  #include <Wire.h>
8  #include <LiquidCrystal.h>
9
10 #pragma endregion libraries
11
12 #pragma region Config
13
14 // Secret key configuration
15 const char secretKey[]="1122334455667788990OAABBCCDDEEFF";
16
17 // WiFi configuration
18 const char *SSID     = "*******";
19 const char *PASSWORD = "*******";
20 const char *SERVERNAME = "***.***.***.***:****";
21 byte IP[] = {000,000,000,000};
22 int PORT = 0000;
23
24 // File configuration
25 char SDFILENAME[] = "/original.csv";
26 char SERVERFILENAME[] = "cifrado.csv";
27
28 // Chaotic map parameters
29 double R1 = 3.999999999999999;
30 double R2 = 3.999999999999999;
31 #pragma endregion
32
33 #pragma region Constants
34
35 // Division factor
36 #define E15 1000000000000000
```

```
37
38  // Digital input for button
39  const uint8_t BUTTON = 15;
40
41  // ***** Configuration of micro-SD memory reading module ***** /////
42
43  // Name to temporarily save cryptogram in micro-SD
44  char TEMPNAME[] = "/Criptograma_Biosignal.csv";
45
46  // Create a File object to handle the SD file
47  File myFile;
48  // File size
49  long fileSize;
50
51  // ******* Variables for data transmission to server ********** /////
52
53  #define MTU_SIZE  2*1760
54
55  byte clientBuf[MTU_SIZE];
56  int clientCount, count, chunks;
57
58  // Create a "client" object to handle the server connection
59  WiFiClient client;
60
61  // ***************** Variables for encryption ***************** /////
62
63  double _min = 200.0, _max = 0.0;
64  char key[4][8];
65  unsigned long A_dec, B_dec, C_dec, D_dec, hexValues[4];
66  double A_dec_norm, B_dec_norm, C_dec_norm, D_dec_norm;
67
68  // Normalized clear signal
69  double normSignal[1000];
70  int signalLen = 0;
71
72  // Chaotic map parameters and initial conditions
73  double x1[1100], x1_opt[1100], x2[1100], y[1100];
74  int signalLenExtended;
75
76  // Permutation vector
77  int permutation_opt[1000], permutation[1000];
78
79  // Cryptogram vector
80  double crypto[1000+5];
81
82  // ********************* LCD Configuration ****************** /////
83
84  // LCD-ESP32 Connection
85  LiquidCrystal lcd(12, 14, 32, 33, 25, 26);
86
87  #pragma endregion
88
89  #pragma region Functions
90
91  /**
92   * Displays a message on the LCD screen.
93   *
94   * @param line1 The first line of the message.
95   * @param line2 The second line of the message.
```

```
 */
void displayMessage(const String& line1, const String& line2) {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(line1);
  lcd.setCursor(0, 1);
  lcd.print(line2);
}

/**
 * Calculate the modulus of a number.
 *
 * @param X The number for which the modulus is to be calculated.
 * @param Y The divisor to calculate the modulus.
 * @return The result of the modulus calculation.
 */
double mod(double X, float Y){
  double z = X - ( floor(X) / Y ) * Y;
  return z;
}

/**
 * Reset the variables used in the program.
 */
void resetVariables() {
  signalLen = 0;
  count = 0;
  clientCount = 0;
}

/**
 * Read the biosignal stored in the micro-SD memory.
 *
 * Read the file containing the biosignal stored in the micro-SD memory.
 * Normalize the signal between 0 and 1 and store it in a vector.
 */
void readBioSignal() {
  char c;
  String bufferString = "";
  double var;

  myFile = SD.open(SDFILENAME);
  if (myFile) {
    while (myFile.available()) {
      c = (char)myFile.read();
      if (c != '\n') {
        bufferString += c;
      }
      else {
        var = bufferString.toDouble();
        _min = min(_min, var);
        bufferString = "";
        signalLen++;
      }
    }
  }
  else {
    while (1);
  }
```

```
155     myFile.close();
156
157     // Perform first scaling and calculate max(N)
158     myFile = SD.open(SDFILENAME);
159     if (myFile) {
160       while (myFile.available()) {
161         c = (char)myFile.read();
162         if (c != '\n') {
163           bufferString += c;
164         }
165         else {
166           var = bufferString.toDouble() - (_min - 0.01);
167           _max = max(_max, var);
168           bufferString = "";
169         }
170       }
171     }
172     else {
173       while (1);
174     }
175     myFile.close();
176
177     // Normalize signal between 0 and 1 (transformed signal)
178     double val;
179     int i = 0;
180     myFile = SD.open(SDFILENAME);
181     while (myFile.available()) {
182       c = (char)myFile.read();
183       if (c != '\n') {
184         bufferString += c;
185       }
186       else {
187         normSignal[i] = (bufferString.toDouble() - (_min - 0.01)) / (_max + 0.01);
188         bufferString = "";
189         i++;
190       }
191     }
192     myFile.close();
193   }
194
195   /**
196    * Encrypts the biosignal read from the micro-SD memory.
197    *
198    * Calculates the parameters and initial conditions for the chaotic maps.
199    * Generates a permutation sequence and a diffusion sequence.
200    * Performs the permutation and diffusion process to encrypt the biosignal.
201    */
202   void encryptSignal() {
203     // Key handling
204     // Determine key ranges by dividing key into segments of 8
205     for (int i = 0; i < 4; i++) {
206       for (int j = 0; j < 8; j++) {
207         key[i][j] = secretKey[i * 8 + j];
208       }
209       // Convert hex to dec
210       hexValues[i] = strtoul(key[i], 0, 16);
211     }
212
213     A_dec = hexValues[0];
```

```
214     B_dec = hexValues[1];
215     C_dec = hexValues[2];
216     D_dec = hexValues[3];
217
218     // Determine the value to add in the initial conditions and parameters
219     A_dec_norm = A_dec / (4294967296 + 1.0);
220     B_dec_norm = B_dec / (4294967296 + 1.0);
221     C_dec_norm = C_dec / (4294967296 + 1.0);
222     D_dec_norm = D_dec / (4294967296 + 1.0);
223
224     // Initial conditions
225     x1[0] = mod(A_dec_norm + B_dec_norm, 1.0);
226     signalLenExtended = signalLen + 100;
227
228     delay(5000);
229     // First Logistic map
230     for (int i = 0; i < signalLenExtended - 1; i++) {
231       x1[i + 1] = R1 * x1[i] * (1 - x1[i]);
232
233       // Optimization of sequence X (removes first 3 decimals)
234       x1_opt[i] = mod(x1[i] * 1000, 1.0);
235     }
236
237     // Calculate the value of Z
238     double Z = 0;
239     for (int i = 0; i < signalLen; i++) {
240       Z = Z + ((normSignal[i] + 1) * x1_opt[signalLenExtended - 1 - i]) + x1_opt[
              signalLenExtended - 1 - i];
241     }
242     // Keep values between 0 - 1
243     Z = floor(mod(Z, 1.0) * E15) / E15;
244
245     // Initial conditions
246     x2[0] = mod(C_dec_norm + D_dec_norm + Z, 1.0);
247
248     for (int i = 0; i < signalLenExtended - 1; i++) {
249       // Logistic map
250       x2[i + 1] = mod((R2 * x2[i] * (1 - x2[i])) * 1000000 * x1[i], 1.0);
251     }
252
253     // Generate the initial permutation sequence
254     for (int i = 0; i < signalLen; i++) {
255       // Calculate the permutation index for each signal element
256       permutation[i] = round((x2[(signalLenExtended) - signalLen + i] * (signalLen
              - 1)));
257     }
258
259     // Optimization of permutation sequence
260     // Find positions of duplicates and put them in a vector
261     permutation_opt[0] = permutation[0];
262     int duplicateCount = 0, duplicatePositions[500];
263     byte isDuplicate;
264     for (int i = 1; i < signalLen; i++) {
265       isDuplicate = 0;
266       permutation_opt[i] = permutation[i];
267       // Check for duplicates in the permutation sequence
268       for (int j = 0; j < (i); j++) {
269         if (permutation_opt[i] == permutation_opt[j] && isDuplicate == 0) {
270           // Save the position of the duplicate
```

```
271           duplicatePositions[duplicateCount] = i;
272           duplicateCount = duplicateCount + 1;
273           isDuplicate = 1;
274         }
275       }
276     }
277
278     // Find numbers that are not in the permutation vector
279     int currentNumber = 0, notInPermutationCount = 0, notInPermutation[500];
280     byte skip = 0;
281     for (int i = 0; i < signalLen; i++) {
282       // Check if it's in the permutation vector
283       for (int j = 0; j < signalLen; j++) {
284         if (currentNumber == permutation[j]) {
285           skip = 1; // If it's in, skip.
286         }
287       }
288       // If it's not in duplicates and permutation vector, enter
289       if (skip == 0) {
290         notInPermutation[notInPermutationCount] = currentNumber; // Save number.
291         notInPermutationCount = notInPermutationCount + 1;
292       }
293       skip = 0;
294       currentNumber = currentNumber + 1;
295     }
296
297     // Update permutation vector (has all the spaces)
298     int duplicatePositionsSize = duplicateCount;
299     byte halfSize = round(duplicatePositionsSize / 2);
300     if (duplicatePositionsSize % 2 != 0) {
301       halfSize = halfSize + 1;
302     }
303
304     byte change = 1;
305     int S = 0;
306     for (int i = 0; i < duplicatePositionsSize; i++) {
307       if (change == 1) {
308         permutation[duplicatePositions[i]] = notInPermutation[S];
309         change = 0;
310       }
311       else {
312         permutation[duplicatePositions[i]] = notInPermutation[S + halfSize];
313         S = S + 1;
314         change = 1;
315       }
316     }
317
318     // SEQUENCE FOR DIFFUSION
319     for (int i = 0; i < signalLen; i++) {
320       y[i] = mod((x2[signalLenExtended - signalLen + i] * 1000) + Z, 1.0);
321       y[i] = floor(y[i] * E15) / E15;
322     }
323
324     // PERMUTATION and DIFFUSION PROCESS (ENCRYPTION)
325     for (int i = 0; i < signalLen; i++) {
326       crypto[i] = mod(normSignal[permutation[i]] + y[i] + y[signalLen - i - 1],
              1.0);
327     }
328
```

```
329     // Add the last 5 data to the end of the cryptogram
330     crypto[signalLen] = Z;
331     if (_min < 0) {
332       crypto[signalLen + 1] = 1.000000000000000; // If it's -, it's 1
333     }
334     else {
335       crypto[signalLen + 1] = 0.000000000000000; // If it's +, it's 0
336     }
337     crypto[signalLen + 2] = abs(_min) / 100000.0;
338     if (_max < 0) {
339       crypto[signalLen + 3] = 1.000000000000000; // If it's -, it's 1
340     }
341     else {
342       crypto[signalLen + 3] = 0.000000000000000; // If it's +, it's 0
343     }
344     crypto[signalLen + 4] = abs(_max) / 100000.0;
345
346   }
347
348   /**
349    * Sends the cryptogram to the server.
350    *
351    * Displays a message on the LCD screen indicating that the cryptogram is being
         sent.
352    * Saves the cryptogram to a file in the micro-SD memory.
353    * Calculates the file size and the number of chunks.
354    * Performs the file transmission to the local server.
355    * Displays a message on the LCD screen indicating that the process has finished
         .
356    */
357   void sendCryptogram() {
358     // Display message "Sending to server..."
359     displayMessage("Enviando al", "servidor...");
360     delay(1000);
361
362     // Save cryptogram to a file in micro-SD memory
363     myFile = SD.open(TEMPNAME, FILE_WRITE);
364     for (int i = 0; i < signalLen + 5; i++) {
365       myFile.println(crypto[i], 15);
366     }
367     myFile.close();
368
369     // Calculate file size
370     myFile = SD.open(TEMPNAME, FILE_READ);
371     fileSize = myFile.size();
372     chunks = fileSize / (16 * MTU_SIZE);
373     count = 0;
374
375     // Perform file transmission to local server
376     if (client.connect(IP, PORT)) {
377       // HTTP POST request:
378       String head = "----84989444e2484915a216e1718e0f93f0\r\nContent-Disposition:
              form-data; name=\"FileGDF\"; filename=\"" + String(SERVERFILENAME) + "
              \"\r\nContent-Type: application/octet-stream\r\n\r\n";
379       String tail = "\r\n----84989444e2484915a216e1718e0f93f0--\r\n";
380
381       client.println("POST /tesis/upload.php?up=1&ACM=1 HTTP/1.1");
382       client.print("Host: ");
383       client.println(SERVERNAME);
```

```
384       client.println("Connection: close");
385       client.println("Content-Type: multipart/form-data; boundary=--84989444
              e2484915a216e1718e0f93f0");
386       client.print("Content-Length: "); client.println(myFile.size() + head.length
              () + tail.length());
387       client.println();
388       client.println("----84989444e2484915a216e1718e0f93f0");
389       client.println("Content-Disposition: form-data; name=\"FileGDF\"; filename
              =\"" + String(SERVERFILENAME) + "\"");
390       client.println("Content-Type: application/octet-stream");
391       client.println();
392
393       client.setNoDelay(1);
394       while (myFile.available()) {
395         if (myFile.available() >= MTU_SIZE) {
396           clientCount = MTU_SIZE;
397         }
398         else {
399           clientCount = myFile.available();
400         }
401
402         myFile.read(&clientBuf[0], clientCount);
403
404         client.write((const uint8_t*)&clientBuf[0], clientCount);
405         count++;
406         if (count == chunks) {
407           count = 0;
408         }
409       }
410
411       client.println();
412       client.println("----84989444e2484915a216e1718e0f93f0--"); // form end
413       client.println();
414
415       myFile.close();
416       // Remove cryptogram from micro-SD memory
417       SD.remove(TEMPNAME);
418     }
419     else {
420       Serial.println("fail");
421     }
422
423     // Server response
424     while (client.available()) {
425       String line = client.readStringUntil('\r');
426       Serial.println(line);
427     }
428     client.stop();
429
430     // Display message "Process finished!"
431     displayMessage("Proceso finalizado", "exitosamente");
432
433     delay(5000);
434   }
435
436   #pragma endregion
437
438   #pragma region Main
439
```

```
440
441  /**
442   * @brief Initializes the system and configures the necessary components.
443   *
444   * This function is called once at the beginning of the program execution.
445   * It initializes the LCD, creates custom characters, displays startup messages,
446   * configures inputs and outputs, initializes the SD card, and connects to WiFi.
447   *
448   * @return void
449   */
450  void setup() {
451    // LCD Initialization
452    Serial.begin(115200);
453    lcd.begin(16,2); // LCD 16 columns 2 lines
454
455    // Display "ENCRYPTION SYSTEM"
456    displayMessage("Sistema de", "Cifrado");
457    delay(4000);
458
459    // Display "Initializing system...."
460    displayMessage("Iniciando", "sistema....");
461    delay(2000);
462
463    // Configure inputs and outputs
464    pinMode(BUTTON, INPUT);
465
466    // Initialize SD card
467    if (!SD.begin()) {
468      while(1);
469    }
470
471    // Initialize WiFi
472    WiFi.begin(SSID, PASSWORD);
473    while(WiFi.status() != WL_CONNECTED)
474    {
475      delay(300);
476    }
477
478    delay(2000);
479  }
480
481  /**
482   * @brief The main loop of the program.
483   *
484   * This function is responsible for executing the main logic of the program.
485   * It resets the variables, displays a message to prompt the user to press a
        button,
486   * waits for the button to be pressed, reads the bio signal, encrypts the signal
        ,
487   * and sends the cryptogram.
488   */
489  void loop() {
490    resetVariables();
491
492    // Display message "Press the button to encrypt"
493    displayMessage("Presiona el btn", "para cifrar");
494    Serial.println("Press the button to encrypt");
495    // Wait for the button to be pressed
496    while (!(digitalRead(BUTTON) == true));
```

```
497
498     // Display message "Reading bio signal..."
499     displayMessage("Leyendo", "ECG signal...");
500     delay(1000);
501
502     readBioSignal();
503     Serial.println("Signal read");
504     encryptSignal();
505     Serial.println("Signal encrypted");
506     sendCryptogram();
507     Serial.println("Cryptogram sent");
508   }
509
510   #pragma endregion
```

Apéndice B

Programa para Servidor Web

En este apéndice se presenta la programación realizada para el funcionamiento del servidor web. Para ello, se implementaron dos archivos:

El primer archivo, llamado *upload.php*, contiene código en lenguaje PHP para administrar la carga del criptograma enviado por el módulo transmisor y almacenarlo en el servidor web. El código del archivo se muestra a continuación.

```
<?php //Summary
  /**
  * This script handles file uploads to the "uploads" directory.
  * It checks if the "up" and "ACM" parameters are set in the GET request.
  * If the parameters are set, it moves the uploaded file to the target directory.
  * If the file upload is successful, it displays a success message.
  * If there is an error during the file upload, it displays an error message along with
      the error code.
  * Finally, it displays the temporary name of the uploaded file.
  */
?>
<?php
$target_dir = "uploads/";
$filename = basename(@$_FILES["FileGDF"]["name"]);
$target_file = $target_dir . $filename;
$FileType = pathinfo($target_file,PATHINFO_EXTENSION);
if(isset($_GET["up"])&&(isset($_GET["ACM"]))) {
  if (move_uploaded_file(@$_FILES["FileGDF"]["tmp_name"], $target_file))
  {
    echo "The file ". basename($filename). " has been uploaded.";
  }
  else
  {
    echo "Sorry, there was an error uploading your file. Error #";
    echo $_FILES["FileGDF"]["error"];
  }
}
echo ". TempName: ";
echo $_FILES["FileGDF"]["tmp_name"];
?>
```

El segundo archivo, llamado *index.php*, contiene código en lenguaje PHP para el proceso de desencriptación. Además, permite la interacción con el usuario mediante una página web para la visualización de la señal clara y la desencriptada. El código del archivo se muestra a continuación.

```
<?php //Summary
  /**
```

```
3       *
4       * This file contains PHP code for decrypting a CSV file using a custom algorithm.
5       * The algorithm involves performing mathematical operations on the data read from the
            CSV file.
6       * The decrypted data is then written to a new CSV file.
7       *
8       * The code reads the CSV file 'cifrado.csv' and retrieves certain values from it.
9       * It then performs calculations based on user input and the retrieved values to decrypt
            the data.
10      * The decrypted data is stored in the array $dataDec and written to the file '
            descifrado.csv'.
11      *
12      * Note: This code assumes that the CSV file has a specific format and that the necessary
             input is provided via a form submission.
13      */
14    ?>
15
16    <?php // Functions
17
18      /**
19      * Calculates the modulus of two numbers.
20      *
21      * @param float $X The dividend.
22      * @param float $Y The divisor.
23      * @return float The modulus of $X divided by $Y.
24      */
25      function modul($X, $Y)
26      {
27        return ($X - (floor($X) / $Y) * $Y);
28      }
29
30      /**
31      * Reads a CSV file and returns its contents as an array.
32      *
33      * @param string $csvFile The path to the CSV file.
34      * @return array The CSV data as a multidimensional array.
35      */
36      function readCSV($csvFile)
37      {
38        $dataEnc = array();
39        $file_handle = fopen($csvFile, 'r');
40        while (!feof($file_handle)) {
41          $dataEnc[] = fgetcsv($file_handle);
42        }
43        fclose($file_handle);
44        return $dataEnc;
45      }
46
47      /**
48      * Decrypts a signal using the provided key.
49      *
50      * @param string $key The encryption key.
51      * @param string $csvFile The path to the CSV file containing the encrypted signal.
52      * @return array The decrypted signal data.
53      */
54      function decryptSignal($key, $csvFile)
55      {
56        $file = new SplFileObject($csvFile, 'r');
57        $file->seek(PHP_INT_MAX);
58        $signalLen = $file->key();
59        $signalLen = $signalLen - 5;
60
61        $normSignal = array();
62        if (($file = fopen($csvFile, "r")) !== FALSE) {
63          while (($value = fgets($file)) !== FALSE) {
64            $normSignal[] = $value;
65          }
66          fclose($file);
67        }
68
69        // Recuperar valor de Z, Min y Max, ubicados en los últimos 5 datos
70        $Z = $normSignal[$signalLen - 1 + 1];
```

```
71        if ($normSignal[$signalLen - 1 + 2] == 1) { // ES NEGATIVO
72          $_min = -1 * $normSignal[$signalLen - 1 + 3];
73        } else { // ES POSITIVO
74          $_min = $normSignal[$signalLen - 1 + 3];
75        }
76
77        if ($normSignal[$signalLen - 1 + 4] == 1) { // ES NEGATIVO
78          $_max = -1 * $normSignal[$signalLen - 1 + 5];
79        } else { // ES POSITIVO
80          $_max = $normSignal[$signalLen - 1 + 5];
81        }
82
83        $_min = $_min * 100000;
84        $_max = $_max * 100000;
85
86        $hexValues = str_split($key, 8);
87        $A_dec = hexdec($hexValues[0]);
88        $B_dec = hexdec($hexValues[1]);
89        $C_dec = hexdec($hexValues[2]);
90        $D_dec = hexdec($hexValues[3]);
91
92        // Determinar el valor a sumar en las condiciones iniciales y parámetros
93        $val_AE = $A_dec / (4294967296 + 1.0);
94        $val_BE = $B_dec / (4294967296 + 1.0);
95        $val_CE = $C_dec / (4294967296 + 1.0);
96        $val_DE = $D_dec / (4294967296 + 1.0);
97
98        //Condiciones iniciales
99        $x1[0] = modul($val_AE + $val_BE, 1.0);
100       $x2[0] = modul($val_CE + $val_DE + $Z, 1.0);
101       $signalLenExtended = $signalLen + 100;
102       // Parameter R for logistic map
103       $r = 3.999999999999999;
104       for ($i = 0; $i < $signalLenExtended - 1; $i++) {
105         $x1[$i + 1] = $r * $x1[$i] * (1 - $x1[$i]);
106       }
107
108       for ($i = 0; $i < $signalLenExtended - 1; $i++) {
109         $x2[$i + 1] = modul($r * $x2[$i] * (1 - $x2[$i]) * 1000000 * $x1[$i], 1.0);
110       }
111
112       // SECUENCIA PARA PERMUTACION
113
114       for ($i = 0; $i < $signalLen; $i++) {
115         $permutation[$i] = round($x2[$signalLenExtended - $signalLen + $i] * ($signalLen -
              1));
116       }
117       // Optimización de secuencia de permutación
118       // Busca posiciones de repetidos y los pone en un vector
119       $permutation_opt[0] = $permutation[0];
120       $duplicateCount = 0;
121       for ($i = 1; $i < $signalLen; $i++) {
122         $isDuplicate = 0;
123         $permutation_opt[$i] = $permutation[$i];
124         for ($j = 0; $j < $i; $j++) {
125           if ($permutation_opt[$i] == $permutation_opt[$j] && $isDuplicate == 0) {
126             $duplicatePositions[$duplicateCount] = $i;
127             $duplicateCount = $duplicateCount + 1;
128             $isDuplicate = 1;
129           }
130         }
131       }
132
133       // Buscar números que no estén en vector de permutación
134       $currentNumber = 0;
135       $notInPermutationCount = 0;
136       $skip = 0;
137       for ($i = 0; $i < $signalLen; $i++) {
138         // Pregunta si está en vector de permutación
139
140         for ($j = 0; $j < $signalLen; $j++) {
141           if ($currentNumber == $permutation[$j]) {
```

```
142             $skip = 1; // Si está, entra.
143           }
144         }
145         // Si no está en repetidos y en vector de permutación, entra
146         if ($skip == 0) {
147           $num_no_est2D[$notInPermutationCount] = $currentNumber; // Guarda número.
148           $notInPermutationCount = $notInPermutationCount + 1;
149         }
150         $skip = 0;
151         $currentNumber = $currentNumber + 1;
152       }
153
154       // Se actualiza vector de permutación (tiene todos los espacios)
155       $duplicatePositionsSize = $duplicateCount;
156       $halfSize = round($duplicatePositionsSize / 2, PHP_ROUND_HALF_DOWN);
157       if ($duplicatePositionsSize % 2 != 0) {
158         $halfSize = $halfSize + 1;
159       }
160
161       $change = 1;
162       $S = 0;
163       for ($i = 0; $i < $duplicatePositionsSize; $i++) {
164         if ($change == 1) {
165           $permutation[$duplicatePositions[$i]] = $num_no_est2D[$S];
166           $change = 0;
167         } else {
168           $permutation[$duplicatePositions[$i]] = $num_no_est2D[$S + $halfSize];
169           $S = $S + 1;
170           $change = 1;
171         }
172       }
173
174       // SECUENCIA PARA DIFUSION
175       for ($i = 0; $i < $signalLen; $i++) {
176         $y[$i] = modul(($x2[$signalLenExtended - $signalLen + $i] * 1000) + $Z, 1.0);
177         $y[$i] = floor($y[$i] * pow(10, 15)) / pow(10, 15);
178       }
179       // PROCESO DE PERMUTACION y DIFUSION INVERSOS
180       for ($i = 0; $i < $signalLen; $i++) {
181         $DECIFR_aux = modul($normSignal[$i] - $y[$i] - $y[$signalLen - $i - 1], 1.0);
182         $dataDec[$permutation[$i]] = ($DECIFR_aux * ($_max + 0.01)) + ($_min - 0.01);
183       }
184
185       $result = array();
186       for ($i = 0; $i < $signalLen; $i++) {
187         $result[$i] = $dataDec[$i];
188       }
189
190       return $result;
191     }
192
193     /**
194     * Writes data to a CSV file.
195     *
196     * @param string $csvFile The path to the CSV file.
197     * @param array $dataDec The data to be written to the CSV file.
198     * @return void
199     */
200     function writeCSV($csvFile, $dataDec)
201     {
202       $fp = fopen($csvFile, 'w');
203       foreach ($dataDec as $fields) {
204         fputcsv($fp, [$fields]);
205       }
206       fclose($fp);
207     }
208   ?>
209
210   <?php //Main code
211     $path = 'uploads/cifrado.csv';
212     $pathResult = 'uploads/descifrado.csv';
213
```

```
214     // Variables to graph in javascript
215     $dataDec = array();
216     $dataEnc = array();
217
218     // Read the encrypted signal from the CSV file
219     if (file_exists($path)) {
220       $dataEnc = readCSV($path);
221
222
223       // Decrypt the signal and write the decrypted data to a new CSV file
224       if (isset($_POST['BtnDescifrar'])) {
225         if (isset($_POST['InputClave'])) {
226           $key = $_POST['InputClave'];
227           $dataDec = decryptSignal($key, $path);
228           writeCSV($pathResult, $dataDec);
229         }
230       }
231     }
232     else{
233       echo "<script>alert('There is no file to decrypt.');</script>";
234     }
235   ?>
236
237   <!DOCTYPE html>
238   <html lang="en">
239
240   <head>
241     <meta charset="UTF-8">
242     <meta name="viewport" content="width=device-width, initial-scale=1.0">
243     <title>ECG Graph</title>
244     <script src="https://cdn.plot.ly/plotly-latest.min.js"></script>
245     <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/
          bootstrap.min.css">
246     <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.3/
          css/all.min.css">
247     <style>
248       body {
249         background-color: #f8f9fa;
250       }
251
252       .navbar {
253         background-color: #343a40;
254       }
255
256       .navbar-brand {
257         color: #ffffff;
258       }
259
260       .card-header {
261         background-color: #343a40;
262         color: #ffffff;
263       }
264
265       .card-body {
266         background-color: #ffffff;
267       }
268
269       .form-control {
270         border-color: #ced4da;
271       }
272
273       .btn-primary {
274         background-color: #007bff;
275         border-color: #007bff;
276       }
277
278       .btn-primary:hover {
279         background-color: #0069d9;
280         border-color: #0062cc;
281       }
282
283       .text-muted {
```

```
284           color: #6c757d;
285         }
286
287         .footer {
288           background-color: #343a40;
289           color: #ffffff;
290         }
291
292       </style>
293     </head>
294
295     <body>
296       <header>
297         <nav class="navbar navbar-expand-md navbar-dark bg-dark">
298           <a href="#" class="navbar-brand">Sistema de Telemedicina Seguro con Encriptamiento
                  Caótico</a>
299           <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#
                  navbarNav">
300             <span class="navbar-toggler-icon"></span>
301           </button>
302           <div class="collapse navbar-collapse" id="navbarNav">
303             <ul class="navbar-nav ml-auto">
304               <li class="nav-item active"><a href="#" class="nav-link">Home</a></li>
305               <li class="nav-item"><a href="#" class="nav-link disabled">Servicios</a></li>
306               <li class="nav-item"><a href="#" class="nav-link disabled">Nosotros</a></li>
307               <li class="nav-item"><a href="#" class="nav-link disabled">Contáctanos</a></li>
308             </ul>
309           </div>
310         </nav>
311       </header>
312       <main class="container-fluid my-5">
313         <div class="row">
314           <div class="col-12 mb-3">
315             <div class="card">
316               <div class="card-header font-weight-bold text-center">Electrocardiograma Cifrado
                      </div>
317               <div class="card-body">
318                 <div id="ecg-graph" data-toggle="modal" data-target="#exampleModal"></div>
319               </div>
320             </div>
321           </div>
322         </div>
323         <div class="row text-center">
324           <div class="col-12 mb-3">
325             <div class="d-flex justify-content-center">
326               <div class="card container">
327                 <div class="card-body">
328                   <form method="post">
329                     <div class="form-group">
330                       <label for="InputClave" class="font-weight-bold">Clave de descifrado</
                              label>
331                       <input type="password" class="form-control" name="InputClave" id="
                              InputClave" maxlength="32" minlength="32" aria-describedby="
                              claveHelp" placeholder="Ingresa la clave">
332                     </div>
333                     <button type="submit" name="BtnDescifrar" id="btnDescifrar" class="btn btn
                          -primary">Descifrar</button>
334                   </form>
335                 </div>
336               </div>
337             </div>
338           </div>
339         </div>
340         <div class="row">
341           <div class="col-12 mb-3">
342             <div class="card">
343               <div class="card-header font-weight-bold text-center">Electrocardiograma
                      Descifrado</div>
344               <div class="card-body">
345                 <div id="ecg-graph2" data-toggle="modal" data-target="#exampleModal2"></div>
346               </div>
347             </div>
```

```
348         </div>
349       </div>
350     </main>
351     <footer class="bg-dark text-white py-3">
352       <div class="container">
353         <div class="row">
354           <div class="col-md-4">
355             <h3>Contáctanos</h3>
356             <p>Empresa</p>
357             <p>Dirección</p>
358             <p>Ciudad, Estado CP</p>
359             <p>Teléfono: XXX-XXX-XXXX</p>
360             <p>Email: correo@gmail.com</p>
361             <ul class="list-inline">
362               <li class="list-inline-item"><a href="#" class="text-white"><i class="fab fa-
                      facebook fa-2x"></i></a></li>
363               <li class="list-inline-item"><a href="#" class="text-white"><i class="fab fa-
                      twitter fa-2x"></i></a></li>
364               <li class="list-inline-item"><a href="#" class="text-white"><i class="fab fa-
                      instagram fa-2x"></i></a></li>
365               <li class="list-inline-item"><a href="#" class="text-white"><i class="fab fa-
                      linkedin fa-2x"></i></a></li>
366             </ul>
367           </div>
368           <div class="col-md-4">
369             <h3>Enlaces útiles</h3>
370             <ul class="list-unstyled">
371               <li><a href="#" class="text-white disabled">Home</a></li>
372               <li><a href="#" class="text-white disabled">Servicios</a></li>
373               <li><a href="#" class="text-white disabled">Nosotros</a></li>
374               <li><a href="#" class="text-white disabled">Contáctanos</a></li>
375             </ul>
376           </div>
377           <div class="col-md-4">
378             <h3>Newsletter</h3>
379             <form>
380               <fieldset disabled>
381                 <div class="form-group">
382                   <input type="email" class="form-control" placeholder="Email Address">
383                 </div>
384                 <button type="submit" class="btn btn-primary">Suscríbete</button>
385               </fieldset>
386             </form>
387           </div>
388         </div>
389       </div>
390     </footer>
391
392     <!-- Modal -->
393     <div class="modal fade" id="exampleModal" tabindex="-1" role="dialog" aria-labelledby="
          exampleModalLabel" aria-hidden="true">
394       <div class="modal-dialog modal-lg" role="document">
395         <div class="modal-content">
396           <div class="modal-header">
397             <h5 class="modal-title" id="exampleModalLabel">Electrocardiograma cifrado</h5>
398             <button type="button" class="close" data-dismiss="modal" aria-label="Close">
399               <span aria-hidden="true">&times;</span>
400             </button>
401           </div>
402           <div class="modal-body">
403             <div id="ecg-graph3"></div>
404           </div>
405           <div class="modal-footer">
406             <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</
                    button>
407           </div>
408         </div>
409       </div>
410     </div>
411
412     <!-- Modal2 -->
413     <div class="modal fade" id="exampleModal2" tabindex="-1" role="dialog" aria-labelledby="
```

```
          exampleModalLabel2" aria-hidden="true">
414       <div class="modal-dialog modal-lg" role="document">
415         <div class="modal-content">
416           <div class="modal-header">
417             <h5 class="modal-title" id="exampleModalLabel">Electrocardiograma descifrado</h5
                    >
418             <button type="button" class="close" data-dismiss="modal" aria-label="Close">
419               <span aria-hidden="true">&times;</span>
420             </button>
421           </div>
422           <div class="modal-body">
423             <div id="ecg-graph4"></div>
424           </div>
425           <div class="modal-footer">
426             <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</
                    button>
427           </div>
428         </div>
429       </div>
430     </div>
431
432     <script>
433       var arrayx = [...Array(1000).keys()];
434       arrayx.forEach(function(element, index, array) {
435         array[index] = element * 0.01 + 0.01;
436       });
437
438       var data = {
439         x: arrayx,
440         y: [
441           <?php
442           foreach ($dataEnc as $row) {
443             if (is_array($row)) {
444               echo $row[0] . ", ";
445             }
446           }
447           ?>
448         ],
449         type: 'scatter',
450         mode: 'lines',
451         line: {
452           color: 'red',
453           width: 1
454         }
455       };
456       console.log(data);
457       var data2 = {
458         x: arrayx,
459         y: [
460           <?php
461           foreach ($dataDec as $row) {
462             echo $row . ", ";
463           }
464           ?>
465         ],
466         type: 'scatter',
467         mode: 'lines',
468         line: {
469           color: 'red',
470           width: 2
471         }
472       };
473
474       var layout = {
475         xaxis: {
476           title: 'Time (s)'
477         },
478         yaxis: {
479           title: 'Voltage (mV)'
480         }
481       };
482
```

```
483         Plotly.newPlot('ecg-graph', [data], layout);
484         Plotly.newPlot('ecg-graph3', [data], layout);
485         Plotly.newPlot('ecg-graph2', [data2], layout);
486         Plotly.newPlot('ecg-graph4', [data2], layout);
487       </script>
488       <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"></script>
489       <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js
              "></script>
490       <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"></
              script>
491     </body>
492
493     </html>
```