UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

FACULTAD DE INGENIERÍA, ARQUITECTURA Y DISEÑO ENSENADA

SEGURIDAD EMBEBIDA EN TELEMEDICINA BASADA EN CRIPTOGRAFÍA CAÓTICA

TESIS

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

INGENIERO EN ELECTRÓNICA

presenta:

ARMANDO CESEÑA VILLA

Ensenada, Baja California, México, Noviembre de 2021.

UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

FACULTAD DE INGENIERÍA, ARQUITECTURA Y DISEÑO ENSENADA

SEGURIDAD EMBEBIDA EN TELEMEDICINA BASADA EN CRIPTOGRAFÍA CAÓTICA

TESIS

Que para obtener el grado de Ingeniero en Electrónica presenta:

ARMANDO CESEÑA VILLA

Aprobada por el siguiente comité:

Dr. Miguel Ángel Murillo Escobar
Director del comité

Dra. Rosa Martha López Gutiérrez
Codirector

Dr. César Cruz Hernández
Miembro del comité

Dra. Liliana Cardoza Avendaño
Miembro del comité

Dr. José Antonio Michel Macarty
Miembro del comité

Ensenada, Baja California, México. Noviembre de 2021

RESUMEN de la tesis de **Armado Ceseña Villa**, presentada como requerimiento parcial para obtener el grado de INGENIERO en ELECTRÓNICA, del programa de Licenciatura de la Universidad Autónoma de Baja California. Ensenada, Baja California, México. Noviembre de 2021.

SEGURIDAD EMBEBIDA EN TELEMEDICINA BASADA EN CRIPTOGRAFÍA CAÓTICA

Resumen aprobado por:

Dr. Miguel Ángel Murillo Escobar
Director de tesis

Dra. Rosa Martha López Gutiérrez
Codirector de tesis

En este trabajo de tesis se diseña un sistema embebido para la transmisión de señales biomédicas de forma remota y segura mediante la implementación de un algoritmo de cifrado caótico el cual utiliza internet como canal de comunicación inseguro entre un módulo transmisor y receptor, con un servidor local que es ejecutado en un ordenador Raspberry Pi Zero W como intermediario entre los módulos.

Se estudian seis mapas caóticos, de los cuales, se selecciona el mapa SLIM y Seno-Hénon para la implementación del algoritmo de cifrado debido a su velocidad de procesamiento y su nivel de sensibilidad a condiciones iniciales evaluado con el cálculo del máximo exponente de Lyapunov. El algoritmo de cifrado propuesto es implementado en un sistema embebido de bajo costo basado en un microcontrolador de 32 bits para conformar los módulos transmisor y receptor del sistema, con los que se realiza la transmisión exitosa tanto de una señal de electrocardiograma (ECG) como de una señal de presión sanguínea (BP).

Finalmente se realizan distintos análisis de seguridad de sensibilidad a la clave secreta y a la señal clara, análisis de correlación, entropía y autocorrelación, análisis que demuestran que el sistema diseñado no es solo capaz de transmitir las señales de forma exitosa sino que además proporciona un nivel de seguridad adecuado para utilizar un canal de transmisión inseguro como lo es el internet, lo cual, lo hace potencialmente viable en aplicaciones de telemedicina.

Palabras clave: cifrado caótico, sistema embebido, telemedicina, mapa SLIM, mapa Seno-Hénon, sistema remoto, análisis de seguridad.

Abstract of the thesis presented by **Armando Ceseña Villa**, as a partial requirement to obtain the degree in ELECTRONICS ENGINEER, of the program of the Autonomous University of Baja California. Ensenada, Baja California, Mexico. November, 2021.

EMBEDDED SECURITY IN TELEMEDICINE BASED ON CHAOTIC CRYPTOGRAPHY

Abstract approved by:

Dr. Miguel Ángel Murillo Escobar
Thesis director

Dra. Rosa Martha López Gutiérrez
Thesis codirector

In this thesis work, an embedded system for the transmission of biomedical signals remotely and securely is designed through implementing a chaotic encryption algorithm, that uses Internet as communication channel between the transmitter and receiver module, with a local sever based on a Raspberry Pi Zero W computer as an intermediary between the modules.

Six chaotic maps are analyzed, from which the SLIM and Sine-Hénon maps are selected for the implementation of the encryption algorithm due to their processing speed and sensitivity to initial conditions evaluated with maximum Lyapunov exponent. The proposed encryption algorithm is implemented in a low-cost embedded system based on a 32-bit microcontroller that integrate the transmitter and receiver modules of the system, modules that carried out the successful transmission of both an electrocardiogram signal (ECG) and a blood preasure signal (BP).

Finally, different security analyzes of sensitivity tests to the secret key and the clear signal, correlation, entropy and autocorrelation analysis are carried out, analyzes that show that the designed system is not only capable of transmitting signals successfully but also provides an adequate level of security to use an insecure transmission channel such as the internet, which makes it potentially viable in telemedicine applications.

Keywords: chaotic cipher, embedded system, telemedicine, SLIM map, Henon-Sine map, remote system, security analysis.

A Dios y mi familia

Agradecimientos

A mi familia, mis padres Armando y María así como a mi hermana Melissa, por su amor y apoyo incondicional en cada etapa de mi vida, por siempre motivarme a seguir adelante y superarme, además de ser mi guía para convertirme en la persona que soy ahora.

A la Dra. Rosa Martha López Gutiérrez, por motivarme a llevar a cabo este trabajo de tesis, además de no solo contribuir a mi desarrollo profesional apoyándome durante toda la carrera universitaria como coordinadora de la carrera de Ingeniería en Electrónica, sino también por brindarme su cariño y consejo que me ayudó a crecer como persona.

Al Dr. Miguel Ángel Murillo Escobar, por guiarme a lo largo del desarrollo de este trabajo de tesis, compartiendo su experiencia y consejo para realizarlo de la mejor forma posible. También por sus consejos y el tiempo compartido durante este periodo.

A mi comité de tesis, Dr. César Cruz Hernández, Dra. Liliana Cardoza Avendaño y Dr. José Antonio Michel Macarty, por haber contribuido a mi formación académica, además de sus comentarios y correcciones para mejorar este trabajo de tesis.

A la Universidad Autónoma de Baja California (UABC), por haberme brindado un espacio para llevar a cabo mis estudios universitarios, en especial a la Facultad de Ingeniería, Arquitectura y Diseño que se convirtió en mi casa a lo largo de la carrera, además de brindarme maestros de calidad que me ayudaron concluir mis estudios universitarios de forma satisfactoria.

Al Consejo Nacional de Ciencia y Tecnología (CONACyT), por el apoyo económico brindado a través del Proyecto de Grupos de Investigación en Ciencia Básica, Referencia 166654 y A1-S-3628.

Ensenada, B.C., México.
Noviembre de 2021

Armando Ceseña Villa

Tabla de Contenido

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción

En la última década, el desarrollo de las telecomunicaciones ha impactado en la sociedad de una forma tan profunda que hoy en día estas juegan un papel fundamental en casi todos los ámbitos de la vida de las personas, ya sea en el trabajo, la escuela o, evidentemente, en la vida social de las mismas, lo cual, ha llevado a que la cantidad de información que se comparte a través de medios de comunicación como internet vaya en aumento año con año. De igual manera, cada vez más servicios están siendo emigrados hacia el área cibernética, desde los servicios escolares, trabajos de oficina, tiendas comerciales y, por su puesto, servicios públicos como lo es el servicio médico [1].

La telemedicina (medicina a distancia) es un concepto que ya se ha manejado desde años atrás pero que se ha visto fuertemente impulsado el último año por la aparición de la enfermedad por coronavirus 2019 (COVID-19) [2, 3] debido a que se han buscado e implementado nuevas estrategias para que las personas sigan contando con servicio médico mientras que al mismo tiempo se busca evitar esparcir el virus, lo cual, implica evitar realizar consultas médicas de forma presencial en la medida de lo posible. En ese sentido, la telemedicina ha surgido como una solución al permitir que dichas consultas sean efectuadas remotamente a través de una red como lo es internet para que de esa forma el personal médico mantenga un contacto mínimo con los pacientes, al mismo tiempo que se aprovechan otros aspectos de la telemedicina como lo es el monitoreo de signos vitales a distancia.

Esto ha llevado a que la cantidad de información médica que se comparte por internet haya aumentado enormemente, y si bien los pacientes han podido estar en contacto con sus médicos de una forma rápida y segura (en cuestión de salud) también es cierto que compartir este tipo de información por medios de comunicación inseguros como lo es internet puede llegar a ser peligroso puesto que cualquier persona lo suficientemente capacitada podría interceptar dicha información y hacer mal uso de ella para estafas y fraudes. Como consecuencia, la seguridad y privacidad de la información en las telecomunicaciones se ha vuelto una de las mayores preocupaciones de la sociedad hoy en día [4].

Para dar solución al problema se ha recurrido al uso la criptografía, una de las

técnicas más antiguas en lo que a la protección de la confidencialidad de un mensaje se refiere y la cual involucra distintas técnicas y métodos para modificar la estructura del mensaje a transmitir mediante un algoritmo de cifrado con la finalidad de lograr que una persona no autorizada no consiga comprender el mensaje enviado sino que, en caso de que logre interceptar el mismo, reciba un mensaje completamente distinto al mensaje que se ha querido transmitir en un inicio mientras que los usuarios autorizados podrán descifrar y comprender correctamente el mensaje sin problema alguno.

Entre los métodos de encriptado que se han utilizado desde hace ya algunos años se encuentran los *convencionales* como lo son el AES (Advanced Encryption Standart) y el DES (Data Encryption Standard), sin embargo, a pesar de estar comprobados como métodos de encriptado seguros también se ha encontrado que pueden llegar a ser ineficientes cuando se trata, por ejemplo, de encriptar archivos multimedia en tiempo real [5]. Esto ha llevado a la creación de nuevos métodos de cifrado *no convencionales* que involucran no solo un complejo algoritmo criptográfico sino que involucran algunos fenómenos con propiedades que aumentan no solo la complejidad del algoritmo, sino también la seguridad del mensaje a transmitir.

Algunos de los métodos no convencionales que se han desarrollado en gran medida han sido aquellos que involucran a la teoría del caos como parte de su proceso de cifrado y son considerados como unos de los más seguros a la hora de proteger información confidencial puesto que toman ventaja de las propiedades intrínsecas del caos como lo son la alta sensibilidad a condiciones iniciales, ergodicidad y comportamiento impredecible (pseudoaleatoriedad) [6] para aumentar la complejidad de los métodos, de manera que al utilizar el caos en conjunto con los procesos de confusión y difusión se genere un algoritmo que sea muy resistente ante ataques *criptoanalíticos*, esto es, que difícilmente se pueda recuperar el mensaje original por una persona no autorizada.

Considerando dichas ventajas, este trabajo se basa en el desarrollo de un sistema para transmitir señales biomédicas a través de internet de forma inalámbrica, implementando un algoritmo de cifrado caótico que le proporcione a las señales un nivel de seguridad tal que en caso de ser interceptadas estas sean información incomprensible para el intruso, protegiendo la integridad de los pacientes.

1.1. Motivación

Las nuevas tecnologías, como lo son internet y los diferentes dispositivos móviles que año con año mejoran sus características, han proporcionado a la sociedad las herramientas necesarias para reducir el tiempo invertido en las diferentes actividades cotidianas, así como para reducir la distancia entre las personas pues mediante el uso de plataformas en internet es posible intercambiar información sin la necesidad de un traslado físico. Sin embargo, todo sistema puede presentar fallas o "huecos" que pueden comprometer la información que viaja a través de sus canales, haciéndolos canales inseguros para

compartir información confidencial como lo es la información médica, pues se pueden llegar a presentar filtraciones en el canal de comunicación.

Una filtración puede entenderse como una exposición de información sin el consentimiento del propietario de la misma, y que puede llegar a perjudicar al mismo. 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 [7].

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 el educativo, el gubernamental, el de los negocios y, evidentemente el médico. Sin embargo, como se muestra en [8], es en este último donde más se han presentado casos de filtraciones en los últimos 15 años, abarcando entre el 60 % y 80 % (ver figura 1.1) y teniendo más presencia en los últimos 5 años [9]. Dentro de los casos que se han dado a conocer se han reportado casos [10], 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 los servidores o directamente del correo electrónico.

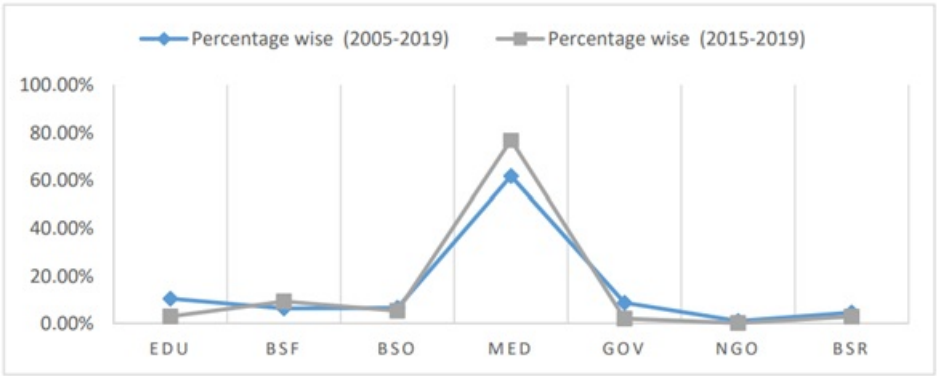


Figura 1.1: Incidentes reportados sobre filtraciones de datos.

Con lo mencionado anteriormente queda de manifiesto la importancia de seguir mejorando los sistemas de seguridad empleados para proteger la información médica, con la finalidad proteger la integridad de los pacientes y para que los mismos adquieran un sentimiento de confianza y tranquilidad cuando se vean en la necesidad de realizar consultas médicas a distancia, las cuales indican ser el futuro de los servicios médicos. Por otro lado, resulta conveniente contemplar los métodos criptográficos basados en caos como una solución al problema planteado puesto que, gracias a sus particulares características, han mostrado buenos resultados frente a pruebas de seguridad criptoanalíticas, por lo que resulta conveniente seguir desarrollando este tipo de algoritmos con la finalidad de mejorar sus propiedades y aumentar su nivel de seguridad.

1.2. Objetivos y alcances de la tesis

Considerando el interés en incrementar la seguridad y privacidad de la información que es transmitida por internet en el área de telemedicina, surge este trabajo de tesis de licenciatura en el que se plantea alcanzar el siguiente *objetivo general*:

Implementar seguridad embebida basada en criptografía caótica para aplicaciones de telemedicina.

Que para cumplir con el objetivo general, se plantea alcanzar los siguientes *objetivos particulares*:

1. Implementar mapa caótico en software Matlab y en sistema embebido de 32 bits.
2. Verificar dinámica caótica con exponente de Lyapunov.
3. Implementar algoritmo criptográfico basado en caos en Matlab y en sistema embebido.
4. Transmitir y recibir criptogramas de forma remota mediante una red inalámbrica.
5. Realizar análisis de seguridad a criptogramas en software Matlab.

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 a futuro.

Capítulo 2

Telemedicina

En este capítulo se presenta el concepto de telemedicina, así como un breve marco histórico y clasificaciones (o áreas de aplicación) de la misma. Finalmente se hace mención al nivel de seguridad de los sistemas de telemedicina.

2.1. Introducción

La palabra telemedicina es utilizada para hacer referencia a la prestación de servicios médicos de forma remota mediante el uso de las nuevas tecnologías como lo es internet. Concretamente, la Organización Mundial de la Salud (OMS) define la telemedicina como: "aportar servicios de salud, donde la distancia es un factor crítico, por cualquier profesional de la salud, usando las nuevas tecnologías de la comunicación para el intercambio válido de información en el diagnóstico, el tratamiento y la prevención de enfermedades o lesiones, investigación y evaluación, y educación continuada de los proveedores de salud, todo con el interés de mejorar la salud de los individuos y sus comunidades" [11, 12].

En la actualidad los sistemas de telemedicina pueden tener diferentes estructuras que los hacen más o menos complejos dependiendo de la aplicación de los mismos, sin embargo éstos tienen una representación básica (ver figura 2.1) en la que es posible observar que su principal objetivo es comunicar al personal y a las instituciones médicas con otros usuarios que podrían no encontrarse en la misma zona geográfica y que bien podrían ser pacientes a los cuales atender u otro personal médico. Por otro lado, el canal de transmisión de la información puede variar dependiendo el tipo de sistema implementado, dentro de los cuales podrían destacarse las redes móviles (3G, 4G) o internet.

La telemedicina ofrece múltiples beneficios de los cuales disfrutan todos aquellos que toman parte en el proceso de intercambio de información: el sistema, los profesionales y los pacientes [13]. Los beneficios que obtiene el sistema van encaminados a la eficiencia del servicio médico a través de algunos factores como lo son la disminución de desplazamientos por parte de los profesionales y una mejora de la gestión de la demanda general.

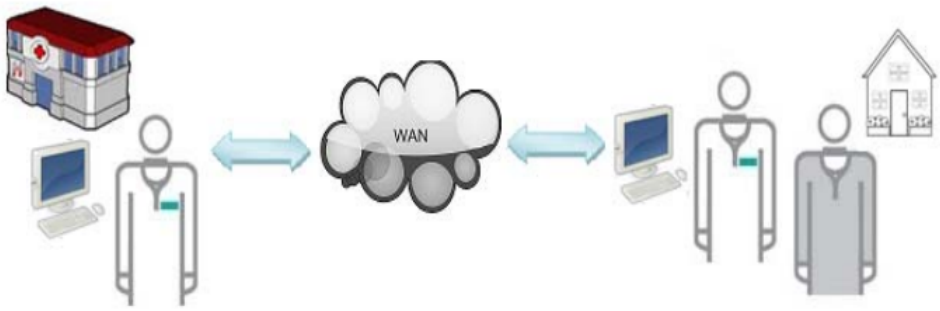


Figura 2.1: Representación básica de una red de telemedicina.

Para los profesionales (los médicos, por ejemplo) se facilita la comunicación entre ellos mismos para realizar cualquier tipo de consulta y se tiene mayor accesibilidad a los datos del paciente así como a artículos de investigación.

Por su parte, los pacientes reciben principalmente el beneficio de no tener que desplazarse hasta el centro médico cada vez que requieran una consulta o una toma de signos vitales, mayor accesibilidad a su información médica e, inclusive, permite el monitoreo en tiempo real por medio de equipos terminales para captar señales biomédicas en caso de requerirlo sin tener que estar presentes en el centro médico. De esta forma la telemedicina permite aumentar la calidad del servicio médico para beneficio de todos, no solo de forma cotidiana sino que al mismo tiempo permite actuar de forma más eficiente ante situaciones de emergencia.

2.2. Antecedentes

La historia de la telemedicina se ha dividido en tres épocas históricas: la era de las telecomunicaciones (en la década de los setenta), la era digital (en la década de los 80) y la era internet que hace referencia a la actualidad [14] . A pesar de que no se conoce con exactitud un punto en la historia que indique el origen de la telemedicina, se conocen algunos hechos históricos que dieron pie al desarrollo de la telemedicina a como la conocemos hoy en día. Algunos de estos acontecimientos son mencionados en [15].

En 1903, Einthoven (inventor del ECG), realizó experimentos para transmitir señales de ECG a través de las líneas telefónicas, y no fue sino hasta 1906 que logró transmitir los datos desde un hospital hasta su propio laboratorio.

Con la llegada de las comunicaciones por radio, en 1924 se dio inicio a los llamados telediagnósticos cuando se llevó a cabo un diagnóstico a distancia para niños utilizando imágenes directas, mientras que en 1950 Gershon-Cohen publicó un artículo en el cual reportó un sistema para el diagnóstico de rayos X, el cual era transmitido por radio o líneas telefónicas a través de cortas o largas distancias, el cual podría proporcionar a las zonas rurales acceso a dicho tipo de servicio sin la necesidad de contar con un radiólogo [16].

La llegada de la televisión y de los satélites también ayudó al desarrollo de la telemedicina, siendo que Wittson y Benschoter, en 1959, utilizaron televisión de circuito cerrado para dirigir sesiones de terapia de grupo entre el Instituto Psiquiátrico de Nebraska y el Hospital Estatal de Norfolk a una distancia de 112 millas. Por otro lado, el primer prototipo de un sistema interactivo de telemedicina fue establecido utilizando el primer satélite para comunicaciones, Early Bird, el cual en 1967 conectó una estación médica en Boston con el hospital general de Massachusetts [17].

Durante la misma década de los sesenta, la Administración Nacional de Aeronáutica y el Espacio (NASA), bajo la preocupación de los efectos fisiológicos de la gravedad cero en el espacio sobre los astronautas y los efectos que conllevarían los vuelos espaciales, logró desarrollar un sistema de asistencia médica que les permitía vigilar constantemente las funciones fisiológicas de los astronautas en el espacio.

Otros acontecimientos fueron los que a continuación se mencionan en [18]:

- En 1972 se da inicio al programa de asistencia médica STARPAHC para nativos de Papago, Arizona, donde se realizó electrocardiografía y radiología que a su vez se transmitió por medio de microondas.
- En 1995, la clínica Mayo establece una conexión con el Hospital Real de Ammán, en Jordania. Mediante esta conexión, médicos de Estados Unidos fueron capaces de realizar consultas médicas en directo mediante el apoyo de un médico hachemita.
- En 2003, la Universidad de Chile da inicio al Proyecto Argonauta, proyecto de telemedicina en la Antártica.

Finalmente, dentro de los acontecimientos más recientes se tiene el proyecto Tele-Ictus, proyecto que fue puesto en marcha en 2007 y que vinculaba al Hospital General de Vic con el Hospital Vall d'Hebron, en España. Dicho vínculo permitió al Hospital General de Vic contar con atención neurológica las 24 horas de los 7 días de la semana, de forma que un neurólogo de Vall d'Hebron puede evaluar a un paciente e indicar las terapias adecuadas en las primeras horas de la fase aguda de la enfermedad [19].

2.3. Clasificación de la telemedicina

La telemedicina ha sido clasificada en diferentes divisiones que involucran el tipo de servicio que ofrecen o el área específica a la que está dirigida o en la que se aplica, sin embargo pueden destacarse cuatro ramas importantes [20]:

1. **Teleconsulta:** La teleconsulta constituye la práctica más común en la telemedicina pues involucra tanto la búsqueda de información médica como el asesoramiento de personal médico, y puede ser aplicada tanto entre pacientes y profesionales de

la salud como entre estos mismos. Por otro lado, la teleconsulta puede ser llevada a cabo de dos formas distintas:

- Modalidad asíncrona: Involucra el envío de información clínica (que puede estar acompañada por imágenes de apoyo) a través de servicios como el correo electrónico para su posterior evaluación por un especialista. La principal ventaja de esta modalidad reside en el hecho de que los participantes (pacientes y especialistas) no tienen que estar presentes cuando se realice el envío de información.

- Modalidad síncrona: Esta modalidad desarrolla la teleconsulta en tiempo real con la participación entre el paciente y el médico mediante el uso de las nuevas tecnologías de la informática y las telecomunicaciones, como lo pueden ser las videoconferencias.

2. **Teleducación:** Abarca el uso de las telecomunicaciones y las tecnologías de la información para la educación médica a distancia, enfocándose en el aumento de experiencias educativas mediante la relación de estudiantes con especialistas, ofreciendo además oportunidades de entrenamiento.

3. **Telemonitoreo:** El telemonitoreo permite obtener los signos vitales de un paciente de forma continua y permanente, evitándole al mismo tener que desplazarse hasta el centro de salud para dicho propósito. Por lo general se lleva a cabo desde el hogar del paciente por medio de dispositivos terminales que permiten, por ejemplo, el monitoreo de ECG (electrocardiogramas), niveles de insulina, variables fisiológicas, entre otros tipos de señales.

4. **Telecirugía:** Constituye el llevar a cabo operaciones quirúrgicas de forma remota por medio de un especialista (cirujano), de manera que el mismo no actúa en las inmediaciones del paciente. Estas prácticas suelen llevarse a cabo con el uso de sistemas robotizados [21] con la finalidad de solucionar problemas de accesibilidad o de realizar cirugías en ambientes peligrosos.

Además de las ya mencionadas, la telemedicina es clasificada por su aplicación en las diferentes especialidades del medio médico como lo son la telecardiología, la telerradiología, tele-bioingeniería y la teleendoscopía.

2.4. Seguridad en telemedicina

Uno de los puntos medulares de la telemedicina en la actualidad es el cuidar la información que se transmite en este tipo de sistemas, pues es sabido que si bien la tecnología ha facilitado la comunicación entre las personas a lo largo de todo el mundo con el desarrollo, por ejemplo, de internet, también se debe hacer evidente que a la par de estos desarrollos han surgido técnicas que permiten a terceros acceder a cualquier

tipo de información que se encuentre en la red.

Muchos son los motivos por los cuales terceras personas desean obtener registros médicos de las bases de datos de hospitales o sistemas de telemedicina, sin embargo el que más sobresale evidentemente es el interés económico. En [22] se reportó que en 2017 el precio promedio por un registro médico se encontraba entre los 20 y los 50 dólares en el mercado negro, para posteriormente ser revendido en la misma web. A pesar de que el interés financiero es considerada como la principal causa del robo de registros médicos, éstos acontecimientos normalmente se ven ligados a otras intenciones como lo es el robo de identidad, extorsión, o simplemente por diversión.

Además de lo anteriormente mencionado, debe considerarse que para llevar a cabo la telemedicina es necesario un sistema que involucra múltiples etapas por las cuales viajan los datos para lograr la transmisión de los mismos de un punto a otro, lo cual a su vez genera múltiples puntos en los cuales la seguridad de la información podría verse comprometida. En [23] 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:** Al no haber recibido ningún tipo de capacitación relacionada a la ciberseguridad, los usuarios o pacientes podrían comprometer la seguridad de los datos de distintas formas, como lo podría ser utilizando claves débiles o cometiendo errores de uso de los dispositivos de telemedicina.

2. **Dispositivos de telemedicina:** Los sistemas operativos de propósito general utilizados en telemedicina, tales como los smartphones, se encuentran expuestos a amenazas de seguridad debido al uso de aplicaciones móviles externas con las que el dispositivo comparte el almacenamiento de información o diversas funcionalidades de los mismos, dejando expuesta la información a las mismas vulnerabilidades a las que dichas aplicaciones se encuentran expuestas (como el robo de información).

3. **Red de casa:** Los sistemas de telemedicina en primera instancia involucran todo tipo de conectividad utilizada en el hogar (tales como Wi-Fi y Bluetooth) para la transmisión de datos, por lo que la red se encuentra expuesta a amenazas asociadas a la transmisión de texto claro y ataques del tipo "Hombre en el medio".

4. **Dispositivos de puerta de enlace:** Las puertas de enlace son un intermediario entre el paciente y el sistema de telemedicina, por lo que se ve expuesto a amenazas similares a las redes de casa.

5. **Internet:** Este el es medio en el cual la seguridad de la información es más vulnerable debido a que es un medio de comunicación público, siendo que la información médica (junto con cualquier tipo de prescripción) es de tipo privada.

6. **Sistema de telemedicina:** Este sistema se compone principalmente de una PC y un software que se encarga de las consultas remotas y se encuentra con el proveedor del servicio de telemedicina. Debido a que este sistema es el que maneja todos los

datos de los pacientes con servicio de telemedicina es también el punto que más atrae amenazas de seguridad de todo tipo, como lo puede ser código malicioso, accesos ilegales de forma física, ataques del tipo "Hombre en el medio", entre otros tipos de amenazas.

7. **Proveedor del servicio de telemedicina:** En los sistemas de telemedicina se establecen conexiones que pueden ser del tipo médico-paciente o inclusive médico-médico y, al existir intercambio de información médica, puede atraer amenazas de seguridad como las establecidas anteriormente.

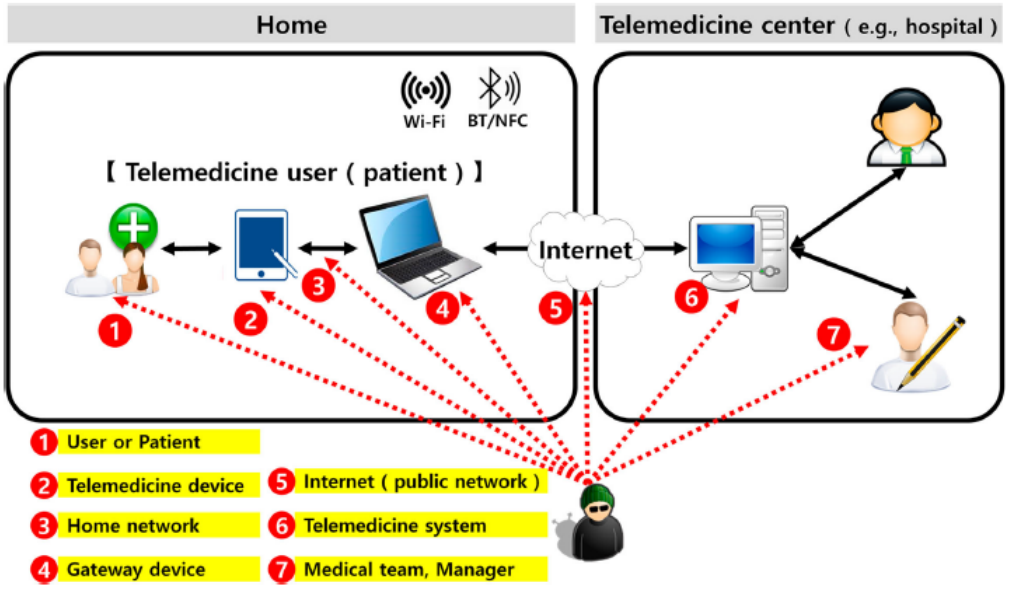


Figura 2.2: Áreas en las que la seguridad se ve amenazada en un sistema de telemedicina.

Como se puede observar, durante el proceso de transmisión de datos en los sistemas de telemedicina existen muchos puntos en los cuales la información podría verse comprometida y, por lo tanto, de igual forma se ve comprometida la seguridad del paciente. Debido a lo anterior, en lo últimos años se ha realizado mucha investigación en el campo de la seguridad en telemedicina con la finalidad de disminuir las áreas en las cuales la información médica puede verse amenazada para ofrecer mayor confidencialidad y seguridad a los pacientes.

2.5. Conclusiones del capítulo

La telemedicina ha sido un paso muy grande en el camino de mejorar los servicios de asistencia médica no solo por la accesibilidad a la información entre el cuerpo médico, sino también porque otorga mayor comodidad al paciente mismo pues no se ve obligado a desplazarse constantemente hasta el centro médico para ser atendido.

Por otro lado, el ámbito de la seguridad en telemedicina es un aspecto sensible que debe tratarse con un cierto grado de prioridad puesto que dicha información tiende a ser muy susceptible a ser robada para diferentes propósitos que podrían atentar contra la seguridad de los pacientes. Además, hoy en día ha incrementado notablemente el uso de los sistemas de telemedicina debido a la actual situación mundial en relación al virus COVID-19, por lo que se debe de seguir investigando en nuevos métodos para aumentar la seguridad en este tipo de sistemas.

Capítulo 3

Caos y criptografía

En este capítulo se presentan los conceptos básicos y propiedades de la teoría del caos, así como un repaso de sus antecedentes y algunos ejemplos de mapas caóticos con sus respectivos análisis de tiempo de ejecución y de dinámicas caóticas, en base a los cuales se seleccionan los mapas a utilizar en este trabajo de tesis. De igual forma se presenta a la criptografía con sus diferentes fundamentos, clasificaciones, un breve relato de su historia y un análisis acerca de la seguridad de los sistemas criptográficos.

3.1. Caos

El término *caos* ha sido utilizado de forma coloquial desde hace muchos años para hacer referencia a situaciones que son difíciles de explicar o comprender, por lo que tradicionalmente se relaciona al caos con el concepto de desorden. Sin embargo, hoy en día el caos se presenta con un concepto diferente pues se ha convertido en un completo campo de estudio desde que se descubrió su estrecha relación con algunos fenómenos naturales.

3.1.1. Antecedentes

Hasta el siglo XIX, la mayoría de los físicos tenían la idea de que los cuerpos mecánicos debían comportarse de forma regular y predecible, sin embargo, Henri Poincaré demostró durante sus estudios del *problema de los n cuerpos* [24] que ciertos sistemas podían evolucionar en el tiempo de forma irregular y aperiódica. Así mismo, Poincaré descubrió que pequeños cambios en las condiciones iniciales del sistema generaban diferencias muy grandes en el resultado final, por lo que una predicción a futuro del sistema era imposible [25].

Casi setenta años después, en 1963, el meteorólogo y matemático estadounidense Edward Lorenz logró observar el mismo fenómeno primeramente apreciado por Poincaré, pero de forma numérica. Lorenz trabajaba con un modelo meteorológico conformado por un conjunto de ecuaciones diferenciales no lineales que describirían el comportamiento de la atmósfera, permitiendo así predecir el clima y, pensando que obtendría

los mismos resultados con un pequeño margen de error, decidió reducir la cantidad de decimales utilizados en los cálculos de seis a tres con el objetivo de reducir el tiempo de procesamiento de la computadora.

Para su sorpresa, el resultado de dicha reducción de decimales fue que no solo los valores numéricos resultantes eran diferentes a los originales, sino que también divergían muy rápido de estos últimos. Además, Lorenz observó que los resultados de sus cálculos no presentaban un comportamiento periódico, y al estudiar más el fenómeno descubrió que la evolución del sistema (su trayectoria) describía una *figura extraña* [26], pues se asemejaba a las alas de una mariposa. Debido a los grandes efectos que tuvieron sobre el sistema pequeños cambios en las condiciones iniciales del mismo, Lorenz le dio a este descubrimiento el nombre de *dependencia sensitiva a condiciones iniciales*, sin embargo, este efecto sería más conocido como el *efecto mariposa*, el cual hace referencia a que el aleteo de una mariposa en Brasil podría provocar un tornado en Texas.

Con los resultados obtenidos, Lorenz continuó con su investigación estudiando la convección de un sistema cerrado, fenómeno fundamental de comportamiento atmosférico, el cual le llevó a reducir el modelo original con el cual trabajaba que se conformaba por un conjunto de 12 ecuaciones diferenciales no lineales, a un sistema de tan solo 3 grados de libertad que se conformaba por el siguiente conjunto de 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 x, y y z eran los estados del sistema, x_0, y_0 y z_0 las condiciones iniciales, σ, ρ y β los parámetros de control y t correspondía al tiempo. Además, al graficar el comportamiento del sistema en el plano de fase observó que continuaba obteniendo el mismo atractor extraño que había obtenido en primera instancia, atractor al cual más tarde se le conocería como Atractor de Lorenz (ver figura 3.1). Con su trabajo, Lorenz habría dado origen a lo que hoy se conoce como *teoría del caos*.

Desde el punto de vista de las matemáticas y la física, el *caos* se presenta como un comportamiento aparentemente errático e impredecible de algunos sistemas dinámicos no lineales a pesar de que su formulación matemática sea en un principio determinista [27]. Hoy en día, la teoría del caos es ampliamente utilizada en diversas áreas de estudio como lo son la física, las matemáticas, la medicina e inclusive la economía, pues con el paso del tiempo se ha encontrado que algunos fenómenos de la vida cotidiana pueden ser adecuadamente modelados mediante ecuaciones que presentan un comportamiento de tipo caótico como es el caso de los ciclos de deudas económicas [28] y el balanceo del cuerpo humano [29].

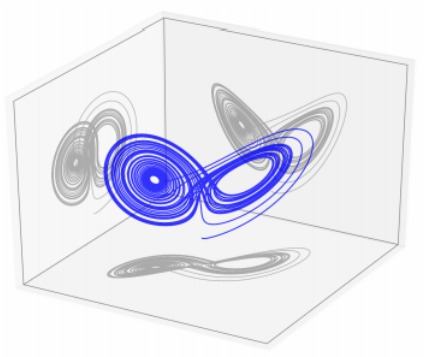


Figura 3.1: Atractor generado por el sistema de Lorenz.

3.1.2. Propiedades de los sistemas caóticos

Un sistema dinámico es aquel cuyo estado evolutivo depende del tiempo, y del cual a su vez se derivan dos clasificaciones: el tiempo continuo y el tiempo discreto. Los sistemas de tiempo continuo son aquellos que pueden ser modelados mediante el uso de ecuaciones diferenciales, mientras que los sistemas de tiempo discreto son representados por ecuaciones en diferencias y, si bien su representación matemática es un poco diferente, existen algunas propiedades que se presentan para ambos tipos de sistemas.

La estabilidad es una de las características más importantes de los sistemas dinámicos [30] pues esta define el comportamiento de los mismos a lo largo del tiempo (ya sea continuo o discreto), comportamiento que se puede clasificar dentro de tres categorías:

- **Estable:** Se dice que un sistema es estable si para un conjunto de condiciones iniciales cercanas la trayectoria del mismo, en el espacio de fase, converge a un punto fijo o presenta un comportamiento oscilatorio periódico (ciclo límite).
- **Inestable:** Un sistema inestable es aquel que para un conjunto de condiciones iniciales cercanas se producen soluciones que divergen de forma exponencial conforme el sistema evoluciona, por lo que no crean un atractor en el espacio de fase.
- **Caótico:** Los sistemas caóticos presentan un comportamiento que resulta de la mezcla entre un comportamiento estable y uno inestable debido a que por un lado su trayectoria se ve afectada por fuerzas que la hacen no converger a ningún punto (fuerzas de repulsión), pero al mismo tiempo se presentan fuerzas de atracción que confinan a la misma dentro de un rango de valores, formando un atractor de *forma extraña*.

Debido a lo anteriormente mencionado, se puede definir un sistema caótico como un sistema dinámico que es representado por medio de un conjunto de ecuaciones diferenciales o en diferencias no lineales que generan trayectorias caóticas deterministas, es decir, que sus estados futuros y pasados dependen del estado actual del sistema. Además, los sistemas caóticos presentan las siguientes propiedades:

- **Sensibilidad exponencial a condiciones iniciales y parámetros de control:** Al variar ligeramente las condiciones iniciales o parámetros de control se modifica de forma drástica la dinámica (o trayectoria) del sistema caótico.
- **No linealidad:** Un sistema con comportamiento no lineal es aquel que está descrito por medio de ecuaciones no lineales y que por consiguiente no está sujeto al principio de superposición, es decir, el comportamiento de la totalidad del sistema no puede ser descrito mediante la suma del comportamiento individual de cada una de las partes del mismo.
- **Ergodicidad:** Hace referencia a que la trayectoria caótica generada se mantiene confinada en un espacio conocido como atractor extraño con respecto al tiempo y que es cubierto en su totalidad para cualquier entrada de condición inicial y parámetro de control.
- **Mezcla de datos:** Un pequeño rango de condiciones iniciales cubre la mayor parte del espectro caótico.
- **Exponente de Lyapunov positivo:** Un sistema de dimensión N tiene N exponentes de Lyapunov; si uno de ellos es positivo se dice que el sistema presenta un comportamiento caótico.
- **Atractor extraño con dimensión fractal:** La gráfica de fase de un sistema caótico se presenta en la forma de un atractor extraño cuya dimensión corresponde a un número fraccionario.

3.1.3. Exponente de Lyapunov

El exponente de Lyapunov es un cálculo utilizado para medir la tasa promedio de divergencia o convergencia exponencial de trayectorias cercanas en el estado de fase. Debido a su fundamento, el exponente de Lyapunov puede ser utilizado para determinar la predictibilidad de un sistema dinámico tomando como base las condiciones iniciales cercanas que generan estados iniciales casi idénticos para el sistema dinámico, de forma que la divergencia exponencial de las órbitas implica pérdida de predictibilidad [31].

El exponente de Lyapunov se determina con la expresión

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

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.

La cantidad de exponentes de Lyapunov de cualquier sistema siempre será equivalente al orden del mismo, es decir, para un sistema de orden N se tendrán N exponentes de Lyapunov. El valor de los exponentes de los sistemas dinámicos permite determinar el comportamiento de los mismos, los cuales se clasifican en cuatro casos posibles [32]:

- Para un valor cero del exponente de Lyapunov se dice que el sistema presenta periodicidad (o cuasiperiodicidad).
- Un valor negativo implica que el sistema presenta un equilibrio estable.
- Si el valor del exponente es positivo significa que el sistema presenta sensibilidad a las condiciones iniciales (presencia de caos).
- Aquellos sistemas que presentan más de un exponente de Lyapunov positivo se les conoce como *hipercaóticos*.

Este cálculo fue realizado con cada uno de los mapas caóticos estudiados para confirmar la presencia de caos en los mismos, y poder seleccionar los más óptimos para ser aplicados en un algoritmo de cifrado caótico en sistema embebido. Debido a la complejidad que supone dicho cálculo para mapas de más de una dimensión, en la literatura se encuentran distintos algoritmos que se encargan de determinar el exponente de Lyapunov de forma numérica a partir de las secuencias numéricas generadas por los sistemas y mapas caóticos. Para este trabajo de tesis se ha utilizado el método presentado en [33] para estimar el máximo exponente de Lyapunov de cada uno de los mapas caóticos estudiados, y se corrobora con el presentado en la literatura.

3.1.4. Mapas caóticos estudiados

Para seleccionar los mapas caóticos a ser utilizados en este trabajo de tesis primeramente se contemplan seis mapas caóticos encontrados en el estado del arte, considerando como parámetro que los mismos pertenecieran al conjunto de mapas de dos dimensiones (conformados por un par de ecuaciones en diferencias) para ser utilizados posteriormente en el algoritmo se cifrado propuesto.

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

El mapa está definido como [34]:

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

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

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).

Para el análisis del máximo exponente de Lyapunov del mapa se utilizaron los valores para los parámetros de $a = 0.9991568421$ y $b = 7.8945162136$ con las condiciones

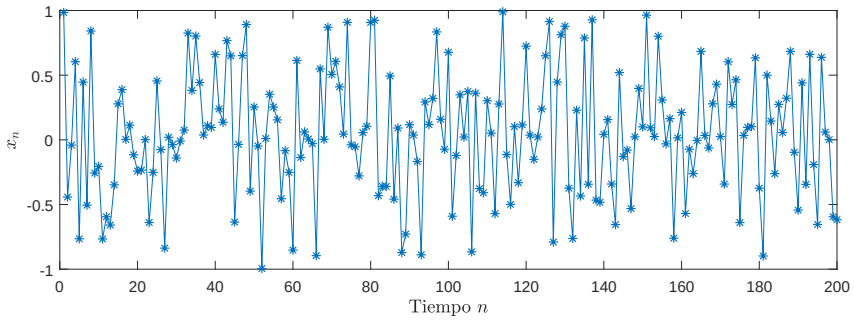


Figura 3.2: Estado x del mapa SLIM con dinámicas caóticas.

iniciales $x_0 = 0.9846216548$ y $y_0 = 0.6548162187$. El resultado es de $\lambda = 5.37$, indicando la presencia de caos.

Función Seno-Chebyshev-Lienal (SCL)

El mapa 2D-SCL surge de la combinación de una función lineal y dos mapas de una dimensión, los cuales corresponden a los mapas caóticos Seno y Chebychev, quedando definido como [35]:

$$x_{i+1} = k \sin(a \cos(b \arccos(x_i))(y_i + c)), \tag{3.4a}$$
$$y_{i+1} = k \sin(a \cos(b \arccos(y_i))(x_{i+1} + c)) \tag{3.4b}$$

donde x_0 y y_0 son las condiciones iniciales y k, a, b y c son los parámetros de control del mapa con $k \in (0, 1]$, $a, b \in (0, \infty)$ y $c \in (-\infty, \infty)$ para la generación de dinámicas caóticas. Utilizando los valores para las condiciones iniciales de $x_0 = 0.9846216548$ y $y_0 = 0.6548162187$ $k = 1$, $a = 2\pi$, $b = \pi$ y $c = 6$ se genera la secuencia mostrada en la figura 3.3, mientras que al calcular el exponente de Lyapunov con los mismos valores para los parámetros y condiciones iniciales se obtiene un valor de $\lambda = 4.55$ el cual, al ser un valor positivo, garantiza la presencia de caos.

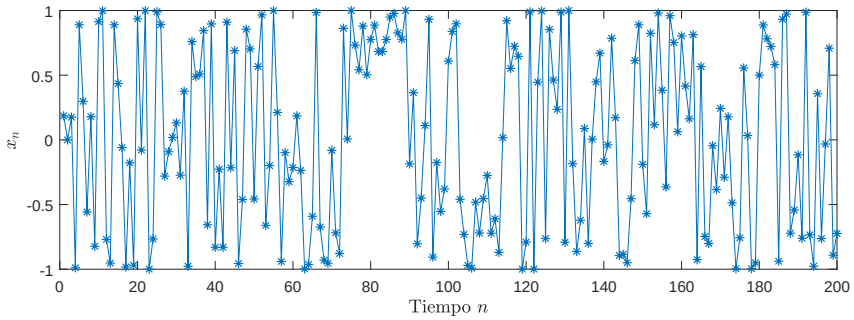


Figura 3.3: Estado x del mapa SCL con dinámicas caóticas.

Mapa Seno-Logístico-Acoplado (LSCM)

Este mapa caótico se deriva del acoplamiento de los mapas caóticos Seno y Logístico con la finalidad de mejorar sus comportamientos caóticos, y mediante una extensión de la dimensión de los mismos de una a dos dimensiones se obtienen las siguientes ecuaciones que lo describen [36] :

$$x_{i+1} = \sin\left(\pi(4\theta x_i(1-x_i) + (1-\theta)\sin(\pi y_i))\right), \tag{3.5a}$$
$$y_{i+1} = \sin\left(\pi(4\theta y_i(1-y_i) + (1-\theta)\sin(\pi x_{i+1}))\right) \tag{3.5b}$$

donde x_0 y y_0 son las condiciones iniciales y θ es el parámetro de control, el cual debe cumplir $\theta \in (0, 1)$ para que el mapa genere secuencias caóticas (ver figura 3.4).

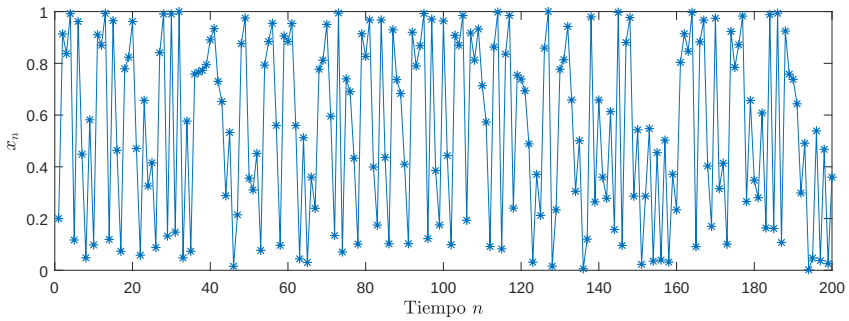


Figura 3.4: Estado x del mapa LSCM con dinámicas caóticas.

Para el análisis del máximo exponente de Lyapunov se considera $x_0 = 0.2$ y $y_0 = 0.9$ como condiciones iniciales y $\theta = 0.98$ como parámetro de control, resultando en un valor para el exponente de $\lambda = 1.34$, garantizando un comportamiento caótico de parte del mapa LSCM.

Mapa caótico logístico iterativo con colapso infinito (LICM)

El mapa caótico LICM surge de la adaptación de los mapas unidimensionales Logístico e ICMIC [37] en un modelo de acoplamiento modulado y con el uso de una función lineal, resultado en el conjunto de ecuaciones [38]:

$$x_{i+1} = \sin\left(\frac{21}{a(y_i+3)kx_i(1-kx_i)}\right), \tag{3.6a}$$
$$y_{i+1} = \sin\left(\frac{21}{a(kx_{i+1}+3)y_i(1-y_i)}\right) \tag{3.6b}$$

donde x_0 y y_0 son las condiciones iniciales mientras que a y k son los parámetros de control del sistema con rangos $a \in (0, \infty)$ y $k \in (0, \infty)$, mientras que cuando se cumple $a = 0.6$ y $k = (0.6, 1.4)$ el mapa presenta un comportamiento hipercaótico (ver figura 3.5). Para el cálculo del exponente de Lyapunov se utiliza $x_0 = 0.9$ y $y_0 = 0.2$ como

condiciones iniciales del sistema y para el valor de los parámetros se utiliza $a = 0.6$ y $k = 1.2$, dando como resultado un valor de exponente de Lyapunov de $\lambda = 5.41$, por lo que el sistema efectivamente presenta un comportamiento caótico.

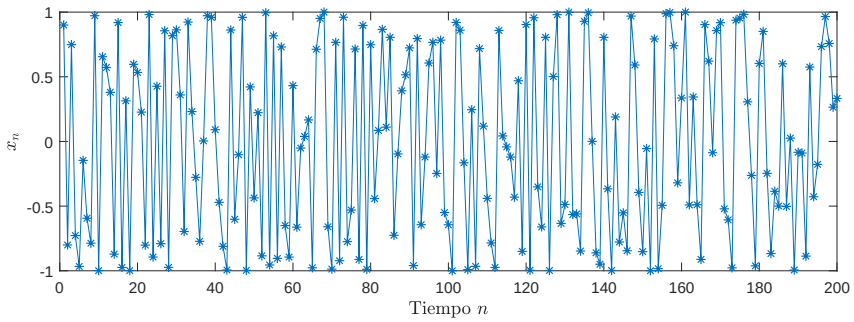


Figura 3.5: Estado x del mapa LICM con dinámicas caóticas.

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:

$$x_{n+1} = (1 - a\sin^2(x_n) + y_n) \ mod \ 1, \tag{3.7a}$$

$$y_{n+1} = bx_n \ mod \ 1 \tag{3.7b}$$

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 3.6). Para confirmar lo anterior se realiza el cálculo del máximo exponente de Lyapunov utilizando $x_0 = 0.35864321$ y $y_0 = 0.89546841$ como valores para las condiciones iniciales así como $a = 8.546198165$ y $b = 9.235646218$ como valores para los parámetros de control, resultando en un valor para el exponente de Lyapunov de $\lambda = 2.17$, confirmando su comportamiento caótico.

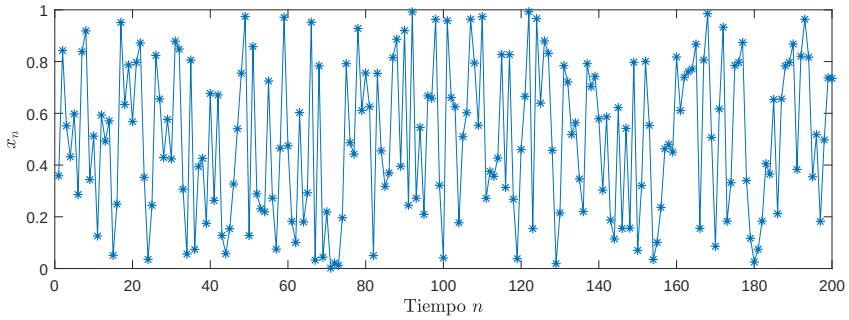


Figura 3.6: Estado x del mapa Seno-Hénon con dinámicas caóticas.

Mapa Logístico Acoplado

El mapa Logístico Acoplado de dos dimensiones conforma un sistema diseñado con la finalidad de aumentar la complejidad del mapa Logístico unidimensional, y se encuentra definido por el siguiente par de ecuaciones [40]:

$$x_{n+1} = r(3y_n + 1)x_n(1 - x_n), \tag{3.8a}$$
$$y_{n+1} = r(3x_{n+1} + 1)y_n(1 - y_n) \tag{3.8b}$$

donde x_0 y y_0 corresponden a las condiciones iniciales del sistema mientras r corresponde al parámetro de control del mismo, el cual otorga un comportamiento caótico al sistema cuando se cumple $r > 0$, comportamiento corroborado al evaluar el sistema con $x_0 = 0.681651$ y $y_0 = 0.916506$ como condiciones iniciales y $r = 1.190154$ (ver figura 3.7), obteniendo además un exponente de Lyapunov de $\lambda = 0.43$ en el estudio realizado.

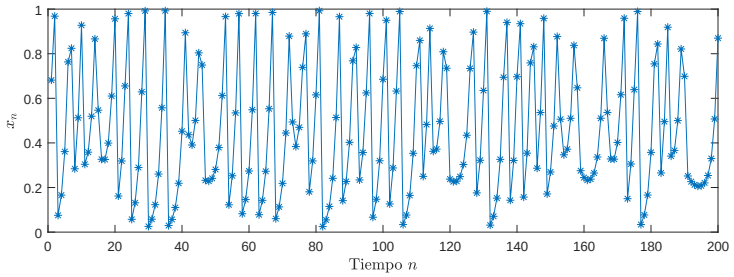


Figura 3.7: Estado x del mapa Logístico Acoplado con dinámicas caóticas.

3.1.5. Mapas caóticos seleccionados

Una vez estudiados los mapas caóticos recuperados del estado del arte se seleccionan los mapas caóticos a utilizar en este trabajo de tesis, para lo cual se considera el valor del exponente de Lyapunov calculado para cada mapa caótico. En la tabla 3.1 se resume el valor del máximo exponente de Lyapunov para los mapas caóticos estudiados con los valores expuestos en la literatura y los valores estimados con el método utilizado.

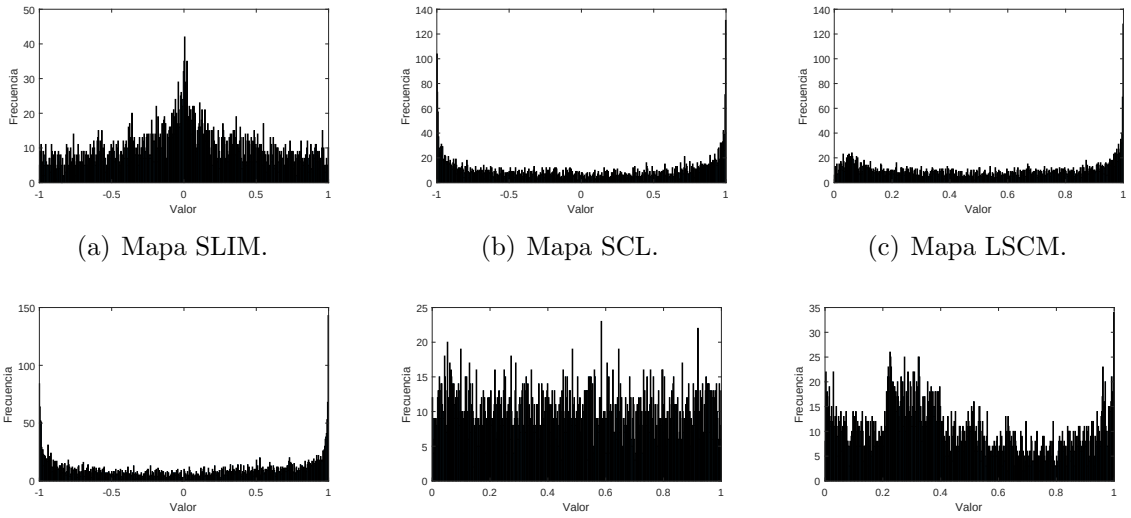
Tabla 3.1: Máximo exponente de Lyapunov de los mapas caóticos estudiados.

Mapa caótico	Máximo exponente de Lyapunov	
	Literatura	Estimado
SLIM	6	5.37
SCL	4.8	4.55
LSCM	1.4	1.34
LICM	5	5.41
Seno-Hénon	3	2.17
Logístico Acoplado	0.56	0.43

Por otro lado, con la finalidad de seleccionar los mapas caóticos más apropiados en términos de eficiencia se realiza el cálculo del tiempo de procesamiento para 5000 muestras de cada uno de los mapas estudiados mediante el uso del software Matlab (ver tabla 3.2) así como un análisis de los histogramas correspondientes a las secuencias numéricas de cada uno (ver figura 3.8) para seleccionar los mejores en cuanto a distribución de datos se refiere.

Tabla 3.2: Tiempo de procesamiento de los mapas caóticos estudiados.

Mapa caótico	Tiempo de procesamiento [ms]
SLIM	5.333
SCL	9.596
LSCM	7.051
LICM	6.845
Seno-Hénon	4.826
Logístico Acoplado	4.446



(a) Mapa SLIM. (b) Mapa SCL. (c) Mapa LSCM.

(d) Mapa LICM. (e) Mapa Seno-Hénon. (f) Mapa Logístico Acoplado.

Figura 3.8: Histogramas correspondientes al estado x de cada uno de los mapas caóticos estudiados.

Acorde a los resultados obtenidos se han seleccionado los mapas caóticos SLIM y Seno-Hénon para la implementación del algoritmo de cifrado propuesto en este trabajo de tesis pues ambos presentan un bajo tiempo de procesamiento (5.333 ms y 4.826 ms respectivamente), generan secuencias caóticas con un exponente de Lyapunov elevado (5.37 para el mapa SLIM y 2.17 para el mapa Seno-Hénon) y además presentan una distribución de datos más uniforme que los demás mapas caóticos, como se puede observar en las figuras 3.8(a) y 3.8(e).

3.2. Criptografía

Enviar mensajes privados es una actividad que toda persona ha necesitado en algún punto de su vida por lo que se han buscado métodos para ocultar la existencia del mensaje mismo, sin embargo en ocasiones eso no es suficiente pues al ser descubierta la existencia del mismo su contenido queda expuesto, por lo que ha entrado en juego otro procedimiento más sofisticado: la *criptografía*.

3.2.1. Definición y antecedentes

La palabra *criptografía* tiene su origen en los vocablos griegos *kriptos* que significa "escondido" y *gráhos* que significa "escritura" [41] por lo que como concepto general se puede entender a la criptografía como la acción de ocultar la escritura, aunque en un concepto más amplio se podría interpretar como el proceso de ocultar un mensaje.

Como definición, hoy en día se entiende a la criptografía como el estudio de técnicas matemáticas dedicadas a proteger la divulgación de información secreta a usuarios no autorizados [42] y cuyo objetivo principal no es esconder la existencia de un mensaje sino su significado, de forma que en caso de que el mensaje lo reciba algún usuario no autorizado este no sea capaz de comprender el mismo a pesar de poder ver su contenido. En ese sentido, existen cuatro objetivos que la criptografía busca satisfacer [43]:

- *Privacidad*: Hace referencia a que el contenido de la información únicamente puede ser conocido por usuarios autorizados, manteniéndolo desconocido para aquellos usuarios no autorizados o *intrusos*. Existen numerosas técnicas para garantizar la privacidad de la información, las cuales van desde protección física hasta complejos algoritmos matemáticos que se encargan de que la información sea incomprensible.
- *Integridad de la información*: Es un servicio que garantiza que la información no puede ser manipulada durante el proceso de transmisión, por lo que se debe ser capaz de detectar dicha manipulación por usuarios no autorizados. Existen distintos tipos de manipulación de los datos, entre los cuales se encuentran:
  - *Inserción*: Se refiere a que se ha agregado información al mensaje original.
  - *Sustitución*: Relacionado con la acción de modificar el contenido del mensaje original sin modificar la longitud del mismo.
  - *Eliminación*: Como su nombre lo indica, hace alusión a la acción de eliminar parte del mensaje original, modificando su significado.
- *Autenticidad*: Se refiere a la capacidad de confirmar que el mensaje recibido fue realmente enviado por quien dice que lo envió, además de que el mensaje recibido es el que se esperaba (integridad de la información).
- *No rechazo*: Garantiza que no se pueda negar la autoría de un mensaje previamente enviado.

El origen de la criptografía no se conoce por completo, no obstante existen autores que relacionan sus inicios a los orígenes del hombre, desde que aprendió a comunicarse y tuvo que encontrar medios de asegurar la confidencialidad de una parte de sus comunicaciones. Sin embargo, el primer testimonio de uso deliberado de métodos técnicos que permitieran cifrar los mensajes proviene de Grecia sobre el siglo V a.C, donde los lacedemonios utilizaban un instrumento llamado escítala o escítalo (ver figura 3.9) durante las guerras entre Esparta y Atenas. Esta herramienta consistía en un palo o bastón en el cual se enrollaba en espiral una tira de cuero sobre la que se escribía el mensaje en columnas paralelas al eje del palo. La tira desenrollada mostraba un texto sin relación aparente con el texto inicial, pero que podía leerse volviendo a enrollar la tira sobre un palo del mismo diámetro que el primero, por lo que el receptor debía tener un palo igual al de la persona que mandaba el mensaje para poder leerlo [44]. De esta forma se dificultaba la lectura de un mensaje en caso de que fuera interceptado.

Figura 3.9: Escítala espartana usada por los griegos para cifrar mensajes.

Años más tarde en el siglo I a.C llegaría Julio César, un militar Romano quien en primera instancia, durante una campaña con el ejército de Roma, "codificó" un mensaje cambiándolo del latín al griego con la finalidad de que si caía en manos enemigas estos no pudieran descifrar el mensaje. Sin embargo su mayor aportación provino con el que después se llamaría cifrado Julio César, un sistema que diseñó para comunicarse con el ejército romano en el que sustituía cada letra del mensaje por la que estaba en el alfabeto tres posiciones hacia adelante, mientras que a las tres últimas letras les asignaba las primeras tres letras del alfabeto [45]. A este y otros métodos de cifrado desarrollados hasta los años 1900 como lo fueron el cifrado por medio del disco de Alberti, el cifrado Vigenère [46] y el cifrador de discos de Jefferson hoy en día se consideran dentro del campo de la *criptografía clásica*.

Con la llegada de las máquinas mecánicas y eléctricas inició una edad de oro para las máquinas de cifrado con el nacimiento de la Enigma (ver figura 3.10) creada por el ingeniero alemán Arthur Scherbius, mientras en Japón inventaron a Purple, en Estados Unidos tenían a SIGABA y la versión inglesa llevaba el nombre de Typex, dando inicio de igual manera a una nueva era para la criptografía a la que se le denominó como *criptografía moderna*. Finalmente, la creación de la computadora ha provocado un gran avance en el desarrollo de algoritmos criptográficos enfocados a ecuaciones matemáticas que proporcionan un mayor nivel de seguridad como lo son el cifrado Lucifer, el cifrado DES y el sistema RSA.

Figura 3.10: Máquina Enigma utilizada por los alemanes para cifrar mensajes.

3.2.2. Criptosistema y su clasificación

Se puede entender un *criptosistema* como un conjunto de bases criptográficas utilizadas para proporcionar seguridad a la información y que suelen componerse de algún tipo de algoritmo matemático. Formalmente, se puede definir a un criptosistema (o esquema de encriptación) como un conjunto (**m**,**C**,**K**,**E**,**D**) [47] donde:

- **m**: Mensaje claro transmitido.
- **C**: Mensaje claro cifrado (también conocido como *criptograma*).
- **K**: Conjunto de claves que se utilizan en el proceso de encriptado.
- **E**: Familia de funciones que se utilizan para encriptar el mensaje claro.
- **D**: Familia de funciones que se utilizan para desencriptar el mensaje claro.

Tomando en cuenta estos elementos, todo criptosistema cumple con la condición

$$D_K(E_K(m)) = m \tag{3.9}$$

la cual describe que al cifrar un mensaje claro m por medio de la función E utilizando la clave K y al descifrarlo con una función D utilizando la misma clave K se recupera el mensaje claro m (ver figura 3.11), y a su vez indica que la función D es la función inversa de E.

Por su parte, los sistemas criptográficos pueden ser clasificados de distintas formas de acuerdo a las distintas características o propiedades que pueden presentar en su estructura [48]:

1. **Tipo de clave secreta.**
    - *Cifrado con clave simétrica o de clave secreta*: En este esquema se utiliza una única clave K para encriptar el mensaje claro y desencriptar el criptograma generado al encriptar. Esto lleva consigo algunas implicaciones pues es necesario que tanto el emisor como el receptor del mensaje conozcan la

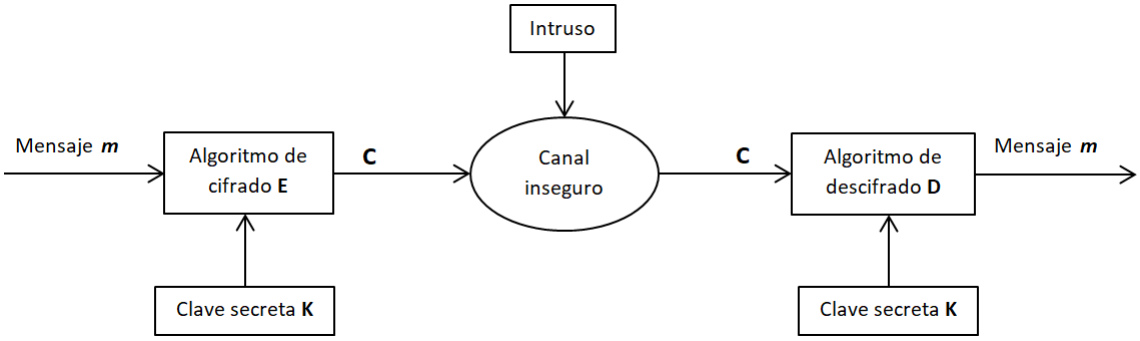


Figura 3.11: Esquema de sistema criptográfico.

clave secreta común para que este esquema sea implementado, por lo que ambas partes deben acordar la clave a utilizar de alguna forma segura, como lo podría ser de forma física.

- *Cifrado con clave asimétrica o de clave pública*: Por definición, es aquella que utiliza una clave, que se denomina de dominio público, y una segunda clave de tipo privada. El concepto general de este esquema es que el emisor utiliza la clave pública del receptor para encriptar el mensaje claro y únicamente el receptor será capaz de desencriptarlo utilizando la clave privada, de forma que si el mensaje es interceptado en el canal de comunicación el intruso no podrá conocer su contenido aunque conozca la clave pública. Este tipo de cifrado proporciona algunas ventajas respecto al esquema de cifrado simétrico como lo es el problema de compartir la clave secreta a utilizar puesto que en este caso, aunque está matemáticamente relacionada con la clave privada, la clave pública puede ser enviada por un canal inseguro (como internet). Sin embargo, al mismo tiempo presenta la desventaja de que es más lento debido a que los algoritmos utilizados tienden a ser más complejos.

2. **Tipo de algoritmo.**

   - *Cifrado en flujo*: Se basa en realizar un cifrado bit a bit (o byte a byte) del mensaje claro. Esto se logra generando una cadena de la misma longitud que el mensaje claro por medio de la clave secreta y utilizando la operación or-exclusiva (X-OR).
   - *Cifrado en bloque*: El mensaje claro es dividido en bloques de una longitud fija (64 o 128 bits, por ejemplo)y se va cifrando bloque a bloque hasta completarlo.

3. **Método de cifrado.**

   - *Métodos tradicionales o convencionales*: Comprenden todos aquellos algoritmos que hacen uso de bases matemáticas para cifrar al mensaje claro, como es la teoría de números, curvas elípticas, transformaciones lineales, entre otros.

- *Métodos no convencionales*: Utiliza algoritmos basados en áreas que se encuentran en investigación, formando nuevos campos de estudio como lo es la criptografía cuántica, la criptografía basada en ADN y la criptografía caótica.

Adicionalmente, los criptosistemas pueden ser identificados por el tipo de operaciones que el algoritmo de cifrado realiza sobre el mensaje claro, de las cuales se tienen dos [49]:

- *Confusión*: Describe la relación entre el criptograma y el mensaje claro, por lo que debe de ser lo más confuso y complejo posible. La operación asociada es la de *permutación*, en la que se cambia la posición de cada elemento del texto claro de manera desordenada sin agregar nuevos elementos, es decir, se mantiene la longitud inicial del mensaje claro.

- *Difusión*: Hace referencia a la propiedad de difundir la redundancia del mensaje claro en las estadísticas del criptograma, consistiendo en modificar el valor de cada elemento del mensaje claro de manera desordenada.

A pesar de que se presentan de forma separada, las operaciones de confusión y difusión suelen utilizarse en conjunto en los algoritmos de sistemas criptográficos, llegando incluso a realizar el proceso en repetidas ocasiones con el objetivo de agregar mayor complejidad al proceso de cifrado y de esa manera aumentar el nivel de seguridad del sistema.

3.2.3. Seguridad en los criptosistemas y cifrado caótico

La seguridad es evidentemente el principal objetivo de los sistemas criptográficos y por lo tanto uno de los aspectos más estudiados de los mismos. En el siglo XIX, Auguste Kerckhoff expuso algunos puntos clave que debían ser cumplidos para que un sistema criptográfico pudiera ser considerado seguro, dentro de los que destacaban que ni el texto claro ni la clave secreta deben de ser posibles de obtener mediante el análisis del criptograma, siempre considerando que la clave de cifrado es información de tipo privada mientras que el algoritmo de cifrado se debe considerar como información de tipo pública [50].

Contemplando lo establecido por Kerckhoff, para analizar el nivel de seguridad de los sistemas criptográficos se han considerado diferentes tipos de ataques que puede llevar a cabo un criptoanalista para llegar a obtener el mensaje claro a partir de su correspondiente criptograma, de entre los que destacan tres categorías:

1. **Ataque exhaustivo:** El ataque consiste en intentar todas las claves posibles para encontrar aquella que descifra el criptograma, proceso que es llevado a cabo por una computadora y por lo cual el tiempo para encontrar la clave está directamente relacionado con el poder computacional utilizado y con el espacio de clave secreta utilizado en el proceso de encriptado.

2. **Ataque diferencial:** Este tipo de ataque pone a prueba no solo la sensibilidad del sistema criptográfico a la clave secreta, sino también al mensaje claro. Existen cuatro casos diferentes que son considerados cuando se realiza un análisis de seguridad de un sistema criptográfico [51]:

    - **Ataque de criptograma conocido:** Este constituye el ataque más básico por un criptoanalista puesto que se considera que el intruso únicamente tiene acceso al criptograma y a partir del mismo intenta deducir el mensaje claro.
    - **Ataque de mensaje claro conocido:** Aquí el intruso tiene acceso a algunos criptogramas con sus correspondientes mensajes claros, todos generados con la misma clave secreta. El objetivo en este tipo de ataque es realizar deducciones mediante la comparación de los diferentes criptogramas con sus respectivos textos claros.
    - **Ataque de mensaje claro elegido:** En este ataque el intruso se encuentra en la posibilidad de obtener el par mensaje claro/criptograma como en el ataque de mensaje claro conocido, con la diferencia de que el intruso está en la posibilidad de elegir el mensaje claro que desee.
    - **Ataque de criptograma elegido:** Finalmente, en este ataque el intruso selecciona el criptograma que desee con su respectivo mensaje claro para su análisis.

3. **Ataque estadístico:** Como su nombre lo indica, son ataques basados en la estadística que apelan a la uniformidad del criptograma, realizando análisis como lo son los histogramas y la correlación del mensaje.

Tomando como base los ataques que podría realizar un criptoanalista a través de un criptograma, queda evidente que el proceso de cifrado del mensaje claro debe ser lo suficientemente complejo para evitar que sea quebrantado por una computadora, razón por la cual en las últimas dos décadas la investigación del lado de la criptografía no convencional ha crecido con el paso de los años con el objetivo de sustituir a los métodos de cifrado clásicos, pues se ha comprobado que estos últimos se han vuelto cada vez más fáciles de quebrantar con el avance del poder computacional.

Si bien se han presentado distintos campos de estudio en la criptografía no convencional, uno de los más estudiados actualmente son los criptosistemas basados en caos debido a que presenta propiedades superiores a los demás en lo que respecta a ergodicidad, complejidad dinámica y una característica en particular como lo es la sensibilidad a condiciones iniciales, lo cual los hace muy buenos candidatos para la generación de secuencias pseudoaleatorias para llevar a cabo los procesos de confusión y difusión en los algoritmos de cifrado [52].

Por otro lado, se ha encontrado que resulta más conveniente el uso de sistemas caóticos en tiempo discreto a la hora de implementar un algoritmo de cifrado puesto que no solo presentan una degradación mínima respecto a los sistemas caóticos de tiempo continuo sino que además, al ser ejecutados comúnmente por una computadora, resultan

mucho más eficientes pues generan secuencias que pueden ser utilizadas en el algoritmo al realizar simples iteraciones del sistema.

A pesar de lo ya mencionado, un sistema criptográfico no se puede llamar seguro por el simple hecho de estar basado en caos pues en la literatura se ha registrado que aún estos se encuentran vulnerables frente a algunos ataques criptoanalíticos como lo son el ataque de criptograma elegido y el ataque de mensaje claro elegido [53]. Debido a esto, en la literatura se han registrado algunas reglas que deben ser consideradas a la hora de implementar un sistema criptográfico basado en caos con al finalidad de diseñarlo de la mejor forma posible [54]:

1. Se deben tener previstos los detalles de implementación del sistema caótico.
2. Facilidad de implementación sin comprometer la seguridad, el costo y la velocidad.
3. La definición de la clave secreta debe ser precisa, estableciendo el rango de la misma para evitar regiones no caóticas.
4. La clave debe presentar una distribución uniforme, de forma que una mínima información de la clave no debe revelar información acerca del mensaje claro.
5. El proceso de generación de claves debe estar bien definido.
6. Un mínimo cambio en el mensaje claro o en la clave debe dar como resultado un criptograma completamente diferente para impedir cualquier ataque estadístico.
7. Debe ser analizado en busca de cualquier debilidad para evitar cualquier tipo de ataque.
8. Las secuencias aleatorias generadas por medio de un generador pseudoaleatorio debe ser evaluado utilizando pruebas estadísticas.

3.3. Conclusiones del capítulo

La criptografía ha probado ser una herramienta fundamental en las comunicaciones entre los seres humanos a lo largo de la historia donde los métodos de cifrado simplemente se han adecuando de acuerdo a la tecnología presente en cada época, pero siempre cumpliendo con su objetivo fundamental como herramienta para transmitir un mensaje de forma segura, es decir, que solamente el destinatario reciba el mensaje deseado. Por otro lado, si bien hoy en día existen diversos métodos de cifrado entre los más prometedores se encuentran sin duda aquellos que involucran a la teoría del caos para cada uno de los procesos a realizar pues se aprovechan propiedades que son intrínsecas al caos como lo son la sensibilidad a condiciones iniciales y la ergodicidad, disminuyendo de esa forma la complejidad del algoritmo y disminuyendo por consiguiente el tiempo de procesamiento.

En este capítulo se presentaron diversos mapas caóticos como potenciales candidatos para aplicaciones criptográficas y a los cuales se les realizaron pruebas de tiempo de ejecución, exponente de Lyapunov y de histograma para verificar la existencia de caos. Finalmente se ha optado por utilizar los mapas SLIM y Seno-Hénon para este trabajo de tesis debido a su buen comportamiento caótico y eficiencia.

Capítulo 4

Algoritmo de cifrado propuesto

En este capítulo se presenta el algoritmo de cifrado caótico propuesto en este trabajo de tesis el cual consiste en utilizar una llave de 32 caracteres hexadecimales (128 bits) con la que se generan un par de secuencias caóticas a partir de los mapas SLIM y Seno-Hénon, las cuales son utilizadas para los procesos de confusión y difusión aplicados en las señales biomédicas.

4.1. Introducción

Hoy en día las señales biométricas empiezan a tomar un papel más relevante pues ya no solo son utilizadas para la detección o tratamiento de enfermedades sino que en años recientes han comenzado a utilizarse para la identificación de cada individuo por lo que, como se mencionó en capítulos anteriores, es de vital importancia mantener la privacidad y seguridad en los procesos de transmisión y almacenamiento de la información médica de cada paciente en los sistemas de telemedicina. Para ello, en la literatura se han registrado en los últimos años distintos algoritmos de cifrado basados en mapas caóticos para el encriptado de señales biomédicas [55, 56] , de tal forma que si la señal es interceptada en algún punto del sistema esta no pueda ser utilizada.

Para este trabajo de tesis, el algoritmo propuesto está basado en [57] realizando algunas modificaciones relacionadas con los mapas caóticos utilizados. Contemplando las diferentes clasificaciones de los sistemas criptográficos mencionadas en la sección 3.2.2, el algoritmo de cifrado propuesto presenta las siguientes características:

- *Cifrado con clave simétrica*: El algoritmo propuesto utiliza la misma clave secreta tanto para el proceso de cifrado como para el proceso de descifrado.
- *Cifrado en flujo*: El algoritmo toma el mensaje claro (señal clara) y cifra un elemento a la vez hasta cifrar el mensaje completo.
- *Arquitectura de confusión y difusión*: Tomando todo el mensaje claro (señal clara) se intercambia la posición de cada uno de los elementos al igual que el valor de los mismos.

- *Método de cifrado no convencional*: Se utilizan los mapas caóticos SLIM (3.3) y Seno-Hénon (3.7) para la generación de las secuencias pseudoaleatorias utilizadas en los procesos de confusión y difusión.

En la figura 4.1 se puede observar un diagrama a bloques del proceso de cifrado, el cual se lleva a cabo de la siguiente manera: primeramente se itera el mapa SLIM con los parámetros y condiciones iniciales en base a la clave secreta de 128 bits para después realizar el cálculo de Z, que se relaciona con las secuencias caóticas generadas con el mapa SLIM y la señal clara. Posteriormente se itera el mapa Seno-Hénon en base al valor de Z, la clave secreta y las secuencias caóticas generadas con el mapa SLIM para continuar con los procesos de confusión y difusión sobre la señal clara (previamente transformada en amplitud) para finalmente agregar el valor de Z al criptograma para que el receptor pueda descifrar correctamente la señal enviada.

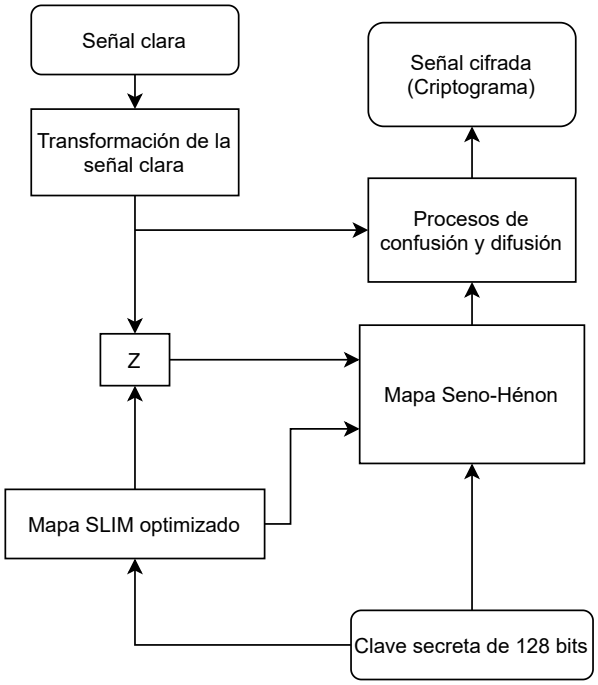


Figura 4.1: Diagrama a bloques del proceso de cifrado propuesto.

El proceso de descifrado se realiza al invertir el proceso de cifrado realizado, tal como se muestra en la figura 4.2. Primeramente se extrae el valor de Z del criptograma para después iterar el mapa SLIM utilizando la clave secreta. Con las secuencias caóticas generadas y el valor de Z se procede a iterar el mapa Seno-Hénon para posteriormente ejecutar el proceso de descifrado (el cual consiste en los procesos de confusión y difusión de forma inversa) sobre el criptograma para finalmente realizar el proceso de transformación de amplitud de manera inversa para obtener la señal clara.

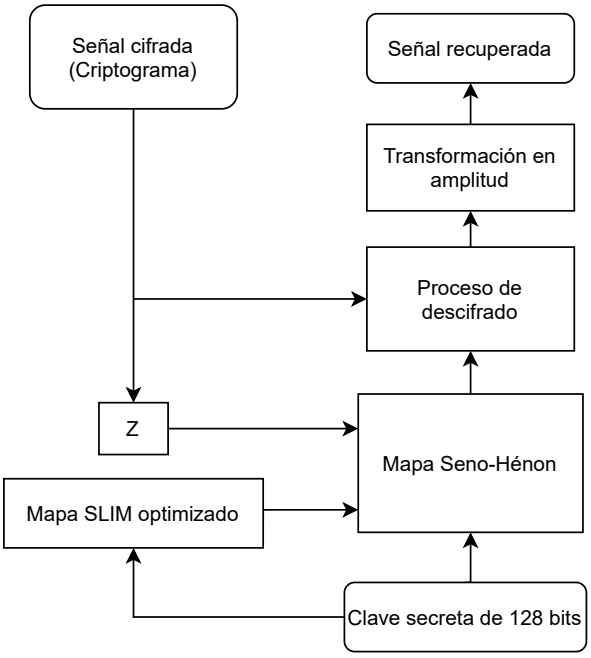


Figura 4.2: Diagrama a bloques del proceso de descifrado propuesto.

4.2. Definición de la clave secreta

La clave secreta K se encuentra definida por una secuencia de 128 bits representada por 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	$A = \frac{(H_1,H_2,\ldots,H_8)_{10}}{2^{32}+1}$	$B = \frac{(H_9,H_{10},\ldots,H_{16})_{10}}{2^{32}+1}$	$C = \frac{(H_{17},H_{18},\ldots,H_{24})_{10}}{2^{32}+1}$	$D = \frac{(H_{25},H_{26},\ldots,H_{32})_{10}}{2^{32}+1}$
SLIM	$a_1 = 0.999 + [((A+B)\ mod\ 1) * 0.001]$		$x_{1_0} = (C+D)\ mod\ 1$	
	$b_1 = 7.999 + [((A+C)\ mod\ 1) * 0.001]$		$y_{1_0} = (B+D)\ mod\ 1$	
Seno-Hénon	$a_2 = 9.999 + [((A+B+Z)\ mod\ 1) * 0.001]$		$x_{2_0} = (C+D+Z)\ mod\ 1$	
	$b_2 = 8.999 + [((A+C+Z)\ mod\ 1) * 0.001]$		$y_{2_0} = (B+D+Z)\ mod\ 1$	
Rango	$0.999 < a_1 < 1$	$9.999 < a_2 < 10$	$7.999 < b_1 < 8$	$8.999 < b_2 < 9$
	$0 < x_{1_0,2_0}, y_{1_0,2_0} < 1$			
Precisión	10^{-15}			
donde $(a\ mod\ b) = (a-b) \times (a/b)$ con $b \neq 0$				

4.3. Transformación de la señal clara

Para realizar el procesamiento de la señal clara se realiza una transformación de la señal clara para que la misma se encuentre en un rango de entre 0 y 1. Para ello se

determina la transformación

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

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

4.4. Cálculo de Z

El valor de Z ayuda a relacionar a la señal clara con el algoritmo de cifrado para aumentar la sensibilidad del criptograma a la señal clara así como a la clave secreta, de forma que pequeños cambios en estos generen un criptograma diferente. Para calcular el valor de Z primeramente se itera el mapa SLIM para obtener una secuencia de datos caóticos, sin embargo como se observó en la figura 3.8(a) el mapa SLIM presenta una distribución de datos *no uniforme* pues los datos generados al iterar el mapa, si bien son caóticos, se encuentran cargados en la zona central del rango de valores por lo que la seguridad de la información cifrada podría verse comprometida. Esto es debido a que al encontrarse los datos concentrados en una zona específica del histograma ocasiona que los procesos de confusión y difusión se lleven a cabo de forma inadecuada, haciendo al criptograma susceptible a ataques diferenciales y estadísticos (tal como se menciona en la sección 3.2.2).

Como solución a este problema, se realiza un proceso de optimización a la secuencia de datos generada por el mapa SLIM que consiste en eliminar los tres primeros dígitos después del punto decimal de cada uno de los datos de la secuencia con la operación

$$x_i^S = \left(x_i^S * 1000\right)\ mod\ 1, \quad para\ i = 1, 2, 3, \ldots, I_2 \tag{4.2}$$

donde I_2 es el número de iteraciones del mapa SLIM y *mod* es la operación de módulo. De esta forma se logra obtener una mejor distribución de los datos como se observa en la figura 4.3.

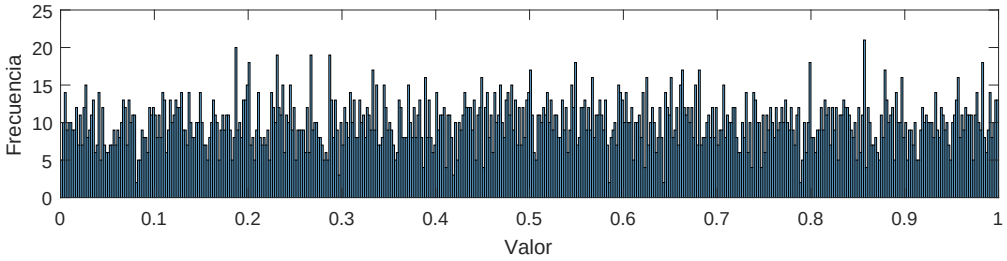


Figura 4.3: Histograma de la secuencia caótica optimizada generada con el mapa SLIM.

Una vez obtenida la secuencia de datos caóticos optimizados del mapa SLIM se realiza la suma de todos los elementos de la señal transformada junto con la secuencia de datos caóticos generados para obtener finalmente el valor de Z, tal como se observa en la siguiente expresión

$$Z = \left\{ S + [PT_i * x^S_{I_2+1-i}] + x^S_{I_2+1-i} \right\} \ mod\ 1, \quad para\ i = 1, 2, 3, \ldots, I_2 \tag{4.3}$$

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.

4.5. Proceso de cifrado

Para el proceso de cifrado primeramente se itera $I_1 = \ell + 100$ veces el mapa caótico Seno-Hénon de acuerdo a los parámetros y condiciones iniciales establecidos en la tabla 4.1, donde ℓ corresponde a la longitud de la señal clara, para generar la secuencia caótica $x^{SH} = x^{SH}_1, x^{SH}_2, x^{SH}_3, \ldots, x^{SH}_{I_1}$ con $x^{SH} \in (0, 1)$, sin embargo se propone una realimentación del mapa SLIM al mapa Seno-Hénon para mejorar las dinámicas caóticas de este último. Para ello se modifica el mapa Seno-Hénon de la siguiente manera:

$$x^{SH}_{i+1} = (1 - a \sin^2(x^{SH}_i) + y^{SH}_i + x^S_i)\ mod\ 1, \tag{4.4a}$$

$$y^{SH}_{i+1} = (bx^{SH}_i + y^S_i)\ mod\ 1 \tag{4.4b}$$

donde x^{SH}_i y y^{SH}_i corresponden a los estados del mapa Seno-Hénon, a y b a los parámetros de control, x^{SH}_0 y y^{SH}_0 a las condiciones iniciales, mientras que x^S_i y y^S_i corresponden a los dos estados del mapa SLIM.

Al igual que con los mapas caóticos estudiados en la sección 3.1.4 se realiza el cálculo del exponente de Lyapunov para el mapa caótico modificado, resultando en que las secuencias numéricas generadas presentan dinámicas caóticas con un exponente de Lyapunov de 2.295. Además, como se puede apreciar en la figura 4.4, el mapa modificado genera secuencias numéricas bien distribuidas por lo que los procesos de confusión y difusión se pueden realizar de forma adecuada.

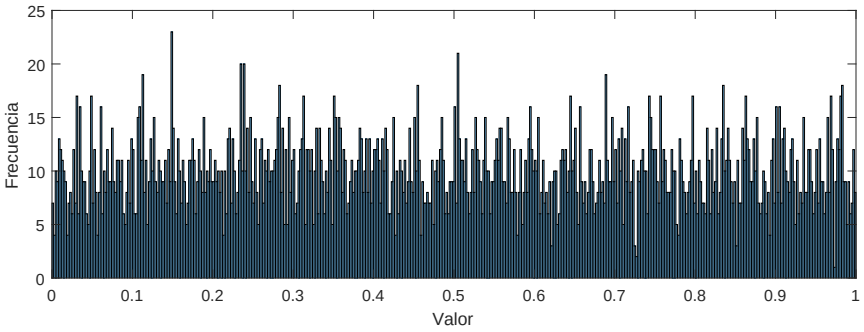


Figura 4.4: Histograma de la secuencia numérica generada con el mapa caótico modificado.

A partir de la secuencia x^{SH} se procede a determinar dos secuencias secundarias, una para realizar el proceso de confusión y otra para el proceso de difusión. Para el proceso de confusión se crea un vector de posición de acuerdo a la siguiente operación

$$CF_i = \ round\left[x^{SH}_{I_1-\ell+i} * (\ell - 1)\right] + 1, \quad para\ i = 1, 2, 3, \ldots, \ell \tag{4.5}$$

donde ℓ es la longitud de la señal clara y $CF \in [1, \ell]$ corresponde al vector para realizar el proceso de confusión. Como se puede observar, con el objetivo de incrementar las características pseudoaleatorias del vector de posiciones se ha optado por contemplar a partir del valor número 100 de la secuencia x^{SH}_i pues es sabido que en los sistemas caóticos el nivel de divergencia entre dos trayectorias con condiciones iniciales muy cercanas incrementa con el paso del tiempo, ya sea continuo o discreto.

Por otro lado, la expresión (4.5) presenta la desventaja de generar valores repetidos por lo que a pesar de tener un vector de la misma longitud que la señal clara se tendrán posiciones repetidas. Para dar solución al problema se realiza un ajuste al vector CF por medio de programación, cambiando los valores repetidos de la secuencia de acuerdo como sigue

$$G_h = [K_h], \quad con\ h \ll \ell \tag{4.6}$$

donde K_h es el valor de la posición faltante en CF de menor a mayor. El vector de valores repetidos G se divide en dos secciones y cada valor se asigna a CF de manera alternada donde un valor repetido aparece, de forma que cuando este proceso termina, se tiene un vector para confusión con todas las posibles posiciones de acuerdo a la longitud de la señal clara (confusión optimizada).

Para el proceso de difusión se vuelve a tomar la secuencia x^{SH} y se crea un vector con valores pseudoaleatorios como sigue

$$DF_i = \left(x^{SH}_{I_1-\ell+i} + Z\right)\ mod\ 1, \quad para\ i = 1, 2, 3, \ldots, \ell \tag{4.7}$$

donde $DF_i \in (0, 1)$ es el vector con valores pseudoaleatorios para el proceso de difusión con longitud ℓ, correspondiente a la longitud de la señal clara.

Habiendo obtenido los vectores para los procesos de confusión y difusión se procede a realizar el proceso de cifrado de la señal transformada de acuerdo a la expresión

$$E_i = PT(CF_i) + DF_i + DF_{\ell-i+1}, \quad para\ i = 1, 2, 3, \ldots, \ell \tag{4.8}$$

donde PT es la señal clara transformada y E la señal cifrada (criptograma).

Finalmente es necesario agregar los valores de Z, P_{min} y N_{max} al final del criptograma E pues son valores que no pueden ser determinados a partir del mismo y es necesario para que el receptor pueda llevar a cabo el proceso de descifrado. Dichos valores se agregan de la siguiente forma

$$E_{\ell+1} = Z \tag{4.9}$$

$$E_{\ell+2} = \begin{cases} 0 & \text{si } P_{min} < 0 \\ 1 & \text{si } P_{min} \geq 0 \end{cases} \tag{4.10}$$

$$E_{\ell+3} = \frac{abs(P_{min})}{1000} \tag{4.11}$$

$$E_{\ell+4} = \begin{cases} 0 & \text{si } N_{max} < 0 \\ 1 & \text{si } N_{max} \geq 0 \end{cases} \tag{4.12}$$

$$E_{\ell+5} = \frac{abs(N_{max})}{1000} \tag{4.13}$$

donde $abs(a)$ representa el valor absoluto de a.

4.6. Proceso de descifrado

Para realizar el proceso de descifrado es necesario aplicar en el criptograma el proceso inverso al efectuado en el proceso de cifrado, es decir, invertir cada uno de los pasos realizados en este. Para ello, se debe hacer uso de la misma clave utilizada en el proceso de cifrado (exactamente los mismos 32 caracteres hexadecimales) puesto que de no ser así no se podrá recuperar de forma correcta la señal clara debido a las operaciones realizadas.

Primeramente se recuperan los valores de Z, P_{min} y N_{max} del criptograma para posteriormente determinar la longitud ℓ del mismo. Después se procede a generar los vectores de confusión y difusión, CF y DF respectivamente, mediante la iteración de los mapas caóticos SLIM y Seno-Hénon utilizando la clave secreta de 128 bits y el valor de Z recuperado. Posteriormente se lleva a cabo el proceso de descifrado mediante la expresión

$$DT_i(CF_i) = E_i - DF_i - DF_{\ell-i+1}, \quad para\ i = 1, 2, 3, \ldots, \ell \tag{4.14}$$

donde E es la señal cifrada (criptograma) y DT es la señal recuperada escalada. Finalmente se realiza el proceso de transformación de forma inversa utilizando los valores de P_{min} y N_{max} para obtener la señal recuperada D.

4.7. Conclusiones del capítulo

En este capítulo se ha presentado un algoritmo de cifrado basado en caos para la transmisión de señales médicas de forma segura a través de canales inseguros. El algoritmo cuenta con una calve secreta de 128 bits (o 32 dígitos hexadecimales) y aprovecha las propiedades de los mapas caóticos utilizados para aumentar la complejidad de los procesos de confusión y difusión, además de involucrar a la señal médica misma a través del valor de Z para de esa forma hacer más robusto el algoritmo ante ataques criptoanalíticos.

Capítulo 5

Implementación de algoritmo de cifrado caótico en sistema embebido

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) y una señal de presión sanguínea (BP). Finalmente se presentan una serie de análisis de seguridad realizados para evaluar si es segura la transmisión de las señales biomédicas a través de internet.

5.1. Sistemas embebidos y microcontroladores

Los sistemas de telemedicina mencionados anteriormente, especialmente aquellos relacionados con el telemonitoreo, han visto un desarrollo importante con el avance de los sistemas embebidos pues gracias a estos se han desarrollado nuevos dispositivos médicos menos invasivos que permiten la captura y almacenamiento de bioseñales en lugares remotos o inclusive en la *nube* [58].

Un *sistema embebido* es un sistema de procesamiento de información y que es integrado a productos para realizar tareas específicas [59] que pueden ser de tiempo real, por lo que tienen cabida en muchas aplicaciones de distintos sistemas desarrollados hoy en día como lo son los sistemas de seguridad, televisores, videojuegos, instrumentos musicales, impresoras, microondas, equipos médicos, sistemas del área automotriz, entre muchos otros (ver figura 5.1). Evidentemente, los sistemas embebidos suelen ser comparados con las capacidades de una computadora de escritorio debido a sus diversas aplicaciones a pesar de que sus objetivos sean distintos: mientras el objetivo de un sistema embebido es el realizar una tarea en específico, una computadora está diseñada para operar software (como un sistema operativo) y realizar distintas tareas que no están relacionadas entre sí, aunque las computadoras sí pueden hacer uso de dispositivos que están integrados por sistemas embebidos (como lo son los teclados y ratones).

Figura 5.1: Aplicaciones de los sistemas embebidos.

En lo que a su estructura se refiere, los sistemas embebidos requieren de una unidad de control para ejecutar las tareas que la aplicación del sistema requiere y que se compone de un microcontrolador o microprocesador que se encuentra dentro del sistema embebido. A su vez, esta unidad le proporciona ventajas particulares respecto a otros sistemas como lo son las computadoras, entre las que se destacan el tiempo de ejecución de la tarea (debido a que se dedican a una única aplicación), bajo costo y un tamaño reducido, pero por sobre todo se encuentra la ventaja relacionada con el consumo de energía pues aunque actualmente se considera como la mayor limitante en este tipo de sistemas [60] su consumo energético sigue siendo verdaderamente eficiente, sobretodo si su programación lo es de igual forma.

Por su parte, un *microcontrolador* se puede definir como un sistema de microcomputadora completo debido a que se compone de 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) que lo hacen ideal para aplicaciones de propósito específico gracias a su tamaño tan compacto, su bajo costo y su sencillo uso [61].

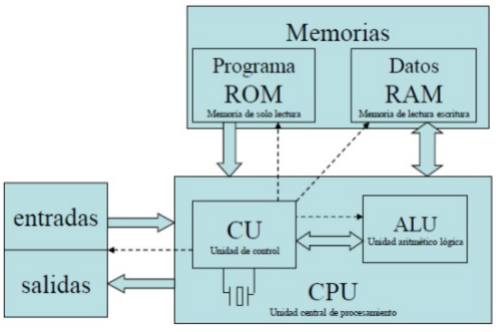


Figura 5.2: Componentes de un microcontrolador.

Los microcontroladores tienen muchas aplicaciones en distintos campos del conocimiento cuando del diseño de dispositivos electrónicos se trata, y si bien en muchas ocasiones son utilizados con orientación hacia el área de control también son ampliamente utilizados para el procesamiento de información como lo es en el campo de la

seguridad, especialmente en el área de la criptografía [62].

5.2. Descripción del sistema embebido propuesto

Con la finalidad de establecer un medio inalámbrico para la transmisión de información médica de forma segura se ha diseñado un sistema que consta esencialmente de un bloque transmisor, un bloque receptor, y el servidor correspondiente que funciona como un intermediario en la transmisión de dicha información.

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 un servidor local donde se almacena el criptograma a la espera de una petición del bloque receptor.

Por parte de la unidad receptora, primeramente se solicita al usuario que presione un botón para iniciar con el proceso de recuperación enviando una petición al servidor a través de una conexión Wi-Fi para recibir por el mismo medio el criptograma enviado por la unidad transmisora y, una vez recibido, ejecuta el algoritmo de descifrado propuesto sobre el mismo para obtener y almacenar la señal recuperada en el dispositivo de memoria flash para su posterior uso.

5.2.1. Descripción del hardware utilizado

Para la implementación del algoritmo de cifrado caótico en los bloques transmisor y receptor, en este trabajo se ha optado por hacer uso de un SoC (System on Chip, por sus siglas en inglés) ESP32 en la forma de una placa de desarrollo ESP32-DevKit (ver figura 5.3) del fabricante *Espressif System*. Los dispositivos ESP32 conforman una serie de bajo costo que funciona como una solución a microcontroladores que no cuenten con conectividad inalámbrica integrada, característica fundamental en implementaciones IoT. Por otro lado, puede existir confusión al hablar de un dispositivo ESP32 debido a que es el nombre utilizado para cualquiera de sus tres presentaciones: chip, módulo o plataforma.

El chip contiene lo correspondiente a los componentes principales del dispositivo como lo es el procesador, la memoria ROM, memoria RAM y módulos de conectividad inalámbrica, mientras que el módulo contiene al chip además de un cristal externo, memoria flash y puede incluir una antena. Por su parte, las placas de desarrollo (o plataformas), como su nombre lo indica, son placas PCB diseñadas que integran al módulo en conjunto con hardware dedicado a la comunicación serie para su programación, re-

Figura 5.3: Placa de desarrollo ESP32-DevKit.

guladores de voltaje para su alimentación y algunos botones como el de *reset*. Por otro lado, la programación de las plataformas puede ser en lenguaje C y, gracias al trabajo de su amplia comunidad, es posible llevar a cabo la programación en el entorno Arduino IDE debido a la gran variedad de librerías que se han desarrollado en los últimos años para dicho propósito.

La plataforma ESP32-DevKit seleccionada incluye el módulo ESP-WROOM-32 el cual se compone de un microcontrolador *Xtensa* de 32 bits con una velocidad de hasta 40 MHz que integra conectividad Wi-Fi y Bluetooth v4.2 y BLE (Bluetooth Low Energy), además de contar con una memoria Flash de 4 MB, 520 KB de memoria SRAM y comunicación SPI, características que lo hacen una buena opción para la implementación de los bloques transmisor y receptor puesto que no solo cuenta con la memoria necesaria para almacenar el código del algoritmo de cifrado, sino que también cuenta con los puertos de entrada/salida suficientes para permitir su conexión con otros dispositivos auxiliares.

Como se describió en la sección anterior, los módulos transmisor y receptor requieren del mismo hardware como estructura básica para su funcionamiento: un sistema embebido que realice los procesos de cifrado o descifrado junto con el manejo de los datos (ESP32-DevKit), una memoria flash que se comunique con el sistema embebido para almacenar las señales clara y recuperada, y una interfaz gráfica para comunicarse con el usuario.

Para la lectura de la memoria flash se ha hecho uso de un módulo genérico para lectura de tarjetas de memoria micro SD (ver figura 5.4(a)) el cual está diseñado para comunicarse a través de una interfaz SPI con un voltaje de alimentación de entre 3.3V y 5V, por lo que es ideal para aplicaciones en sistema embebido que requieran el acceso o almacenamiento de datos. En cuanto a la interfaz con el usuario se refiere, se ha utilizado una pantalla LCD de 2 filas con 16 columnas (16x2) para desplegar los mensajes correspondientes en cada etapa del proceso, ya sea en el módulo de transmisión o de recepción (ver figura 5.4(b)).

Los componentes descritos anteriormente se han interconectado con el módulo ESP32-DevKit para formar cada uno de los bloques del sistema. En la figura 5.5 se

(a) (b)

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

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.

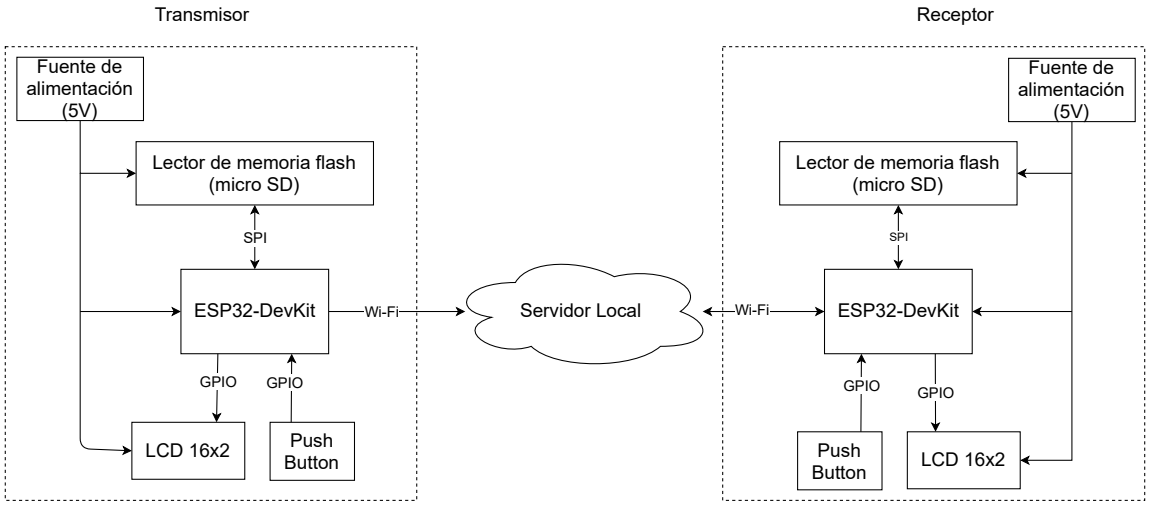


Figura 5.5: Diagrama a bloques del sistema.

En la implementación física los módulos transmisor y receptor se ensamblaron como en la figura 5.6, donde se observa que la pantalla LCD se encuentra conectada a los pines de propósito general de la plataforma ESP32-DevKit al igual que el módulo de lectura de memoria micro SD, con la diferencia de que los pines utilizados para este último son los correspondientes a la comunicación SPI del SoC.

Por otro lado, se ha seleccionado un ordenador Raspberry Pi Zero W (ver figura 5.7) para la implementación del servidor local tipo LAMP que es utilizado para la transmisión de los mensajes encriptados a través de Internet. Es importante mencionar que Raspberry Pi no es tan solo un microprocesador o un microcontrolador sino que es lo

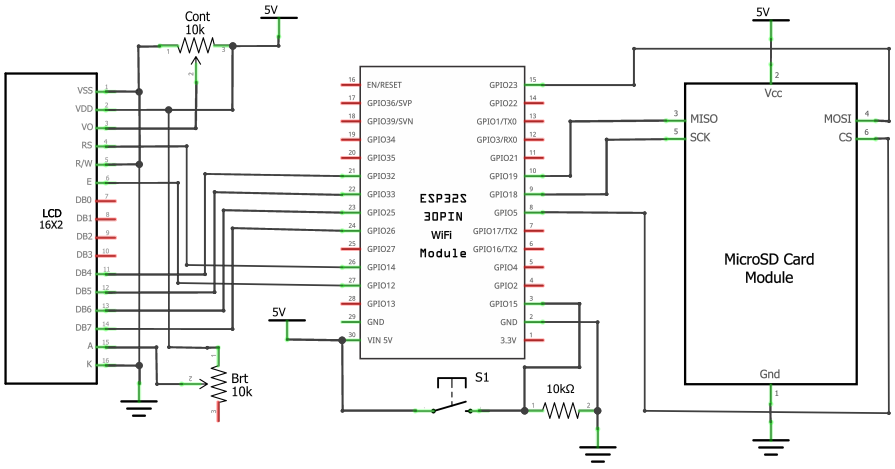


Figura 5.6: Diagrama de conexión eléctrica utilizado para los bloques transmisor y receptor.

que se conoce como un *ordenador de placa reducida*, es decir, ya cuenta con todas las características básicas de una computadora de escritorio como lo es un procesador (o CPU) de tipo ARM con GPU integrada, memoria SRAM que va desde los 256 MB hasta los 8 GB (varía según el modelo), comunicaciones inalámbricas, lector de tarjetas SD o micro SD así como otros periféricos de entrada y salida como salida HDMI, puertos USB, puerto Ethernet y, por su puesto, terminales digitales de propósito general (GPIO).

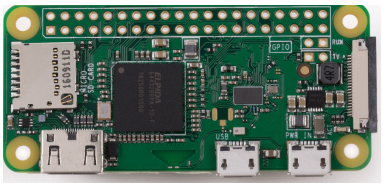


Figura 5.7: Raspberry Pi Zero W.

Al ser técnicamente una computadora pequeña, los Raspberry Pi operan a través de un sistema operativo con una base GNU/Linux cuya distribución oficial es el sistema operativo Raspbian, un sistema operativo que ha sido optimizado para ser utilizado por el hardware de los Raspberry Pi y que, si bien integra algunas herramientas para ser utilizadas dentro del mismo sistema (como lo son entornos de programación y de navegación por Internet), la realidad es que obliga al usuario a tener por lo menos conocimientos básicos del sistema operativo Linux.

En lo que a la versión Raspberry Pi Zero W se refiere, se ha seleccionado esta plataforma debido a que cuenta con los requerimientos mínimos para cumplir con su función, la cual es operar como un *servidor local*. Por un lado, presenta una memoria SDRAM de 512 MB para ejecutar el servidor, mientras que por otro lado cuenta con la

ranura para memoria micro SD para almacenar los archivos que le sean enviados desde el bloque transmisor. Además, cuenta con protocolo de comunicación Wi-Fi 802.11n que es necesario para la conexión a Internet para recibir y enviar archivos según sean solicitados desde el bloque transmisor o receptor.

Para su programación se ha utilizado el lenguaje de programación PHP5, un lenguaje de programación utilizado para el desarrollo web como intermediario entre el servidor y los dispositivos conectados al mismo que permite el manejo de archivos de forma local mientras que se atienden las peticiones de los mismos dispositivos conectados, haciendo posible tanto la recepción del criptograma enviado por el módulo transmisor en un solo archivo como la espera de la petición del módulo receptor para que le sea enviado el mismo.

5.3. Resultados experimentales

Para la evaluación de resultados experimentales se ensambló la electrónica correspondiente para cada uno de los bloques del sistema, cargando los microcontroladores con su respectivo programa en lenguaje C a través del software Arduino IDE. En la figura 5.8 se puede observar el entorno de programación del software Arduino IDE con una parte del código perteneciente al módulo transmisor. En el Apéndice A se presenta el código utilizado para el módulo transmisor, mientras que en el Apéndice B se presenta el código utilizado para el módulo receptor.

Figura 5.8: Interfaz de programación en software Arduino IDE.

Por otro lado, para el servidor local se realizó la programación correspondiente en lenguaje PHP 5 para el manejo de los archivos (los programas son presentados en el Apéndice C), ejecutando el servidor con Apache2 desde el Raspberry Pi (ver figura 5.9). En la figura 5.10 se puede observar cada una de las partes del sistema, encontrándose tanto los bloques transmisor y receptor como el servidor local realizado en el Raspberry Pi Zero W.

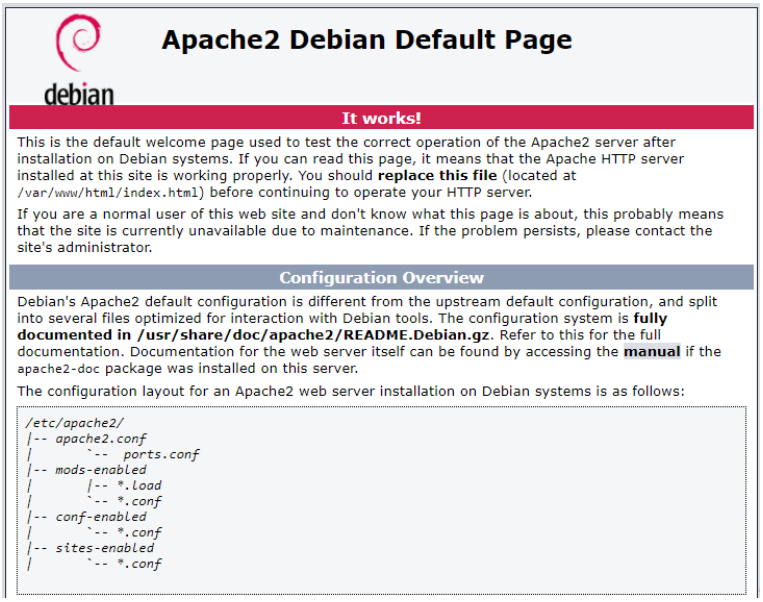


Figura 5.9: Página predeterminada del servidor local ejecutado en Raspberry Pi.

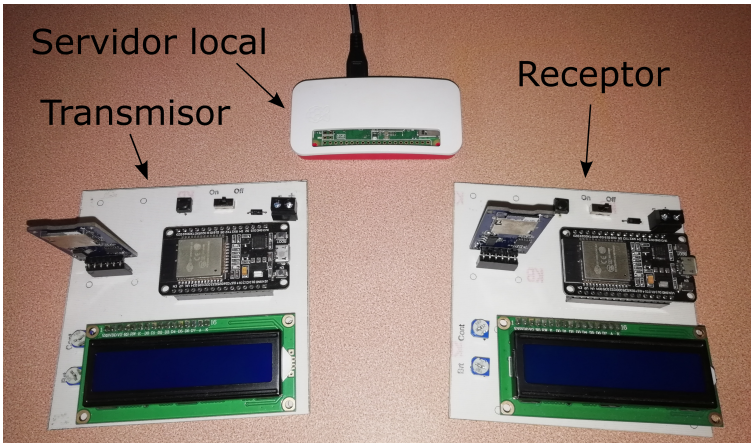


Figura 5.10: Componentes del sistema.

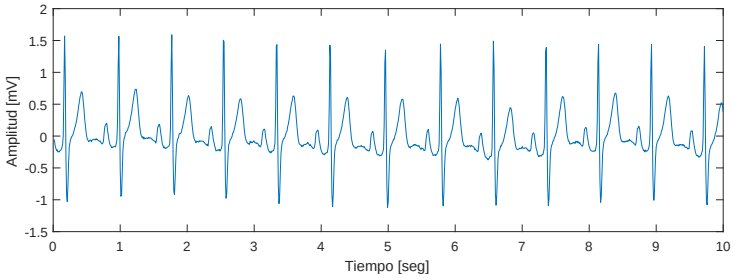
Como señales a transmitir por el sistema para las pruebas experimentales se han utilizado dos señales biomédicas: una señal de electrocardiograma (ECG) y una señal de presión sanguínea (BP), las cuales deberán ser transmitidas de un punto a otro de forma remota y segura.

5.3.1. Transmisión de señal de ECG y BP

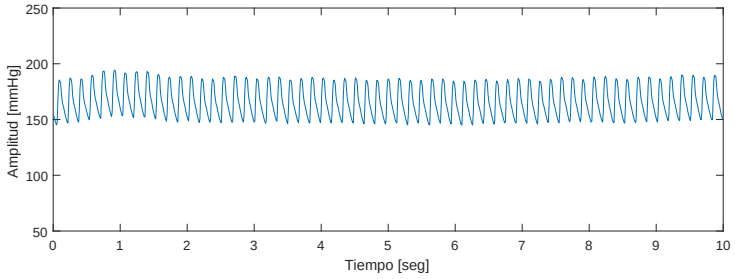
Las señales clínicas a ser transmitidas se obtuvieron de la base de datos PhysioBank ATM, ambas con una duración de 10 segundos. Las señales utilizadas fueron las siguientes:

1. Electrocardiograma (ECG): Adquirida de la base de datos "Apnea-ECG Database (apnea-ecg)", grabación "a17", señal "ECG" con duración de 10 segundos a una frecuencia de muestreo de $F_s = 100$ Hz (ver figura 5.9a).

2. Presión sanguínea (BP): Adquirida de la base de datos "Blood Pressure in Salt-Sensitive Dahl Rats (bpssrat)", grabación "sshs03", señal "BP" con duración de 10 segundos a una frecuencia de muestreo de $F_s = 100$ Hz (ver figura 5.9b).



(a)

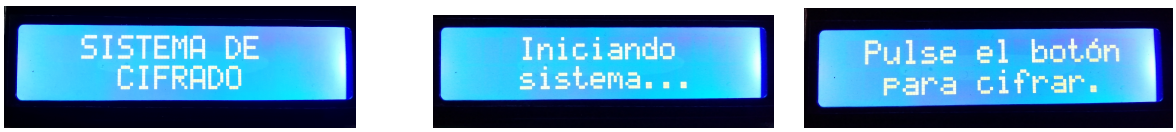


(b)

Figura 5.11: Señales a transmitir: a)Señal ECG, b)Señal BP.

Para la transmisión de las señales primeramente el módulo transmisor muestra a través de la pantalla LCD unos mensajes de inicialización durante el arranque del microcontrolador y que muestran cierta información al usuario (ver figura 5.10).

Una vez inicializado el sistema, las señales clínicas son almacenadas en una memoria micro-SD para después ser leídas, cifradas y enviadas por el módulo ESP32 al servidor local en el Raspberry a través de la conexión a Internet, como se mencionó anteriormente, utilizando dos claves secretas diferentes para cada señal: para la transmisión de la señal ECG se utiliza en el proceso de cifrado la clave secreta **112233445566778899OOAABBCCDDEEFF**, mientras que para la señal BP se utiliza la clave secreta **112233445566778899OOAABBCCDDEEF5**. Por otro lado, el



(a) (b) (c)

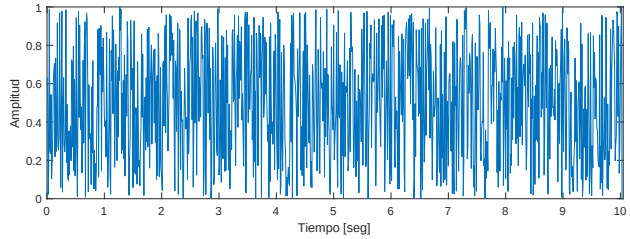
Figura 5.12: Mensajes de inicio del módulo transmisor.

sistema despliega al usuario la etapa en la que se encuentra a través de la pantalla LCD durante el proceso completo (ver figura 5.11). Los criptogramas generados a partir de la señal ECG y la señal BP pueden ser observados en las figuras 5.12(a) y 5.12(b) respectivamente, criptogramas que fueron extraídos del servidor mediante la memoria micro-SD de la Raspberry por lo que sería lo que un intruso lograría recuperar.

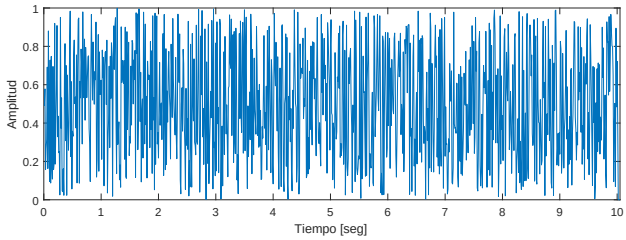
(a) (b) (c)

(d)

Figura 5.13: Mensajes desplegados durante la ejecución del módulo transmisor.



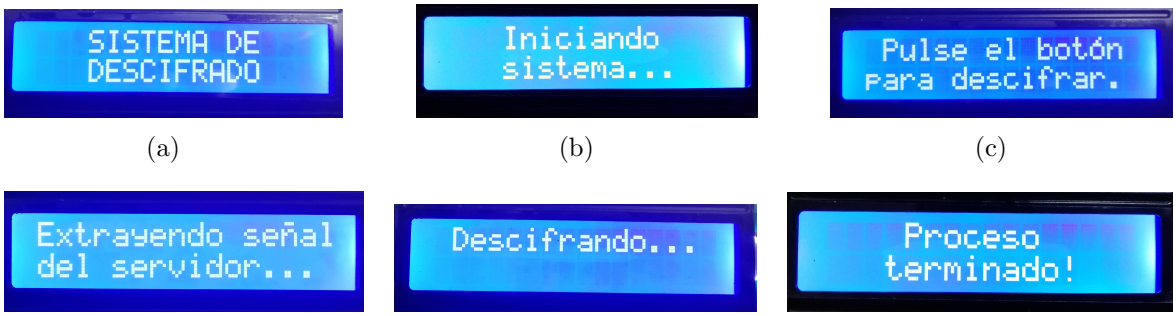
(a)



(b)

Figura 5.14: Criptogramas generados a partir de: a)Señal ECG, b)Señal BP.

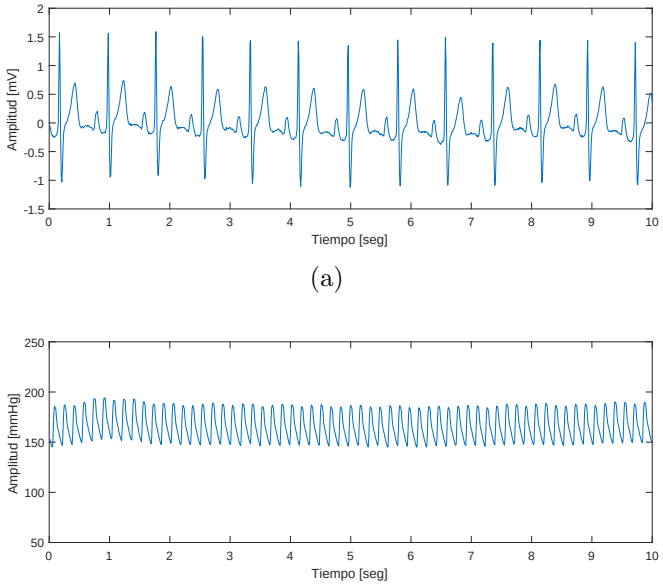
Una vez almacenada la señal cifrada en el servidor local se utiliza el bloque receptor para extraer los criptogramas y se procesan para obtener las señales recuperadas. El módulo receptor ESP32 muestra en un inicio los mismos mensajes de inicialización que el módulo transmisor (ver figuras 5.13(a) - 5.13(c)) para posteriormente iniciar con el proceso de recuperación mostrando a través de la pantalla LCD cada una de las etapas del proceso (ver figuras 5.13(d) - 5.13(f)). En las figuras 5.14(a) y 5.14(b) se muestran las señales recuperadas mediante memoria micro-SD y graficadas en Matlab; se puede observar que ambas señales recuperadas corresponden a las bioseñales originales desde un análisis visual.



(a) (b) (c)

(d) (e) (f)

Figura 5.15: Mensajes desplegados por el módulo receptor.



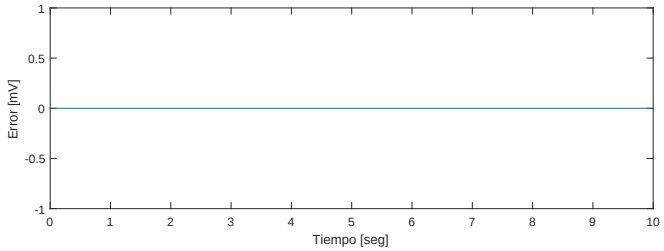
(a)

(b)

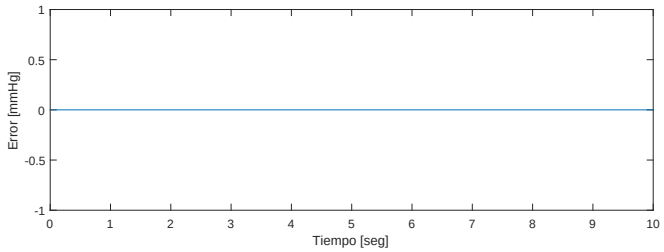
Figura 5.16: Señales recuperadas por el módulo receptor de: a)Señal ECG, b)Señal BP.

Finalmente, para comprobar que las señales claras han sido transmitidas de forma exitosa se calcula el error entre las señales recuperadas y las señales claras utilizando

el software Matlab y, como se aprecia en las figuras 5.15(a) y 5.15(b), dicho error es nulo tanto para la señal ECG como para la señal BP por lo que se puede decir que el cifrado, transmisión, recepción y descifrado de las señales médicas se realizó de forma correcta.



(a)



(b)

Figura 5.17: Error entre la señal clara y la señal recuperada por el bloque receptor: a)Señal ECG, b)Señal BP.

5.4. Análisis de seguridad

Si bien en la sección anterior se comprobó que las señales biomédicas son correctamente cifradas, es importante realizar distintos análisis de seguridad que confirmen que el paso de los criptogramas por los canales de comunicación es seguro, es decir, que a pesar de que los criptogramas sean extraídos por una persona no autorizada esta no será capaz de obtener las señales claras utilizando algún método criptoanalítico conocido.

Para ello se han realizado seis pruebas a diferentes criptogramas generados por el sistema embebido de cifrado caótico y señales recuperadas en el receptor, extrayendo ambos con memoria micro-SD para ser analizadas con ayuda de Matlab. Estos análisis involucran tanto análisis de sensibilidad como análisis estadísticos para cubrir un amplio margen de ataques criptoanalíticos con los que se podrían recuperar las señales biomédicas.

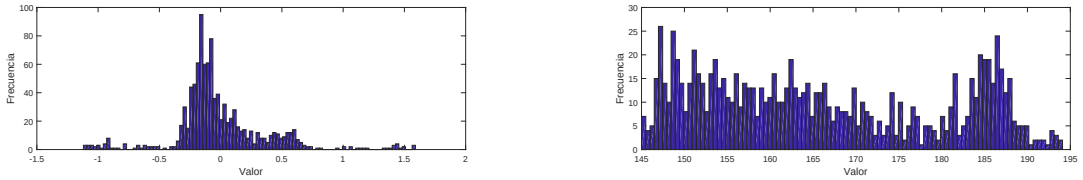
5.4.1. Espacio de claves

Con la finalidad de soportar un ataque exhaustivo, un algoritmo de cifrado seguro debe tener un amplio espacio de claves para que de esa forma sean muchas las claves a intentar antes de encontrar aquella utilizada para cifrar la señal. Numéricamente hablando, se dice que para que un sistema criptográfico pueda resistir un ataque exhaustivo este debe tener un espacio de claves mayor a 2^{100} [63].

Como se mencionó en la sección 4.2, la clave secreta propuesta está definida por 128 bits representados en 32 caracteres hexadecimales, implicando que el espacio de claves del algoritmo de cifrado propuesto es de 2^{128}, por lo que el mismo puede resistir un ataque de fuerza bruta.

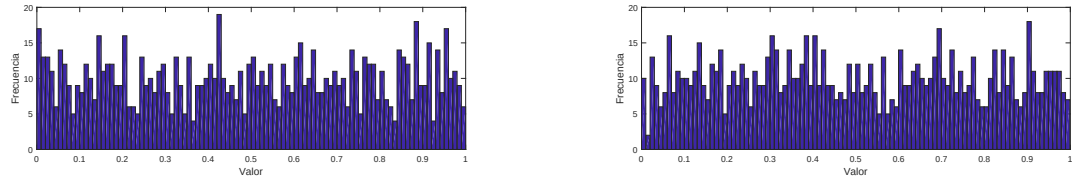
5.4.2. Histogramas

El análisis de histogramas tanto de la señal clara como del criptograma permite apreciar la robustez del algoritmo de cifrado ante ataques estadísticos, resaltando que el criptograma debe presentar uniformidad en sus datos para ello. En la figura 5.16 se puede observar la frecuencia de los datos de las señales claras de ECG y BP respectivamente, donde se puede ver que cada una de ellas tiene una distribución de datos no uniforme, mientras que sus respectivos criptogramas en la figura 5.17 se observan más uniformes, de forma que el algoritmo criptográfico puede resistir un ataque de histograma.



(a) Señal ECG. (b) Señal BP.

Figura 5.18: Histogramas correspondientes a las señales claras transmitidas.



(a) Señal ECG. (b) Señal BP.

Figura 5.19: Histogramas correspondientes a los criptogramas generados a partir de las señales claras.

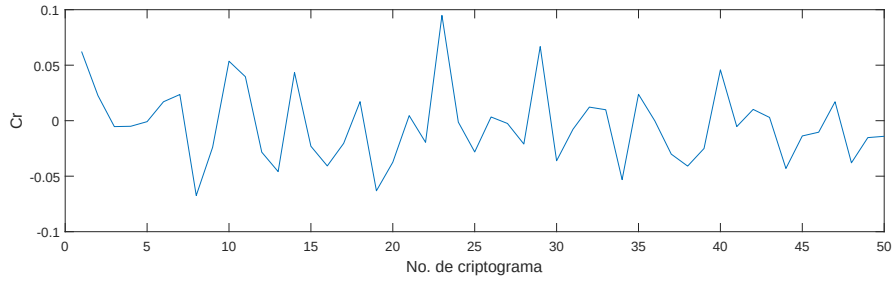
5.4.3. Correlación

La correlación indica la relación lineal entre el criptograma y la señal clara, es decir, determina que tan diferentes son uno del otro. La importancia de este análisis radica en el hecho de que si la señal clara y el criptograma llegan a ser similares un criptoanalista podría recuperar tanto la señal clara como la clave secreta mediante un ataque estadístico. La correlación entre dos secuencias x e y se puede calcular como

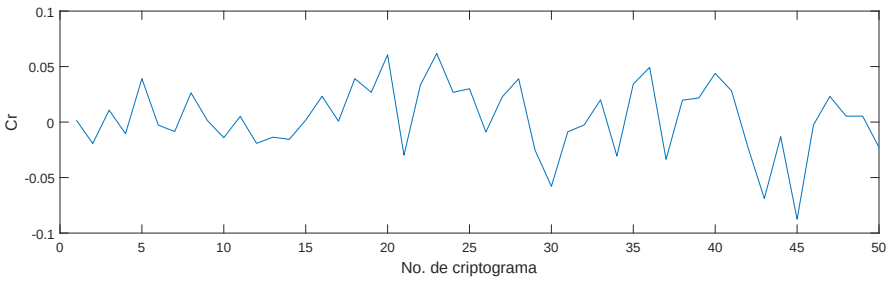
$$Cr = \frac{N \times \sum_{i=0}^{N}(x_i \times y_i) - \sum_{i=0}^{N} x_i \times \sum_{i=0}^{N} y_i}{\sqrt{\left(N \times \sum_{i=0}^{N}(x_i)^2 - \left(\sum_{i=0}^{n} x_i\right)^2\right) \times \left(N \times \sum_{i=0}^{N}(y_i)^2 - \left(\sum_{i=0}^{n} y_i\right)^2\right)}} \tag{5.1}$$

siendo $Cr \in (-1, 1)$ el coeficiente de correlación donde $Cr = 0$ significa correlación nula.

Para el análisis realizado se generaron 50 criptogramas a partir de la señal ECG y 50 criptogramas a partir de la señal de BP, cada uno de ellos con una clave distinta para posteriormente calcular la correlación de cada uno con su respectiva señal clara (ver figura 5.18). Como resultado se tiene que el coeficiente de correlación promedio para los criptogramas de ECG fue de **-0.003929709061263**, mientras que el coeficiente de correlación promedio para los criptogramas de BP fue de **0.003650443111846**, presentando ambos aproximadamente correlación nula por lo que se puede decir que en promedio los criptogramas generados son completamente diferentes a la señal clara.



(a)



(b)

Figura 5.20: Coeficiente de correlación para 50 criptogramas distintos: a)Criptogramas generados a partir de la señal ECG, b)Criptogramas generados a partir de la señal BP.

5.4.4. Sensibilidad a la clave secreta

Un buen sistema criptográfico debe presentar una alta sensibilidad a la clave secreta de forma que el cambio de un solo bit de la misma genere un criptograma completamente diferente a partir de la misma señal médica. Para la prueba se generan dos criptogramas distintos de ECG y dos criptogramas de BP utilizando una CLAVE 1 y una CLAVE 2 (ver tabla 5.1) las cuales presentan una diferencia de tan solo un bit y posteriormente se lleva a cabo el cálculo del coeficiente de correlación entre los criptogramas generados tal como se realizó en la sección anterior.

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

No. clave	Clave secreta
CLAVE 1	11223344556677889900AABBCCDDEEFF
CLAVE 2	1122334455667**8**889900AABBCCDDEEFF

Los valores de los coeficientes de correlación entre los criptogramas de ECG y BP (Cr_{ECG} y Cr_{BP} respectivamente) generados con la CLAVE 1 y la CLAVE 2 se muestran en la tabla 5.2 y, tal como se puede observar, al ser ambos valores muy cercanos a cero se puede decir que la correlación entre los criptogramas en nula y que por lo tanto 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}	-0.005863504178797
Cr_{BP}	0.000900800074023

5.4.5. Sensibilidad a la señal clara

La sensibilidad a la señal clara es importante en los algoritmos criptográficos para garantizar que pequeños cambios en la señal clara generen criptogramas completamente diferentes a manera de aumentar la robustez del mismo ante ataques de tipo diferencial. Para llevar a cabo este análisis se hace uso de dos métodos de medición como lo son el NPCR (Net Pixel Change Rate), que mide la cantidad de muestras que son diferentes al comparar dos señales, y UACI (Unified Average Changing Intensity) que representa la diferencia en magnitud entre las muestras de las señales comparadas. El primero de ellos se determina con la expresión

$$NPCR = \frac{\sum_{i=1}^{\ell} W(i)}{\ell} \times 100 \tag{5.2}$$

con

$$W(i) = \begin{cases} 0 & \text{if } E_1(i) = E_2(i) \\ 1 & \text{if } E_1(i) \neq E_2(i) \end{cases} \tag{5.3}$$

y el valor de UACI se determina con

$$UACI = \frac{100}{\ell} \sum_{i=1}^{\ell} |E_1(i) - E_2(i)| \tag{5.4}$$

donde ℓ es la longitud del texto, mientras que E_1 y E_2 son dos criptogramas diferentes.

Para generar el criptograma E_1 se hace uso de la CLAVE 1 (mostrada en la tabla 5.1) para cifrar las señales claras de ECG y BP, mientras que para el criptograma E_2 se hace una ligera modificación a las señales claras: a la señal de ECG se le modifica el valor del dato número 600 de **0.425** a **0.426**, mientras que para la señal de BP se modifica el valor del dato número 300 de **147.64** a **147.65**. Finalmente se repite el proceso de cifrado con las señales claras modificadas para generar los criptogramas E_2.

La tabla 5.3 muestra los resultados obtenidos de NPCR y UACI tanto para la señal de ECG como para la señal de BP, donde se observa que entre los criptogramas generados el 100 % de los datos generados son completamente diferentes con una diferencia en magnitud promedio del 33.11 %, por lo que el algoritmo de cifrado propuesto es robusto ante ataques del tipo diferencial puesto que el valor de NPCR está por encima del 90 % y el valor de UACI por encima del 33 % [64] .

Tabla 5.3: Resultados de análisis diferencial NPCR y UACI para cifrado de texto.

Prueba	ECG	BP
NPCR(%)	100	100
UACI(%)	33.1168	33.1126

5.4.6. Entropía de la información

La *entropía* permite determinar que tan caóticos son los datos generados de una forma estadística, donde un alto valor de la misma significa que los datos generados para los criptogramas son suficientemente caóticos, mientras que un valor bajo implica que los datos generados tienden a ser repetitivos por lo que los criptogramas pueden volverse predecibles.

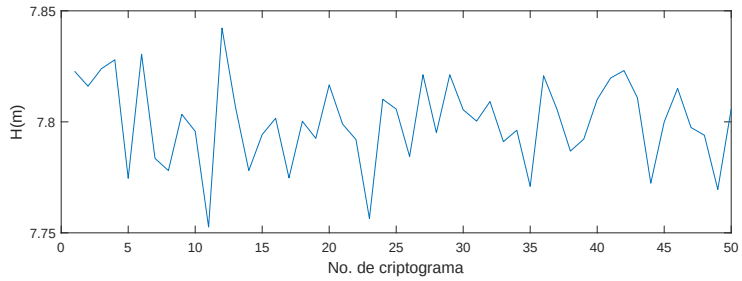
La entropía se determina mediante la expresión

$$H(m) = \sum_{i=0}^{2^N-1} p(m_i)\ log_2(1/p(m_i)), \tag{5.5}$$

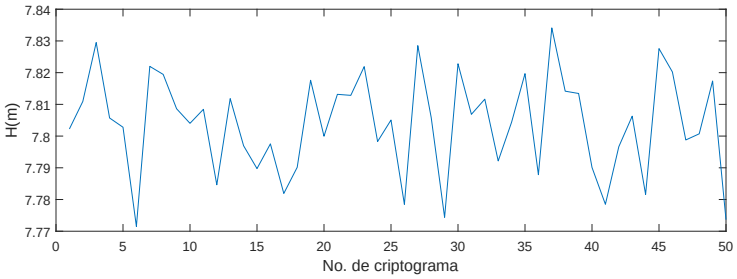
donde N es el número de bits que representan la unidad básica del mensaje m, 2^N son todas las combinaciones de la unidad básica, $p(m_i)$ representa una probabilidad de m_i, $\log_2$ es el logaritmo base 2 y la entropía esta expresada en bits, donde la máxima entropía es N. Si un mensaje m es cifrado con 2^N posibles valores, la entropía debería

ser idealmente $H(m) = N$, si m es puramente aleatorio.

Como los datos a analizar son los correspondientes a los criptogramas generados, se toma en cuenta que los datos generados en los mismos se encuentran en el intervalo $[0, 1]$ por lo que se realiza una transformación a datos de 8 bits, es decir, datos en el intervalo $[0, 255]$. De esta forma la máxima entropía posible es de 8. Para el análisis se determinó la entropía de 50 criptogramas obtenidos de 50 claves secretas distintas para las señales de ECG y BP (ver figura 5.19) donde se obtuvo una entropía promedio de **7.79996** para los criptogramas generados a partir de la señal de ECG y una entropía promedio de **7.80384** para los criptogramas generados a partir de la señal de BP, y debido a que en ambos casos el resultado es un valor cercano a 8 (el valor de entropía ideal) se puede decir que los criptogramas generados pueden resistir un ataque de entropía pues sus datos presentan un nivel alto de desorden.



(a) Señal ECG.



(b) Señal BP.

Figura 5.21: Entropía para 50 criptogramas distintos.

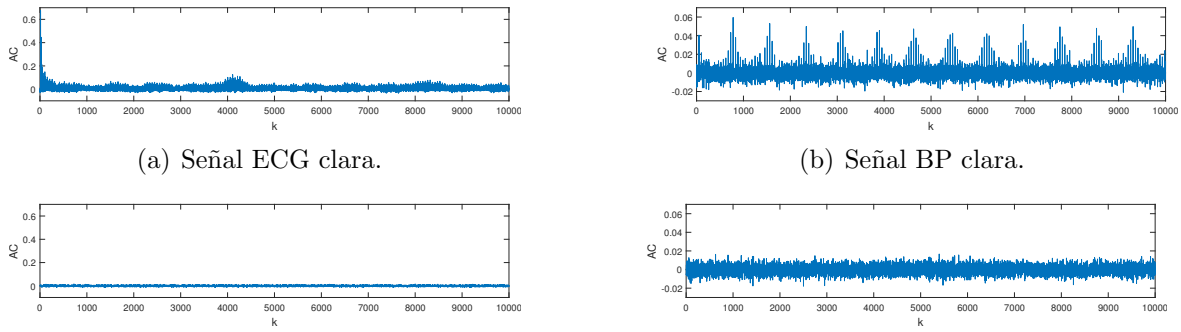
5.4.7. Autocorrelación

La autocorrelación es un cálculo realizado en procesamiento de señales que consiste en obtener la correlación de una señal consigo misma siendo desplazada k posiciones, para determinar si existe algún tipo de patrón o inclusive periodicidad a lo largo de la señal. Para calcularla se utiliza la expresión

$$AC(k) = \frac{A - D}{T}, \tag{5.6}$$

donde $AC \in [-1, 1]$ es la autocorrelación de la señal desplazada k posiciones, A es el número de elementos que concuerdan entre el la señal original y la señal desplazada, D es el número de elementos que no concuerdan y T es la longitud de la señal. En cuanto al valor del cálculo de la autocorrelación se tiene que valores altos suponen la existencia de muchos datos idénticos a nivel de bit y valores negativos significa que existen muchos valores opuestos, mientras que cuando existe autocorrelación nula ($AC = 0$) significa que existe un equilibrio entre la cantidad de 1's y 0's en la señal, caso deseado a la hora de analizar criptogramas.

Para el cálculo de autocorrelación tanto de las señales claras como de los criptogramas en primer lugar se realiza una transformación de los datos con resolución de 10^{15} (variables de tipo *double*) a datos de 64 bits cada uno para después proceder a determinar la autocorrelación de cada señal. En la figura 5.20 se puede observar la autocorrelación de las señales claras y de los criptogramas para las señales de ECG y BP con un corrimiento circular a la derecha de hasta $k = 10000$, donde se muestra que mientras que las señales claras presentan algunos patrones con valores tanto positivos como negativos los criptogramas por su parte presentan una autocorrelación muy cercana a 0, por lo que se puede decir que el algoritmo de cifrado genera números pseudoaleatorios de forma uniforme.



(a) Señal ECG clara.

(b) Señal BP clara.

(c) Señal ECG cifrada.

(d) Señal BP cifrada.

Figura 5.22: Autocorrelación de señales utilizadas.

5.4.8. Tiempo de cifrado y transmisión

La eficiencia es un aspecto importante a considerar tanto en los algoritmos de cifrado como en los sistemas de telemedicina pues de ello depende su posible aplicación en sistemas de tiempo real. En la tabla 5.4 se muestran los tiempos requeridos para los procesos de cifrado, transmisión y descifrado para las bioseñales de 10 segundos de duración con una frecuencia de muestreo de 100 Hz, observándose que los tiempos de cifrado y descifrado son prácticamente idénticos, mientras que la transmisión del criptograma al servidor se realiza en un tiempo muy corto.

Tabla 5.4: Tiempo requerido en los distintos procesos del sistema.

Proceso	Tiempo [seg]
Cifrado	0.168
Transmisión	0.188
Descifrado	0.164

5.5. Conclusiones del capítulo

Se presentó la descripción del hardware utilizado en la implementación del algoritmo de cifrado propuesto en la forma de un sistema de telemedicina basado en sistema embebido de 32 bits para la transmisión de señales biomédicas de forma inalámbrica a través de una conexión a Internet, así como su construcción física. De igual forma se expusieron dos pruebas de transmisión con una señal de ECG y una señal de BP que resultaron exitosas sin pérdida de información, comprobando además que las señales almacenadas en el servidor local se encontraban efectivamente encriptadas, presentando además un buen tiempo de ejecución en cada uno de los procesos.

El sistema diseñado demostró no solo ser un sistema funcional de bajo costo, sino que además es de fácil uso pues permite monitorear la etapa del proceso de cifrado o descifrado en la que se encuentra (según sea el bloque transmisor o receptor) por medio de la pantalla LCD, aspecto que es importante considerar cuando el sistema lo va a utilizar el público general.

Además se presentaron un conjunto de análisis de seguridad aplicados a distintos criptogramas generados con el sistema a partir tanto de la señal de ECG como de la señal de BP, los cuales demostraron que el algoritmo de cifrado propuesto es robusto ante distintos ataques criptoanalíticos como lo son los diferenciales y los estadísticos, por lo cual puede ser utilizado para la transmisión de forma segura de distintas señales biomédicas dentro de un sistema de telemedicina.

Capítulo 6

Conclusiones

6.1. Conclusiones generales

En este trabajo de tesis se desarrolló un sistema de telemedicina en sistema embebido de 32 bits en el cual se implementó un algoritmo de cifrado basado en caos para la transmisión de señales biomédicas a través de Internet con una comunicación Wi-Fi, es decir, de manera inalámbrica, proporcionando seguridad tanto en el canal de comunicación durante la transmisión de las señales como en el servidor durante su almacenamiento, implementado en un Raspberry Pi Zero W.

Para la selección de los mapas caóticos a utilizar en el algoritmo de cifrado, se estudiaron seis mapas bidimensionales presentados en el estado del arte: SLIM, SCL, LSCM, LICM, Seno-Hénon y Logístico acoplado, mapas a los que se les realizaron pruebas de eficiencia así como de dinámicas caóticas mediante el cálculo del exponente de Lyapunov para finalmente elegir los mapas SLIM y Seno-Hénon por su desempeño en las pruebas. Por otro lado, el algoritmo de cifrado caótico implementado consiste en un algoritmo de clave simétrica de 128 bits por medio de la cual se determina el valor de las condiciones iniciales y de los parámetros de cada uno de los mapas caóticos para posteriormente generar las secuencias numéricas utilizadas en los procesos de confusión y difusión para cifrar las señales claras y transmitirlas de forma segura.

Para corroborar el funcionamiento del sistema se realizó la trasmisión de una señal de ECG y una señal de BP, las cuales se ingresaron al sistema a través de un dispositivo de memoria (micro SD) y fueron extraídas de la misma forma en el receptor de forma exitosa, demostrando el correcto funcionamiento del sistema. Por otro lado, los resultados de los análisis estadísticos y de sensibilidad demostraron que el algoritmo de cifrado propuesto proporciona a los criptogramas un buen nivel de robustez ante distintos ataques criptoanalíticos, garantizando la seguridad de las señales en el proceso de transmisión al mismo tiempo que se mantiene la eficiencia del sistema al requerir poco tiempo de ejecución en cada proceso, convirtiendo al sistema en una opción viable para ser implementado en redes o sistemas que requieran de la transmisión de señales biomédicas, como lo es el caso de las redes de hospitales o dispositivos utilizados en el área médica como relojes o distintos instrumentos para telemonitoreo.

6.2. Principales contribuciones del trabajo de tesis

- Estudio y selección de dos mapas caóticos para implementarse en sistema embebido.
- Diseño y construcción de sistema embebido de bajo costo para la transmisión de criptogramas por WiFi.
- Algoritmo de cifrado caótico para cifrar señales biomédicas con aplicación en telemedicina.

6.3. Trabajo a futuro

Como trabajo a futuro para el sistema desarrollado se presenta lo siguiente:

- Implementar el sistema para la transmisión de otros archivos médicos que requieren ser transmitidos por medios como Internet de forma segura, como lo son las fichas médicas de los pacientes o imágenes tanto a blanco y negro como a color.
- Implementar en el sistema, la electrónica necesaria para la lectura de señales biomédicas en tiempo real como electrocardiogramas, electromiogramas, electroencefalogramas, señales de presión sanguínea, entre otras.
- Modificar el sistema de forma que se habilite la función multi-usuario para crear un sistema más completo de telemedicina en el que múltiples usuarios puedan transmitir señales biomédicas de forma remota.
- Realizar otros análisis de seguridad como lo es la prueba del Instituto Nacional de Estándares y Tecnología, NIST 800-22, para cuantificar de una mejor manera el nivel de pseudoaleatoriedad de los criptogramas generados por el algoritmo de cifrado propuesto.
- Implementar en el sistema, la lectura de múltiples señales biomédicas en el transmisor mediante comunicación por Bluetooth para el cifrado de múltiples bioseñales.

Bibliografía

[1] Almazyad, I., Rao, A., & Rozenblit, J. (2020). A framework for secure data management for medical devices. *2020 Spring Simulation Conference*, **2020**: 1-12.

[2] Sinsky, C. A., Jerzak, J. T., & Hopkins, K. D. (2021). Telemedicine and Team-Based Care: The Perils and the Promise. *Mayo Clinic Proceedings*, **96**(2): 429-437.

[3] Ryu, W. H. A., Kerolus, M. G., & Traynelis, V. C. (2021). Clinicians' user experience of telemedicine in neurosurgery during COVID-19. *World neurosurgery*, **146**: 359-367.

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

[5] Zhang, W., Zhu, Z., & Yu, H. (2019). A symmetric image encryption algorithm based on a coupled logistic–bernoulli map and cellular automata diffusion strategy. *Entropy*, **21**(5): 504-526.

[6] García-Guerrero, E. E., Inzunza-González, E., López-Bonilla, O. R., Cárdenas-Valdez, J. R., & Tlelo-Cuautle, E. (2020). Randomness improvement of chaotic maps for image encryption in a wireless communication scheme using PIC-microcontroller via Zigbee channels. *Chaos, Solitons y Fractals*, **133**: 109646-109657.

[7] Wikina, S. B. (2014). What caused the breach? An examination of use of information technology and health data breaches. *Perspectives in health information management*, **11**: 1-16.

[8] Seh, A. H., Zarour, M., Alenezi, M., Sarkar, A. K., Agrawal, A., Kumar, R., & Khan, R. A. (2020). Healthcare data breaches: Insights and implications. *Healthcare*, **8**(2): 133-150.

[9] Ping, W., Jin-Gang, W., Xiao-Bo, S., & Wei, H. (2006). The research of telemedicine system based on embedded computer. *2005 IEEE Engineering in Medicine and Biology 27th Annual Conference*, 2005, 114-117.

[10] Hathaliya, J. J., & Tanwar, S. (2020). An exhaustive survey on security and privacy issues in Healthcare 4.0. *Computer Communications*, **153**: 311-335.

[11] Castillejo, J. A. P. (2013). Telemedicina, una herramienta también para el médico de familia. *Atención primaria*, **45**(3): 129-132.

[12] Herrera, F. P., & Periche, F. F. (2017). 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*, **1**(1): 4-7.

[13] Alvez, R. (2011). Aplicación de telemedicina para la mejora de los sistemas de emergencias y diagnósticos clínicos. *Memoria Investigaciones en Ingeniería*, **9**: 91-97.

[14] Novillo-Ortiz, D. (2016). Marco de Implementación de un Servicio de Telemedicina. *Organización Panamericana de Salud, Washington D. C., Estados Unidos*, 2016, pp. 12-15

[15] Ramos, V. (2010). Contributions to the history of Telemedicine of the TICs. *IEEE Conference on the History of Communications*, 2010, pp. 1-5.

[16] Zundel, K. M. (1996). Telemedicine: history, applications, and impact on librarianship. *Bulletin of the Medical Library Association*, **84**(1): 71-79.

[17] Bashshur, R. L., Reardon, T. G., & Shannon, G. W. (2000). Telemedicine: a new health care delivery system. *Annual review of public health*, **21**(1): 613-637.

[18] Cánovas, L. P. L., Cánovas, L. B. L., & Forcelledo, A. H. (2018). Telemedicina, impacto y perspectivas para la sociedad actual. *Universidad Médica Pinareña*, **14**(3): 289-303.

[19] Roig, F., & Saigí, F. (2009). Dificultades para incorporar la telemedicina en las organizaciones sanitarias: perspectivas analíticas. *Elsevier España*, **23**(2): 147e1-147e4.

[20] Ruiz, C., Zuluaga, A., & Trujillo, A. (2007). TELEMEDICINA: Introducción, aplicación y principios de desarrollo. *CES Medicina*, **21**(1): 77-93.

[21] Peñafiel, J. M., Gil, A., Azorín, J. M., Sabater, J. M., Pérez, C., & Morales, R. (2007). INTERFAZ AVANZADA DE UN SISTEMA DE TELECIRUGÍA. *Universidad Miguel Hernández de Elche*, 2007, pp. 1-7.

[22] Conaty-Buck, S. (2017). Cybersecurity and healthcare records. *American Nurse Today*, **12**(9): 62-64.

[23] Kim, D. W., Choi, J. Y., & Han, K. H. (2020). Risk management-based security evaluation model for telemedicine systems. *BMC Medical Informatics and Decision Making*, **20**(1): 1-14.

[24] Pérez E. (1997). Caos en el sistema solar. *Miscelánea Matemática*, **27**: 59-69.

[25] Oestreicher, C. (2007). A history of chaos theory. *Dialogues in clinical neuroscience*, **9**(3): 279-289.

[26] Davies, B. (2018). Exploring chaos: Theory and experiment. *CRC Press*, pp. 6-11.

[27] Tachiquín, M. (2010). Teoría del Caos: una visión de su historia y actualidad *Revista del Centro de Investigación de la Universidad la Salle*, **9**(34): 41-47.

[28] Schmitt-Grohé, S., & Uribe, M. (2021). Deterministic debt cycles in open economies with flow collateral constraints. *Journal of Economic Theory*, **192**: 105-195.

[29] West, B. J. (2010). Fractal physiology and the fractional calculus: a perspective. *Frontiers in physiology*, **1**:12.

[30] Anzurez, J., Padilla, J. & Cuevas, O. (2008). Estabilidad de Sistemas No-lineales: Sistema de Nivel de Líquidos de Dos Tanques Interconectados *Revista de Ingeniería Eléctrica, Electrónica y Computación*, **5**(2): 6-12.

[31] Dubeibe, F. (2013). Cálculo del máximo exponente de Lyapunov con Mathematica. *Revista Colombiana de Fısica*, **45**(1): 151-155.

[32] J. C. Sprott (2010). Elegant Chaos: Algebraically Simple Chaotic Flows. *World Scientific*, pp. 20-29.

[33] Hrothgar (2015). Lyapunov exponent of the Lorenz system. *Chebfun*, http://www.chebfun.org/examples/ode-nonlin/LyapunovExponents.html.

[34] Xu, Q., Sun, K., Cao, C., & Zhu, C. (2019). A fast image encryption algorithm based on compressive sensing and hyperchaotic map. *Optics and Lasers in Engineering*, **121**: 203-214.

[35] Chen, C., Sun, K., & He, S. (2020). An improved image encryption algorithm with finite computing precision. *Signal Processing*, **168**: 107340-107349.

[36] Hua, Z., Jin, F., Xu, B., & Huang, H. (2018). 2D Logistic-Sine-coupling map for image encryption. *Signal Processing*, **149**: 148-161.

[37] Liang, X., Liang, W., & Xiong, J. (2020). Intelligent diagnosis of natural gas pipeline defects using improved flower pollination algorithm and artificial neural network. *Journal of Cleaner Production*, **264**: 121655-121665.

[38] Cao, C., Sun, K., & Liu, W. (2018). A novel bit-level image encryption algorithm based on 2D-LICM hyperchaotic map. *Signal Processing*, **143**: 122-133.

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

[40] Liao, X., Kulsoom, A., & Ullah, S. (2016). A modified (Dual) fusion technique for image encryption using SHA-256 hash and multiple chaotic maps. *Multimedia Tools and Applications*, **75**(18): 11241-11266.

[41] Fernández, S. (2004). La criptografía clásica. *Sigma: Revista de matemáticas*, **24**: 119-142.

[42] Dodis, Y. (2000). Exposure-resilient cryptography. *Tesis Doctoral, Instituto de tecnología de Massachusetts*, pp. 11-14.

[43] Menezes, A. J., Van Oorschot, P. C., & Vanstone, S.A. (2018). Handbook of applied cryptography. *CRC press*, pp. 4-6.

[44] Hermosilla, J. C. H. (2012). Breve historia del espionaje. *Ediciones Nowtilus SL*, pp. 38-42.

[45] Arnau, M. G. (2003). Criptografía clásica. ¿Cómo romper cifrados monoalfabéticos y polialfabéticos? Análisis de frecuencias y método Kasiski. *Burán*, **19**: 95-97.

[46] Darma Nasution, Surya & Ginting, Guidio & Syahrizal, Muhammad & Rahim, Robbi. (2017). Data Security Using Vigenere Cipher and Goldbach Codes Algorithm. *International Journal of Engineering Research & Technology*, **6**: 360-363.

[47] Baig, M. (2001). Criptografía Cuántica. *Universidad Autónoma de Barcelona, España*, pp. 6-7.

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

[49] Borda M. (2011). Cryptography Basics. *Fundamentals in Information Theory and Coding*, Springer, Berlin, Heidelberg, pp. 121-208.

[50] Padrón, A., Prieto, R., Herrera, A., & Calva, G. (2014). Implementación de Protocolos de Comunicación Seguros: Escenarios. *Congreso de Instrumentación*, **29**: 1-9.

[51] Katz, J., & Lindell, Y. (2020). Introduction to modern cryptography. *CRC press*, 2da Ed., pp. 18-21.

[52] Li, H., Deng, L., & Gu, Z. (2020). A robust image encryption algorithm based on a 32-bit chaotic system. *IEEE Access*, **8**: 30127-30151.

[53] Munir, N., Khan, M., Jamal, S. S., Hazzazi, M. M., & Hussain, I. (2021). Cryptanalysis of hybrid secure image encryption based on Julia set fractals and three-dimensional Lorenz chaotic map. *Mathematics and Computers in Simulation*, **190**: 826-836.

[54] Pushpalatha, G. S., & Ramesh, S. (2021). Chaotic based encryption algorithms for speech signal and cryptographic requirements: A brief survey. *Materials Today: Proceedings*, pp. 1-5.

[55] Wang, J., Han, K., Fan, S., Zhang, Y., Tan, H., Jeon, G., Pang, Y., & Lin, J. (2020). A logistic mapping-based encryption scheme for Wireless Body Area Networks. *Future Generation Computer Systems*, **110**: 57-67.

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

[57] Murillo-Escobar, M. A., Cardoza-Avendaño, L., López-Gutiérrez, R. M., & Cruz-Hernández, C. (2017). A double chaotic layer encryption algorithm for clinical signals in telemedicine. *Journal of medical systems*, **41**(4) : 59-75.

[58] Michel-Macarty, J. A., Murillo-Escobar, M. A., López-Gutiérrez, R. M., Cruz-Hernández, C., & Cardoza-Avendaño, L. (2018). Multiuser communication scheme based on binary phase-shift keying and chaos for telemedicine. *Computer methods and programs in biomedicine*, **162** : 165-175.

[59] Marwedel, P. (2021). Embedded system design: embedded systems foundations of cyber-physical systems, and the internet of things. *Springer Nature*, 4ta Ed., pp. 2-3.

[60] Eggermont, L. (2002). Embedded systems roadmap. *Citeseer*, **30**: 20-23.

[61] Rossano, V. (2009). Electrónica & microcontroladores PIC. *Gradi*, 1ra Ed., pp. 14-19.

[62] Janakiraman, S., Thenmozhi, K., Rayappan, J. B. B., & Amirtharajan, R. (2018). Lightweight chaotic image encryption algorithm for real-time embedded system: Implementation and analysis on 32-bit microcontroller. *Microprocessors and Microsystems*, **56**: 1-12.

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

[64] Li, T., Du, B., & Liang, X. (2020). Image encryption algorithm based on logistic and two-dimensional Lorenz. *IEEE Access*, **8** : 13792-13805.

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 local a través de la conexión Wi-Fi.

```
/*************************************************************
  Código para el cifrado caótico y transmisión de bioseñales
  por internet.

  Elaborado por: Armando Ceseña Villa
*************************************************************/

// Librerías
#include <WiFi.h> // Wi-Fi
#include <SPI.h> // Comunicación módulo SD
#include <SD.h> // SD
#include <LiquidCrystal.h> // LCD

// Factor de división
#define e15 1000000000000000

// Entrada digital para botón
const uint8_t button = 15;

// ***** Configuración módulo de lectura  de memoria micro-SD ***** /////
// Crea un objeto File para manejar el archivo de la SD
File myFile;

// Nombre del archivo con bioseñal a leer en memoria micro-SD
char archivo_Encriptar[] = "/Biosignal";

// Nombre para guardar temporalmente criptograma en micro-SD
char fileName[25] = "/Criptograma_Biosignal.csv";

// Tamaño de archivo
long fileSize;

// ******* Variables para envío de datos a servidor ********** /////
// Constantes para configuración WiFi
const char *ssid     = "*******";
const char *password = "*******";
```

```
37 const char *servername = "***.***.***.***";
38
39 // Variables para transmisión de datos a servidor
40 #define MTU_Size  2*1760
41 byte clientBuf[MTU_Size];
42 int clientCount, count, chunks;
43 char symbol[] = {0xE2, 0x96, 0xA0};
44
45 char nombre_archivo[] = "Criptograma.csv";
46
47 // Crea un objeto "client" para manejar la conexión al servidor
48 WiFiClient client;
49
50 // ***************** Variables para encriptado ***************** /////
51 // Clave secreta
52 const char cadena[]="1122334455667788990AABBCCDDEEFF";
53
54 char Llave[4][8];
55 unsigned long dec_AE, dec_BE, dec_CE, dec_DE, x[4];
56 double val_AE,val_BE,val_CE,val_DE;
57
58 // Señal clara normalizada
59 double norm[1000];
60 int LONG = 0;
61
62 // Valores para calcular parámetros y condiciones iniciales
63 double V_PA2, V_CI2, V_PA2_b, V_CI2_b, mod_PA, mod_CI, mod_PA_b, mod_CI_b;
64
65 // Parámetros y condiciones iniciales de mapas caóticos
66 double a2_L1, b, a_LL1, b1;
67 double y[1100], x2_L1_1[1100], x2_L1[1100], x1_ENC[1100], y2;
68 int long_caos2;
69
70 // Vector de permutación
71 int f[1000], per[1000];
72
73 // Vector para criptograma
74 double CRIPTO[1000+5];
75
76 // ********************* Configuración LCD ****************** /////
77 // Conexión LCD-ESP32
78 LiquidCrystal lcd(12, 14, 32, 33, 25, 26);
79 //            lcd(Rs, E,  D4, D5, D6, D7)
80
81 // Caracter para escribir la letra "ñ"
82 byte eneMin[8] =
83  {
84    0b00001110,
85    0b00000000,
86    0b00010110,
87    0b00011001,
88    0b00010001,
89    0b00010001,
90    0b00010001,
91    0b00000000
92  };
93
94 // Caracter para escribir la letra "é"
95  byte eAcento[8] =
```

```
96  {
97     0b00000010,
98     0b00000100,
99     0b00001110,
100    0b00010001,
101    0b00011111,
102    0b00010000,
103    0b00001110,
104    0b00000000
105 };
106
107 // Caracter para escribir la letra "ó"
108 byte oAcento[8] = // o
109 {
110    0b00000010,
111    0b00000100,
112    0b00001110,
113    0b00010001,
114    0b00010001,
115    0b00010001,
116    0b00001110,
117    0b00000000
118 };
119
120 // ********************** Funciones utilizadas ***************** /////
121
122 // Función para realizar la operación mod(x,y)
123 double mod(double X, float Y){
124   double z = X - ( floor(X) / Y ) * Y;
125   return z;
126 }
127
128 void setup()
129 {
130   // Inicialización LCD
131   lcd.begin(16,2); // LCD 16 columnas 2 lineas
132   lcd.createChar(1, eneMin); // Inicializa caracter "ñ"
133   lcd.createChar(2, eAcento); // Inicializa caracter "é"
134   lcd.createChar(3, oAcento); // Inicializa caracter "ó"
135
136   // Despliega mensaje "SISTEMA DE CIFRADO"
137   lcd.clear();
138   lcd.setCursor(3,0);
139   lcd.print("SISTEMA DE");
140   lcd.setCursor(4,1);
141   lcd.print("CIFRADO");
142   delay(4000);
143
144   // Despliega mensaje "Iniciando sistema...."
145   lcd.clear();
146   lcd.setCursor(3,0);
147   lcd.print("Iniciando");
148   lcd.setCursor(3,1);
149   lcd.print("sistema....");
150   delay(2000);
151
152   // Configuración de entradas y salidas
153   pinMode(button, INPUT);
154
```

```
155   // Inicializar memoria SD
156   if (!SD.begin()) {
157     while(1);
158   }
159
160   // Inicializar WiFi
161   WiFi.begin(ssid, password);
162   while(WiFi.status() != WL_CONNECTED)
163   {
164     delay(300);
165   }
166
167   delay(2000);
168 }
169
170 void loop() {
171   // Despliega mensaje "Pulse el botón para cifrar"
172   lcd.clear();
173   lcd.setCursor(1,0);
174   lcd.print("Pulse el bot");
175   lcd.write(3);
176   lcd.write("n");
177   lcd.setCursor(2,1);
178   lcd.print("para cifrar");
179
180   //Permite que el ESP32 realice funciones indispensables
181   yield();
182
183   // Espera a que se presione el botón
184   while(!(digitalRead(button)==true));
185
186   // Despliega mensaje "Leyendo señal médica..."
187   lcd.clear();
188   lcd.setCursor(1,0);
189   lcd.print("Leyendo se");
190   lcd.write(1);
191   lcd.write("al");
192   lcd.setCursor(5,1);
193   lcd.print("m");
194   lcd.write(2);
195   lcd.write("dica...");
196   delay(1000);
197
198   // Abre el archivo CSV que contiene la bioseñal para su lectura
199   // y determina min(P)
200   char c;
201   String bufferString = "";
202   double _min=200.0, _max=0.0, var;
203
204   myFile = SD.open(archivo_Encriptar);
205   if (myFile) {
206     //Calcula el valor mínimo del ECG
207     while (myFile.available()) {
208       c = (char)myFile.read();
209       if(c != '\n')
210       {
211         bufferString += c;
212       }
213       else
```

```
214       {
215         var = bufferString.toDouble();
216         _min = min(_min,var);
217         bufferString = "";
218         LONG++;
219       }
220     }
221   }
222   else {
223     while(1);
224   }
225   myFile.close();
226
227   //Primer escalamiento y calcular max(N)
228   myFile = SD.open(archivo_Encriptar);
229   if(myFile) {
230     while (myFile.available()) {
231       c = (char)myFile.read();
232       if(c != '\n')
233       {
234         bufferString += c;
235       }
236       else
237       {
238         var = bufferString.toDouble()-(_min-0.01);
239         _max = max(_max,var);
240         bufferString = "";
241       }
242     }
243   }
244   else {
245     while(1);
246   }
247   myFile.close();
248
249   // Normalizar señal entre 0 y 1 (señal transformada)
250   double val;
251   int i=0;
252   myFile = SD.open(archivo_Encriptar);
253   while (myFile.available())
254   {
255     c = (char)myFile.read();
256     if(c != '\n')
257     {
258       bufferString += c;
259     }
260     else
261     {
262       norm[i] = (bufferString.toDouble()-(_min-0.01))/(_max+0.01);
263       bufferString = "";
264       i++;
265     }
266   }
267   myFile.close();
268
269   // Despliega mensaje "Cifrando..."
270   lcd.clear();
271   lcd.setCursor(2,0);
272   lcd.print("Cifrando...");
```

```
273
274   // MANEJO DE LLAVE
275   // Determinar rangos de llaves dividiendo llave en segmentos de 8
276   for(int i=0; i<4; i++)
277   {
278     for(int j=0; j<8; j++)
279     {
280       Llave[i][j] = cadena[i*8+j];
281     }
282     // Convertir hex - dec
283     x[i] = strtoul(Llave[i], 0, 16);
284   }
285
286   dec_AE = x[0];
287   dec_BE = x[1];
288   dec_CE = x[2];
289   dec_DE = x[3];
290
291   // Determinar el valor a sumar en las condiciones iniciales y parámetros
292   val_AE = dec_AE / (4294967296 + 1.0);
293   val_BE = dec_BE / (4294967296 + 1.0);
294   val_CE = dec_CE / (4294967296 + 1.0);
295   val_DE = dec_DE / (4294967296 + 1.0);
296
297   // ITERACIÓN DE MAPA SLIM
298
299   V_PA2 = mod(val_AE + val_BE,1.0) * 0.001;
300   V_CI2 = mod(val_CE + val_DE,1.0);
301
302   V_PA2_b = mod(val_AE + val_CE,1.0) * 0.001;
303   V_CI2_b = mod(val_BE + val_DE,1.0);
304
305   //Parámetros
306   a2_L1 = 0.999 + V_PA2;
307   b = 7.999 + V_PA2_b;
308
309   //Condiciones iniciales
310   x2_L1_1[0] = V_CI2;
311   y[0] = V_CI2_b;
312
313   // Número de iteraciones de mapa SLIM
314   long_caos2 = LONG + 100;
315
316   for(i=0; i < long_caos2-1; i++)
317   {
318     // Mapa SLIM
319     x2_L1_1[i+1] = sin(b*y[i])*sin(50/x2_L1_1[i]);
320     y[i+1] = a2_L1*(1-2*x2_L1_1[i+1]*x2_L1_1[i+1])*sin(50/y[i]);
321
322     // Optimización de secuencia X
323     x2_L1[i] = mod(x2_L1_1[i]*1000,1.0);
324   }
325
326   // Calcular el valor de Z
327   double SUMA;
328   for(i=0;i<LONG;i++)
329   {
330     SUMA = SUMA + ( (norm[i]+1) * x2_L1[long_caos2-1-i]) + x2_L1[long_caos2-1-i];
331   }
```

```
332  // Solo valores entre 0 - 1
333  SUMA = floor(mod(SUMA,1.0)*e15)/e15;
334
335  // ITERACIÓN DE MAPA SENO-HÉNON REALIMENTADO PARA SECUENCIAS DE PERMITACIÓN-
         DIFUSIÓN
336
337  mod_PA = mod(val_AE + val_BE + SUMA,1.0) * 0.001;
338  mod_CI = mod(val_CE + val_DE + SUMA,1.0);
339
340  mod_PA_b = mod(val_AE + val_CE + SUMA,1.0) * 0.001;
341  mod_CI_b = mod(val_BE + val_DE + SUMA,1.0);
342
343  //Parámetros
344  a_LL1 = 9.999 + mod_PA;
345  b1 = 8.999 + mod_PA_b;
346
347  //Condiciones iniciales
348  x1_ENC[0] = mod_CI;
349  y2 = mod_CI_b;
350
351  for(i=0; i < long_caos2-1; i++)
352  {
353    // Mapa Seno-Hénon realimentadocon secuencias de mapa SLIM
354    x1_ENC[i+1] = mod(1-a_LL1*(sin(x1_ENC[i])*sin(x1_ENC[i]))+y2+ x2_L1_1[i],1.0);
355    y2 = mod(b1*x1_ENC[i] + y[i] ,1.0);
356  }
357
358  // SECUENCIA PARA PERMUTACION
359
360  for(i=0;i<LONG;i++)
361  {
362    per[i]=round((x1_ENC[(long_caos2)-LONG + i]*(LONG-1)));
363  }
364
365  // Optimización de secuencia de permutación
366  // Busca posiciones de repetidos y los pone en un vector
367  f[0] = per[0];
368  int s=0,j, k, vect_r2D[500];
369  byte en;
370  for(k=1;k<LONG;k++)
371  {
372    en=1;
373    f[k]=per[k];
374    for(j=0;j<(k);j++)
375    {
376      if(f[k] == f[j] && en == 1)
377      {
378        vect_r2D[s]=k;
379        s=s+1;
380        en=0;
381      }
382    }
383  }
384
385  // Buscar numeros que no esten en vector de permutación
386  int preg_num=0, no=0, num_no_est2D[500];
387  byte saltar=0;
388  for(int v=0; v<LONG; v++)
389  {
```

```
390     // Pregunta si esta en vector de permutación
391     for(int vvv=0; vvv<LONG; vvv++)
392     {
393       if(preg_num == per[vvv])
394       {
395         saltar=1; // Si esta entra.
396       }
397     }
398     // Si no esta en repetidos y en vector de permutación, entra
399     if(saltar == 0)
400     {
401       num_no_est2D[no]=preg_num; // Guarda número.
402       no=no+1;
403     }
404     saltar=0;
405     preg_num=preg_num+1;
406   }
407
408   // Se actualiza vector de permutación (tiene todos los espacios)
409   int tam_vect=s;
410   byte en_dos=round(tam_vect/2);
411   if(tam_vect %2 != 0)
412   {
413     en_dos = en_dos +1;
414   }
415
416   byte cambio=1;
417   int S=0;
418   for(s=0; s<tam_vect; s++)
419   {
420     if(cambio == 1)
421     {
422      per[vect_r2D[s]]=num_no_est2D[S];
423      cambio=0;
424     }
425     else
426     {
427      per[vect_r2D[s]]=num_no_est2D[S+en_dos];
428      S=S+1;
429      cambio=1;
430     }
431   }
432
433   // SECUENCIA PARA DIFUSION
434   for(i=0; i<LONG; i++)
435   {
436     y[i] = mod((x1_ENC[long_caos2-LONG + i]*1000) + SUMA,1.0);
437     y[i] = floor(y[i]*e15)/e15;
438   }
439
440   // PROCESO DE PERMUTACION y DIFUSION (CIFRADO)
441   for(i=0; i<LONG; i++)
442   {
443     CRIPTO[i] = mod(norm[per[i]] +  y[i] +  y[LONG-i-1],1.0);
444   }
445
446   // Se añaden los últimos 5 datos al final del criptograma
447   CRIPTO[LONG] = SUMA;
448   if(_min < 0)
```

```
449   {
450     CRIPTO[LONG + 1] = 1.000000000000000; //Si es -, es 1
451   }
452   else
453   {
454     CRIPTO[LONG + 1] = 0.000000000000000; //Si es +, es 0
455   }
456   CRIPTO[LONG + 2] = abs(_min)/100000.0;
457   if(_max < 0)
458   {
459     CRIPTO[LONG + 3] = 1.000000000000000;; //Si es -, es 1
460   }
461   else
462   {
463     CRIPTO[LONG + 3] = 0.000000000000000; //Si es +, es 0
464   }
465   CRIPTO[LONG + 4] = abs(_max)/100000.0;
466
467   // ******************** Transmisión de criptograma *************** /////
468
469   // Despliega mensaje "Enviando a servidor..."
470   lcd.clear();
471   lcd.setCursor(2,0);
472   lcd.print("Enviando a");
473   lcd.setCursor(2,1);
474   lcd.print("servidor...");
475   delay(1000);
476
477   // Guarda cripgrama en un archivo en memoria micro-SD
478   myFile = SD.open(fileName, FILE_WRITE);
479   for(i=0; i<LONG+5; i++)
480   {
481     myFile.println(CRIPTO[i],15);
482   }
483   myFile.close();
484
485   // Calcula tamaño de archivo
486   myFile = SD.open(fileName, FILE_READ);
487   fileSize = myFile.size();
488   chunks = fileSize/(16*MTU_Size);
489   count = 0;
490
491   // Reaiza transmisión de archivo a servidor local
492   if(client.connect(servername, 80))
493   {
494     //HTTP POST request:
495     client.println("POST /upload.php?up=1&ACM=1 HTTP/1.1");
496     client.print("Host: ");
497     client.println(servername);
498     client.println("Connection: close");
499     client.println("Content-Type: multipart/form-data; boundary=--84989444
          e2484915a216e1718e0f93f0");
500     client.print("Content-Length: ");client.println(myFile.size()+188);
501     client.println();
502
503     client.println("----84989444e2484915a216e1718e0f93f0");
504     client.println("Content-Disposition: form-data; name=\"FileGDF\"; filename=\""
           + String(nombre_archivo) + "\"");
505     client.println("Content-Type: application/octet-stream");
```

```
506     client.println();
507
508     client.setNoDelay(1);
509
510     while(myFile.available())
511     {
512       if(myFile.available() >= MTU_Size)
513       {
514         clientCount = MTU_Size;
515       }
516       else
517       {
518         clientCount = myFile.available();
519       }
520
521       myFile.read(&clientBuf[0],clientCount);
522
523       client.write((const uint8_t *)&clientBuf[0], clientCount);
524       count++;
525       if(count == chunks)
526       {
527         count = 0;
528       }
529     }
530
531     client.println();
532     client.println("----84989444e2484915a216e1718e0f93f0--"); //form end
533     client.println();
534
535     myFile.close();
536     // Elimina criptograma de la memoria micro-SD
537     SD.remove(fileName);
538   }
539
540   // Respuesta del servidor
541   while(client.available())
542   {
543   }
544   client.stop();
545
546   // Despliega mensaje "Proceso terminado!"
547   lcd.clear();
548   lcd.setCursor(4,0);
549   lcd.print("Proceso");
550   lcd.setCursor(3,1);
551   lcd.print("terminado!");
552
553   delay(5000);
554 }
```

Apéndice B

Programa para módulo receptor

En este apéndice se presenta el código en lenguaje C con el que se programó la placa ESP32 correspondiente al módulo receptor para solicitar y recibir el criptograma desde el servidor local, por medio de la conexión WiFi, para finalmente recuperar y almacenar la señal médica en la memoria micro-SD.

```
/*************************************************************
  Algoritmo de descifrado caótico para recepción de bioseñales
  por internet.

  Elaborado por: Armando Ceseña Villa
*************************************************************/

// Librerías para WiFI y SD
#include <WiFi.h> // Wi-Fi
#include <SPI.h> // Comunicación módulo SD
#include <SD.h> // SD
#include <LiquidCrystal.h> // LCD

// Factor de división
#define e15 1000000000000000

// Entrada digital para botón
const uint8_t button = 15;

// ***** Configuración módulo de lectura  de memoria micro-SD ***** /////
// Crea un objeto File para manejar el archivo de la SD
File medFile;

// Nombre para guardar archivo recuperado en SD
char fileName[25] = "/Recuperado_Biosignal.csv";

// *********** Variables para envío de datos a servidor ************ /////
// Constantes para configuración WiFi
const char *ssid     = "*******";
const char *password = "*******";
const char *servername = "***.***.***.***";

// Instrucción HTTP para descargar criptograma de servidor
String url = "/download.php?name=Criptograma.csv";

```

```
// Crea un objeto "client" para manejar la conexión al servidor
WiFiClient client;

// ***************** Variables para descifrado ***************** /////

// Clave secreta
const char cadena[]="1122334455667788990OAABBCCDDEEFF";

char Llave[4][8];
unsigned long dec_AE,dec_BE,dec_CE,dec_DE, x[4];
double val_AE,val_BE,val_CE,val_DE;

// Criptograma recuperado
double norm[1005];
int LONG = 0;

// Valores para calcular parámetros y condiciones iniciales
double V_PA2, V_CI2, V_PA2_b, V_CI2_b, mod_PA, mod_CI, mod_PA_b, mod_CI_b;

// Parámetros y condiciones iniciales de mapas caóticos
double a2_L1, b, a_LL1, b1;
double x1_ENC[1100], y2, x2_L1_1_2, y3, x2_L1_1[1100], y[1100];
int long_caos2, i;

// Vector de permutación
int f[1000], per[1000];

//Vector para señal descifrada
double DECIFR[1000];

// ********************* Configuración LCD ****************** /////
// Conexión LCD-ESP32
LiquidCrystal lcd(12, 14, 32, 33, 25, 26);
//            lcd(Rs, E,  D4, D5, D6, D7)

// Caracter para escribir la letra "ñ"
byte eneMin[8] =
 {
   0b00001110,
   0b00000000,
   0b00010110,
   0b00011001,
   0b00010001,
   0b00010001,
   0b00010001,
   0b00000000
 };

// Caracter para escribir la letra "é"
 byte eAcento[8] =
 {
    0b00000010,
    0b00000100,
    0b00001110,
    0b00010001,
    0b00011111,
    0b00010000,
    0b00001110,
    0b00000000
```

```
95  };
96
97  // Caracter para escribir la letra "ó"
98  byte oAcento[8] = // o
99  {
100    0b00000010,
101    0b00000100,
102    0b00001110,
103    0b00010001,
104    0b00010001,
105    0b00010001,
106    0b00001110,
107    0b00000000
108 };
109
110 // ********************** Funciones utilizadas ***************** /////
111
112 // Función para realizar la operación mod()
113 double mod(double X, float Y){
114   double z = X - ( floor(X) / Y ) * Y;
115   return z;
116 }
117
118 void setup() {
119   // Inicialización LCD
120   lcd.begin(16,2); // LCD 16 columnas 2 lineas
121   lcd.createChar(1, eneMin); // Inicializa caracter "ñ"
122   lcd.createChar(2, eAcento); // Inicializa caracter "é"
123   lcd.createChar(3, oAcento); // Inicializa caracter "ó"
124
125   // Despliega mensaje "SISTEMA DE DESCIFRADO"
126   lcd.clear();
127   lcd.setCursor(3,0);
128   lcd.print("SISTEMA DE");
129   lcd.setCursor(3,1);
130   lcd.print("DESCIFRADO");
131   delay(4000);
132
133   // Despliega mensaje "Iniciando sistema...."
134   lcd.clear();
135   lcd.setCursor(3,0);
136   lcd.print("Iniciando");
137   lcd.setCursor(3,1);
138   lcd.print("sistema....");
139   delay(2000);
140
141   // Configuración de entradas y salidas
142   pinMode(button, INPUT);
143
144   // Inicializar memoria SD
145   if (!SD.begin()) {
146     while(1);
147   }
148
149   // Inicializar WiFi
150   WiFi.begin(ssid, password);
151   while(WiFi.status() != WL_CONNECTED)
152   {
153     delay(300);
```

```
154   }
155
156   delay(2000);
157 }
158
159 void loop() {
160   // Despliega mensaje "Pulse el botón para descifrar"
161   lcd.clear();
162   lcd.setCursor(1,0);
163   lcd.print("Pulse el bot");
164   lcd.write(3);
165   lcd.write("n");
166   lcd.setCursor(1,1);
167   lcd.print("para descifrar");
168
169   //Permite que el ESP32 realice funciones indispensables
170   yield();
171
172   // Espera a que se presione el botón
173   while(!(digitalRead(button)==true));
174
175   // Despliega mensaje "Extrayendo señal del servidor..."
176   lcd.clear();
177   lcd.setCursor(0,0);
178   lcd.print("Extrayendo se");
179   lcd.write(1);
180   lcd.write("al");
181   lcd.setCursor(0,1);
182   lcd.print("del servidor...");
183   delay(1000);
184
185   // Se conecta con el servidor del Raspberry
186   if(client.connect(servername, 80))
187   {
188     client.setNoDelay(1);
189   }
190
191   //HTTP GET request, solicita criptograma al servidor
192   int bytesSent = client.print(String("GET ") + url + " HTTP/1.1\r\n" +
193                "Host: " + servername + "\r\n" +
194                "Connection: close\r\n\r\n");
195
196   // Espera inicio de recepción de datos
197   char c;
198   String bufferString = "";
199   double _min = 0.0, _max = 0.0, var;
200   while(true)
201   {
202     c = client.read();
203     if(c == '\t')
204     {
205       break;
206     }
207   }
208
209   // Almacenamiento del criptograma enviado por el servidor
210   while(client.available())
211   {
212     c = client.read();
```

```
213     if(c != '\n')
214     {
215       bufferString += c;
216     }
217     else
218     {
219       norm[LONG] = bufferString.toDouble();
220       bufferString = "";
221       LONG++;
222     }
223   }
224   client.stop();
225
226   // Despliega mensaje "Descifrando..."
227   lcd.clear();
228   lcd.setCursor(1,0);
229   lcd.print("Descifrando...");
230
231   // Longitud de criptograma
232   LONG = LONG-5;
233
234   // Recuperar valor de Z, Min y Max, ubicados en los últimos 5 datos
235   double SUMA = norm[LONG-1+1];
236   if(norm[LONG-1+2] == 1) // ES NEGATIVO
237   {
238     _min = -1*norm[LONG-1+3];
239   }
240   else // ES POSITIVO
241   {
242     _min = norm[LONG-1+3];
243   }
244
245   if(norm[LONG-1+4] == 1) // ES NEGATIVO
246   {
247     _max = -1*norm[LONG-1+5];
248   }
249   else // ES POSITIVO
250   {
251     _max = norm[LONG-1+5];
252   }
253
254   _min = _min*100000;
255   _max = _max*100000;
256
257   // MANEJO DE LLAVE
258   // Determinar rangos de llaves dividiendo llave en segmentos de 8
259   for(int i=0; i<4; i++)
260   {
261     for(int j=0; j<8; j++)
262     {
263       Llave[i][j] = cadena[i*8+j];
264     }
265     // Convertir hex - dec
266     x[i] = strtoul(Llave[i], 0, 16);
267   }
268
269   dec_AE = x[0];
270   dec_BE = x[1];
271   dec_CE = x[2];
```

```
272   dec_DE = x[3];
273
274   // Determinar el valor a sumar en las condiciones iniciales y parámetros
275   val_AE = dec_AE / (4294967296 + 1.0);
276   val_BE = dec_BE / (4294967296 + 1.0);
277   val_CE = dec_CE / (4294967296 + 1.0);
278   val_DE = dec_DE / (4294967296 + 1.0);
279
280   // MAPA SLIM: Condiciones iniciales y parámetros
281
282   V_PA2 = mod(val_AE + val_BE,1.0) * 0.001;
283   V_CI2 = mod(val_CE + val_DE,1.0);
284
285   V_PA2_b = mod(val_AE + val_CE,1.0) * 0.001;
286   V_CI2_b = mod(val_BE + val_DE,1.0);
287
288   //Parámetros
289   a2_L1 = 0.999 + V_PA2;
290   b = 7.999 + V_PA2_b;
291
292   //Condiciones iniciales
293   x2_L1_1_2 = V_CI2;
294   y2 = V_CI2_b;
295
296   // MAPA SENO-HÉNON: Condiciones iniciales y parámetros
297
298   mod_PA = mod(val_AE + val_BE + SUMA,1.0) * 0.001;
299   mod_CI = mod(val_CE + val_DE + SUMA,1.0);
300
301   mod_PA_b = mod(val_AE + val_CE + SUMA,1.0) * 0.001;
302   mod_CI_b = mod(val_BE + val_DE + SUMA,1.0);
303
304   //Parámetros
305   a_LL1 = 9.999 + mod_PA;
306   b1 = 8.999 + mod_PA_b;
307
308   //Condiciones iniciales
309   x1_ENC[0] = mod_CI;
310   y3 = mod_CI_b;
311
312   // Número de iteraciones
313   long_caos2 = LONG + 100;
314
315   double aux1, aux2;
316   for(i=0; i < long_caos2-1; i++)
317   {
318     // Mapa Seno-Hénon
319     x1_ENC[i+1] = mod(1-a_LL1*(sin(x1_ENC[i])*sin(x1_ENC[i]))+y3+ x2_L1_1_2,1.0);
320     y3 = mod(b1*x1_ENC[i] + y2 ,1.0);
321
322     //Mapa SLIM
323     x2_L1_1_2 = sin(b*y2)*sin(50/x2_L1_1_2);
324     y2 = a2_L1*(1-2*x2_L1_1_2*x2_L1_1_2)*sin(50/y2);
325   }
326
327   // SECUENCIA PARA PERMUTACION
328
329   for(i=0;i<LONG;i++)
330   {
```

```
331     per[i]=round((x1_ENC[(long_caos2)-LONG + i]*(LONG-1)));
332   }
333
334   // Optimización de secuencia de permutación
335   // Busca posiciones de repetidos y los pone en un vector
336   f[0] = per[0];
337   int s=0,j, k, vect_r2D[500];
338   byte en;
339   for(k=1;k<LONG;k++)
340   {
341     en=1;
342     f[k]=per[k];
343     for(j=0;j<(k);j++)
344     {
345       if(f[k] == f[j] && en == 1)
346       {
347         vect_r2D[s]=k;
348         s=s+1;
349         en=0;
350       }
351     }
352   }
353
354   // Buscar numeros que no esten en vector de permutación
355   int preg_num=0, no=0, num_no_est2D[500];
356   byte saltar=0;
357   for(int v=0; v<LONG; v++)
358   {
359     // Pregunta si esta en vector de permutación
360     for(int vvv=0; vvv<LONG; vvv++)
361     {
362       if(preg_num == per[vvv])
363       {
364         saltar=1; // Si esta entra.
365       }
366     }
367     // Si no esta en repetidos y en vector de permutación, entra
368     if(saltar == 0)
369     {
370       num_no_est2D[no]=preg_num; // Guarda número.
371       no=no+1;
372     }
373     saltar=0;
374     preg_num=preg_num+1;
375   }
376
377   // Se actualiza vector de permutación (tiene todos los espacios)
378   int tam_vect=s;
379   byte en_dos=round(tam_vect/2);
380   if(tam_vect %2 != 0)
381   {
382     en_dos = en_dos +1;
383   }
384
385   byte cambio=1;
386   int S=0;
387   for(s=0; s<tam_vect; s++)
388   {
389     if(cambio == 1)
```

```
390     {
391      per[vect_r2D[s]]=num_no_est2D[S];
392      cambio=0;
393     }
394     else
395     {
396      per[vect_r2D[s]]=num_no_est2D[S+en_dos];
397      S=S+1;
398      cambio=1;
399     }
400   }
401
402   //  SECUENCIA PARA DIFUSION
403   for(i=0; i<LONG; i++)
404   {
405     y[i] = mod((x1_ENC[(long_caos2)-LONG + i]*1000) + SUMA,1.0);
406     y[i] = floor(y[i]*e15)/e15;
407   }
408
409   // PROCESO DE PERMUTACION y DIFUSION INVERSOS
410   double DECIFR_aux;
411   for(i=0; i<LONG; i++)
412   {
413     DECIFR_aux = mod(norm[i] -  y[i] -  y[LONG-i-1],1.0);
414     DECIFR[per[i]] = (DECIFR_aux*(_max+0.01)) + (_min-0.01);
415   }
416
417   // Grabar bioseñal recuperada en memoria micro-SD
418   medFile = SD.open(fileName, FILE_WRITE);
419   if(medFile)
420   {
421     for(i=0; i<LONG; i++)
422     {
423       medFile.println(DECIFR[i],15);
424     }
425   }
426   medFile.close();
427
428   // Despliega mensaje "Proceso terminado!"
429   lcd.clear();
430   lcd.setCursor(4,0); // Primera fila
431   lcd.print("Proceso");
432   lcd.setCursor(3,1); // Primera fila
433   lcd.print("terminado!");
434
435   delay(5000);
436 }
```

Apéndice C

Programa para servidor en Raspberry Pi

En este apéndice se presenta la programación realizada para el funcionamiento del servidor local en una placa Raspberry Pi Zero W. Para ello, se implementaron dos archivos:

- El primer archivo, llamado *upload.php*, contiene código en lenguaje PHP 5 para administrar la carga del criptograma enviado por el módulo transmisor y mantenerlo almacenado en la memoria micro-SD de la placa Raspberry Pi Zero W. El código del archivo se muestra a continuación.

```
<?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.";
  }
}
// para raspberry pi chown www-data:www-data /var/www/html -R && chmod 0775 /var/www/html/AutoPost1.0/De
?>
```

- El segundo archivo, llamado *download.php*, contiene código en lenguaje PHP 5 para atender la petición del módulo receptor y enviar el criptograma almacenado en la memoria de la placa Raspberry Pi Zero W al mismo cuando lo solicite. El código del archivo se muestra a continuación.

```
<?php
$target_dir = "uploads/";
$filename = $_GET["name"];
$target_file = $target_dir . $filename;

```

```
6
7  if (file_exists($target_file)) {
8      header('Content-Description: File Transfer');
9      header('Content-Type: application/octet-stream');
10     header('Content-Disposition: attachment; filename="'.basename($target_file).'"');
11     header('Content-Length: ' . filesize($target_file));
12     echo("\t");
13     readfile($target_file);
14 }
15 ?>
```