INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE INGENIERÍA MECÁNICA Y ELECTRICA

SECCIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

PROPUESTA DE UN ALGORITMO DE CIFRADO HÍBRIDO EMPLEANDO EL MÉTODO S.P.P.S. CON ROTACÓN CUATERNIÓNICA

TESIS

QUE PARA OBTENER EL TITULO DE:
MAESTRO EN CIENCIAS
EN INGENIERÍA DE TELECOMUNICACIONES

PRESENTA:

ING. HÉCTOR ÁVILA CASTRO

DIRECTORES DE TESIS:

DR. HÉCTOR OVIEDO GALDEANO
DR. FRANCISCO GALLEGOS FUNES

UNIDAD ZACATENCO

2020

SIP-14
REP 2017

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México siendo las 10:00 horas del día 10 del mes de Marzo del 2020 se reunieron los miembros de la Comisión Revisora de la Tesis, designada por el Colegio de Profesores de Posgrado de: E.S.I.M.E ZACATENCO

para examinar la tesis titulada:

PROPUESTA DE UN ALGORITMO DE CIFRADO HÍBRIDO BASADO EN EL MÉTODO S.P.P.S. CON ROTACIÓN CUATERNIÓNICA

por el (la) alumno (a):

Apellido Paterno:	ÁVILA	Apellido Materno:	CASTRO	Nombre (s):	HÉCTOR

Número de registro: A 1 8 0 8 9 9

Aspirante del Programa Académico de Posgrado: MAESTRÍA EN CIENCIAS EN INGENIERÍA DE TELECOMUNICACIONES

Una vez que se realizó un análisis de similitud de texto, utilizando el software antiplagio, se encontró que el trabajo de tesis tiene 17 % de similitud. **Se adjunta reporte de software utilizado**.

Después que esta Comisión revisó exhaustivamente el contenido, estructura, intención y ubicación de los textos de la tesis identificados como coincidentes con otros documentos, concluyó que en el presente trabajo **SI** ☐ **NO** ☒ **SE CONSTITUYE UN POSIBLE PLAGIO.**

JUSTIFICACIÓN DE LA CONCLUSIÓN: *(Por ejemplo, el % de similitud se localiza en metodologías adecuadamente referidas a fuente original)*

Con base en el análisis realizado en la plataforma "Turnitin" se concluye que el presente trabajo no constituye de ninguna manera en plagio ya que el porcentaje de similitud es Del 17%

****Es responsabilidad del alumno como autor de la tesis la verificación antiplagio, y del Director o Directores de tesis el análisis del % de similitud para establecer el riesgo o la existencia de un posible plagio.**

Finalmente y posterior a la lectura, revisión individual, así como el análisis e intercambio de opiniones, los miembros de la Comisión manifestaron **APROBAR** ☒ **SUSPENDER** ☐ **NO APROBAR** ☐ la tesis por **UNANIMIDAD** ☒ o **MAYORÍA** ☐ en virtud de los motivos siguientes:

Que la tesis cumple con los requisitos que marca el reglamento de estudios de posgrado vigente para tesis de maestría

COMISIÓN REVISORA DE TESIS

Director de Tesis
DR. HECTOR OVIEDO GALDEANO
14261-EG-19, ASIGNATURA

Presidente
DR. VLADIMIR RABINOVITCH
13624-EG-18 Y COLEGIADO

Secretario
DR. RAÚL CASTILLO PÉREZ
12922-EE-17 Y COLEGIADO

2do. Director de Tesis
DR. FRANCISCO JAVIER GALLEGOS FUNES
12672-EF-17 Y COLEGIADO

3er Vocal
M. EN C. JOSÉ ANTONIO LÓPEZ TOLEDO
EXTERNO

PRESIDENTE DEL COLEGIO DE PROFESORES
DR. JOSÉ MARTÍNEZ TRINIDAD

I.P.N.
SECCIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN
ZACATENCO

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA DE AUTORIZACIÓN DE USO DE OBRA PARA DIFUSIÓN

En la Ciudad de México el día 18 del mes de Agosto del año 2022, el (la) que suscribe Héctor Ávila Castro alumno(a) del programa Maestría en ciencias en Ingeniería de telecomunicaciones con número de registro A180899, adscrito(a) Sección de estudios de Posgrado e Investigación de la ESIME unidad Zacatenco, manifiesta que es autor(a) intelectual del presente trabajo de tesis bajo la dirección del Dr. Héctor Oviedo Galdeano y el Dr. Francisco Gallegos Funes y cede los derechos del trabajo intitulado Propuesta de un algoritmo de cifrado híbrido basado en el método S.P.P.S. con rotación cuaterniónica, al Instituto Politécnico Nacional, para su difusión con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expresado del autor y/o director(es). Este puede ser obtenido escribiendo a las siguiente(s) dirección(es) de correo. delta5303@hotmail.com. Si el permiso se otorga, al usuario deberá dar agradecimiento correspondiente y citar la fuente de este.

.

Héctor Ávila Castro

Contenido

Capítulo 1

Criptografía y sistemas de seguridad

Capítulo 2

Herramientas matemáticas de los sistemas de seguridad

Capítulo 3

Sistemas Criptográficos

Capítulo 4

Pruebas y resultados

Índice de tablas y figuras

Introducción

Las comunicaciones han sido un factor determinante para el desarrollo del ser humano a través de la historia desde las primeras formas de comunicación escrita, como las cartas, hasta las comunicaciones digitales que se utilizan actualmente para la transferencia de información. Han servido para alertar a las personas de catástrofes naturales, terminar guerras e incluso conocer cómo es el universo a través de imágenes.

Así, como se han desarrollado sistemas de comunicación cada vez más complejos que aseguran que la información que es transmitida llegue a su destino de manera íntegra y privada, existen también individuos cuyo objetivo es interrumpir o distorsionar el mensaje para obtener una ventaja utilizando la información robada. Es por esta razón que para proteger la información que desea ser transmitida se creó la criptografía. Ha sido utilizada a lo largo de la historia para ocultar la información que se transmite a través de un canal inseguro, desde los romanos con el sistema de cifrado César hasta los sistemas que son utilizados hoy en día como AES o RSA.

Los sistemas de cifrado se han vuelto más complejos y más difíciles de romper a medida que el desarrollo tecnológico avanza, de igual forma, el mismo desarrollo de tecnologías permitirá que en un futuro próximo estos sistemas puedan ser quebrados, es por esto que el desarrollo de sistemas criptográficos debe estar en constante innovación para que las comunicaciones seguras puedan estar garantizadas.

La forma de crear nuevos y mejores sistemas de cifrado consiste en la aplicación de métodos matemáticos cada vez más complejos para que los procesos de cifrado incrementen el nivel de seguridad.

Actualmente las comunicaciones digitales permiten manejar y compartir información tan importante como documentos confidenciales, realizar compras, transferir grandes sumas de dinero de manera electrónica y hasta monitorear o controlar distintos aparatos electrónicos en nuestro hogar. Aunque no represente un mayor problema debido a que las cosas que se mencionan anteriormente son elementos meramente materiales que pueden ser reemplazados, la situación se vuelve considerablemente preocupante al pensar que en el futuro próximo con el desarrollo de “el internet de las cosas” con el cual se podrán controlar

no solo aspectos del hogar sino también aparatos que impliquen la seguridad e incluso la salud de un ser humano, se denota que los sistemas criptográficos necesitan ser mejorados e inquebrantables.

Debido al constante enfrentamiento que existe entre quienes desarrollan sistemas de cifrado y quienes buscan romperlos, la criptografía tiene un campo de desarrollo interminable.

Justificación

De acuerdo con la CONDUSEF los ataques o fraudes cibernéticos han crecido exponencialmente en los últimos años. Sólo en el año 2019 los ciber fraudes han aumentado en un 35% en comparación con los ciber ataques realizados en el año 2018. En la siguiente tabla realizada por el CONDUSEF [33] se observa el aumento de los ataques digitales y la disminución de los ataques tradicionales.

	2015	2016	2017	2018	2019	VAR. (2019 vs 2018)
TOTALES	1,683,661	2,674,023	3,345,664	3,515,712	4,318,853	
CIBERNÉTICOS	304,256	836,532	1,578,000	2,074,554	2,807,819	35%
	18%	31%	47%	59%	65%	-
TRADICIONALES	1,379,287	1,835,409	1,762,805	1,441,115	1,511,022	5%
	82%	69%	53%	41%	35%	-
Por definir	118	2,082	4,859	43	12	-

Como se aprecia en la tabla los fraudes cibernéticos aumentan de manera exponencial al pasar los años. En el 2019 el monto reclamado debido a ciber fraudes ascendió a $5,908 MDP. Por esto es imperativo crear nuevos y mejores sistemas de cifrado de la información para que los fraudes electrónicos disminuyan cada vez más hasta incluso desaparecer.

Objetivo general

Desarrollar un algoritmo de cifrado que proporcione una transmisión segura de datos por medio de canales inseguros empleando el método de potencias formales de parámetro espectral para solucionar funciones pseudoanalíticas; y combinándolo con rotación cuaternionica.

Prologo

Es necesario explicar cómo es que la criptografía utiliza las matemáticas para desarrollar los procesos de cifrado y descifrado. Es importante explicar cómo la criptografía ha evolucionado a través de los años debido a la implementación de procesos matemáticos cada vez más complejos y a la implementación de distintas máquinas que han facilitado el desarrollo de los sistemas de cifrado.

La presente investigación tiene como objetivo el desarrollo de un sistema de cifrado híbrido empleando el método S.P.P.S. (*spectral parameter power series*) conocidas en nuestro idioma como potencias formales de parámetro espectral [1][2]. Este método matemático será combinado con la rotación cuaterniónica desarrollada por William Hamilton para obtener un sistema de cifrado efectivo eficiente [2][3].

Se describe cómo es que las culturas antiguas empleaban distintos métodos para ocultar o encriptar los mensajes de manera que la información sólo fuese obtenida por quien estuviese designado. También se describe cómo a media que la tecnología de la humanidad se desarrolla, es aprovechada para la implementación de sistemas de cifrado cada vez más complejos [4].

Se explican las herramientas matemáticas básicas empleadas por todos los sistemas criptográficos y cómo es que dichas herramientas funcionan para realizar el proceso de cifrado y descifrado de manera correcta y que no exista pérdida de información [5][6].

También son descritos distintos métodos de cifrado que son actualizados actualmente para mantener segura la información transmitida. De igual manera se explica cómo funcionan distintos métodos que emplean cuaterniones para realizar el cifrado, se observa cómo es que los sistemas que emplean cuaterniones ofrecen un gran nivel de seguridad y eficiencia y por lo tanto demuestran que es una buena opción utilizar este método matemático para el desarrollo de nuevos sistemas [1][7][8].

Debido a que el objetivo de este trabajo es el desarrollo de un sistema de cifrado hibrido, se describe cómo se emplean y mezclan dos métodos matemáticos diferentes para realizar una transferencia de información segura.

Resumen

En el presente trabajo se presenta la propuesta de un algoritmo de cifrado simétrico combinando el método matemático S.P.P.S. con la rotación cuaterniónica. Mediante la combinación de estos métodos y analizando los resultados obtenidos fue posible obtener un sistema de cifrado que es seguro y eficiente.

Los esquemas de cifrado simétrico son la base de cualquier sistema de cifrado ya sea simétrico o asimétrico. Los cifrados utilizados actualmente ofrecen un buen nivel de seguridad y eficiencia, para diseñar e implementar un nuevo esquema de cifrado es necesario tomar en cuenta estos dos factores ya que son los elementos principales que la industria requiere.

El esquema propuesto en la presente investigación posee un buen nivel de seguridad, comparado con otros sistemas, debido a que se ha empleado el método matemático de potencias formales de parámetro espectral. Al proponer problemas matemáticos que solo pueden solucionarse empleando el método S.P.P.S. se garantiza la seguridad del sistema criptográfico ya que si alguien intenta resolver el problema empleando cualquier otro método no obtendrá los valores que se necesitan para descifrar el sistema.

Como medida adicional para aumentar la seguridad y mejorar la eficiencia del sistema, se aplicó la rotación cuaterniónica al cifrado S.P.P.S. La rotación cuaterniónica agrega otra operación al cifrado aumentando su complejidad y seguridad, y debido a la cantidad de datos que se pueden cifrar a la vez utilizando la matriz de rotación la eficiencia también es mejorada.

Abstract

In the present investigation the proposal of a new symmetric cryptographic system using the S.P.P.S. mathematical method combined with the quaternion rotation is presented. Due to the combination of these two methods it is possible to obtain a secure and efficient cipher method.

The symmetric cipher systems are the base of any cryptographic system. The cipher methods used today offer a good security and efficiency level, in order to design a new cipher scheme these two elements must be considered.

The scheme proposed in this investigation has a good security level due to the implementation of the mathematical method of spectral parameter power series. The proposal of a problem that can only be solved using S.P.P.S. ensures that this is the only method that can cipher and decipher information.

As an additional step to increase the security and efficiency of the proposal system the quaternion rotation is added. The rotation adds another operation to the method and therefore the security of the cipher is increased and due to the data, the quaternion rotation can encrypt at once the efficiency of the system is improved.

Capítulo 1

Criptografía y seguridad en redes

1.1. Criptografía clásica

La palabra criptografía tiene raíces griegas, proviene de las palabras *kryptos* y *grapehin*, que significan “escondido” y “escritura” respectivamente, por lo que la palabra criptografía significa literalmente “escritura escondida” y actualmente se refiere al proceso de desarrollar y utilizar herramientas mediante las cuales hacer que la información o datos importantes no puedan ser leídos más que por las personas a las que se le es enviada dicha información [1][4].

La criptografía es el arte y ciencia de encriptación, para poder desarrollar un buen algoritmo de encriptación se requieren conocimientos científicos, así como del entendimiento de distintos procesos matemáticos. También es importante tener cierta experiencia y la mentalidad adecuada para enfrentar y resolver problemas relacionados con la seguridad [9][10].

La criptografía es uno de los campos técnicos más antiguos que se han estudiado. Existen registros que datan de hasta hace cuatro mil años [4].

Los registros más antiguos de la criptografía vienen de Egipto aproximadamente hace unos 2000 años antes de Cristo en donde los jeroglíficos eran usados para decorar las tumbas de los reyes. Estos símbolos contaban la historia de vida de los reyes. Los jeroglíficos tenían el propósito de ser criptográficos, pero sin esconder el texto. En los años siguientes estos escritos se volvieron más complicados de descifrar y eventualmente se perdió el interés por descifrar estos símbolos [4].

En el pasado (y en algunos casos actualmente) la gente relacionaba la criptografía con las artes ocultas. Se pensaba que la criptografía era utilizada para comunicarse con espíritus obscuros, y en general se creó una mala imagen de la criptografía. Sin embargo, los primeros criptógrafos eran científicos, aun así, mucha gente también creía que estas personas eran seguidores de lo oculto.

En la antigua China se ocultaba el significado de las palabras debido a la naturaleza de su lenguaje. Los mensajes eran transformados en símbolos para obtener privacidad, aun así, la encriptación de mensajes en la antigua China parecía ser involuntaria y solo el resultado de

su lenguaje. No existen pruebas de que se haya utilizado un sistema de encriptación más complejo en conquistas militares.

Por otro lado, en la India se utilizaba una forma de escritura escondida más avanzada, el gobierno utilizaba códigos secretos para comunicarse con agentes que tenía distribuidos por todo el país. Los primeros sistemas criptográficos de la India hacían uso de la sustitución alfabética y estaban basados en fonética.

En Mesopotamia al igual que en Egipto la forma de encriptar los textos era sustituyendo los caracteres en los textos por figuras cuneiformes, esta técnica también era utilizada en Babilonia y Asiria [4].

Existe un cifrado hebreo que es utilizado en la Biblia algunas veces. El método conocido como "*cifrado atbash*", consiste en sustituir la última letra del alfabeto por la primera y viceversa. En la imagen se observa un ejemplo del cifrado *atbash*.

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

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

Figura 1. Cifrado "atbash" [11]

En Grecia se desarrollaron novedosos métodos de encriptación como la idea de enrollar un pergamino alrededor de una vara de cierto grosor y longitud y escribir el mensaje sobre el pergamino enrollado. Cuando el pergamino se desenrollaba de la vara el texto escrito sobre el papel no tendría sentido. El receptor del mensaje tendría una vara de las mismas dimensiones y así podría descifrar el mensaje.

Otra forma de encriptación griega fue el "cuadro de Polybius". Este método consistía en ordenar las letras del alfabeto en un cuadro de cinco por cinco con la i y la j ocupando el mismo cuadro. Las columnas y son numeradas del uno al cinco para que cada letra tenga un número de dos dígitos [4].

Otra forma de encriptación griega fue el "cuadro de Polybius", el cual se muestra en la figura 2.

	1	2	3	4	5
1	P	O	L	Y	B
2	I/J	U	S	A	C
3	D	E	F	G	H
4	K	M	N	Q	R
5	T	V	W	X	Z

Figura 2. Cuadro de Polybius [12]

Este método consistía en ordenar las letras del alfabeto en un cuadro de cinco por cinco con la i y la j ocupando el mismo cuadro. Las columnas y filas son numeradas del uno al cinco para que cada letra tenga un número de dos dígitos.

El método criptográfico utilizado por los romanos consistía en intercambiar las letras del alfabeto por la letra que se encontraba tres posiciones adelante. Este método era utilizado por Julio César para comunicarse secretamente con los generales de su ejército. Es por esta razón que este método es llamado “cifrado César” y es tal vez el método de cifrado histórico más mencionado en la literatura de criptografía. El cifrado César puede observarse en la figura 3.

Figura 3. Cifrado Cesar [13]

El siguiente aporte importante a la criptografía fue hecho por los árabes. El autor árabe Qalqashandi desarrolló una técnica para resolver métodos de cifrado que es utilizada actualmente. La técnica consiste en escribir todos los símbolos de un texto encriptado y contar la frecuencia con la que aparece cada símbolo. Después se obtiene el promedio de frecuencia de cada letra del lenguaje en el que es más probable que el mensaje fue escrito, utilizando estos promedios es posible obtener el texto plano [4].

Más adelante en la edad media, la criptografía comenzó a ser un tema de interés para muchos investigadores y gobiernos por lo que el desarrollo y descubrimientos en la criptografía fue en aumento.

En la Europa del Este los códigos para encriptar mensajes se hicieron más populares. La mayoría de los gobiernos de esta región utilizaban la criptografía de alguna forma u otra para comunicarse con sus respectivos embajadores.

El primer hecho de gran importancia sucedió en Venecia, Italia en donde fue creada en 1452 una organización de elaboración, cuyo único propósito era el de estudiar la criptografía. La organización contaba con tres secretarios de cifrado quienes resolvían y creaban métodos de cifrado que era utilizados por el gobierno.

Fue en Italia donde después de años de investigación Leon Battista Alberti desarrolló el cifrado por substitución poli-alfabética. Este método consta de dos discos de cobre que tienen escritos encima el alfabeto y que son puestos uno encima del otro. El proceso de cifrado consiste en alinear una letra del disco interior con una letra del disco exterior la cual es escrita como la primera letra del texto cifrado. Los discos se mantienen inmóviles haciendo que cada letra del texto plano este alineada con el disco interior que a su vez está alineada con el disco exterior. Después de escribir algunas palabras, los discos son rotados de tal manera que las letras del disco interior estén alineadas con otras letras del disco exterior. El proceso se repite hasta que todo el texto plano ha sido cifrado. Está técnica de cifrado hace más difícil que el mensaje pueda ser desencriptado utilizando la técnica de la frecuencia [4]. En la figura siguiente se muestra un ejemplo de encriptación poli-alfabética.

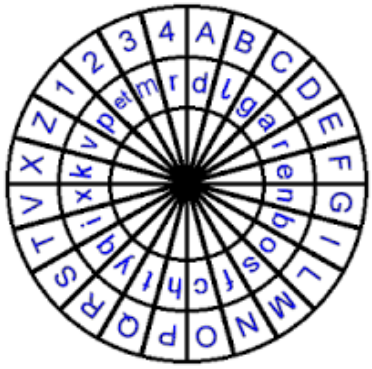


Figura 4. Disco utilizado para encriptación poli-alfabetica [14]

La encriptación poli-alfabética pasó por una serie de cambios y los más notables se atribuyen a Vigenere, quien fue la primera persona en diseñar lo que parecía un algoritmo de encriptación que hacía uso de una llave para encriptar y desencriptar los mensajes.

En 1682 el gobierno francés había formado un grupo de criptógrafos conocido como “La Cámara Negra”, el cual se encargaba de descifrar códigos para el gobierno. Para los años 1700s los grupos “Cámara Negra” eran comunes en Europa. Cada país europeo tenía su propio grupo dedicado al desarrollo e investigación de criptografía [4].

Con la llegada del telégrafo en 1844, el interés público por la criptografía tuvo un cambio abrupto, debido a que el telégrafo no era un medio de comunicación seguro, la criptografía se presentaba como una opción para encriptar los mensajes enviados por telégrafo y así asegurar las comunicaciones hechas por este nuevo medio. Así las personas comenzaron a desarrollar sus propios algoritmos de encriptación.

En 1864 el oficial prusiano Friedrich Kasiski presentó un método que descifró casi todos los métodos de encriptación que existían hasta entonces. El método consistía en encontrar cadenas de caracteres repetidos en el texto cifrado. La distancia entre las cadenas repetidas se utiliza para encontrar la longitud de la llave. Dado que el cifrado de cadenas del texto plano ocurre a distancias que son múltiplo de la longitud de la llave, si se encuentran los divisores comunes más grandes de las distancias de repetición nos llevaran a encontrar la longitud de la llave. Al conocer la longitud de la llave se utilizan métodos estadísticos en cada carácter y la frecuencia de uso implica a qué carácter se representa.

En el siglo XX Inglaterra comenzó a darle prioridad a la investigación y desarrollo de la criptografía, debido a esto los ingleses fueron capaces de descifrar la mayor parte de los cifrados enemigos durante las guerras del siglo.

El mayor logro de los descifradores ingleses fue romper con el código de encriptación de la fuerza naval alemana. La solución a este código se simplifico bastante debido al hecho de que las comunicaciones alemanas siempre utilizaban palabras nacionalistas o políticas como llave de encriptación.

El desarrollo de máquinas diseñadas para encriptar mensajes cambió drásticamente la naturaleza de la criptografía y el criptoanálisis. El desarrollo de la criptografía se relacionó

entonces con el diseño de la maquinaria. Los métodos de encriptación se volvieron mecánicos y electromecánicos [4].

La máquina Enigma fue inventada por el ingeniero alemán Arthur Scherbius en el final de la primera guerra mundial. Los mensajes alemanes enviados durante la segunda guerra mundial eran codificados con esta máquina. Enigma utilizaba 4 o más rotores. Los rotores giran a distintas velocidades conforme se teclea el mensaje y codifica el texto plano según la posición de los rotores. En este caso la llave de este criptosistema era la posición inicial de los rotores.

El criptosistema usado por la máquina Enigma fue quebrado por un grupo de criptoanálisis británico en conjunto con un grupo de criptoanalistas polacos refugiados.

El grupo británico de criptoanálisis era dirigido por el brillante matemático inglés Alan Turing que desde muy temprana edad demostró una enorme capacidad y talento para resolver problemas matemáticos. Turing logró romper la máquina Enigma haciendo uso de otra máquina que él mismo diseño, su máquina "Bombe" fue diseñada y construida exclusivamente para romper el código Enigma [4]. La figura 5 muestra la máquina enigma y su funcionamiento.

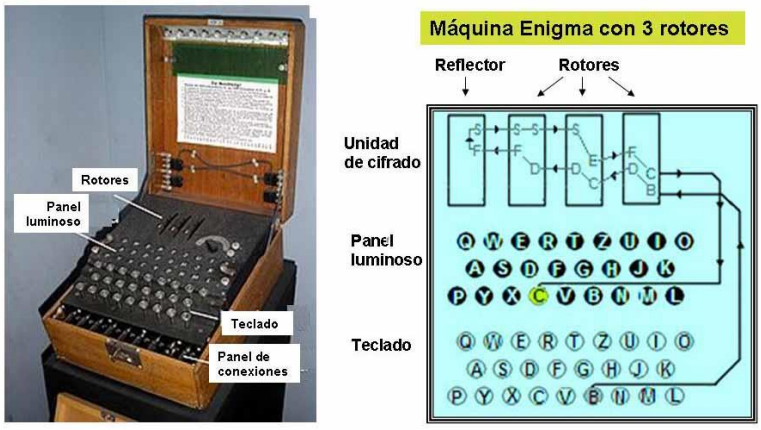


Figura 5. Máquina enigma [15]

La invención de la máquina de Turing para romper el criptosistema Enigma marcó el inicio de la criptografía moderna. La máquina "Bombe" se observa en la figura 6.

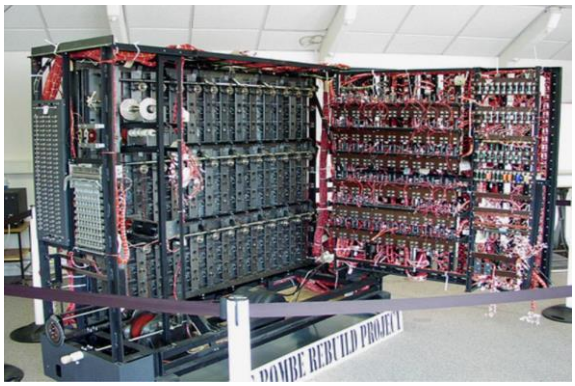


Figura 6. Máquina "Bombe" diseñada por Alan Turing [16]

En 1948 el matemático americano Claude Shannon públicó su artículo "*A Communication Theory of Secret Systems*". Fue uno de los primeros criptógrafos modernos que aplicó técnicas avanzadas de matemáticas a los algoritmos de cifrado. La mayor aportación del escrito de Shannon es el descubrimiento de la medida de fortaleza criptográfica llamada "distancia de unicidad". Esta indica la cantidad de texto cifrado necesario para determinar el texto plano del mensaje. Teóricamente, con el tiempo suficiente es posible que cada algoritmo de cifrado puede ser quebrado dada una longitud de texto cifrado tal que su distancia de unicidad sea 1.

En 1950 Turing con su estudio "*Cimputering Machinering and Intelligence*", sentó las bases para el desarrollo de la inteligencia artificial. Alan Turing es considerado el padre de la informática debido a los estudios que hizo relacionados con la inteligencia artificial que ayudaron al desarrollo de la primera computadora.

1.2. Criptografía moderna: La criptografía en los sistemas de seguridad

Un sistema de seguridad es un conjunto de elementos dedicados cuyo propósito principal es mantener la información resguardada. La criptografía es sólo una pequeña parte de este sistema, sin embargo, es una parte crítica del sistema, es la parte que le da acceso a las personas autorizadas y rechaza a las personas no autorizadas. Es importante mencionar que la criptografía por sí sola en un sistema de seguridad es inútil. Cuando existe una brecha en la seguridad no es porque la criptografía sea la parte débil del sistema, en algunos casos incluso un mal sistema criptográfico puede ser mucho mejor que el resto de los elementos en

el sistema de seguridad. La criptografía solo demuestra todo su valor y utilidad cuando el resto del sistema es lo suficientemente seguro para resistir el ataque de cibercriminales [9].

A diferencia de cualquier otro tipo de ingeniería los adversarios que existen para la seguridad en sistemas no son predecibles y la mayoría son inteligentes, astutos, tienen malicia y están dispuestos a probar cosas que nadie ha intentado. No se puede conocer la razón, el motivo, ni el momento del ataque. Los ataques pueden ocurrir mucho tiempo después de cuando se diseñó el sistema de seguridad por lo que el atacante tendría una gran ventaja para irrumpir el sistema.

No existe un sistema que sea completamente seguro. Cada sistema criptográfico tiene sus puntos débiles y es por esto que es imprescindible fortalecerlo ya que "Un sistema es tan fuerte como su eslabón más débil"[1].

La criptografía puede ser empleada de muchas formas. Es una parte esencial de muchos sistemas de seguridad. También puede ser utilizada para debilitar sistemas de seguridad si se usa de forma apropiada. La criptografía no es la solución completa para los problemas de seguridad de un sistema, puede ser parte de la solución o puede ser parte del problema [9][17].

1.3. Objetivos de los sistemas de seguridad

Existen distintos aspectos que deben de cumplirse para que un sistema de seguridad pueda ser considerado apto y utilizable.

- Integridad
  Los elementos enviados o la información que es transmitida no presentan ninguna modificación a menos que esta haya sido hecha por alguno de los entes autorizados.

- Disponibilidad
  La información transmitida debe ser utilizable por los usuarios autorizados cuando lo necesiten.

[5] Ferguson, Schneier y Kohno. *Cryptography Engeneering.* 1ª edición. Wiley Publishing. Estados Unidos de América. 2010. 1-22 .

- Privacidad
  Los datos enviados sólo deben ser accesibles por los usuarios autorizados.

- Autenticidad
  Se debe asegurar que la información enviada y recibida sea la información correcta y que dicha información llegue en el tiempo establecido.

- No repudio
  Este aspecto sirve para evitar que la entidad que envía o modifica la información alegue que no lo hizo.

1.4. Tipos de cifrado

El objetivo principal de la criptografía es la encriptación. Cuando existe comunicación entre dos o más entidades es común suponer que los canales de comunicación no son seguros. Con el trabajo suficiente cualquier sistema criptográfico práctico puede ser atacado. La eficacia de un algoritmo depende de cuanto trabajo se requiere para romper este algoritmo [9].

Actualmente para que un sistema sea considerado seguro necesita tener por lo menos un nivel de seguridad de 128 bits. Esto quiere decir que para que un ataque intente romper la seguridad del sistema va a requerir ejecutar al menos 2^{128} pasos [18].

Existen dos esquemas de encriptación: la encriptación de llave pública y la encriptación de llave privada.

- Cifrado de llave privada o cifrado simétrico. Probablemente la criptografía de llave privada es la forma más tradicional de encriptar un mensaje. Este tipo de cifrado consiste en usar una única llave para encriptar la información que va a ser enviada a través del canal inseguro. Utilizando la misma llave es como el receptor desencripta la información para poder utilizarla. Es importante que dicha llave sea muy difícil de descubrir ya que hoy en día los ordenadores pueden descifrar llaves velozmente. Se debe tener en cuenta que los algoritmos criptográficos son públicos, por lo que su fortaleza debe depender de su complejidad interna y de la longitud de la llave

empleada para evitar los ataques de fuerza bruta. En la figura 7 se muestra el esquema del cifrado simétrico [17].

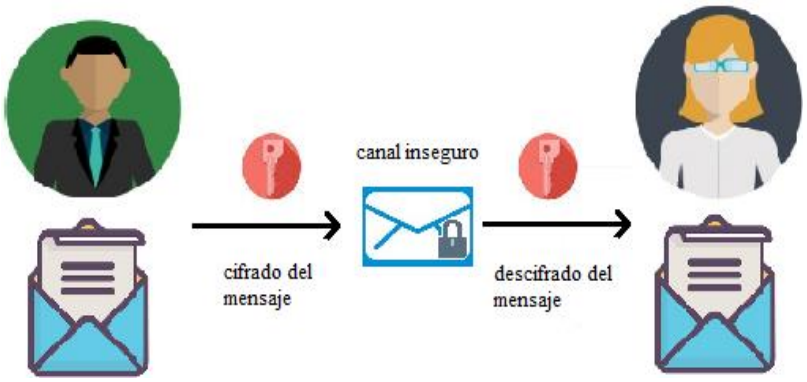


Figura 7. Esquema de cifrado simétrico

- Cifrado de llave privada o cifrado asimétrico. El sistema de encriptación por llave pública fue presentado por primera vez en 1976 por matemáticos de la universidad de Stanford en Estados Unidos. La idea principal de este sistema es que cada persona o entidad tenga dos claves relacionadas de manera que sean complementarias; una de éstas será divulgada o expuesta libremente, la cual será conocida como clave pública. La segunda, llamada clave privada es la que deberá permanecer secreta y sólo el receptor tendrá conocimiento de ésta.

  Gracias a que la clave pública ha sido expuesta y divulgada, cualquier persona podrá codificar mensajes utilizándola y únicamente el poseedor de la segunda clave, la cual es complemento de la llave pública y será capaz de decodificar el mensaje.

  El principal problema de la criptografía simétrica es la distribución de llaves ya que si este esquema es implementado para la comunicación entre muchas personas, cada persona tendría que memorizar la clave del otro y dependiendo de con quién quiera compartir la información y usar la clave correspondiente. La criptográfica asimétrica soluciona este problema ya que cada usuario solo debe compartir la clave pública que todos pueden utilizar [18].

  Por otro lado, la desventaja que tiene la criptografía asimétrica con respecto al cifrado de llave privada es la eficiencia. Encriptar con el esquema simétrico puede, en la mayoría de los casos, ser un proceso más rápido.

Actualmente existen algoritmos de cifrado que combinan los estilos de criptografía para aprovechar las ventajas que cada esquema ofrece. En la figura se observa el esquema de encriptación asimétrico.

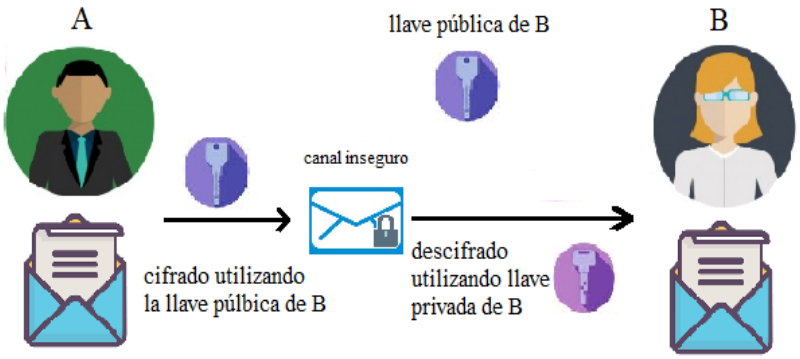


Figura 8. Esquema de cifrado asimétrico

1.5. Tipos de ataques

Existen principalmente dos tipos de ataques que un cibercriminal podría realizar para vulnerar un sistema de seguridad; el ataque pasivo y el ataque activo. El primer ataque, de tipo pasivo, es muy difícil de detectar ya que el atacante no altera la comunicación, únicamente revisa la información que está siendo transmitida. Los objetivos de este tipo de ataque son la intercepción de datos y el análisis de tráfico. Al realizar un ataque pasivo el cibercriminal sabrá cuál es el origen y el destino de la comunicación, también podrá extraer información acerca de los periodos de actividad de una determinada entidad. Los ataques pasivos no alteran de ninguna manera los datos enviados, es por esto que son tan difíciles de detectar. Una forma de evitar que este tipo de ataque funcione es mediante el cifrado de la información.

El segundo ataque, de tipo activo, consiste en modificar de alguna forma los datos, el flujo de datos transmitidos o la creación de flujo de datos y de esta forma se podrá obtener la información deseada [9][18]. Los ataques pasivos son técnicamente más difíciles de detectar que los ataques activos, y éstos pueden ser clasificados en cuatro categorías:

1. Suplantación de identidad

   El intruso se hace pasar por otra persona o logra acceder a la información que le está destinada. El atacante no altera el contenido de la información solamente obtiene los datos transmitidos, de esta manera la confidencialidad de los datos se rompe ya que aunque la información llegue a su destino ésta ya ha sido obtenido por una persona no autorizada.

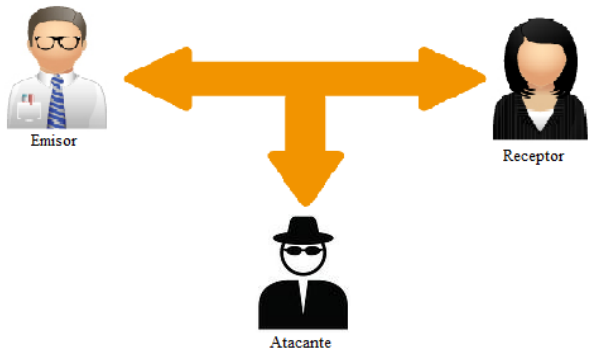


Figura 9. Esquema del ataque de suplantación de identidad

2. Degradación fraudulenta del servicio o interrupción

   Inhibe el uso normal o la gestión de recursos de comunicaciones. El atacante suprime todos los mensajes dirigidos a una determinada entidad o interrumpe el servicio de una red.

Figura 10. Esquema del ataque de interrupción de servicio

3. Modificación

   El intruso consigue interceptar el mensaje y alterar la información que éste contiene, también puede hacer que el mensaje llegue tiempo después o que varios mensajes lleguen con un orden diferente para producir un efecto diferente al que era esperado.

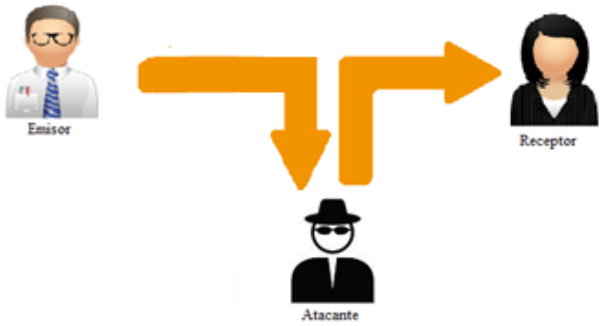


Figura 11. Modificación de mensaje

4. Creación de mensajes

   El atacante genera información falsa y la envía a una o varias entidades diferentes.

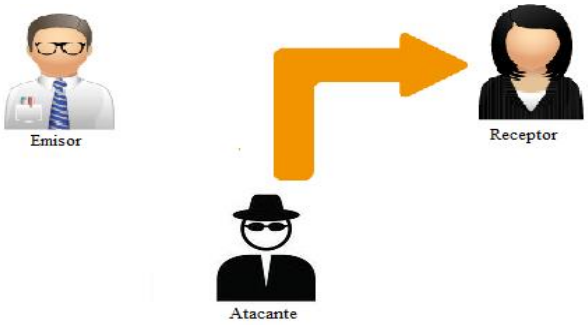


Figura 12. Creación de mensajes

1.6. Ataques a esquemas de cifrado

Existen distintos tipos de ataques que un cibercriminal puede ejecutar para romper un esquema de cifrado y cada uno tiene un nivel diferente de efectividad, sin embargo, el objetivo principal de estos ataques es obtener el texto plano a partir del texto cifrado.

- *Ataque conociendo el texto cifrado*

La mayoría de la gente cuando habla acerca de romper un sistema de seguridad se refiere a un ataque de este tipo en el cual el cibercriminal sólo puede ver el texto cifrado. Gracias a que el atacante solamente tiene acceso al texto cifrado es muy difícil que pueda romper la llave del criptosistema y obtener el texto plano.

- *Ataque conociendo el texto plano y el texto cifrado*

Aquí el atacante conoce el texto plano y el texto cifrado. El objetivo para este tipo de ataque es obtener la llave utilizada para cifrar el texto plano. Conociendo el texto cifrado y el texto plano es posible obtener la llave que se ha utilizado para cifrar el mensaje, de esta forma una vez conociendo la llave utilizada para codificar mensajes es posible descubrir la información de otros mensajes que utilicen la misma llave. Si no se conoce todo el texto plano es posible incluso deducir la llave si se conoce una parte del mensaje sin cifrar como puede ser la firma de la persona o empresa que envía dicha información.

Este tipo de ataque es más peligroso que el *Ataque conociendo el texto cifrado* ya que el cibercriminal tiene más información y la información extra beneficia el proceso de hackeo del atacante.

- *Ataque eligiendo una parte del mensaje plano*

Este es un tipo de ataque más peligroso que los descritos anteriormente. Durante este tipo de ataque, el atacante selecciona textos planos que son preparados especialmente para vulnerar al sistema fácilmente. Se puede elegir un número de textos planos, con su correspondiente texto cifrado, a conveniencia del cibercriminal.

Existen dos variaciones de este ataque. La primera variación es de forma desconectada; se prepara una serie de textos planos que se quiere encriptar antes de obtener los textos cifrados. La segunda variación es de forma conectada; se escogen textos planos nuevos dependiendo en los textos cifrados que ya se han recibido.

- *Ataques eligiendo una parte del texto cifrado*

En este tipo de ataque se utilizan partes tanto del texto plano como del texto cifrado. Aquí el atacante puede escoger los valores de los textos planos y de los textos cifrados. Por cada texto plano que se adquiera también se obtendrá su texto cifrado correspondiente y viceversa. Los atacantes tienen más libertad, esto hace que este ataque sea más peligroso que el *Ataque eligiendo una parte del texto plano*. La meta es descubrir la llave utilizada para cifrar los mensajes.

Todos los ataques descritos anteriormente tienen el potencial de descubrir o desencriptar un mensaje especifico. Sin embargo, cualquier criptosistema con un diseño razonable no debería tener problemas para resistir a cualquiera de estos ataques. Si un algoritmo de encriptación en desarrollo falla al enfrentarse a estos ataques es mejor revisar y mejorar su esquema [19].

Capítulo 2

Herramientas matemáticas de los sistemas de seguridad

2.1. Aritmética modular

Cuando se quiere encriptar un mensaje es necesario que la información del mensaje sea representada en forma numérica; esto no representa una complicación ya que de hecho las computadoras representan los caracteres y letras con números de acuerdo con lo establecido en el código ASCII, por ejemplo; la letra ‘o’ es representada por el número ‘111’ y el número ‘2’ es representado en el código ASCII como ‘50’ [20].

El cifrado consiste en transformar la información representada de forma numérica de manera que la información sea modificada y difiera del texto original, o texto plano, tanto que resulte irreconocible. Para que estas transformaciones puedan ser utilizadas al momento de cifrar información debe tener las siguientes propiedades:

- Ser invertible: ser posible obtener el texto original a partir del texto cifrado.
- Tener un proceso de cifrado y descifrado sencillo: esto dirigido hacia los entes que posean las claves necesarias para realizarlo.
- Tener un proceso para descifrado altamente funcional e indescifrable sin la clave necesaria.

Un algoritmo de cifrado es considerado seguro si el método aplicado para intentar romperlo es un ataque de Fuerza Bruta. Éste sólo es utilizado si no existe otra opción para romper un algoritmo de cifrado; este ataque puede romper cualquier sistema si se le da el tiempo necesario. No obstante, los algoritmos de seguridad actuales requieren de cantidades de tiempo medidas en cientos de años para poder ser descifrados. Si la única opción para romper un cripto sistema es el ataque de fuerza bruta, el sistema es considerado seguro [20].

2.2. Congruencias lineales

Si a, b y m son números enteros tales que $a - b$ es un múltiplo de m, que es positivo, se dice que a y b son congruentes respecto del módulo m, si la diferencia dividida por él produce el mismo resto. La relación de congruencia se expresa como $a \equiv b(mod(m))$.

Se puede observar que si *m* divide *a* – *b*, esto supone que ambos *a* y *b* tienen el mismo resto al ser divididos por el módulo *m*. Algunos ejemplos de congruencia son: 23 ≡ 2 mod (7) (23 = 3 • 7 + 2) y -6 ≡ 1 mod (7) (-6 = -7 • 1 + 1).

Cuando a y b no sean congruentes respecto del módulo m, escribiremos $a \not\equiv b(mod(m))$. De la propia definición se deduce que $a - b \equiv 0(mod(m))$. La congruencia puede expresarse como $a = mt + r$ con $0 \leq r < m$ en donde t es un número entero. A esta expresión se le llama división euclídea en el conjunto $\mathbb{N}$ de los números naturales.

A partir de la definición dada anteriormente, a continuación, se indican las propiedades de las congruencias:

1. Para todo $a \equiv a(mod(m))$, es decir, todo número es congruente consigo mismo respecto a cualquier módulo.
2. Si $a \equiv b(mod(m))$ entonces $b \equiv a(mod(m))$.
3. Si $a \equiv b(mod(m))$ y $b \equiv c(mod(m))$, entonces $a \equiv c(mod(m))$.
4. Si un número *a* es primo con m, todo $b \equiv a(mod(m))$ será también primo con m.
5. Si $a \equiv b(mod(m))$ y $c \equiv d(mod(m))$, también $a + c \equiv b + d\ (mod(m))$.
6. Si $a \equiv b(mod(m))$ y *c* es distinto a cero, entonces $a \cdot c \equiv b \cdot c(mod(m))$.
7. Si $a \equiv b(mod(m))$ y *d* es un divisor cualquiera de $a \equiv b(mod(d))$.
8. Si $a \cdot c \equiv b \cdot c(mod(m))$ y el $mcd(c, m)=d$ entonces, $a \equiv b(mod(\frac{m}{d}))$.
9. Si *k* es un número natural y $a \equiv b(mod(n))$ también $a^k \equiv b^k(mod(m))$.
10. Si *b es 1* b^2 entonces $(m - b)^2 \equiv b^2(mod(m))$.
11. Si *p* es un número primo entonces $(m + b)^2 \equiv m^p + n^p(mod(p))$.
12. Si *p* es número primo, $(m_1 + m_2 + \cdots + m_n)^2 \equiv m_1^p + m_2^p + \ldots + m_n^p(mod(p))$.

La relación de congruencia módulo *m* es una relación de equivalencia para todo $m \in Z$. Se le llama Z_m al conjunto cociente de *Z* respecto de la relación de congruencia módulo *m* [21][22]. Con las congruencias es posible establecer el siguiente conjunto de operaciones aritméticas;

Siendo $a, b, c, d \in Z$ y m ∈ N, tales que $a \equiv b$ (mod (*m*)) y $c \equiv d$ (mod (*m*)), entonces:

- $a + c \equiv b + d\ (mod(m))$
- $a \cdot c \equiv b \cdot d\ (mod(m))$

A partir de estas expresiones se pueden definir algunas propiedades aritméticas para las sumas de congruencias:

- Propiedad asociativa: $a + (b + c) \pmod{(m)} = (a + b) + c \pmod{(m)}$.
- Elemento neutro: existe un elemento $0 \in Zm$, tal que $a + 0 \pmod{(m)} = a \pmod{(m)}$.
- Elemento opuesto: existe un elemento $b \in Zm$, tal que $a + b = 0$.
- Propiedad conmutativa: $a + b \pmod{(m)} = b + a \pmod{(m)}$.

De la misma manera se pueden definir las propiedades aritméticas para el producto de congruencias:

- Propiedad cancelativa; $a \bullet c \equiv b \bullet c \pmod{(m)}$ y $\text{mcd}(m, c) = 1$, entonces $a \equiv b \pmod{(m)}$.
- Propiedad asociativa; $a \bullet (b \bullet c) \pmod{(m)} = (a \bullet b) \bullet c \pmod{(m)}$.
- Elemento neutro; existe un elemento $1 \in Z_m$, tal que $a \bullet 1 \pmod{(m)} = a \pmod{(m)}$.
- Elemento inverso; existe un elemento $a^{-1} \in Z_m$ para todo $a \in Zm$ con $\text{mcd}(a, m)=1$, tal que $a \bullet a^{-1} = 1$.
- Además de todas estas propiedades se tiene también la propiedad distributiva: $a \bullet (b + c) \pmod{(m)} = (a \bullet b) + (a \bullet c) \pmod{(m)}$.

2.3. Algoritmo de Euclides

El algoritmo de Euclides es utilizado para obtener el máximo común divisor de dos números. Para calcular el máximo común divisor entre dos números enteros positivos a y b se divide el número más grande, dígase a, entre el número más pequeño, dígase b; esta división proporcionará un cociente, c_1, y un resto, r_1. Si $r_1=0$, entonces $\text{mcd}(a, b)=b$, sino es cero se divide c_1, entre el resto, r_1, obteniendo otro cociente, c_2, y otro resto, r_2. Si $r_2=0$, entonces $\text{mcd}(a, b)= r_1$, sino es cero se repite la división entre el resto hasta que se logre un resultado sin divisor. Esto es, entonces; el máximo común divisor entre a y b es el último resto distinto de cero que se obtenga con el procedimiento anterior.

El teorema del algoritmo de Euclides establece: Sean a, b dos números enteros no nulos. Entonces $\text{mcd}(a, b) = \text{mcd}(b, r)$ donde $0 \leq r < b$ tal que existe un entero q con $a = b \bullet q + r$.

Para obtener el máximo común divisor de dos enteros utilizando el algoritmo de Euclides se definen q_i y r_i recursivamente mediante las siguientes ecuaciones:

$$a = b \bullet q_1 + r_1 \quad (0 \leq r_1 < b)$$

$$b = r_1 \bullet q_2 + r_2 \quad (0 \leq r_2 < r_1)$$

$$r_1 = r_2 \bullet q_3 + r_3 \quad (0 \leq r_3 < r_2)$$

y así sucesivamente hasta:

$$r_{k-3} = r_{k-2} \bullet q_{k-1} + r_{k-1} \quad (0 \leq r_{k-1} < r_{k-2})$$

$$r_{k-2} = r_{k-1} \bullet q_k \quad (r_k = 0)$$

A continuación, se muestra un ejemplo del cómo se emplea el algoritmo de Euclides para calcular el mcd(721, 448):

$$\begin{aligned} 721 &= 448 \bullet 1 + 273 \\ 448 &= 273 \bullet 1 + 175 \\ 273 &= 175 \bullet 1 + 98 \\ 175 &= 98 \bullet 1 + 77 \\ 98 &= 77 \bullet 1 + 21 \\ 77 &= 21 \bullet 3 + 14 \\ 21 &= 14 \bullet 1 + 7^* \\ 14 &= 7 \bullet 2 + 0 \end{aligned}$$

El número marcado con *, que es el último divisor que no es nulo, es el máximo común divisor de (721, 448).

2.4. Inverso multiplicativo

Existe también un algoritmo conocido como Euclides extendido; lo que busca este algoritmo extendido, además de calcular el máximo común divisor de dos números, es demostrar que

el mcd se puede expresar como una combinación lineal de dos números y encontrar cuáles son estos números. Esta propiedad resulta muy útil cuando *a* y *b* son primos relativos ya que permite obtener el inverso multiplicativo, el cual es utilizado en diversos algoritmos de cifrado como en el algoritmo asimétrico RSA [23][24].

2.5. Números primos

Decimos que un número entero $p > 1$ es primo si sus únicos divisores positivos son 1 y p. Un número entero $q > 1$ que no es primo es llamado número compuesto. Por lo que un número entero q es compuesto si y sólo si existen a, b enteros positivos, menores a q, tales que $q=a \bullet b$. Si el máximo común divisor de a, b es igual a 1 se dice entonces que a y b son números primos entre sí o que son primos relativos [20].

Los números primos y compuestos tienen las siguientes propiedades:

I. Si el número p es primo entonces, $p|x_1 \bullet x_2 \ldots x_n \Rightarrow p|x_1$ para algún i.
II. Teorema fundamental de la aritmética: Todo número natural mayor a 1 o bien es un número primo, o bien se puede descomponer como producto de números primos.
III. Si un número n es compuesto, se verifica que ha de tener un divisor primo menor o igual que su raíz cuadrada.
IV. Existen infinitos números primos.

2.6 *Test* de primalidad Miller-Rabin

Es posible descomponer números enteros grandes en factores primos; sin embargo, el tiempo para realizar esta operación puede resultar ser no justificable. Una factorización de un número $n = p \bullet q$, donde p y q son números primos y n tiene una longitud de 2048 bits, no se puede hacer en un tiempo que sea aceptable o eficiente para poder ser empleado en el desarrollo de un algoritmo de cifrado.

En 1978 fue presentado el algoritmo de cifrado RSA cuya seguridad reside en la dificultad para factorizar. Dicho algoritmo hace uso de dos números primos muy grandes para generar la llave de encriptación. La mayoría de los algoritmos de cifrado utilizan números primos muy grandes y la dificultad de descifrar un mensaje radica en determinar si un determinado es primo.

El *test* de primalidad Miller-Rabin fue desarrollado por Michael Rabin en 1980 y se basó en el trabajo de Gari Miller de 1976. Este es el *test* de primalidad más utilizado en la práctica.

Rabin demostró que para todo número entero compuesto la probabilidad de que el *test* lo detecte como un número primo es inferior a $(\frac{1}{4})^t$ donde t > 1 es un parámetro que se introduce para determinar la exactitud del *test* repetido t veces. Esto es; si el resultado del *test* indica que el número es compuesto, entonces, ese número realmente es compuesto. Si el número introducido es detectado como un número primo por el algoritmo, existe una gran probabilidad de que sí lo sea, la probabilidad aumenta dependiendo del número de veces t que se repita el *test*.

El teorema del algoritmo Miller-Rabin dice lo siguiente: sea $n > 3$ un número primo y se escriba $n - 1 = 2^r m$ donde m es impar y r un número natural no negativo. Sea cualquier número entero con $1 \leq a < n$, tal que mcd(a, n) = 1. Entonces $a^m \equiv 1 \pmod n$ ó $a^{2^j m} \equiv -1 \pmod m$, para algún j, $0 \leq j \leq r - 1$ [22].

Para determinar si un número n impar grande dado es un número primo o compuesto se siguen los siguientes pasos:

1. Se elige de manera arbitraria e independiente k números a tal que $2 \leq a \leq n - 1$.
2. Se determina con el algoritmo de Euclides mcd(a , n), si mcd(a, n) es distinto a 1 entonces n es compuesto.
3. Se prueba si bajo el a elegido se encuentra un resultado distinto a 1. Si el primer encuentro es 1 finaliza el algoritmo y n es compuesto. Si no se puede encontrar tal número a entonces el algoritmo termina como fallido.

Si el resultado del *test* es fallido entonces *n* tiene la apariencia de un número primo, pero no necesariamente lo es. La probabilidad para que resulte fallido un número compuesto *n* el *test* de Miller-Rabin es menor a $(\frac{1}{4})^t$. Si la *k* propuesta es un número muy grande, entonces la probabilidad de que *n* sea un número primo es alta.

Existe también un algoritmo empleado en procesos computacionales para generar un número primo *p* con *n* bits el cual consta de los siguientes pasos:

1. Se genera un número aleatorio de *n* bits.
2. Se pone 1 en el bit más significativo y en el menos significativo.
3. Se intenta dividir el número propuesto por una tabla de números primos que ha sido previamente propuesta.
4. Se ejecutan los *test* de primalidad sobre el número propuesto para verificar que sea un número primo.
5. Si el *test* de primalidad falla, se incrementa en dos unidades el número propuesto y se regresa al paso tres.

2.7. Funciones pseudoanalíticas

Las funciones pseudoanalíticas son soluciones generalizadas de las ecuaciones de Cauchy-Riemann. El análisis de funciones analíticas de variables complejas ocupa un lugar importante en el desarrollo y el estudio de las matemáticas; es por esto que en la literatura matemática existen muchas generalizaciones de estas funciones. En algunas se puede extender el dominio de las funciones que han sido consideradas o el rango o ambas [2][3].

Las funciones pseudoanalíticas poseen la propiedad de continuidad y cada clase de funciones pseudoanalíticas tiene tanta estructura como las funciones analíticas. Las operaciones como la diferenciación compleja y la integración compleja tienen contrapartes significativas en las funciones pseudoanalíticas.

Un par de funciones complejas *F* y *G* que poseen derivadas parciales en un dominio Ω con respecto a las variables reales *x* y *y* se dice que son una pareja generadora si satisfacen la condición:

$$Im\,(\bar{F}\,G) > 0$$

En donde $\bar{F}$ es el complejo conjugado de *F*: $\bar{F}$= ReF - *i*ImF. Así cada función compleja W puede ser representada por los elementos de las funciones generadoras:

$$W = \Phi F + \Psi G$$

Donde Φ y Ψ son funciones valuadas en los reales. La pareja (F, G) generaliza la pareja (1, i) la cual corresponde a la usual función analítica compleja [2][3]. A continuación, se introduce la notación de la derivada de *W*:

$$\partial_{(\mathrm{F,G})}\mathrm{W} = (\partial_{(\mathrm{z})}\,\Phi)\mathrm{F} + (\partial_{(\mathrm{z})}\,\Psi)\mathrm{G}$$

Donde $\partial_{(\mathrm{z})} = \frac{\partial}{\partial_{(\mathrm{x})}} - i\,\frac{\partial}{\partial_{(\mathrm{x})}}$ e *i* denota la unidad imaginaria i^2= -1. La derivada de la función W existe si y solo si:

$$(\partial_{(\bar{z})}\Phi)\mathrm{F} + (\partial_{(\bar{z})}\,\Psi)\mathrm{G} = 0 \qquad (1)$$

Las siguientes expresiones son conocidas como coeficientes característicos de la pareja generadora (*F*, G)

$$a_{(F,G)} = -\frac{\bar{F}\,\partial_{(\overline{z})}G - \partial_{(\overline{z})}F\bar{G}}{F\bar{G} - \bar{F}G} \qquad b_{(F,G)} = \frac{F\,\partial_{(\overline{z})}G - \partial_{(z)}FG}{F\bar{G} - \bar{F}G}$$

$$A_{(F,G)} = -\frac{\bar{F}\,\partial_{(z)}G - \partial_{(z)}F\bar{G}}{F\bar{G} - \bar{F}G} \qquad B_{(F,G)} = -\frac{F\,\partial_{(z)}G - \partial_{(z)}FG}{F\bar{G} - \bar{F}G}$$

Con las notaciones anteriores se puede reescribir la ecuación (1) como:

$$\partial_{(\overline{z})}W = a_{(F,G)}W + b_{(F,G)}W$$

La cual es conocida como ecuación de Vekua. Las soluciones de esta ecuación son llamadas (*F*, G) pseudoanalíticas. La solución general de esta ecuación puede ser presentada en forma de una serie de Taylor de potencias formales [2]:

$$W = \sum_n^{\infty} z^{(n)} \left(a_{n,}\, z_0; z\right)$$

Si existen dos parejas generadoras (*F*, *G*) y ($\tilde{F}$, $\tilde{G}$) tal que:

$$\tilde{F} = a_{11}F + a_{12}G$$

$$\tilde{G} = a_{21}F + a_{22}G$$

Son llamadas parejas generadoras, en donde a_{ij} son constantes reales. También se consideran como equivalentes si dos parejas generadoras definidas en el mismo dominio tienen los mismos coeficientes característicos. Si (*F*, *G*) y ($\tilde{F}$, $\tilde{G}$) son generadoras equivalentes entonces cada función ($\tilde{F}$, $\tilde{G}$) pseudoanalítica de primera clase es (*F*, *G*) pseudoanalítica de primera clase, entonces:

$$\frac{\partial_{(F,G)}\mathrm{W}}{\partial_{(z)}} = \frac{\partial_{(\tilde{F},\tilde{G})}\mathrm{W}}{\partial_{(z)}}$$

2.8. Potencias Formales

La potencia formal $z_0^{(0)}(a_{0,}\, z_0; z)$ con centro en z_0, coeficiente a_0 y exponente 0 es definida como la combinación lineal de las generadoras F_m y G_m se expresa como [1][25]:

$$z_0^{(0)}(a_{0,}\, z_0; z) = \lambda F_0 + \mu G_0$$

Donde λ y μ son constantes reales tal que

$$\lambda\, F_0(z_0) + \mu\, G_0(z_0) = a_0$$

Las potencias formales con exponentes mayores a 0 se definen con la fórmula recursiva:

$$Z_m^{(n)}(a_{n,}\, z_0; z) = \mathrm{n} \int_{z_0}^{z} Z_{m+1}^{(n-1)}\,(a_{n,}\, z_0; z) d_{(F_m\, G_m)}\mathrm{z}$$

Esta definición tiene distintas propiedades que implican:

1. $\lim_{z \to z_0} (z)^n \left(a_{n,}\, z_0; z\right) = a_n(\mathrm{z} - z_0)^n$

2. Considerando a_n = a' + a'' y siendo a', a'' constantes reales entonces:

$$Z_m^{(n)}(a' + a''_{,}\, z_0; z) = a' Z_m^{(n)}(1, z_0; z) + a'' Z_m^{(n)}(i, z_0; z)$$

3. Las potencias formales satisfacen las relaciones diferenciales:

$$\frac{\partial_{(F,G) Z_m^{(n)}(a,\, z_0; z)}}{\partial_{(z)}} = \mathrm{n}\, Z_{m+1}^{(n-1)}(a_{,}\, z_0; z)$$

4. Las fórmulas asintóticas $Z_m^{(n)}(a_{,}\, z_0; z) \sim a(z_0 - z)^n$, $\mathrm{z} \to z_0$ se cumplen.

Estas propiedades son válidas para todas las potencias formales; lo que significa que cualquier potencia formal $z^{(n)}(a_{n,}\, z_0; z)$ puede ser aproximada por la combinación lineal $z^{(n)}(1, z_0; z)$ y $z^{(n)}(i, z_0; z)$, así los cálculos numéricos serán realizados para aproximar estas dos clases de potencias formales [1][3].

Si se considera un par de funciones generadoras con la forma:

$$\mathrm{F}\ (\mathrm{x}, \mathrm{y}) = \frac{\sigma(x)}{\tau(y)} \qquad \mathrm{G}\ (\mathrm{x}, \mathrm{y}) = i\ \frac{\tau(y)}{\sigma(x)}$$

donde σ y τ son funciones valuadas en los reales de sus variables correspondientes y asumiendo que $\mathrm{z}_0 = 0$ y F (0) = 1 para que los cálculos sean más simples; de esta manera la potencia formal cero está construida de la siguiente forma:

$$X^{(0)}(x) = \tilde{X}^{(0)}(x) = Y^{(0)}(y) = \tilde{Y}^{(0)}(y) = 1$$

y para potencias mayores a cero [7]:

$$X^{(\mathrm{n})}(x) = \begin{cases} n \int_0^x X^{(\mathrm{n}-1)}(\xi) \frac{d\xi}{\sigma^2(\xi)} & para\ n\ impar \\ n \int_0^x X^{(\mathrm{n}-1)}(\xi) \sigma^2(\xi) d\xi & para\ n\ par \end{cases}$$

$$\tilde{X}^{(\mathrm{n})}(x) = \begin{cases} n \int_0^x \tilde{X}^{(\mathrm{n}-1)}(\xi) \sigma^2(\xi) d\xi & para\ n\ impar \\ n \int_0^x \tilde{X}^{(\mathrm{n}-1)}(\xi) \frac{d\xi}{\sigma^2(\xi)} & para\ n\ par \end{cases}$$

$$Y^{(\mathrm{n})}(y) = \begin{cases} n \int_0^y Y^{(\mathrm{n}-1)}(\eta)\frac{d\eta}{\tau^2(\eta)} & para\ n\ impar \\ n \int_0^y Y^{(\mathrm{n}-1)}(\eta)\tau^2(\eta)d\eta & para\ n\ par \end{cases}$$

$$\tilde{Y}^{(\mathrm{n})}(x) = \begin{cases} n \int_0^y \tilde{Y}^{(\mathrm{n}-1)}(\eta)\tau^2(\eta)d\eta & para\ n\ impar \\ n \int_0^y \tilde{Y}^{(\mathrm{n}-1)}(\eta)\frac{d\eta}{\tau^2(\eta)} & para\ n\ par \end{cases} \quad (2)$$

Entonces para a_n = a' + a'' se tiene:

$$z^{(\mathrm{n})}(a_n, 0; z) = \frac{\sigma(x)}{\tau(y)} \mathrm{Re}\ {}_*z^{(n)}(a_n, 0; z) + i\ \frac{\tau(y)}{\sigma(x)} \quad \mathrm{Im}\ {}_*z^{(n)}(a_n, 0; z) \quad (3)$$

donde

$${}_*z^{(n)}(a_n, 0; z) = a' \sum_{j=0}^{n} \binom{n}{j} X^{(n-j)} i^j Y^j + \mathrm{i}a'' \sum_{j=0}^{n} \binom{n}{j} \tilde{X}^{(n-j)} i^j \tilde{Y}^j \quad \text{para n impar;}$$

$${}_*z^{(n)}(a_n, 0; z) = a' \sum_{j=0}^{n} \binom{n}{j} \tilde{X}^{(n-j)} i^j Y^j + \mathrm{i}a'' \sum_{j=0}^{n} \binom{n}{j} X^{(n-j)} i^j \tilde{Y}^j \quad \text{para n par} \quad (4)$$

2.9. Cuaterniones

El conjunto de números complejos existe debido a la necesidad de dar sentido a la extracción de raíces cuadradas de números negativos. Con los números complejos es posible obtener las raíces cuadradas o de cualquier otro orden, así como logaritmos de cualquier número.

La representación de los números complejos en el plano y la representación de las rotaciones en el plano que eran el producto de complejos unitarios hizo pensar a los matemáticos si existía alguna forma de generalizar los números complejos de manera que se obtuviese un sistema de números hipercomplejos que se pudiesen representar en el espacio tridimensional y de donde fuera posible obtener una representación numérica de las rotaciones el espacio tridimensional.

William Rowan Hamilton fue considerado un matemático brillante desde niño y que hizo importantes contribuciones a los campos de la física y la astronomía; llevaba un buen tiempo intentando encontrar la manera de generalizar los números complejos a tres dimensiones. Hamilton trabajaba con la idea de introducir una segunda unidad imaginaria *j* a los números

complejos de manera que tanto i^2 como j^2 fueran iguales a -1. De esta manera los hipercomplejos estarían formados por tres partes: una real y dos imaginarias; la suma y resta de las tripletas no presentaba mayor dificultad, ya que se reducía a una suma formal componente a componente. El verdadero problema surgía al querer multiplicarlos y en especial asignar un valor al producto de *ij*. Si el resultado de la multiplicación era un número real se concluía que *j* era múltiplo real de *i*. Si el resultado del producto era un número imaginario la conclusión era que *j* era real y por lo tanto no era una unidad nueva [26].

El 16 de octubre de 1843 Hamilton se encontraba paseando con su esposa a través del puente de Brougham y súbitamente se le ocurrió introducir una nueva unidad imaginaria *k* de forma que $ij = k$, y renunciar a la conmutatividad del producto, que era hasta ese momento una propiedad fundamental tanto de los números reales como de los números complejos. Así las tres unidades imaginarias cumplirían con:

$$i^2 = j^2 = k^2 = ijk = -1$$

Y el producto de las unidades imaginarias sería:

$$ij = -ji = k$$

$$jk = -kj = i$$

$$ki = -ik = j$$

Un cuaternión tiene cuatro componentes: Una componente real; denominada parte escalar por Hamilton. Y tres componentes imaginarias; denominadas por Hamilton como parte vectorial del cuaternión. Un cuaternión $\boldsymbol{q}$ puede ser visto de la siguiente forma [1][7][8]:

$$\boldsymbol{q} = s + u\boldsymbol{i} + v\boldsymbol{j} + w\boldsymbol{k}$$

Donde *s*, *u*, *v* y *w* son números reales. La parte escalar del cuaternión $\boldsymbol{q}$ seria *s* mientras que la parte vectorial serian *u*, *v* y *w*.

Sea $\mathbb{H}(\mathbb{R})$el conjunto de cuaterniones reales, un cuaternión $\boldsymbol{q} \in \mathbb{H}(\mathbb{R})$ será representado como:

$$\boldsymbol{q} = q_0 + \sum_{j=0}^{3} q_j i_j = q_0 + q_1 i_1 + q_2 i_2 + q_3 i_3$$

Donde q_j son los coeficientes del cuaternión y i_j son las unidades imaginarias.

Un cuaternión es un objeto cuatridimensional y no pude ser representado en un espacio tridimensional. Hamilton quería generalizar a los complejos en tres dimensiones por lo que al principio que un cuaternión sea cuatridimensional puede parecer un problema. Sin embargo, Hamilton se dio cuenta de que existían ventajas en el hecho de que existiera una parte escalar a parte de la parte vectorial.

2.9.1. Propiedades de los cuaterniones

Un cuaternión sea real o complejo se puede representar como la suma de un escalar y un vector[1][7][8]:

$$\boldsymbol{q} = q_0 + \vec{q} = Sc(q) + Vec(q)$$

donde $Sc(q) = q_0$ y $Vec(q) = \vec{q} = \sum_{j=0}^{3} q_j i_j$

Los cuaterniones poseen ciertas propiedades utilizadas al realizar operaciones entre dos o más cuaterniones, algunas propiedades de los cuaterniones son:

- Norma de los cuaterniones: $\boldsymbol{|q|} = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} = \sqrt{q\bar{q}}$
- Conjugado cuaternionico: $\bar{\boldsymbol{q}} = q_0 - q_1 i_1 - q_2 i_2 - q_3 i_3 = q_0 - q$
- Inverso cuaternionico: $\boldsymbol{q}^{-1} = \frac{\bar{q}}{|q|^2}$
- Producto de unidades imaginarias: $i_1 i_2 i_3 = i_1^2 = i_2^2 = i_3^2 = -1$
- Cuaternión puro: $\boldsymbol{q} = q_1 i_1 + q_2 i_2 + q_3 i_3$
- Cuaternión real: $\boldsymbol{q} = q_0$

2.9.2. Operaciones con cuaterniones

Existen distintas operaciones que se pueden realizar entre dos o más cuaterniones:

- Suma o resta: $q \pm p = (q_0 \pm p_0) + \sum_{k=1}^{n} (qk \pm pk) ik$

  La suma de cuaterniones es asociativa, el 0 es un elemento neutro en la suma de

cuaterniones. Para todo cuaternión q existe un cuaternión opuesto -q y la suma de cuaterniones es conmutativa.

- Producto: $qp = q_o p_0 + p_o\vec{q} + q_0\vec{p} - \vec{q}\cdot\vec{p} + \vec{q}\times\vec{p}$

  El producto entre cuaterniones es asociativo, distributivo, existe un elemento neutro que es el 1, para todo cuaternión $q \neq 0$ existe un cuaternión inverso denotado como q^{-1} y que multiplicado por q se obtiene 1.

  Es posible representar el producto de cuaterniones de manera matricial. Si se considera que los cuaterniones $p, q \in \mathbb{H}(\mathbb{R})$ y suponiendo que el producto de dichos cuaterniones sea s donde $s = s_0 + s_1 i_1 + s_2 i_2 + s_3 i_3$ esto es $pq = s$ por lo tanto:

$$
\begin{aligned}
s_0 &= (p_0 q_0 - p_1 q_1 - p_2 q_2 - p_3 q_3) \\
s_1 &= (p_1 q_0 + p_0 q_1 - p_3 q_2 + p_2 q_3) \\
s_2 &= (p_2 q_0 + p_3 q_1 + p_0 q_2 - p_1 q_3) \\
s_3 &= (p_3 q_0 - p_2 q_1 + p_1 q_2 + p_0 q_3)
\end{aligned}
$$

Lo anterior puede ser representado de forma matricial:

$$
\begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} = \begin{bmatrix} p_0 & -p_1 & -p_2 & -p_3 \\ p_1 & p_0 & -p_3 & p_2 \\ p_2 & p_3 & p_0 & -p_1 \\ p_3 & -p_2 & p_1 & p_0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix},
$$

La matriz cuadrada de los coeficientes de p es conocida como $B_l(p)$, esto es:

$$
B_l(p) = \begin{bmatrix} p_0 & -p_1 & -p_2 & -p_3 \\ p_1 & p_0 & -p_3 & p_2 \\ p_2 & p_3 & p_0 & -p_1 \\ p_3 & -p_2 & p_1 & p_0 \end{bmatrix}
$$

para el producto $pq = t$ el resultado es la matriz $B_r(p)$:

$$
B_r(p) = \begin{bmatrix} p_0 & -p_1 & -p_2 & -p_3 \\ p_1 & p_0 & p_3 & -p_2 \\ p_2 & -p_3 & p_0 & p_1 \\ p_3 & p_2 & -p_1 & p_0 \end{bmatrix}.
$$

Si se siguen las reglas del álgebra de matrices, se obtiene la matriz opuesta de $B_r(p)$:

$$B_r^T(p) = \begin{bmatrix} p_0 & p_1 & p_2 & p_3 \\ -p_1 & p_0 & p_3 & -p_2 \\ -p_2 & -p_3 & p_0 & p_1 \\ -p_3 & p_2 & -p_1 & p_0 \end{bmatrix}.$$

- División: $qp^{-1} = \frac{p\bar{q}}{|q|^2}$
- Producto vectorial: $\vec{q} \times \vec{p} = \begin{vmatrix} i_1 & i_2 & i_3 \\ q_1 & q_2 & q_3 \\ p_1 & p_2 & p_3 \end{vmatrix} = (q_2p_3 - q_3p_2)\, i_1 - (q_1p_3 - q_3p_1)\, i_2 + (q_1p_2 - q_2p_1)\, i_3$.
- Producto escalar: $p \bullet q = p_0q_0 + p_1q_1 + p_2q_2 + p_3q_3$.
- Cuadrado de un cuaternión: $q^2 = (q_0^2 - q_1^2 - q_2^2 - q_3^2, 2q_0(q_1i_1 + q_2i_2 + q_3i_3))$.

2.9.3. Rotación cuaterniónica

Considerando p y q cuaterniones con $p_0 = 0$ y q unitario, haciendo el triple producto $q\vec{p}\bar{q}$:

$$q\vec{p}\bar{q} = (q_0 + \vec{q})\,\vec{p}(q_0 - \vec{q}) = (q_0 + \vec{q})(q_0\vec{p} - \vec{p}\vec{q})$$

$$= q_0^2\vec{p} + q_0\vec{p}\vec{q} - q_0\vec{p}\vec{q} - \vec{q}\vec{p}\vec{q}$$

$$= q_0^2\vec{p} + q_0\,(- \langle\vec{q},\vec{p}\rangle + [\vec{q} \times \vec{p}]) - q_0\,(- \langle\vec{p},\vec{q}\rangle + [\vec{p} \times \vec{q}]) - \vec{q}\vec{p}\vec{q}$$

como $\langle\vec{p},\vec{q}\rangle = \langle\vec{q},\vec{p}\rangle$ y $[\vec{p} \times \vec{q}] = - [\vec{q} \times \vec{p}]$, entonces:

$$q\vec{p}\bar{q} = q_0^2\vec{p} + 2q_0[\vec{q} \times \vec{p}] + \vec{q}\langle\vec{p},\vec{q}\rangle - \vec{q}\,[\vec{p} \times \vec{q}]$$

debido a que:

$$\vec{q}\,[\vec{p} \times \vec{q}] = - \langle\vec{q}, \vec{p} \times \vec{q}\rangle + [\vec{q} \times \vec{p} \times \vec{q}] = 0$$

por lo tanto,

$$q\vec{p}\bar{q} = q_0^2\vec{p} + 2q_0[\vec{q} \times \vec{p}] + \vec{q}\langle\vec{p} \bullet \vec{q}\rangle - \vec{p}\,\langle\vec{q},\vec{q}\rangle + \vec{q}\langle\vec{q},\vec{p}\rangle$$

entonces,

$$\vec{r} = q\vec{p}\bar{q} = (q_0^2 - |q|^2)\,\vec{p} + 2q_0[\vec{q} \times \vec{p}] + 2\vec{q}\langle\vec{p},\vec{q}\rangle.$$

Como se observa el resultado del triple producto de dos cuaterniones por un vector es

otro vector; si el cuaternión q es unitario entonces la norma del producto es la norma del vector $\vec{p}$. Esto es:

$$|\vec{r}| = |q\vec{p}\bar{q}| = |q||\vec{p}||\bar{q}| = |\vec{p}|$$

también se puede representar a $\vec{r}$ como:

$$r = q\vec{p}\bar{q} = (q_0^2 - |q|^2) + 2q_0[\vec{q} \times \vec{p}] + 2\vec{q}\langle\vec{p}, \vec{q}\rangle$$

$$= (2q_0^2 - q_0^2 - |q|^2)\,\vec{p} + 2q_0[\vec{q} \times \vec{p}] + 2\vec{q}\langle\vec{p}, \vec{q}\rangle$$

$$= (2q_0^2 - |q|^2)\,\vec{p} + 2q_0[\vec{q} \times \vec{p}] + 2\vec{q}\langle\vec{p}, \vec{q}\rangle$$

$$= (2q_0^2 - 1)\,\vec{p} + 2q_0[\vec{q} \times \vec{p}] + 2\vec{q}\langle\vec{p}, \vec{q}\rangle$$

la última expresión conformada por vectores distintos se puede escribir en forma matricial de la siguiente manera [7][8][26]:

$$(2q_0^2 - 1)\,\vec{p} = \begin{bmatrix} (2q_0^2 - 1)\,\vec{p} & 0 & 0 \\ 0 & (2q_0^2 - 1)\,\vec{p} & 0 \\ 0 & 0 & (2q_0^2 - 1)\,\vec{p} \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix}$$

$$2\vec{q}\langle\vec{p}, \vec{q}\rangle = \begin{bmatrix} 2q_1^2 & 2q_1q_2 & 2q_1q_3 \\ 2q_1q_2 & 2q_2^2 & 2q_2q_3 \\ 2q_1q_3 & 2q_2q_3 & 2q_3^2 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix}$$

$$2q_0[\vec{q} \times \vec{p}] = \begin{bmatrix} 0 & -2q_0q_3 & 2q_0q_2 \\ 2q_0q_3 & 0 & -2q_1q_1 \\ -2q_0q_2 & 2q_0q_1 & 0 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix}$$

sumando los tres vectores:

$$\begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \begin{bmatrix} R_1 \\ R_2 \\ R_3 \end{bmatrix},$$

entonces, se puede decir que el vector $\vec{p}$ se ha rotado hacia el vector $\vec{r}$ y que la matriz cuaterniónica de rotación es la matriz R [7][8],

$$R = \begin{bmatrix} (2q_0^2 - 1) + 2q_1^2 & 2q_1q_2 - 2q_0q_3 & 2q_1q_3 + 2q_0q_2 \\ 2q_0q_3 + 2q_1q_2 & (2q_0^2 - 1) + 2q_2^2 & 2q_2q_3 - 2q_1q_1 \\ 2q_1q_3 - 2q_0q_2 & 2q_2q_3 + 2q_0q_1 & (2q_0^2 - 1) + 2q_3^2 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} \qquad (5)$$

Para descifrar la información después de utilizar este método es necesario obtener la matriz inversa de rotación. La matriz de rotación inversa se obtiene de la siguiente manera:

$$R(q)^{-1} = \frac{1}{\det(R(q))}\, adj(R(q)) \tag{6}$$

donde:

$\det(R(q))$ es el determinante de la matriz *R(q)*

$adj(R(q)) = cof(R(q))^T$ es la adjunta de la matriz o de la matriz transpuesta de la matriz de cofactores de *R(q).*

Por lo que la matriz *R* se puede recuperar mediante la operación:

$$P = R(q)^{-1} x\, P'.$$

2.10. Cifrado asimétrico RSA

Un algoritmo asimétrico emplea claves secretas con longitudes mucho mayores a los sistemas simétricos. Los sistemas basados en llave pública son relativamente recientes ya que los primeros algoritmos desarrollados bajo este estándar empezaron a aparecer en 1975.

El algoritmo RSA fue presentado en 1978 por Rivest, Shamir y Adlman, y actualmente es el criptosistema más utilizado para cifrar información. Este sistema se basa en el hecho matemático de la dificultad de factorizar números muy grandes. El proceso para factorizar un número consiste en empezar a dividir sucesivamente éste entre 2, entre 3, entre 4, y así sucesivamente buscando que el resultado de la división sea exacto, es decir, dé resto 0, con lo que se obtiene un divisor del número. Si el número considerado es un número primo, para factorizarlo habría que empezar por 1, 2, 3, hasta llegar a él mismo, ya que por ser primo ninguno de los números anteriores es divisor suyo. Y si el número primo es lo suficientemente grande, el proceso de factorización es complicado y lleva mucho tiempo [19].

El sistema RSA crea sus claves basado en la exponenciación modular de exponente y

módulo fijos de la siguiente forma:

- Se buscan dos números primos lo suficientemente grandes: p y q (de entre 100 y 300 dígitos).
- Se obtienen los números n = p * q y Ø = (p-1) * (q-1).
- Se busca un número *e* tal que no tenga múltiplos comunes con Ø.
- Se calcula d = e-1 mod Ø, con mod = resto de la división de números enteros.
- Y ya con estos números obtenidos, n es la clave pública y d es la clave privada. Los números p, q y Ø se destruyen. También se hace público el número e, necesario para alimentar el algoritmo.

El cálculo de estas claves se realiza en secreto por el usuario que tiene la clave privada, y una vez generada ésta conviene protegerla mediante un algoritmo criptográfico simétrico.

En cuanto a las longitudes de claves; el sistema RSA permite longitudes variables, siendo aconsejable actualmente el uso de claves de no menos de 1024 bits (se han roto claves de hasta 512 bits, aunque se necesitaron más de 5 meses y casi 300 ordenadores trabajando juntos para hacerlo).

La seguridad del sistema RSA se basa en la longitud de su clave y el número de operaciones que se necesitan realizar si no se conoce la clave correcta, ya que, si bien realizar la exponenciación modular es fácil, su operación inversa la extracción de raíces de módulo Ø no es factible a menos que se conozca la factorización de e, clave privada del sistema.

El sistema de cifrado RSA es el más rápido, conocido y usado de los sistemas de clave pública. Presenta todas las ventajas de los sistemas asimétricos, incluyendo la firma digital, aunque resulta más útil a la hora de implementar la confidencialidad el uso de sistemas simétricos, por ser más rápidos. Se suele usar también en los sistemas mixtos para encriptar y enviar la clave simétrica que se usará posteriormente en la comunicación cifrada.

A continuación, se muestra un ejemplo simple del funcionamiento del cifrado RSA.

Para este ejemplo p = 3 y q = 11, dando n = 11 y z = 20. Un valor adecuado de d es d = 7, puesto que 7 y 20 no tienen factores comunes.

Con estas selecciones, *e* puede encontrarse resolviendo la ecuación 7*e*=1(mod 20), que produce *e*=3. El texto cifrado C de un mensaje de texto normal P se da por la regla C=P3(mod 33). El texto cifrado lo descifra el receptor de acuerdo con la regla P= C7(mod 33).

2.11. Coeficiente de correlación

El coeficiente de correlación tiene como finalidad examinar la fuerza de la asociación entre dos variables cuantitativas. Así se conocerá la intensidad de la relación entre ellas. Con el coeficiente de correlación será posible determinar qué tan eficiente es un sistema criptográfico.

Existen dos coeficientes de correlación que se usan frecuentemente: el de Pearson y el de Spearman. El coeficiente de correlación de Pearson evalúa específicamente la adecuación a la recta lineal que defina la relación entre dos variables cuantitativas. El coeficiente no paramétrico de Spearman mide cualquier tipo de asociación, no necesariamente lineal.

2.12. Covarianza

El numerador del coeficiente de correlación es la covarianza muestral s_{XY} entre *X* e *Y*, que nos indica si la posible relación entre dos variables es directa o inversa. Es una medida que nos habla de la variabilidad conjunta de dos variables cuantitativas se tiene que

$$s_{XY} = \frac{1}{n}\sum_i (x_i - \bar{x})\,(y_i - \bar{y})$$

si los valores altos (o bajos) de *X* tienden a asociarse con valores altos (o bajos) de *Y*, el producto de las desviaciones tenderá a ser positivo y la covarianza será positiva. Por el contrario, si valores altos de una variable se relacionan con valores bajos de la otra variable, el producto de las desviaciones tenderá a ser negativo y la covarianza será

negativa.

De tal modo que:

- Si s_{XY} >0 las dos variables crecen o decrecen a la vez (nube de puntos creciente).
- Si s_{XY} <0 cuando una variable crece, la otra tiene tendencia a decrecer (nube de puntos decreciente).
- Si los puntos se reparten con igual densidad alrededor del centro de gravedad, s_{XY} = 0 (no hay relación lineal).

El signo de la covarianza indica si el aspecto de la nube de puntos es creciente o no, pero no muestra nada sobre el grado de relación entre las variables.

Capítulo 3

Sistemas criptográficos

3.1. Algoritmos de cifrado simétrico

La criptografía de llave simétrica o también llamada criptografía de llave privada, consiste en que sólo dos usuarios tienen la clave secreta única necesaria para cifrar o descifrar la información que será enviada a través de un canal inseguro.

Un sistema de cifrado seguro pone toda la seguridad en la clave y ninguna en el algoritmo. Así no importa si el atacante conoce el algoritmo; no sería posible para él descifrar el mensaje si no conoce la llave que se utilizó para ocultar el mensaje [9][19].

3.1.1. Algoritmo DES

El DES por sus siglas en inglés *Data Encription Standard* o Estándar de Encriptación de Datos es un algoritmo escogido como estándar FIPS en Estados Unidos; fue presentado en 1976 y aunque es el algoritmo de cifrado simétrico mayormente estudiado, mejor conocido y más empleado en el mundo, actualmente este algoritmo es considerado inseguro y no debe utilizarse. En su lugar se utiliza otra versión de este cifrado llamado 3DES que consiste en aplicar el algoritmo DES tres veces [27].

El algoritmo DES cifra por bloques de 64 bits. Se toma un texto plano de la longitud determinada de bits y mediante una serie de operaciones el texto plano se transforma en un texto cifrado de la misma longitud. La longitud de la llave o clave del sistema de cifrado es también de 64 bits. No permite variar la longitud de la clave pues al tener una clave de 64 bits el sistema puede ser quebrado utilizando criptoanálisis diferencial o un ataque de fuerza bruta, por lo que actualmente este sistema ha sido descartado como un estándar.

Se ha propagado ampliamente por el mundo el uso del algoritmo DES y aunque fue quebrado en 1999, se creó una variante llamada 3DES que tiene una longitud de llave de 168 bits lo cual lo hace apto para ser empleado en sistemas de seguridad.

Funcionamiento del algoritmo DES:

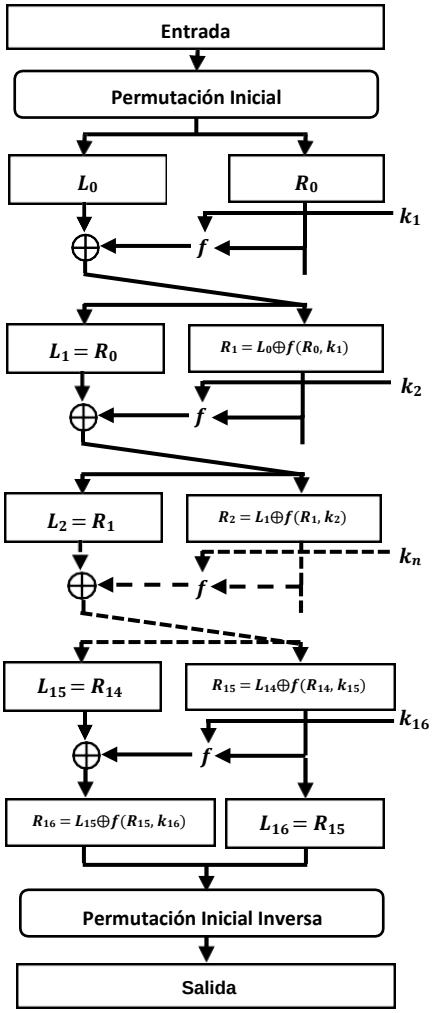


Figura 13. Esquema de cifrado DES

1. El texto plano recibe una primera permutación inicial conocida como *IP* (*Inicial Permutation*).
2. El resultado de la permutación inicial es separado en dos bloques de 32 bits denominados L_0 y R_0.
3. Se aplica la función $f(R_0, K_1)$ mediante la cual el bloque R_0 es expandido debido a un bloque de 48 bits denominado *E*. Posteriormente se realiza la operación XOR bit a bit con K_1 como llave. La obtención de estas llaves se expondrá más adelante.

4. Al obtener $E(R_0) \oplus K_1$ se utilizan las cajas *S* para obtener 32 bits de salida a partir de la entrada de 48 bits. Se tienen ocho entradas de seis bits y se utilizan como se observa en el siguiente diagrama:

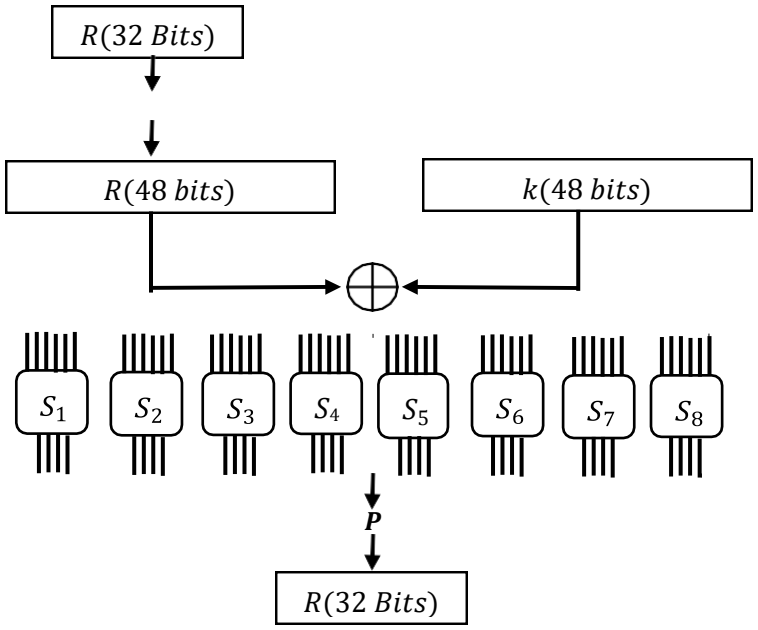


Figura 14. Diagrama de función *f(R,K)*

5. Se realiza una permutación *P* a los 32 bits de las cajas *S*.
6. Al resultado obtenido a la salida de $f(R_0,K_1)$ se le realiza una operación XOR bit a bit con el bloque L_0 y la salida es la entrada R_1 de la siguiente ronda.
7. El bloque R_0 pasa a convertirse en L_0.
8. El proceso descrito se realiza durante 16 rondas, en la última ronda no se intercambian los bloques L_{16} y R_{16}.
9. Se realiza una permutación final IP^{-1} que es la opuesta a la permutación inicial *IP*.

Para la generación de llaves K_n se sigue el esquema de la figura n. Se realiza una primera permutación *PC1* a los 56 bits de la llave, después se separa la cadena de bits obtenida en dos bloques C_0 y D_0, las cadenas se recorren cierto número de bits hacia la izquierda de acuerdo con la tabla de corrimientos. Las salidas K_n se obtienen mediante la salida de cada corrimiento de bits y realizando la permutación $PC2$ que elimina 8 bits entre las cadenas C_0 y D_0 para obtener sub-llaves K_n con longitudes de 48 bits cada una.

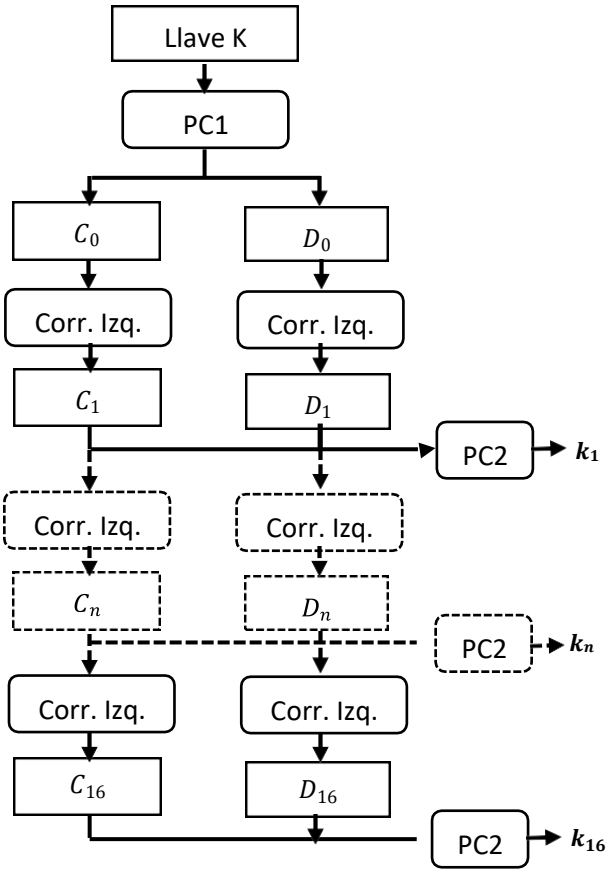


Figura 15. Esquema par obtener las llaves $\mathbf{K_n}$

Las distintas tablas de permutaciones, las tablas de corrimientos para sub-llaves así como las cajas *S* y su funcionamiento se pueden observar en el anexo.

3.1.2. Algoritmo AES

El estándar AES es un algoritmo de cifrado por bloques es la opción para sustituir al estándar DES. El tamaño de los bloques empleados por AES es de 128 bits y tiene claves de longitudes variables; existe AES de 128 bits, de 192 y de 256 bits. Para AES de 128 bits el sistema tiene 10 rondas para llaves, 12 rondas para 192 y 12 rondas de llaves para 25 bits. El cifrado AES puede ser implementado tanto en *hardware* como en *software*.

AES tiene distintas formas de gestionar los bloques de mensaje que se encriptarán. A continuación se explican algunas formas de estos modos de cifrado [28]:

- CBC: A cada bloque de texto se le aplica la operación XOR con el bloque cifrado anterior antes de ser cifrado. Así cada bloque de texto cifrado depende de todo el texto plano procesado hasta ese punto. La desventaja de este modo es que ya que es secuencial no puede ser paralelizado.
- OFB: Se generan bloques de flujo de calves a los cuales se les aplica la operación XOR bit a bit con el texto claro para obtener texto cifrado.
- CFB: Se hace igual que OFB pero para producir el flujo de claves se utiliza el último bloque de cifrado en lugar del último bloque del flujo de llaves como lo hace OFB.

3.2 Algoritmos de cifrado asimétrico

Los sistemas de cifrado asimétrico o de llave pública fueron creados para solucionar el problema de distribución de llaves. En estos sistemas cada persona tiene dos claves, una de estas claves es pública, lo cual significa que puede ser vista por cualquier persona, y la otra clave debe mantenerse en secreto para que nadie tenga acceso a ella. Cuando se quiere enviar información a una persona el remitente debe cifrar la información con la clave pública del destinatario, una vez cifrada la información sólo la clave privada del destinatario podrá descifrar el mensaje.

3.2.1. Algoritmo RSA

El algoritmo de cifrado RSA llamado así en honor a sus creadores Rivest, Shamir y Adelman, es el algoritmo asimétrico de mayor popularidad, es utilizado para encriptar comunicaciones, firmas digitales e intercambiar claves [29].

En los sistemas RSA el tamaño de la calve varía entre 512 y 2048 bits. Como es de esperarse entre mayor sea la clave mayor la seguridad del sistema aumenta, pero la eficiencia de este disminuye y se genera más texto cifrado. Los bloques de texto deben tener un tamaño menor al de la longitud de la clave.

La clave pública tiene la forma (*e*, *n*), donde *e* es el exponente y *n* el módulo. La longitud de la clave es igual al número de bits de *n*. El módulo se obtiene al multiplicar dos números primos grandes, *p* y *q*; los números se seleccionan de forma aleatoria y se mantienen ocultos. La clave privada tiene la forma (*d*, *n*), donde *d* es el producto inverso de *e* módulo (*p-1*) (*q-1*).

El cálculo *d* a partir de *p* y *q* es sencillo, pero es computacionalmente imposible calcular *d* sin conocer *p* y *q* para valores grandes de *n*, ya que obtener sus valores es equivalente a factorizar *n*, que es un problema computacionalmente imposible.

El algoritmo funciona de la siguiente manera:

- Cifrado: Un mensaje se cifra haciendo el cálculo de $c = m^e \ modulo \ n$, donde *m* es el texto plano, *c* es el texto cifrado y (*e*, *n*) es la calve pública del destinatario.
- Descifrado: El mensaje se descifra calculando $c^d \ modulo \ n = (m^e)^d \ modulo \ n = m^{ed} \ modulo \ n = m$ donde (*d*, *n*) es la clave privada del destinario. La última sustitución es posible por el modo en que los números se han elegido debido a que *d* es el producto inverso de *e modulo n*, por lo que $m^{ed} = m$.
- Firma digital: Si el emisor desea enviar el mensaje firmado usa su clave privada para calcular $c = m^d \ modulo \ n$ y el destinatario lo valida calculando $c^e \ modulo \ n = (m^d)^e \ modulo \ n = m^{de} \ modulo \ n = m$ donde (*e*, *n*) es la calve pública del emisor.

El algoritmo emplea operaciones matemáticas complejas que tienen un coste computacional elevado y trabaja con claves de gran tamaño, por lo que el algoritmo puede llegar a ser lento. Este problema se puede solucionar utilizando exponentes fijos lo cual hace que el proceso de cifrado y la verificación de firma se realicen de forma más rápida.

Si se compara a los sistemas de cifrado DES y RSA, el algoritmo RSA es 100 veces más lento en *software* y de 1000 a 10000 veces más lento en *hardware*.

3.3 Algoritmos de cifrado que utilizan funciones pseudoanalíticas y rotación cuaterniónica

3.3.1. Construcción de un nuevo sistema criptográfico empelando la teoría de funciones pseudoanalíticas

El trabajo realizado por A. Bucio R., A. Hernández Becerril, A. Robles G., P. Ramírez T. y A. Arista Jalife, presentado en el año 2013 [1]; propone el uso de funciones pseudoanalíticas para resolver el problema de la impedancia de la tomografía eléctrica. Debido a que el problema de la impedancia de la tomografía eléctrica no puede ser resuelto con métodos matemáticos tradicionales, la aplicación de las potencias formales de parámetro espectral para resolver este problema puede resultar en el desarrollo de un sistema de cifrado que proveerá de una gran seguridad ya que el intentar descifrar dicho sistema significará solucionar un problema que no puede ser resuelto si no empleando las potencias formales de parámetro espectral.

Para construir el sistema criptográfico se considera una serie de puntos localizados sobre varios radios *R* de un círculo de radio unitario con centro $z_0 = 0$, y con un ángulo θ tal que:

$$x|q| = r|q|cos\theta \qquad y|q| = r|q|sen\theta$$

En donde $r|q|$ son puntos localizados sobre un radio, trazados equidistantemente uno de otro. Entonces los elementos del arreglo $Z^{(n)}|q|$ que corresponden a la aproximación numérica de la potencia formal $Z^{(n)}$ (1, 0; z) se define como:

$$Z^{(0)}|q| = \sqrt{\sigma(x|q|, y|q|)}$$

los siguientes elementos del arreglo $Z^{(n)}|q|$ se calculan utilizando el método de integración trapezoidal. Los cálculos se denotan como:

$$Z^{(n)}|q| = \beta[Z^{(n-1)}[q]]$$

donde $n = 1, 2, \ldots N$.

El proceso se realiza para distintos ángulos θ en los que el círculo sea dividido. Se considera un número de ángulos *S* tal que:

$$\left\{\theta\,|s| = \frac{2\pi s}{S}\right\}$$

se consideran *S* ángulos de los cuales se obtendrán *N* potencias formales a partir de cada punto *Q* considerado.

Al obtener los distintos valores de potencias de cada ángulo, el resultado es un arreglo de vectores independientes a los que se les puede aplicar el proceso de ortonormalización Gram-Schmidt con lo que se obtiene una matriz de vectores ortonormalizados:

$$U_{[N, S]}$$

El proceso de cifrado del sistema presentado en [1] consiste en realizar la multiplicación de la matriz ortonormalizada por la matriz de datos que se quieren cifrar. Una vez obtenida esta matriz de datos cifrados el proceso de cifrado consiste en multiplicar la matriz de datos cifrados por la matriz inversa $U^{-1}_{[N, S]}$. Así se obtienen los datos originales de la información ha sido transmitida.

3.3.2. Un nuevo sistema de clave publica cuadripartita

El trabajo [3] presentado por los matemáticos Tomoyuki Nagase, Ryusuke Koide, Takashi Araki y Yoshiei Hasegawa de la universidad de Hirosaki en Japón, desarrollado en 2004, plantea el uso de cuaterniones para crear un sistema de cifrado que utiliza cuatro llaves para encriptar.

El esquema de cifrado consiste en separar señales de voz en bloques de muestras de matrices de 3x3. Los bloques de información son cifrados al ser rotados con la matriz de rotación cuaterniónica y utilizar las cuatro llaves de encriptación para realizar dicha rotación.

Las llaves de cifrado son cuatro cuaterniones *q* están formados por números enteros aleatorios, en este ejemplo el articulo utiliza valores como: *q* = (0, 20, 40, 30) y la señal es mostrada en 16 bits.

El algoritmo de cifrado comienza al construir la matriz de rotación cuaterniónica a través del primer cuaternión *q* de orden 0 de modo que

$$R(q) = \begin{bmatrix} Q_{00} & Q_{01} & Q_{02} \\ Q_{10} & Q_{11} & Q_{12} \\ Q_{20} & Q_{21} & Q_{22} \end{bmatrix}$$

Una vez hecha la matriz inicial, lo cuaterniones de orden *n* son generados utilizados las columnas de la matriz *R(q)*:

$$q_{11} = (0, Q_{00}, Q_{01}, Q_{02})$$

$$q_{12} = (0, Q_{01}, Q_{11}, Q_{21})$$

$$q_{13} = (0, Q_{02}, Q_{12}, Q_{22})$$

Los cuaterniones anteriores generan a su vez sus propias matrices de rotación $R(q_{11})$, $R(q_{12})$ y $R(q_{13})$, que rotarán los datos que se van a encriptar mediante la expresión(). Si se desea se pueden seguir obteniendo cuaterniones con sus respectivas matrices de rotación a partir de las matrices *R(q)*.

Para realizar el proceso de descifrado se debe encontrar la matriz inversa de cada matriz de rotación *R(q)* y utilizar la ecuación (5) para poder recuperar la información original a partir de los datos cifrados.

Este algoritmo de cifrado puede presentar algunas ventajas como:

- Un cifrado más rápido; ya que sólo se debe realizar la multiplicación de la matriz de rotación por los bloques de la información que se quiere cifrar.
- Se pueden generar *n* matrices múltiples a partir de un cuaternión inicial *q*.

El algoritmo también puede presentar problemas como el de la longitud de la llave ya que esa depende de los coeficientes del cuaternión, debido a que este tamaño varia ente 8 o 16 bits, un ataque de fuerza bruta podría quebrar el criptosistema.

3.3.3. Un nuevo método basado en cuaterniones para encriptar imágenes DICOM

Este trabajo presentado en 2015 por los autores Mariusz Dzwonkowski, Michal Papaj y Roman Rykaczewski; propone un algoritmo de cifrado de las imágenes digitales y comunicaciones de medicina (DICOM) utilizando la rotación cuaterniónica (4). Principalmente este esquema de cifrado está diseñado para cifrar imágenes, sin embargo, el artículo también menciona que dicho algoritmo puede ser empleado para cifrar información textual.

Para el proceso de encriptado de una imagen utilizando este sistema primero se debe descomponer la imagen en dos imágenes de 8 bits en escala de grises. Las dos imágenes son la información de entrada para comenzar el cifrado del algoritmo. Cada imagen será encriptada de manera individual. Las imágenes son pasadas a escalas de grises debido a que el algoritmo tiene ciertas limitaciones.

Para realizar el proceso de cifrado y descifrado se sigue el proceso mostrado en las siguientes figuras que fueron obtenidas del artículo[4].

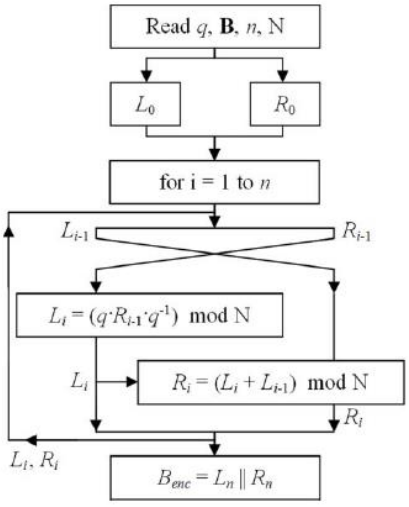


Figura 16. Proceso de cifrado

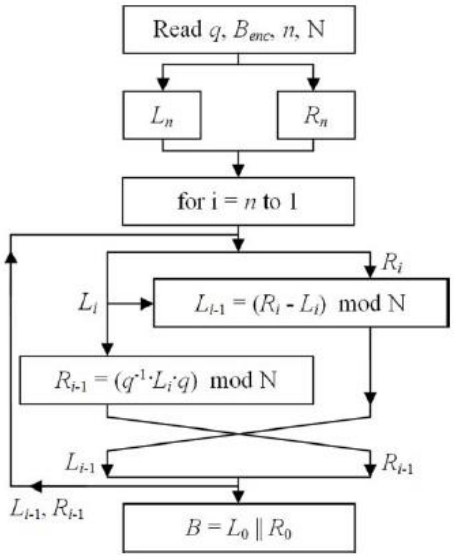


Figura 17. Proceso de descifrado

Las operaciones fundamentales para el proceso de cifrado y descifrado del algoritmo sustituyen la multiplicación de matrices por una multiplicación de cuaterniones:

$$L_i = (q \bullet R_{i-1} \bullet q^{-1}) \bmod N \qquad R_i = (L_{i-1} + L_i) \bmod n$$

para $i = 1$ a n

$$R_{i-1} = (q^{-1} \bullet L_i \bullet q) \bmod N \; L_{i-1} = (R_i - L_i) \bmod N$$

para $i = n$ a 1

El valor de n indica el número de rondas que tendrá el cifrado. Cada ronda es encriptada con un cuaternión diferente q_i $i = 1, 2, \ldots, n$. Estos cuaterniones que funcionan como las llaves únicas del sistema de cifrado son creadas por la matriz de rotación. Para realizar el cifrado y descifrado de cada imagen de 8 bits es necesario obtener una llave cuaterniónica única.

3.3.4. Proceso de cifrado y descifrado utilizando cuaterniones y series de Farey para transmisión segura

Esquema de cifrado propuesto por los autores Vijay Sankar y Arul Lawrance. Este algorimto hace uso de una serie de Farey para obtener los coeficientes de un cuaternión, el cual consiste

en fracciones con valores que oscilan entre 0 y 1 y son ordenadas de manera ascendente. Las fracciones son generadas de acuerdo con los siguientes pasos:

1. Primero se decidirá qué orden tendrá la serie y se denota con *n*.
2. Se obtendrán todas las combinaciones de fracciones posibles tanto para el nominador como para el denominador usando 1, 2, 3…,*n*. El número total de combinaciones es n^2.
3. Si las fracciones obtenidas tienen un valor superior a 1, serán descartadas.
4. Todas las combinaciones obtenidas que tengan los mismos valores serán eliminadas y sólo la que tenga los valores más simples será tomada en cuenta.
5. Las fracciones que sean útiles para el algoritmo son ordenadas de manera ascendente.

El algoritmo utiliza cuatro elementos de la serie de Faray como coeficientes del cuaternión primario *q*.

Para realizar el proceso de cifrado se necesita calcular la matriz de rotación *R(q)* y a continuación se generan 12 cuaterniones secundarios con sus correspondientes matrices de rotación $R(q_{nm})$.

Sea:

$$R(q) = = \begin{bmatrix} Q_{00} & Q_{01} & Q_{02} \\ Q_{10} & Q_{11} & Q_{12} \\ Q_{20} & Q_{21} & Q_{22} \end{bmatrix}$$

Entonces,

$$q_{01} = (0, Q_{00}, Q_{01}, Q_{02})$$

$$q_{02} = (0, Q_{10}, Q_{11}, Q_{12})$$

$$q_{03} = (0, Q_{20}, Q_{21}, Q_{22})$$

$$q_{11} = (Q_{00}, 0, Q_{01}, Q_{02})$$

$$q_{12} = (Q_{10}, 0, Q_{11}, Q_{12})$$

$$q_{13} = (Q_{20}, 0, Q_{21}, Q_{22})$$

$$q_{21} = (Q_{00}, Q_{01}, 0, Q_{02})$$

$$q_{22} = (Q_{10}, Q_{11}, 0, Q_{12})$$

$$q_{23} = (Q_{20}, Q_{21}, 0, Q_{22})$$

$$q_{31} = (Q_{00}, Q_{01}, Q_{02}, 0)$$

$$q_{32} = (Q_{10}, Q_{11}, Q_{12}, 0)$$

$$q_{33} = (Q_{20}, Q_{21}, Q_{22}, 0)$$

Las matrices de rotación son generadas a partir de los 12 cuaterniones generados.

La información que va a ser cifrada es separada y acomodada en matrices de 3x3 y cada una de estas matrices es multiplicada por las doce matrices $R(q_{nm})$ de acuerdo con la ecuación (4). La salida de la primera multiplicación de matrices es la información de entrada para realizar el segundo producto de matrices y así sucesivamente hasta multiplicar cada bloque por las doce matrices de rotación, el resultado de la última multiplicación será el texto cifrado.

Para el proceso de descifrado basta con calcular las matrices inversas de $R(q_{nm})$ y multiplicar los bloques de la información cifrada por estas matrices inversas, pero ahora en el orden opuesto al utilizado para realizar el cifrado, es decir comenzar por la matriz número doce hasta la matriz número uno.

3.3.5. Método de cifrado con cuaterniones para imágenes y video

Este trabajo elaborado por Mariusz Dzwonkowski y Roman Rykacsewski, utiliza dos matrices para ofrecer una mayor aleatoriedad y por lo mismo una mayor seguridad en el cifrado hecho.

Se crean dos matrices de 3x3 conocidas como contador *Ctr* y matriz de inicialización *IM* cada elemento de las matrices es generado de manera aleatoria en un rango que va de 0 a 255. Los elementos de las matrices son sumados elemento a elemento y luego se les aplica módulo 256 debido a que este es el rango de una variable de 8 bits.

$$c_{modfc} = (Ctr_{fc} + IM_{fc}) \bmod (256)$$

donde:

f equivale a la fila y *c* la columna de la matriz C_{mod}, *Ctr* y la matriz *IM*.

La matriz C_{mod} es transformada en un cuaternión c_{mod}, que a su vez es un cuaternión puro c'_{mod} que es empleado para calcular una matriz C'_{mod} de 3x3.

El modo en que el algoritmo cifra la información es separándola en matrices de 3x3 elementos, por lo que existen *n* bloques de elementos a cifrar. Cada elemento de cada matriz representa un píxel de 8 bits cuyo valor está en el intervalo que va de 0 a 255.

Con la matriz C'_{mod} y con las matrices de datos B_n, el cifrado de la primera matriz B_0 se realiza mediante una operación XOR $\oplus$ elemento a elemento entre las matrices, esto es $B'_0 = C'_{mod} \oplus B_0$.

Para continuar con el cifrado a la matriz *Ctr* se le aumenta en 1 el valor de cada uno de sus elementos y después se le aplica módulo 256 antes de realizar la suma con la matriz *IM*, de esta operación se obtiene una matriz C'_{mod1} que es diferente a la matriz utilizada en el primer bloque. El proceso se realiza hasta cifrar todos los bloques *n* de información.

La siguiente figura extraída del artículo[5] muestra el diagrama del cifrado descrito.

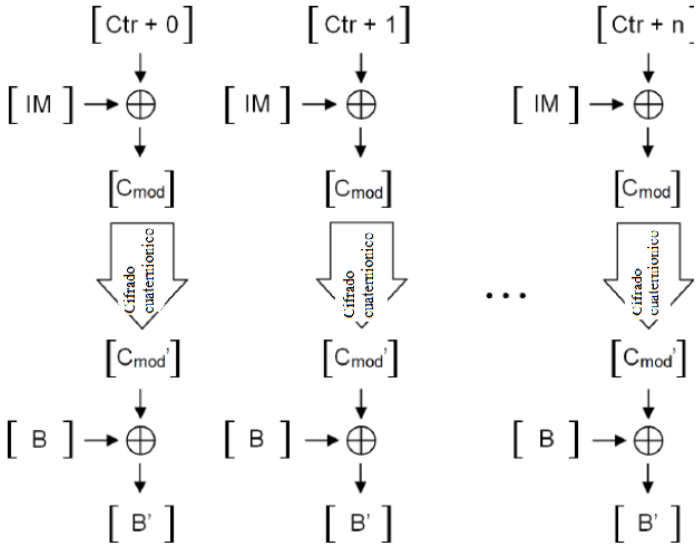


Figura 18. Cifrado de cuaterniones basado en Ctr

3.3.6. Criptosistemas de llave pública cuaterniónicas

Este trabajo propuesto por los matemáticos Maheswara Rao y Shailendra Vikash Narayan, propone el uso de llaves cuaterniónica en un esquema de cifrado de llave pública. La seguridad del esquema radica en la dificultad de resolver la conjugación cuaterniónica.

El articulo propone un protocolo de intercambio de llaves similar al intercambio de llaves realizado en el estándar de cifrado RSA. El protocolo se explica a continuación tomando en cuenta a dos entidades que quieren compartir información, para este ejemplo serán denominados persona A y persona B.

i. La persona A y la persona B acuerdan escoger de manera aleatoria un conjunto de elementos x y z.
ii. La persona A elige dos números secretos $r, s, \epsilon Z$ tal que $1 \leq r \leq p\text{-}1$ y $2 \leq s \leq p\text{-}1$ después se realiza la operación $y_A = z^r x^s z^{-r} (mod\ p)$, y se manda y_A a la persona B.
iii. La persona B elige dos números secretos $u, v, \epsilon Z$ tal que $1 \leq u \leq p\text{-}1$ y $2 \leq v \leq p\text{-}1$ después se realiza la operación $y_B = z^u x^v z^{u\ r} (mod\ p)$, y se manda y_B a la persona A.
iv. La persona A realiza el cálculo $K_A = z^r y^s{}_B z^{-r}\ (mod\ p)$ que es la llave compartida que se utilizara en la sesión de cifrado.
v. La persona B calcula $K_B = z^u y^v{}_A z^{-u}\ (mod\ p)$ que es la segunda llave compartida.

Hablando de manera más práctica los pasos (i) y (ii) se pueden realzar de manera simultánea y solamente necesita de un pase de comunicación entre las dos personas. Después de esto los pasos (i) y (iii) pueden ser finalizados a través de una sola comunicación entre las entidades. Finalmente, los pasos (iv) y (v) pueden ser hechos sin la necesidad de que exista comunicación entre las dos partes.

Para descifrar la información el sujeto A simplemente realiza el cálculo:

$$[c_2(z^r c^s{}_1 z^{-r})^{-1}]\ (mod\ p) = m$$

3.4 Cifrado híbrido utilizando potencias formales de parámetro espectral con rotación cuaterniónica

El algoritmo propuesto está basado en el estudio de las funciones pseudoanalíticas y la solución de estas funciones empleando las potencias formales de parámetro espectral. Las funciones que han sido utilizadas no pueden ser resueltas empleando métodos matemáticos tradicionales, sin embargo, se puede realizar una aproximación a la solución de estas funciones haciendo uso de las potencias formales; de esta manera las soluciones a estas funciones propuestas pueden aproximarse utilizando las potencias formales y entonces se puede concluir que el algoritmo tendrá un alto nivel de seguridad.

El presente trabajo toma en cuenta el trabajo realizado por L. Bers [3] y V. Kravchenko [4], y a partir de eso se toma en cuenta un par de funciones complejas F y G que poseen en un dominio Ω derivadas parciales con respecto a las variables reales x y y, se dice que son una pareja generadora si satisfacen la condición:

$$\mathrm{Im}\,(\bar{F}\mathrm{G}) > 0$$

En donde $\bar{F}$ representa al complejo conjugado de F: $\bar{F}$ = ReF - iImF. Así cada función compleja W puede ser representada por los elementos de las funciones generadoras:

$$\mathrm{W} = \Phi\mathrm{F} + \Psi\mathrm{G}$$

En donde Φ y Ψ son funciones valuadas en los reales. La pareja (F, G) generaliza la pareja (1, i) la cual corresponde a la usual función analítica compleja.

Considerando una pareja generadora con la forma:

$$F\,(x, y) = \frac{\sigma(x)}{\tau(y)} \qquad G\,(x, y) = i\,\frac{\tau(y)}{\sigma(x)}$$

En donde σ y τ son funciones valuadas en los reales de sus variables correspondientes y asumiendo que $z_0 = 0$ y F (0) = 1.

Las funciones σ y τ son definidas como: $\sigma(x) = qcos(\Theta)$, $\tau(y) = qsen(\Theta)$; obtenidas a partir de un círculo unitario con centro en $z_0 = 0$ y considerando un radio con un ángulo Θ.

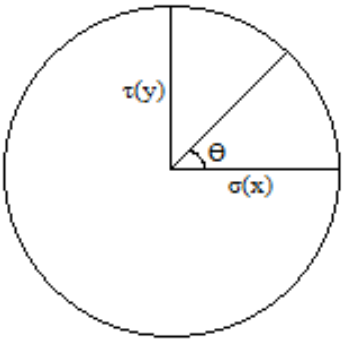


Figura 19. Definición de σ y τ

El radio entoces será dividido en *q* puntos que son distribuidos equidistantemente a lo largo del radio como se obersva en la figura 18.

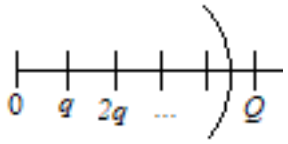


Figura 20. Radio dividido en q puntos

Una vez definidas las funiones σ y τ las potencias formales se construyen de la siguiente manera:

$$X^{(0)}(x) = \tilde{X}^{(0)}(x) = Y^{(0)}(y) = \tilde{Y}^{(0)}(y) = 1$$

y para potencias mayores a 0:

$$X^{(\mathrm{n})}(x) = \begin{cases} n \int_0^x X^{(\mathrm{n}-1)}(\xi)\frac{d\xi}{\sigma^2(\xi)} & \text{para n impar} \\ n \int_0^x X^{(\mathrm{n}-1)}(\xi)\sigma^2(\xi)d\xi & \text{para n par} \end{cases}$$

$$\tilde{X}^{(\mathrm{n})}(x) = \begin{cases} n \int_0^x \tilde{X}^{(\mathrm{n}-1)}(\xi)\sigma^2(\xi)d\xi & \text{para n impar} \\ n \int_0^x \tilde{X}^{(\mathrm{n}-1)}(\xi)\frac{d\xi}{\sigma^2(\xi)} & \text{para n par} \end{cases}$$

$$Y^{(\mathrm{n})}(y) = \begin{cases} n \int_0^y Y^{(\mathrm{n}-1)}(\eta)\frac{d\eta}{\tau^2(\eta)} & \text{para n impar} \\ n \int_0^y Y^{(\mathrm{n}-1)}(\eta)\tau^2(\eta)d\eta & \text{para n par} \end{cases}$$

$$\tilde{Y}^{(\mathrm{n})}(x) = \begin{cases} n \int_0^y \tilde{Y}^{(\mathrm{n}-1)}(\eta)\tau^2(\eta)d\eta & \text{para n impar} \\ n \int_0^y \tilde{Y}^{(\mathrm{n}-1)}(\eta)\frac{d\eta}{\tau^2(\eta)} & \text{para n par} \end{cases} \quad (2)$$

para $a_n = a' + a''$ se tiene:

$$z^{(\mathrm{n})}(a_n, 0; z) = \frac{\sigma(x)}{\tau(y)} Re\ {}_*z^{(n)}(a_n, 0; z) + i\ \frac{\tau(y)}{\sigma(x)} \quad Im\ {}_*z^{(n)}(a_n, 0; z) \quad (3)$$

donde:

$${}_*z^{(n)}(a_n, 0; z) = a' \sum_{j=0}^{n} \binom{n}{j} X^{(n-j)} i^j Y^j + i a'' \sum_{j=0}^{n} \binom{n}{j} \tilde{X}^{(n-j)} i^j \tilde{Y}^j \quad \text{para n impar}$$

$${}_*z^{(n)}(a_n, 0; z) = a' \sum_{j=0}^{n} \binom{n}{j} \tilde{X}^{(n-j)} i^j Y^j + i a'' \sum_{j=0}^{n} \binom{n}{j} X^{(n-j)} i^j \tilde{Y}^j \quad \text{para n par} \quad (4)$$

Utilizando estas fórmulas los valores para realizar el cifrado podrán ser obtenidos.

Considerando $\sigma(x) = qcos(\Theta)$, $\tau(y) = qsen(\Theta)$ se utilizará la interpolación de splines cúbicos para obtener los polinomios que existen entre cada punto *q* de un radio k. Una vez obtenidos los polinomios se puede utilizar cualquier método de integración para resolver las integrales (1). Para el presente trabajo el método de integración utilizado es Simpson 1/3, de esta manera se obtienen los *N* valores de potencias que se requieran.

Después de obtener los valores de las expresiones en (1) para cada potencia *n* de cada punto *q* se utilizan las expresiones (2) y (3) para obtener los valores con los que se construirá la matriz que servirá como la llave del criptosistema.

$$\{z^{(n)}[q,k]\}_{q=0,k=0}^{Q,K}$$

De esta manera se obtendrá un sistema lineal independiente, el cual podrá ser ortonormalizado al utilizar el estándar de ortonormalización Gram-Schmidt obteniendo una matriz $G_{[Q,K]}$ que contiene vectores ortonormalizados.

Después de obtener la matriz ortonormalizada $G_{[Q,K]}$ se aplica la rotación cuaternionica (4) para que el cifrado sea más complejo.

En resumen, se puede describir el proceso de cifrado y descifrado con los siguientes pasos:

Cifrado:

i. Se construye la matriz Se construye la matriz $z^{(n)}$ utilizando el proceso mencionado en la sección III.
ii. Se ortnormaliza la matriz obtenida $z^{(n)}$ para obtener $G_{[Q,K]}$ que es la llave para cifrar la información.
iii. Se cifra la información multiplicando los productos internos de la matriz $G_{[Q,K]}$ y la matriz de datos $D_{[M,N]}$. Lo que da como resultado una matriz con datos encriptados $C_{[M,N]}$.
iv. Se aplica la rotación cuaternionica a la matriz $C_{[M,N]}$ para obtener una matriz de datos con un cifrado hibrido.

Descifrado:

i. Se obtiene la matriz inversa de rotación cuaternionica utilizando la ecuación (5).
ii. Se obtiene la matriz ortonormalizada $G_{[Q,K]}$.
iii. Se multiplica el producto interno de la matriz $G_{[Q,K]}$ y la matriz de datos encriptados $C_{[M,N]}$.

Las ventajas de este cifrado residen en la dificultad de resolver las funciones pseudoanalíticas, debido a que estas funciones no se pueden resolver utilizando métodos matemáticos tradicionales la única manera de resolver el cifrado es aplicando las potencias formales de parámetro espectral. Además, después de aplicar dichas potencias se debe de encontrar el cuaternión que fue utilizado para rotar la matriz de valores ortonormalizados.

Capítulo 4

Pruebas y resultados

4.1 Especificaciones del equipo de cómputo utilizado

Actualmente la tecnología avanza a un ritmo muy acelerado, de esta manera, las pruebas realizadas en este equipo podrán ser mejoradas en poco tiempo. Es importante mencionar también que el equipo utilizado no cuenta con los componentes más avanzados que ofrece el mercado actual (2019); el equipo es considerado como un equipo de gama media, es decir, que sus componentes no tienen más de dos años desde su presentación.

Las especificaciones de dicho equipo, las cuales determinan el tiempo de procesamiento y de ejecución del programa son las siguientes:

Edición de Windows

Windows 10 Home Single Language

Sistema

Fabricante:	ASUSTek Computer Inc.
Procesador:	Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz 2.80 GHz
Memoria instalada (RAM):	8.00 GB
Tipo de sistema:	Sistema operativo de 64 bits, procesador x64
Lápiz y entrada táctil:	La entrada táctil o manuscrita no está disponible para esta pantalla

Para determinar el uso del lenguaje se consideró la adaptabilidad multiplataforma pues fácilmente puede adecuarse a otros sistemas operativos. El programa fue escrito en C++, en el compilador Dev C++ en su versión 5.11.0.0. La ventaja de hacer programas en compiladores de lenguaje C++ es que el tiempo de ejecución es menor en comparación con otros compiladores como MatLab.

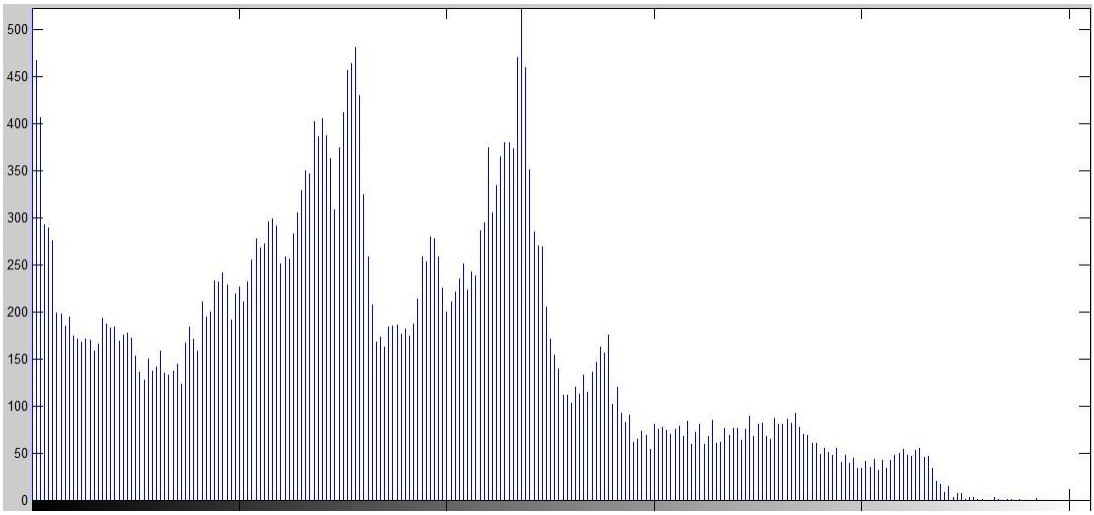
Debido a esto, el lenguaje C++ es la opción más viable al elaborar y probar un programa de cifrado puesto que el tiempo es un factor determinante para reconocer la utilidad de un algoritmo de cifrado.

4.2 Cifrado de imágenes con el método S.P.P.S.

Los resultados mostrados a continuación, incluyendo el cifrado y descifrado de una imagen monocromática utilizando el método S.P.P.S., se calculan con el coeficiente de correlación que existe entre la imagen original y la imagen cifrada haciendo uso de la función *corr2()* en la plataforma MatLab. También se muestran los histogramas tanto de la imagen original como de la imagen cifrada.

Figura 21. Imagen original "Lena.bmp"

El histograma de la imagen original es obtenido utilizando la función *imhist()* también incluida en MatLab:



*22**Figura 22. Histograma de la imagen "Lena.bmp"***

Para comenzar el proceso de cifrado de la imagen *Lena.bmp* lo primero a realizar es obtener valores polinomiales empleando el método de *splines* cúbicos [30][31]. En la figura 23 se observa un ejemplo de cómo se obtienen de manera computacional estos polinomios.

```
i       ai       bi       ci       di
0     0.71     0.77     0.00    -0.06
1     1.41     0.58    -0.19     0.32
2     2.12     1.15     0.76    -1.20
3     2.83    -0.93    -2.84     0.95

i       ai       bi       ci       di
0     0.71     0.77     0.00    -0.06
1     1.41     0.58    -0.19     0.32
2     2.12     1.15     0.76    -1.20
3     2.83    -0.93    -2.84     0.95
```

Figura 23. Polinomios obtenidos con los "splines cúbicos"

Una vez obtenidos los polinomios se procede a obtener los valores de $X^{(n)}$, $\tilde{X}^{(n)}$, $Y^{(n)}$, $\tilde{Y}^{(n)}$, utilizando las ecuaciones (2). De manera computacional se ve de la siguiente manera:

```
X: 0.382812
Xt: 2.65272
Y: 0.382809
Yt: 2.65274
X: 1.01549
Xt: 1.01549
Y: 1.01549
Yt: 1.01549
X: 0.388742
Xt: 2.69381
Y: 0.388739
Yt: 2.69383
X: 1.03122
Xt: 1.03122
Y: 1.03122
Yt: 1.03122
```

Figura 24. Obtención de $X^{(n)}$, $\tilde{X}^{(n)}$, $Y^{(n)}$, $\tilde{Y}^{(n)}$

A continuación, se obtienen los valores de ${}_*z^{(n)}$ utilizando las ecuaciones mostradas en (4) según sea el caso, es decir, según el número de potencia que se desea obtener es par o impar. Con los valores obtenidos ${}_*z^{(n)}$ es posible calcular los valores de potencias formales utilizando la ecuación (3). En las siguientes imágenes se muestra como se obtienen los valores para ${}_*z^{(n)}$ y $z^{(n)}$.

```
factorial: 1
X: 0.382812
Y: 1
*Z: 0.382812
factorial: 1
Xt: 1
Yt: 2.65274
*Z: -2.65274
*Z suma: -2.26993
```

```
z real: -2.26992
z real: -2.03099
z real: -6.16513
z real: -4.12488

z real: -48.8058
z real: -2.50973
z real: -122.618
z real: -6.29868
```

Figura 25. ***Obtención de valores*** $_*z^{(n)}$ ***y*** $z^{(n)}$

Al obtener los valores de potencias $z^{(n)}$ se procede a aplicar el proceso de ortonormalización Gram-Schmidt a la matriz que contiene estos valores de potencias.

```
q10: -0.283075
q20: -0.253279
q30: -0.768836
q40: -0.514402

q11: -0.243776
q21: 0.368249
q31: -0.5205
q41: 0.730784
```

Figura 26. Ortonormalización de matriz de potencias

Con la matriz de datos ortonormalizada es posible realizar el proceso de cifrado de la imagen multiplicando esta matriz de datos por la matriz de la imagen obteniendo una imagen cifrada:

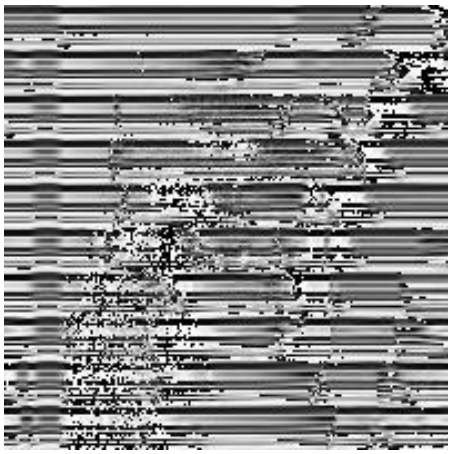


Figura 27. Imagen cifrada con el método S.P.P.S.

El coeficiente de correlación obtenido entre la imagen cifrada con el método S.P.P.S. y la imagen original es de -0.0373. Esto quiere decir que las imágenes son muy poco parecidas. A continuación se muestra el histograma de la imagen cifrada.

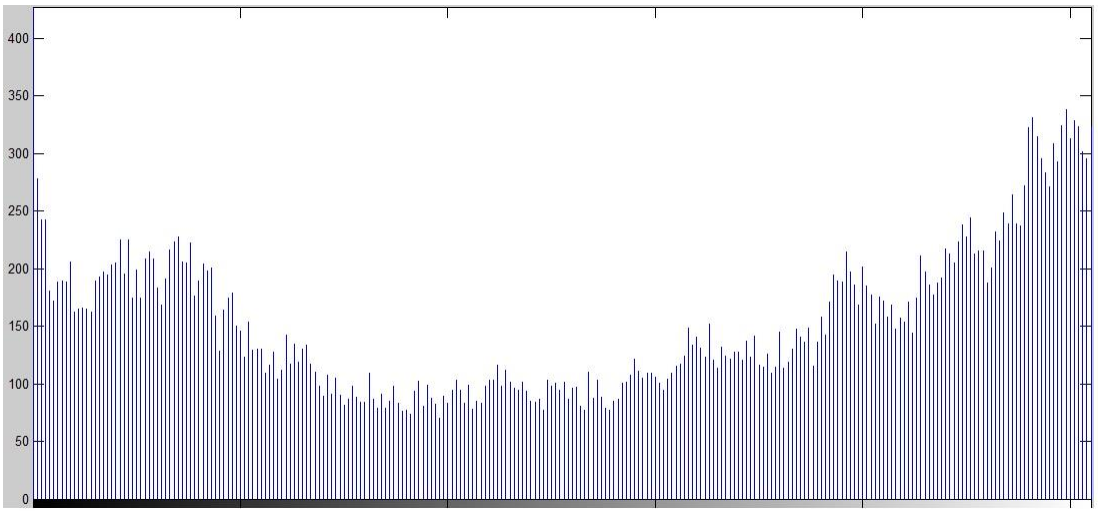


Figura 28. Histograma de la imagen cifrada con el método S.P.P.S.

Se puede observar, que el cifrado empleando el método S.P.P.S., puede ser utilizable. Sin embargo, como lo demuestra el histograma, los valores de la matriz de la imagen son muy disparejos, lo que significa que el sistema puede ser atacado mediante el ataque de fuerza bruta y debido a esta disparidad de valores es sencillo para un atacante distinguir en qué punto debe concentrarse para romper el cifrado.

El valor de la entropía para esta imagen es de $H(X) = 7.8745$ que es un buen valor pero no el óptimo.

4.3 Cifrado híbrido S.P.P.S. con rotación cuaterniónica

Se puede observar que el cifrado de información, utilizando potencias formales, tiene el potencial de ser un algoritmo de cifrado efectivo. En el presente trabajo se busca una forma de mejorar dicho cifrado y la manera que se propone es la de combinar el método de S.P.P.S. con la rotación cuaterniónica. Entonces, continuando con el proceso de cifrado; después de obtener la matriz de datos ortonormalizados se aplicará la rotación cuaterniónica a dicha

matriz para mejorar el proceso de mezcla de la información y obtener un mejor cifrado al momento de multiplicar por la matriz de datos de la imagen.

Para poder hacer la rotación cuaterniónica que se muestra en la ecuación (5) es necesario introducir u obtener un cuaternión. Los coeficientes del cuaternión serán propuestos de manera aleatoria, ya que estos valores sólo sirven para poder realizar la matriz y no poseen un valor fijo en específico.

Los resultados obtenidos al realizar la rotación cuaterniónica son notablemente mejores:

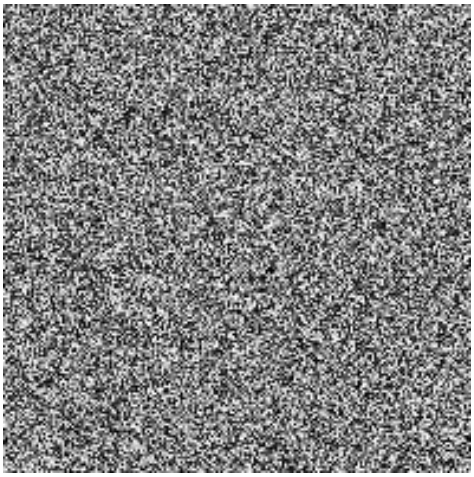


Figura 29. Imagen cifrada utilizando el método S.P.P.S. con rotación cuaterniónica

El histograma de la imagen cifrada con el cifrado híbrido nos muestra lo siguiente:

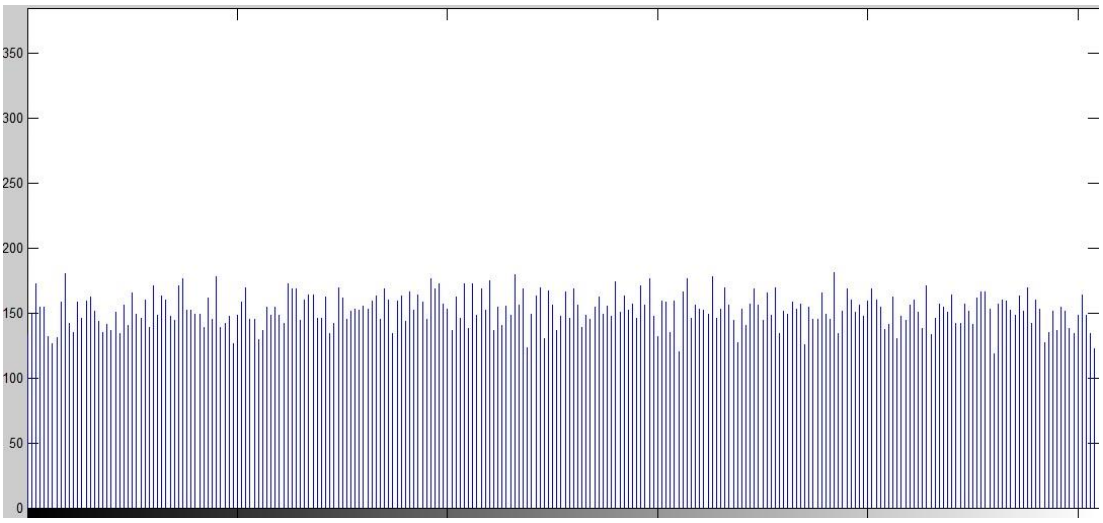


Figura 30. Histograma de la imagen cifrada con el método S.P.P.S. con rotación cuaterniónica

En el histograma se puede notar que el cifrado es mucho mejor si se le aplica la rotación cuaterniónica. Los valores de la matriz de la imagen cifrada son ahora mucho más parecidos, lo que significa que si se quiere romper el cifrado ahora va a ser mucho más complicado debido a la similitud de valores que tiene la información de la imagen. Gracias al uso de la rotación cuaterniónica el sistema de cifrado propuesto ahora es mucho más seguro que el cifrado que solamente utiliza las potencias formales.

El coeficiente de correlación obtenido entre la imagen original y la imagen cifrada con el algoritmo híbrido es de: 0.0046 lo que significa que la información que contiene cada imagen es muy diferente una de la otra.

La entropía de la información de la imagen cifrada tiene el valor $H(X) = 7.995$ el cual es un valor muy cercano a 8 que es el número de bits que utiliza cada píxel de la imagen, esto significa que la información de la imagen cifrada está realmente oculta y es muy complicado que se pueda descifrar la información.

Para probar el algoritmo de cifrado se tomaron en cuenta distintos números de potencias; el programa se ejecutó cinco veces para cada caso obteniendo los siguientes resultados:

Tabla 1. Resultados de pruebas del cifrado

Potencias/ prueba	1	2	3	4	5	Promedio
6	2.82s.	2.802s.	2.79s.	2.87s.	2.826s.	2.82s.
9	2.85s.	2.824s.	2.858s.	2.83s.	2.838s.	2.84s.
18	2.792s.	2.794s.	2.862s.	2.805s.	2.838s.	2.818s.
33	2.785s.	2.827s.	2.813s.	2.862s.	2.818s.	2.821s.

Considerando el tiempo promedio en el que se realiza el cifrado se puede concluir que actualmente el cifrado no sería apto para comunicaciones que son en vivo como llamadas telefónicas ya que el tiempo de respuesta que se necesita para este tipo de comunicaciones es de 140 ms. Sin embargo, el algoritmo de cifrado híbrido puede ser utilizado sin ningún inconveniente para comunicaciones que no requieren de inmediatez de respuesta o en tiempo real.

A continuación se mostrará el proceso necesario para descifrar la información y los resultados obtenidos al descifrar la imagen.

Para el proceso de descifrado de la información lo primero que se hace es obtener la matriz inversa de la matriz de rotación cuaterniónica utilizando la ecuación (6). Al obtener la matriz inversa de rotación, es multiplicada por la matriz de información cifrada. De esta manera se obtiene la matriz de datos ortonormalizados.

Es necesario obtener la matriz inversa de la información ortonormalizada debido a que esta matriz está formada por vectores ortonormales, obtener su matriz inversa es relativamente simple ya que sólo hay que obtener su matriz transpuesta. Entonces, con la matriz transpuesta es posible recuperar los datos originales a partir de la imagen cifrada y de esta manera reconstruir la imagen original.

Resultados del proceso de descifrado de la imagen:

Figura 31. Imagen descifrada

Se puede observar que la imagen que se ha obtenido al descifrar la imagen cifrada es idéntica a la imagen original. Para probar que la imagen se ha recuperado de manera íntegra se obtiene el coeficiente de correlación entre la imagen original y la imagen descifrada.

Utilizando la función *corr2* de MatLab se obtiene que el coeficiente de correlación de estas imágenes es 1.000, esto significa que la señal o la información que ha sido recuperada es exactamente la misma que la información original. También se obtiene el histograma de la

imagen descifrada para comparar y revisar con el histograma de la imagen original y así comprobar si la información se ha recuperado correctamente.

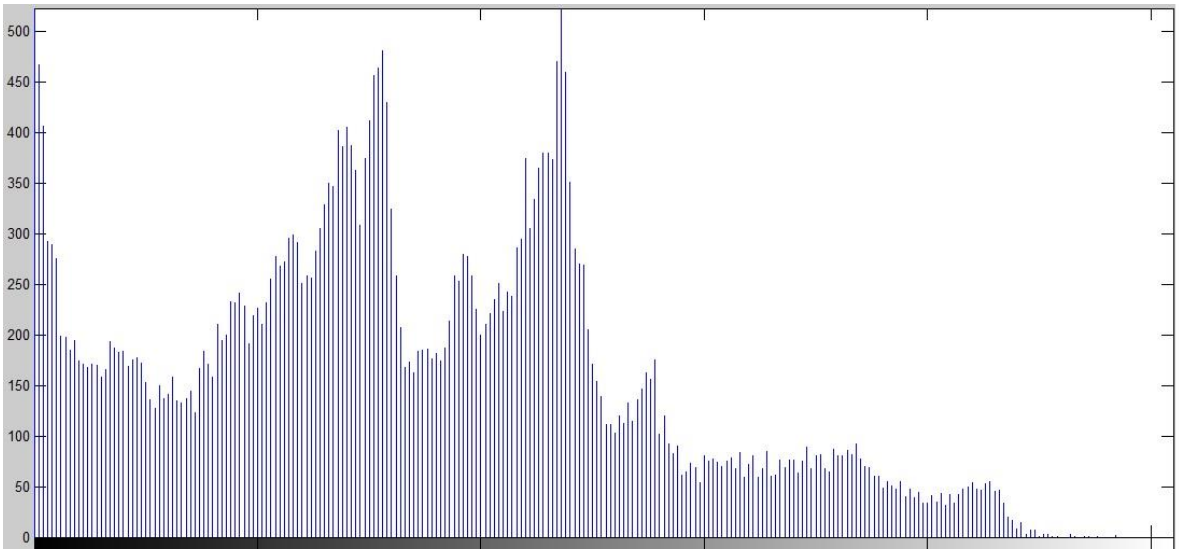


Figura 32. Histograma de la imagen cifrada

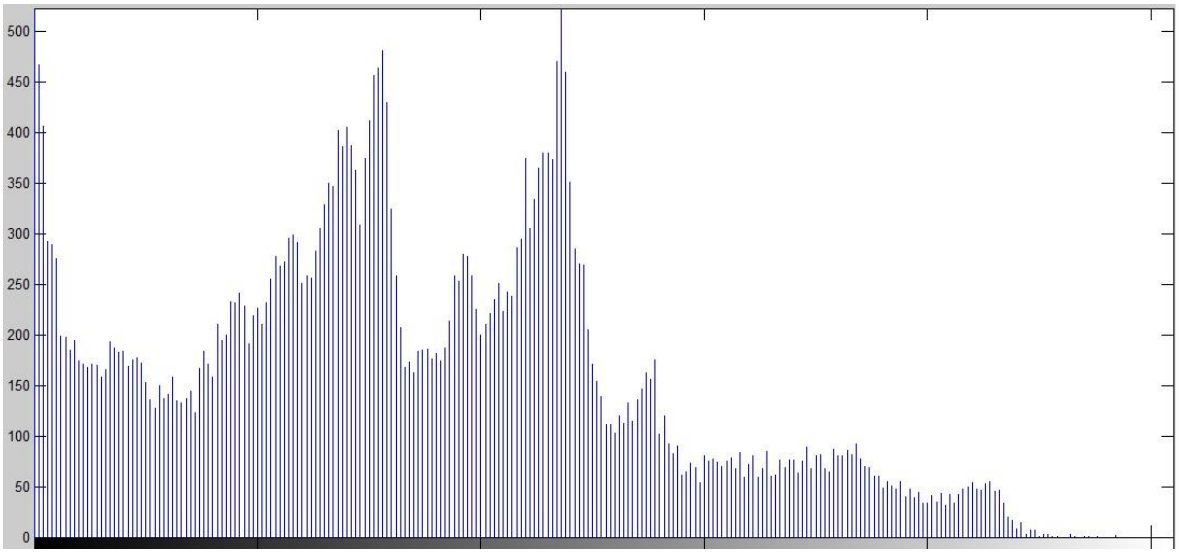


Figura 33, Histograma de la imagen original

Como se puede observar, los histogramas son idénticos, gracias a que el coeficiente de correlación es 1.000 se puede concluir que la información ha sido encriptada, enviada y recuperada de manera íntegra por lo que el sistema tiene un funcionamiento óptimo.

Ahora se compara el algoritmo híbrido con otros algoritmos para determinar si es que existe alguna ventaja al usar el algoritmo híbrido de S.P.P.S. con rotación cuaterniónica en comparación con los otros algoritmos.

Primero se obtendrán los coeficientes de correlación de las imágenes cifradas con distintos números de potencias:

I. El coeficiente correlación cuando se utilizan tres potencias formales es de 0.0357. El histograma que se obtiene es:

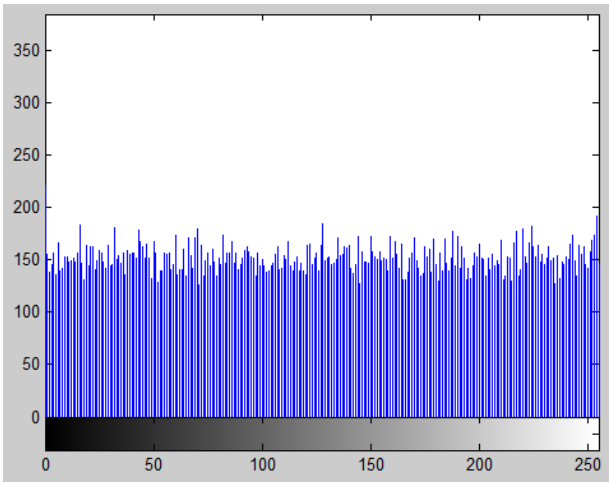


Figura 34. Histograma de la imagen cifrada utilizando tres potencias

II. El coeficiente de correlación cuando se utilizan nueve potencias es 0.0031. Su histograma:

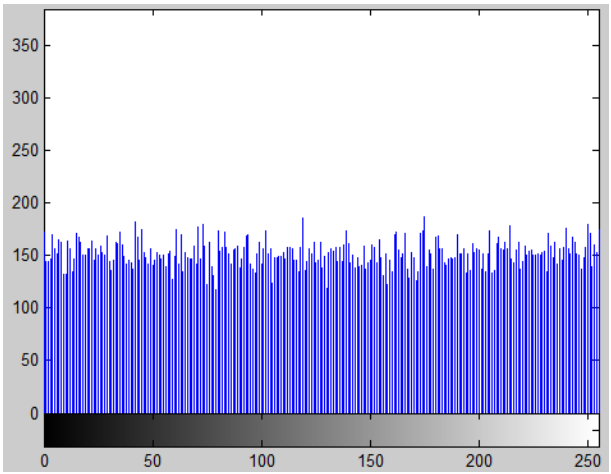


Figura 35. Histograma de la imagen cifrada con 9 potencias

III. El coeficiente de correlación cuando se utilizan dieciocho potencias es 0.0014. EL histograma de este caso es el siguiente:

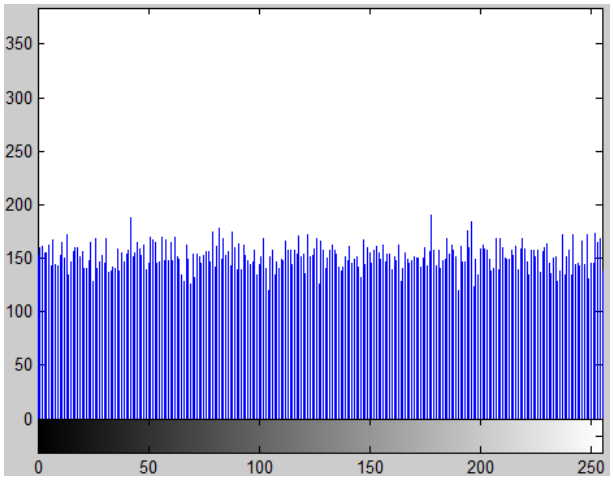


Figura 36. Histograma de la imagen cifrada con 18 potencias

IV. El coeficiente cuando se utilizan treinta y tres potencias es 0.0014 y el histograma es:

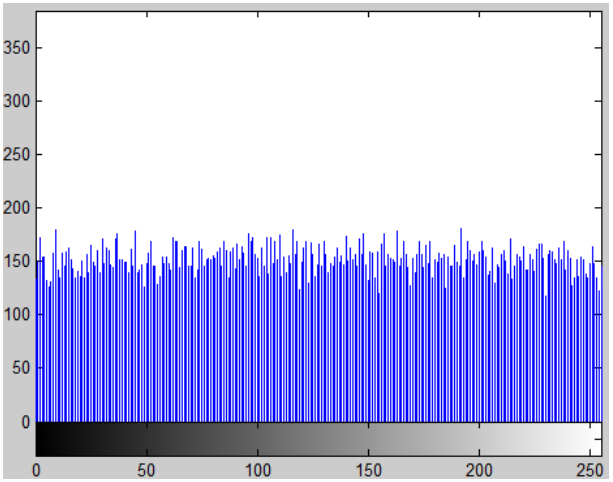


Figura 37. Histograma de la imagen cifrada con 33 potencias

Como se puede observar en los histogramas, los valores de la imagen cifrada se mantienen similares; sólo existen algunas variaciones de valores debido al número de potencias que se están utilizando. Los valores de los coeficientes de correlación van disminuyendo, lo que significa que conforme aumenta el número de potencias utilizadas el cifrado va mejorando ya que los datos de la imagen original y la imagen cifrada son cada vez menos parecidos.

Se pueden comparar estos resultados obtenidos con otros trabajos realizados en el área de la criptografía. Por ejemplo; se compara el algoritmo híbrido con el trabajo realizado por el M.

en C. Carlos Alejandro Solís Órnela [32] que combina la rotación cuaterniónica con el estándar RSA.

El promedio de tiempo de ejecución del trabajo es de 0.8876s. y el coeficiente de correlación de una imagen con su parte cifrada es de 0.0038. En cuanto a tiempo es claro que este trabajo es más eficiente que el propuesto. En cuanto a seguridad se observa que el coeficiente de correlación es superado desde que utilizan nueve potencias formales por lo que se puede deducir que el algoritmo que emplea S.P.P.S. tiene un nivel mayor de seguridad y, que si se utilizan más potencias formales, la posibilidad de que alguien pueda quebrar el sistema disminuye drásticamente.

Ahora bien, muchos autores concuerdan en que al momento de diseñar un nuevo sistema criptográfico lo mejor es dar prioridad al apartado de seguridad, claro que al hacer esto la eficiencia del sistema se ve mermada debido a los recursos máquina que se necesitan para realizar las operaciones.

Es importante recordar que para realizar las pruebas se utiliza un equipo que es considerado de gama media por lo que utilizar los componentes más actuales disminuirá el tiempo necesario para cifrar información. También es importante recordar que actualmente la tecnología aumenta a un ritmo desmesurado, por lo que es posible que en muy poco tiempo los equipos de cómputo más nuevos puedan ejecutar el programa de manera más rápida, por lo que el algoritmo híbrido podría ser utilizado incluso en comunicaciones telefónicas o que necesitan ser inmediatas.

Ahora se compara el algoritmo híbrido empleando S.P.P.S. con rotación cuaterniónica con el trabajo del M. en C. Dante Ulises Contreras Cortés [33] quien plantea el uso de cuaterniones combinado con la norma NTRU.

En el trabajo está especificado que el tiempo total de ejecución de este programa fue de 16 milisegundos. Se cifraron cuatro mensajes obteniendo los siguientes coeficientes de correlación:

Tabla 2. Tiempo de cifrado de cuatro mensajes

Mensaje 1	**0.00282**
Mensaje 2	0.04855
Mensaje 3	0.22179
Mensaje 4	0.36699

Una vez más se observa que el tiempo de ejecución es mejor que el algoritmo propuesto en este trabajo y, también, se observa una vez más que los coeficientes de correlación del algoritmo propuesto son mejores, lo que significa que la seguridad es mejor al utilizar las potencias formales combinadas con la rotación cuaternionica.

4.4 Trabajos a futuro

El desarrollo del algoritmo híbrido ha demostrado que el uso de potencias formales tiene gran potencial para ser considerado para crear numerosos sistemas que integren este método matemático. Al combinar el método S.P.P.S. con la rotación cuaterniónica se obtuvo un sistema que demostró ser seguro.

Tomando en cuenta este proyecto se pueden desarrollar otros trabajos basados en el método aquí descrito; se puede complementar el presente trabajo pasando del método simétrico al asimétrico empleando o adaptando las distintas normas de cifrado asimétrico como el RSA o el NTRU.

Se puede mejorar el algoritmo para que pueda cifrar una mayor cantidad de información o buscar optimizar los tiempos de ejecución para que el sistema pueda ser utilizada en comunicaciones más inmediatas como lo es la comunicación telefónica o por mensajes instantáneos.

También es posible implementar el programa para el cifrado de todo tipo de datos como video, audio, sonido y texto. El realizar pruebas con imágenes comprueba que el sistema es apto para ser adaptado y para cifrar todo tipo de datos.

La codificación de fuente y de canal son aspectos igualmente importantes para ser tomados en cuenta cuando se consideran las comunicaciones por lo que el desarrollo de codificadores utilizando S.P.P.S. es una buena opción.

Con el desarrollo tecnológico que existe en la actualidad y lo que viene en años próximos es importante tomar en cuenta distintos métodos matemáticos para su implementación en sistemas de cifrado. Se debe considerar que en el futuro cercano no sólo los datos financieros o información que se requiere mantener de carácter privado estarán en riesgo al momento de su transmisión, sino que con los avances en los campos de la biomedicina o tecnología militar, la salud y la vida de las personas será también un factor importante a considerar cuando se trate de desarrollar un nuevo sistema de cifrado.

Conclusión

La criptografía ha sido utilizada por muchas civilizaciones como método para establecer comunicaciones seguras. Algunos gobiernos han dedicado e invertido inmensas cantidades de tiempo y dinero al desarrollo de algoritmos de encriptación cada vez más complejos para poder mantener a salvo su privacidad. Es por esto que un sistema criptográfico debe tener como prioridad la confidencialidad y seguridad de la información que será transmitida.

Es importante mencionar que los resultados obtenidos en la presente investigación están basados en cifrado y descifrado de imágenes. El cifrado de imágenes es realizado utilizando multiplicación de matrices; al hacer funcionar un esquema de cifrado con imágenes es más sencillo adaptar el algoritmo para cifrar texto o audio, por lo que es conveniente primero el cifrado y descifrado de imágenes.

Al utilizar funciones pseudoanalíticas la seguridad del sistema puede ser considerada como alta debido al proceso matemático utilizado para resolver dichas funciones. Si alguien no autorizado intenta robar o alterar la información es necesario resolver la función para poder obtener la llave con la que se han encriptado los datos y aun con esta información el intruso deberá descubrir cuántas potencias formales fueron utilizadas para la aproximación, ya que cualquier variación modificará los valores contenidos en las matrices y por lo tanto no será capaz de romper el sistema.

Con los resultados obtenidos en el proyecto se pudo comprobar que un sistema de cifrado que utiliza las potencias formales de parámetro espectral tiene el potencial para convertirse en un cifrado fuerte, no obstante, también se comprobó que con los equipos de procesamiento que existen actualmente un cifrado que utiliza S.P.P.S. puede tener un alto coste computacional.

La combinación de los cuaterniones con las potencias formales ha probado ser una buena opción para solucionar el problema de seguridad y tiempo que presenta el sistema S.P.P.S. por si solo. Gracias a la implementación del a rotación cuaternionica al sistema original desarrollado con las potencias formales, se pudo mejorar notablemente el desempeño del cifrado tanto en el aspecto de seguridad tanto como en el aspecto de eficiencia.

Para desarrollar un algoritmo de encriptación hoy en día es importante tomar en cuenta que en un futuro próximo las nuevas tecnologías tendrán una capacidad de procesamiento mayor, lo que puede ocasionar que sistemas de cifrado desarrollados con tecnología pasada puedan ser quebrados con más facilidad. Debido a esto los algoritmos de encriptación actuales deben enfocarse en ofrecer un nivel de seguridad alto para que estén vigentes ahora y en el futuro.

Es importante recalcar que debido a la velocidad con la que se desarrolla la tecnología con cada vez más capacidad de procesamiento, la criptografía es una ciencia que seguirá evolucionando y que tiene un campo de acción que sigue y seguirá desarrollándose. Hoy en día la criptografía sigue siendo una herramienta extremadamente útil para certificar las comunicaciones seguras ya sea para proteger conversaciones meramente casuales entre dos personas, hasta proteger información bancaria o incluso militar.

Anexos

Anexo A

Estándar DES (FIPS PUS 46-3). Tablas de permutaciones, expansiones y tablas de obtención de llaves de cada ronda del proceso de cifrado.

Tabla de permutación IP.

IP

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Tabla de permutación IP^{-1}.

IP^{-1}

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

Tabla E para expandir una cadena d e 32 bits a una cadena de 48 bits.

E BIT-SELECTION TABLE

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

Tabla de permutación P aplicada después de la sustitución hecha con las cajas S y el resultado de la operación XOR entre las subclaves y la salida de la tabla E.

P

16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

Tablas PC-1 utilizadas para la obtención de subclaves o llaves secundarias. La función elimina el bit menos significativo.

***PC*-1**

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

Permutación PC-2 elimina 8 bits y reduce el tamaño de llave a 48 bits

PC-2

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Cajas S utilizadas para sustituir la función f cada 6 bits de la cadena de salida obtenida de la operación XOR entre el resultado de la expansión E con la cadena R y las subclaves k. La sustitución se realiza de la siguiente manera:

La cadena de salida entre $(E(R),\ k)$ = 48 $bits$, se toman los primeros 6 bits de la cadena, de estos 6 bits, se toman el primer y el ultimo bit (el bit más significativo y el menos significativo), se agrupan de forma binaria y se convierte a decimal, que en total son cuatro posibles combinaciones, que indican la fila en la primer caja S, posteriormente los 4 bits restantes o de en medio se toman en cuenta para la columna de la caja S, en total 16 posibles combinaciones. Se toman los siguientes 6 bits para la segunda caja S, y asi sucesivamente.

S_1

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S_2

15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S_3

10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S_4

7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S_5

2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S_6

12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S_7

4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S_8

13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Tabla de corrimientos para obtención de las sub-llaves.

Iteration Number	Number of Left Shifts
1	1
2	1
3	2
4	2
5	2
6	2
7	2
8	2
9	1
10	2
11	2
12	2
13	2
14	2
15	2
16	1

Estándar AES (FIPS-PUB 197)

Caja S empleada para sustituir bytes.

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Caja inversa de S

		y														
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
	1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
	2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
	3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
	4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
	5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
	6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
	7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
	8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
	9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
	a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
	b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
	c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
	d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
	e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
	f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

Anexo B

Código en lenguaje c++

```
#include <stdio.h>
#include <math.h>
#include <iostream>
#include <iomanip>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <string>
#include <stdlib.h>
#include "bitmap_image.hpp"
int i=0, j=0, contri=0;
using namespace std;
float pi=3.1416;
int main()
{
////////////////////////////////////// CUATERNION Y SU INVERSA
float quate[4] = {3, 170, 24, 89}, tq, qrotada[3][6];
int iq, jq, kq, nq=3;


float qrotacion[3][6] = {{((2*pow(quate[0],2)-1)+2*pow(quate[1],2)),
(2*quate[1]*quate[2]-2*quate[0]*quate[3]), (2*quate[1]*quate[3]+2*quate[0]*quate[2])},
{(2*quate[1]*quate[2]+2*quate[0]*quate[3]), ((2*pow(quate[0],2)-1)+2*pow(quate[2],2)),
(2*quate[2]*quate[3]-2*quate[0]*quate[1])}, {(2*quate[1]*quate[3]-2*quate[0]*quate[2]),
(2*quate[2]*quate[3]+2*quate[0]*quate[1]), ((2*pow(quate[0],2)-
1)+2*pow(quate[3],2))}};


        for (int fila=0;fila<3;fila++)
        {
```

```
                for (int columna=0;columna<3;columna++)
                {

                        qrotada[fila][columna]=qrotacion[fila][columna];
                }
        }

for(iq=0;iq<3;iq++)
{
  for(jq=3;jq<2*3;jq++)
  {
    if(iq==jq-3)
      qrotada[iq][jq]=1;
    else
      qrotada[iq][jq]=0;
   }
}

for(iq=0;iq<3;iq++)
{
  tq=qrotada[iq][iq];
  for(jq=i;jq<2*3;jq++)
    qrotada[iq][jq]=qrotada[iq][jq]/tq;
  for(jq=0;jq<3;jq++)
  {
    if(iq!=jq)
    {
      tq=qrotada[jq][iq];
      for(kq=0;kq<2*3;kq++)
      qrotada[jq][kq]=qrotada[jq][kq]-tq*qrotada[iq][kq];
```

```
        }
      }
    }
  /*  cout<<"\n\nInverse matrix\n\n";
   for(iq=0;iq<3;iq++)
   {
     for(jq=3;jq<2*3;jq++)
       cout<<"\t"<<qrotada[iq][jq];
     cout<<"\n";
    }*/
////////////////////////////////////////  SPLINES  ////////////////////////////
int n,i,j;
float radianes=0;
radianes = (45*pi)/180;
    cout<<"Introduzca el numero de puntos: ";cin>>n;
    n--;
    float x[n + 1], ax[n + 1], hx[n], Ax[n], lx[n + 1],
    ux[n + 1], zx[n + 1], cx[n + 1], bx[n], dx[n], sigma[n], ay[n + 1], hy[n], Ay[n], ly[n + 1],
      uy[n + 1], zy[n + 1], cy[n + 1], by[n], dy[n], tau[n];

    for(int coef=0;coef<n;coef++)
          {
                sigma[coef]= (coef + 1) * cos(radianes);
          }

           for(int coef=0;coef<n;coef++)
          {
                tau[coef]= (coef + 1) * sin(radianes);
          }
```

```
for (i = 0; i < n + 1; ++i)
        {
         x[i]=i + 1;
        }
for (i = 0; i < n + 1; ++i)
        {
                ax[i]=sigma[i];
                ay[i]=tau[i];
        }

/////// PASO 1
for (i = 0; i <= n - 1; ++i)
        {
                hx[i] = x[i + 1] - x[i];
                hy[i] = x[i + 1] - x[i];
        }

///// PASO 2
for (i = 1; i <= n - 1; ++i)
{
        Ax[i] = 3 * (ax[i + 1] - ax[i]) / hx[i] - 3 * (ax[i] - ax[i - 1]) / hx[i - 1];
        Ay[i] = 3 * (ay[i + 1] - ay[i]) / hy[i] - 3 * (ay[i] - ay[i - 1]) / hy[i - 1];
        }

   ///// PASO 3
   lx[0] = 1;
   ux[0] = 0;
   zx[0] = 0;

   ly[0] = 1;
```

```
    uy[0] = 0;
    zy[0] = 0;
//////PASO 4
    for (i = 1; i <= n - 1; ++i) {
        lx[i] = 2 * (x[i + 1] - x[i - 1]) - hx[i - 1] * ux[i - 1];
        ux[i] = hx[i] / lx[i];
        zx[i] = (Ax[i] - hx[i - 1] * zx[i - 1]) / lx[i];

        ly[i] = 2 * (x[i + 1] - x[i - 1]) - hy[i - 1] * uy[i - 1];
        uy[i] = hy[i] / ly[i];
        zy[i] = (Ay[i] - hy[i - 1] * zy[i - 1]) / ly[i];
    }
///////PASO 5
    lx[n] = 1;
    zx[n] = 0;
    cx[n] = 0;

    ly[n] = 1;
    zy[n] = 0;
    cy[n] = 0;
//////PASO 6
    for (j = n - 1; j >= 0; --j) {
        cx[j] = zx[j] - ux[j] * cx[j + 1];
        bx[j] = (ax[j + 1] - ax[j]) / hx[j] - hx[j] * (cx[j + 1] + 2 * cx[j]) / 3;
        dx[j] = (cx[j + 1] - cx[j]) / (3 * hx[j]);

        cy[j] = zy[j] - uy[j] * cy[j + 1];
        by[j] = (ay[j + 1] - ay[j]) / hy[j] - hy[j] * (cy[j + 1] + 2 * cy[j]) / 3;
        dy[j] = (cy[j + 1] - cy[j]) / (3 * hy[j]);
    }
```

```
//////PASO 7
printf("%2s %8s %8s %8s %8s\n", "i", "ai", "bi", "ci", "di");
for (i = 0; i < n; ++i)
   printf("%2d %8.2f %8.2f %8.2f %8.2f\n", i, ax[i], bx[i], cx[i], dx[i]);


      cout<<"\n";


     printf("%2s %8s %8s %8s %8s\n", "i", "ai", "bi", "ci", "di");
for (i = 0; i < n; ++i)
   printf("%2d %8.2f %8.2f %8.2f %8.2f\n", i, ay[i], by[i], cy[i], dy[i]);
      //cout<<"\n";
// CUADRADO DE POLINOMIOS DE SIGMA Y TAU // estas creando un vector para almacenar los valores de los splines y después poder hacer multiplicación de polinomios
   float vectx[n][7], vectx2[4][7], vectxr[n][7], vecty[n][7], vecty2[4][7], vectyr[n][7];


   for(int filas=0;filas<n;filas++)
   {
      for(int columnas=0;columnas<7;columnas++)
      {
            vectx[filas][columnas]=0.0;
            vecty[filas][columnas]=0.0;
                  }
            }

            for(int filas=0;filas<4;filas++)
   {
      for(int columnas=0;columnas<7;columnas++)
      {
            vectx2[filas][columnas]=0.0;
            vecty2[filas][columnas]=0.0;
```

```
                }
            }

            for(int filas=0;filas<n;filas++)
{
   for(int columnas=0;columnas<7;columnas++)
   {
            vectxr[filas][columnas]=0.0;
            vectyr[filas][columnas]=0.0;
                }
            }
for(int k=0;k<n;k++)
{
   vectx[k][0]= ax[k];
   vectx[k][1]= bx[k];
   vectx[k][2]= cx[k];
            vectx[k][3]= dx[k];

            vecty[k][0]= ay[k];
   vecty[k][1]= by[k];
   vecty[k][2]= cy[k];
            vecty[k][3]= dy[k];
            }
     for(int ki=0;ki<n;ki++)
     {
            for(int k2=0;k2<4;k2++)
            {
                  vectx2[k2][k2+0] = vectx[ki][k2]*vectx[ki][0];
   vectx2[k2][k2+1] = vectx[ki][k2]*vectx[ki][1];
   vectx2[k2][k2+2] = vectx[ki][k2]*vectx[ki][2];
```

```
        vectx2[k2][k2+3] = vectx[ki][k2]*vectx[ki][3];

        vecty2[k2][k2+0] = vecty[ki][k2]*vecty[ki][0];
        vecty2[k2][k2+1] = vecty[ki][k2]*vecty[ki][1];
        vecty2[k2][k2+2] = vecty[ki][k2]*vecty[ki][2];
        vecty2[k2][k2+3] = vecty[ki][k2]*vecty[ki][3];
                }

                        for(int k3=0;k3<n;k3++)
                        {
                                vectxr[ki][0] = vectx2[0][0];
                                vectxr[ki][1] = vectx2[0][1] + vectx2[1][1];
                                vectxr[ki][2] = vectx2[0][2] + vectx2[1][2] + vectx2[2][2];
                                vectxr[ki][3] = vectx2[0][3] + vectx2[1][3] + vectx2[2][3] +
vectx2[3][3];
                                vectxr[ki][4] = vectx2[1][4] + vectx2[2][4] + vectx2[3][4];
                                vectxr[ki][5] = vectx2[2][5] + vectx2[3][5];
                                vectxr[ki][6] = vectx2[3][6];

                                vectyr[ki][0] = vecty2[0][0];
                                vectyr[ki][1] = vecty2[0][1] + vecty2[1][1];
                                vectyr[ki][2] = vecty2[0][2] + vecty2[1][2] + vecty2[2][2];
                                vectyr[ki][3] = vecty2[0][3] + vecty2[1][3] + vecty2[2][3] +
vecty2[3][3];
                                vectyr[ki][4] = vecty2[1][4] + vecty2[2][4] + vecty2[3][4];
                                vectyr[ki][5] = vecty2[2][5] + vecty2[3][5];
                                vectyr[ki][6] = vecty2[3][6];
                        }
    }
```

```
//// COMIENZA INTEGRACIÓN CON SIMPSON 1/3
        cout<<"\n";
        int intervalos=60, pot=0;
   cout<<"Cuantas potencias desea obtener: "; cin>>pot;
   float hin[n+1], inic[n+1], xval[intervalos+1], X[pot][n], Xt[pot][n], yvalx[intervalos+1], yparx=0.0, yimparx=0.0, yvalxt[intervalos], yparxt=0.0, yimparxt=0.0,
        Y[pot][n], Yt[pot][n], yvaly[intervalos], ypary=0.0, yimpary=0.0, yvalyt[intervalos], yparyt=0.0, yimparyt=0.0;
   for(int k=0;k<n;k++)
   {
        X[0][k]=1;
        Xt[0][k]=1;
        Y[0][k]=1;
        Yt[0][k]=1;
        }
        for(int k=0,apun=0;k<n;k++)
   {
      hin[apun++] = (x[k+1]-x[k])/intervalos;
        }
        for(int k=0;k<n;k++)
        {
                inic[k]=x[k];

                        for(int k2=0;k2<=intervalos;k2++)
                        {
                                xval[k2]=inic[k];
                                inic[k]+=hin[k];
                        }
                for(int pf=1, aux=0;pf<=pot;pf++)
                {
```

```
for(int m=0;m<=intervalos;m++)
{
    if(pf%2==0)
    {
    yvalx[m]= (vectxr[k][0] + (vectxr[k][1]*xval[m]) +
(vectxr[k][2]*pow(xval[m],2)) + (vectxr[k][3]*pow(xval[m],3)) +
(vectxr[k][4]*pow(xval[m],4)) + (vectxr[k][5]*pow(xval[m],5)) +
(vectxr[k][6]*pow(xval[m],6)));

    yvalxt[m]= (1/(vectxr[k][0] + (vectxr[k][1]*xval[m]) +
(vectxr[k][2]*pow(xval[m],2)) + (vectxr[k][3]*pow(xval[m],3)) +
(vectxr[k][4]*pow(xval[m],4)) + (vectxr[k][5]*pow(xval[m],5)) +
(vectxr[k][6]*pow(xval[m],6))));

    yvaly[m]= (vectyr[k][0] + (vectyr[k][1]*xval[m]) +
(vectyr[k][2]*pow(xval[m],2)) + (vectyr[k][3]*pow(xval[m],3)) +
(vectyr[k][4]*pow(xval[m],4)) + (vectyr[k][5]*pow(xval[m],5)) +
(vectyr[k][6]*pow(xval[m],6)));

    yvalyt[m]= (1/(vectyr[k][0] + (vectyr[k][1]*xval[m]) +
(vectyr[k][2]*pow(xval[m],2)) + (vectyr[k][3]*pow(xval[m],3)) +
(vectyr[k][4]*pow(xval[m],4)) + (vectyr[k][5]*pow(xval[m],5)) +
(vectyr[k][6]*pow(xval[m],6))));

    }
    else
    {
    yvalx[m]= (1/(vectxr[k][0] + (vectxr[k][1]*xval[m]) +
(vectxr[k][2]*pow(xval[m],2)) + (vectxr[k][3]*pow(xval[m],3)) +
(vectxr[k][4]*pow(xval[m],4)) + (vectxr[k][5]*pow(xval[m],5)) +
(vectxr[k][6]*pow(xval[m],6))));

    yvalxt[m]=(vectxr[k][0] + (vectxr[k][1]*xval[m]) +
(vectxr[k][2]*pow(xval[m],2)) + (vectxr[k][3]*pow(xval[m],3)) +
(vectxr[k][4]*pow(xval[m],4)) + (vectxr[k][5]*pow(xval[m],5)) +
(vectxr[k][6]*pow(xval[m],6)));

    yvaly[m]= (1/(vectyr[k][0] + (vectyr[k][1]*xval[m]) +
(vectyr[k][2]*pow(xval[m],2)) + (vectyr[k][3]*pow(xval[m],3)) +
(vectyr[k][4]*pow(xval[m],4)) + (vectyr[k][5]*pow(xval[m],5)) +
(vectyr[k][6]*pow(xval[m],6))));

    yvalyt[m]= (vectyr[k][0] + (vectyr[k][1]*xval[m]) +
(vectyr[k][2]*pow(xval[m],2)) + (vectyr[k][3]*pow(xval[m],3)) +
(vectyr[k][4]*pow(xval[m],4)) + (vectyr[k][5]*pow(xval[m],5)) +
(vectyr[k][6]*pow(xval[m],6)));
```

```
            }
        }

        for(int mpar=1;mpar<=intervalos;mpar+=2)
        {
            yparx+=yvalx[mpar];
            yparxt+=yvalxt[mpar];

            ypary+=yvaly[mpar];
            yparyt+=yvalyt[mpar];
        }

for(int mimpar=2;mimpar<intervalos;mimpar+=2)
        {
            yimparx+=yvalx[mimpar];
            yimparxt+=yvalxt[mimpar];

            yimpary+=yvaly[mimpar];
            yimparyt+=yvalyt[mimpar];
        }

        yparx = yparx*4;
        yimparx = yimparx*2;

        ypary = ypary*4;
        yimpary = yimpary*2;

        yparxt = yparxt*4;
        yimparxt = yimparxt*2;
```

```
                    yparyt = yparyt*4;
                    yimparyt = yimparyt*2;

                    X[pf][k] = X[pf-1][k] * ((hin[k]/3) * (yvalx[0] + yparx + yimparx +
yvalx[intervalos]));
                    yparx=0;
                    yimparx=0;

                    Xt[pf][k] =  Xt[pf-1][k] * (hin[k]/3) * (yvalxt[0] + yparxt + yimparxt
+ yvalxt[intervalos]);
                    yparxt=0;
                    yimparxt=0;

                    Y[pf][k] =  Y[pf-1][k] * (hin[k]/3) * (yvaly[0] + ypary + yimpary +
yvaly[intervalos]);
                    ypary=0;
                    yimpary=0;

                    Yt[pf][k] =  Yt[pf-1][k] * (hin[k]/3) * (yvalyt[0] + yparyt + yimparyt
+ yvalyt[intervalos]);
                    yparyt=0;
                    yimparyt=0;
                }
            }
  for(int k=0;k<n;k++)
   {
        X[0][k]=1;
        Xt[0][k]=1;
        Y[0][k]=1;
        Yt[0][k]=1;
        }
```

```
////////////////////////////////////////////// OBTENIENDO FACTORIALES DE CADA POTENCIA PARA *z //////////////////////////////////////////////
float zasr[pot][n], zasim[pot][n], zasims[pot][n], zasrs[pot][n];
double factorials[pot], result[pot+1][pot];

for(int filas=0;filas<=pot;filas++)
{
   for(int columnas=0;columnas<=n;columnas++)
   {
        zasim[filas][columnas]=0.0;
        zasr[filas][columnas]=0.0;
        zasrs[filas][columnas]=0.0;
        zasims[filas][columnas]=0.0;
        result[filas][columnas] = 0.0;
        }
}

for(int potenf=0; potenf<=pot;potenf++)
{
        double  factorialj=1;
        if (potenf<0)
        {
                factorialj=0;
        }

        else if(potenf==0)
        {
                factorialj=1;
        }
```

```
        else
        {
                for(int fac=1;fac<=potenf;fac++)
                {
                        factorialj= factorialj * fac;
                }
        }
                factorials[potenf]=factorialj;

        if (potenf>0)
        {
                for(int fil=0;fil<=potenf;fil++)
                {
                        result[fil][potenf] =
factorials[potenf]/(factorials[fil]*factorials[potenf-fil]);
                }
        }
}
////////////////////////////////////////// OBTENIENDO *Z PARA POTENCIAS PARES E IMPARES AQUI ERROR EN LAS Z ////////////////////////////////////
for(int n234=0, auxil=0;n234<n;n234++)
{
        for(int potenf=1;potenf<=pot;potenf++)
        {
                if(potenf%2!=0)
                {
                        for(int k=0;k<=potenf;k++)
                        {
                                if(k==0)
                                {
```

```
                    zasim[potenf][k] = result[k][potenf]* ((Xt[potenf-
k][n234] * Yt[k][n234]));
                    zasr[potenf][k] = result[k][potenf] * ((X[potenf-
k][n234] * Y[k][n234]));
                }
                else if(k%4==0 && k%2==0)
                {
                    zasim[potenf][k] = result[k][potenf] * (Xt[potenf-
k][n234] * Yt[k][n234]);
                    zasr[potenf][k] = result[k][potenf] * ((X[potenf-
k][n234] * Y[k][n234]));
                }
                else if(k%2==0)
                {
                    zasim[potenf][k] = -1 * (result[k][potenf] *
(Xt[potenf-k][n234] * Yt[k][n234]));
                    zasr[potenf][k] = -1 * (result[k][potenf] * ((X[potenf-
k][n234] * Y[k][n234])));
                }
                else if(k==3 || k==auxil+4)
                {
                    zasim[potenf][k] = -1 * (result[k][potenf] * (X[potenf-
k][n234] * Y[k][n234]));
                    zasr[potenf][k] = result[k][potenf] * ((Xt[potenf-
k][n234] * Yt[k][n234]));
                    auxil=k;
                }
                else
                {
                    zasim[potenf][k] = result[k][potenf] * (X[potenf-
k][n234] * Y[k][n234]);
                    zasr[potenf][k] = -1 * (result[k][potenf] * ((Xt[potenf-
k][n234] * Yt[k][n234])));
```

```
                    }
                    zasims[potenf][n234] += zasim[potenf][k];
                    zasrs[potenf][n234] += zasr[potenf][k];
                }
            }
            else
            {
                for(int k=0;k<=potenf;k++)
                {
                    if(k==0)
                    {
                        zasim[potenf][k] =result[k][potenf] * ((X[potenf-
k][n234] * Yt[k][n234]));
                        zasr[potenf][k] = result[k][potenf] * ((Xt[potenf-
k][n234] * Y[k][n234]));
                    }
                    else if(k%4==0 && k%2==0)
                    {
                        zasim[potenf][k] = result[k][potenf] * (X[potenf-
k][n234] * Yt[k][n234]);
                        zasr[potenf][k] = result[k][potenf] * ((Xt[potenf-
k][n234] * Y[k][n234]));
                    }
                    else if(k%2==0)
                    {
                        zasim[potenf][k] = -1 * (result[k][potenf] * (X[potenf-
k][n234] * Yt[k][n234]));
                        zasr[potenf][k] = -1 * (result[k][potenf] * ((Xt[potenf-
k][n234] * Y[k][n234])));
                    }
                    else if(k==3 || k==auxil+4)
                    {
```

```
                                        zasim[potenf][k] = -1 * (result[k][potenf] *
(Xt[potenf-k][n234] * Y[k][n234]));
                                        zasr[potenf][k] = result[k][potenf] * ((X[potenf-
k][n234] * Yt[k][n234]));
                                        auxil=k;
                                }
                                else
                                {
                                        zasim[potenf][k] = result[k][potenf] * (Xt[potenf-
k][n234] * Y[k][n234]);
                                        zasr[potenf][k] = -1 * (result[k][potenf] * ((X[potenf-
k][n234] * Yt[k][n234])));
                                }
                                zasims[potenf][n234] += zasim[potenf][k];
                                zasrs[potenf][n234] += zasr[potenf][k];
                        }
                }
        }
}
////////////////////////////////////////////////////////////  Obtención de potencias Z(n)
//////////////////////////////////////////
double znrf[pot][n], znimf[pot][n];

for (int xis=0;xis<n;xis++)
{
        for(int pote=1;pote<=pot;pote++)
        {
                znrf[pote][xis] = (((xis+1)*sigma[xis])/((xis+1)*tau[xis])) * zasrs[pote][xis];
                znimf[pote][xis] = (((xis+1)*sigma[xis])/((xis+1)*tau[xis])) *
zasims[pote][xis];
        }
}
```

```
////////////////////////////////////////////////////////// ORTNORMALIZACIÓN GRAHAM-SCHMIDT ////////////////////////////////////////
double orton[pot][n];
for(int prof=0;prof<n;prof++)
{
        for(int ip=1;ip<=pot;ip++)
        {
                orton[ip][prof] = znrf[ip][prof];
        }
}

double r[pot][n], q[pot][n];

   for (int k=0; k<n; k++){
    r[k][k]=0;
    for (int ii=1; ii<=pot; ii++)
      r[k][k] = r[k][k] + orton[ii][k] * orton[ii][k];
    r[k][k] = sqrt(r[k][k]);

    for (int ii=1; ii<=pot; ii++) {
       q[ii][k] = orton[ii][k]/r[k][k];
    }

    for(int jj=k+1; jj<n; jj++) {
      r[k][jj]=0;
      for(int ii=1; ii<=pot; ii++) r[k][jj] += q[ii][k] * orton[ii][jj];

      for (int ii=1; ii<=pot; ii++) orton[ii][jj] = orton[ii][jj] - r[k][jj]*q[ii][k];
    }
}
```

```
////////////////////////////////////////////// INVERSA DE LA MATRIZ ORTONORMALIZADA
////////////////////////////////
double qreng[pot][n], qinv[pot][n], t;
for(int columna=0,filas=1;columna<n,filas<=pot;columna++,filas++)
{
        for(int fila=1,columnas=0;fila<=pot,columnas<n;fila++,columnas++)
        {
                qreng[filas][columnas] = q[fila][columna];
        }
}

////////////////////////////////////////////  CIFRANDO IMAGEN  ////////////////////////////////////
double temp=0, div=0, sust=0, sust2=0, aux=0, aux2=0; //tempr=0;
std::string file_name("image198.bmp");
  bitmap_image image(file_name);
  if (!image)
  {
    printf("test07() - Error - Failed to open '%s'\n",file_name.c_str());
  }

  else
  {
        unsigned char red, green, blue, blue2;
        unsigned int f=image.height();
        unsigned int c=image.width();
        float bmatriz[c][f], btemp[c][f], btemp2[c][f], bcifr [c][f]; //rtemp[c][f],
rmatriz[c][f], rcifr[c][f], rtemp2[c][f];

        div= c / pot;
        aux=pot;
```

```
        aux2=n;
        //cout<<"\n div: "<<div;

  for(i=0;i<f;i++)
  {
                for(j=0;j<c;j++)
                {
                        image.get_pixel(i,j,red,green,blue);
                        blue=blue;
                        //red=red;
                        bmatriz[i][j]=blue;
                        //rmatriz[i][j]=red;
              //printf("\nplis: %f", bmatriz[i][j]);
                }
        //cout<<"\n";
  }

for(int divs=0;divs<div;divs++)
{
  for (int divi=0;divi<div;divi++)
  {
                for(int sk=sust;sk<aux;sk++)
                {
                        for(int sk2=sust2;sk2<aux2;sk2++)
                        {
                                btemp[sk][sk2]=bmatriz[sk][sk2];
                                //rtemp[sk][sk2]=rmatriz[sk][sk2];
                        }
                }
```

```
                ////// MULTIPLICACIÓN MATRICES
                for(int fila=sust;fila<aux;fila++)
                {
                        for(int columna=sust2, sil=0;columna<aux2, sil<n;columna++, sil++)
                        {
                                temp=0;
                                //tempr=0;
                                for(int cor=sust2, cil=1;cor<aux2,cil<=n;cor++, cil++)
                                {
                                        temp += btemp[fila][cor] * q[cil][sil];
                                        bcifr[fila][columna] = temp;

                                        /*tempr += rtemp[fila][cor] * q[cil][sil];
                                        rcifr[fila][columna] = tempr;*/

                                }
                        }
                }
                sust2=aux2;
                aux2=aux2+n;
  }
  sust=(aux-1);
  aux=(aux-1)+pot;
}
////////////////////////////////// ROTACIÓN CUANTERNIONICA /////////////////
double sustq=0, sustq2=0, auxq=3, auxq2=3, tempq=0;
float btempq[c][f], bcifrq[c][f], btempq2[c][f];
for (int roq=0;roq<66;roq++)
{
        for(int roq2=0;roq2<66;roq2++)
```

```
        {
                for(int mq=sustq;mq<auxq;mq++)
                {
                        for (int mq2=sustq2;mq2<auxq2;mq2++)
                        {
                                btempq[mq][mq2] = bcifr[mq][mq2];
                        }
                }

                for(int filaq=sustq;filaq<auxq;filaq++)
                {
                        for(int columnaq=sustq2, silq=0;columnaq<auxq2,
silq<3;columnaq++, silq++)
                        {
                                tempq=0;
                                //tempr=0;
                                for(int corq=sustq2, cilq=0;corq<auxq2,cilq<3;corq++,
cilq++)
                                {
                                        tempq += btempq[filaq][corq] * qrotacion[cilq][silq];
                                        bcifrq[filaq][columnaq] = tempq;

                                        /*tempr += rtemp[fila][cor] * q[cil][sil];
                                        rcifr[fila][columna] = tempr;*/

                                }
                        }
                }
                sustq2=auxq2;
                auxq2=auxq2+3;
```

```
        }
        sustq=(auxq-1);
  auxq=(auxq-1)+3;
}

  for(i=0;i<f;i++)
  {
                for(j=0;j<c;j++)
                {
                        blue=bcifrq[i][j];
                        //red=rcifr[i][j];
                 image.set_pixel(i,j,blue,blue,blue);
                // image.set_pixel(i,j,red,red,red);
                }

        //cout<<"\n";
  }
      image.save_image("image198cif.bmp");

////////////////////////////////// DESCIFRANDO IMAGEN  //////////////////////////////////
float regresoq[c][f];
sustq=0;
sustq2=0;
auxq=3;
auxq2=3;
tempq=0;
for (int roq=0;roq<66;roq++)
{
        for(int roq2=0;roq2<66;roq2++)
        {
```

```
                for(int mq=sustq;mq<auxq;mq++)
                {
                        for (int mq2=sustq2;mq2<auxq2;mq2++)
                        {
                                btempq2[mq][mq2] = bcifrq[mq][mq2];
                        }
                }

                for(int filaq=sustq;filaq<auxq;filaq++)
                {
                        for(int columnaq=sustq2, silq=0;columnaq<auxq2,
silq<3;columnaq++, silq++)
                        {
                                tempq=0;
                                //tempr=0;
                                for(int corq=sustq2, cilq=0;corq<auxq2,cilq<3;corq++,
cilq++)
                                {
                                        tempq += btempq[filaq][corq] * qrotada[cilq][silq];
                                        regresoq[filaq][columnaq] = tempq;

                                        /*tempr += rtemp[fila][cor] * q[cil][sil];
                                        rcifr[fila][columna] = tempr;*/

                                }
                        }
                }
                sustq2=auxq2;
                auxq2=auxq2+3;
        }
```

```
        sustq=(auxq-1);
  auxq=(auxq-1)+3;
}

float regreso[c][f], //regresor[c][f];
sust=0;
sust2=0;
aux=pot;
aux2=n;

for(int divs=0;divs<div;divs++)
{
  for (int divi=0;divi<div;divi++)
  {
                for(int sk=sust;sk<aux;sk++)
                {
                        for(int sk2=sust2;sk2<aux2;sk2++)
                        {
                                btemp2[sk][sk2]=regresoq[sk][sk2];
                                //rtemp2[sk][sk2]=rcifr[sk][sk2];
                        }
                }

                /// MULTIPLICACIÓN MATRICES
                for(int fila=sust;fila<aux;fila++)
                {
                        for(int columna=sust2, sil=0;columna<aux2, sil<n;columna++, sil++)
                        {
                                temp=0;
                                //tempr=0;
```

```
                                for(int cor=sust2, cil=1;cor<aux2,cil<=n;cor++, cil++)
                                {
                                        temp += btemp2[fila][cor] * qreng[cil][sil];
                                        regreso[fila][columna] = temp;

                                /*      tempr += rcifr[fila][cor] * qreng[cil][sil];
                                        regresor[fila][columna] = tempr;*/

                                }
                        }
                }
                sust2=aux2;
                aux2=aux2+n;
  }
  sust=(aux-1);
  aux=(aux-1)+pot;
}

  for(i=0;i<f;i++)
  {
                for(j=0;j<c;j++)
                {
                        blue=regreso[i][j];
                       // red=regresor[i][j];
                 image.set_pixel(i,j,blue,blue,blue);
                 //image.set_pixel(i,j,red,red,red);
                }

        //cout<<"\n";
  }
```

```
    image.save_image("image198descif.bmp");
  }
        return 0;
}
```

Bibliografía

[1] A. Bucio, A. Hernandez Becerril, C. M. A. Robles, M. P. Ramirez, A. Arista Jalife. (2013), *Construction of a New Cryptographic Method, Employing Pseudoanalytic Function Theory.*

[2] L. Bers (1953), *Theory of Pseudoanalytic Functions*, IMM, Universidad de Nueva York.

[3] V. V. Kravchenko (2009), *Applied Pseudoanalytic Function Theory*, Serie: Frontiers in Mathematics, ISBN: 978-3-0346-0003-3.

[4] D. Khan, (1973), *The Codebrakers*, New American library, Nueva York.

[5] Ferguson, Schneier y Kohno. *Cryptography Engeneering.* 1ª edición. Wiley Publishing. Estados Unidos de América. 2010. 1-22.

[6] J. Katz, (2015), *Introduction to modern cryptography*, 2a edición, Taylor & Francis, Boca Raton.

[7] Tomoyuki N., Ryusuke Koide, Takashi A., Yowhiei H. (2004) *"A new Quadripartite Public- Key Cryptosystem".*

[8] Mariusz Dzwonkowski, Michal Papaj y Roman Rykaczewski. (2015). "*A new quaternion-based encryption method for DICOM images"*.

[9] J. Hoffstein, (2014), *A introduction to mathematical cryptography*, 2a edición, Springer, Nueva York.

[10] O. Goldreich, (1999), *Modern cryptography, probabilistic proof, and pseudorandomness,* Springer, Nueva York.

[11] https://3l3m3nt4l.blogspot.com/2017/01/cifrado-atbash.html

[12] https://sites.google.com/site/clasescecyte/indice/tics/encriptacion?overridemobile=true

[13] https://justcodeit.io/tutorial-cifrado-cesar-en-python/

[14] http://criptografia11.blogspot.com/2017/09/los-cifrados-polialfabeticos.html

[15] https://eltrasterodepalacio.wordpress.com/2015/09/03/descifrando-mensajes-la-maquina-enigma-y-la-2a-guerra-mundial/

[16] https://www.lavanguardia.com/historiayvida/historia-contemporanea/20180611/47312986353/que-aporto-a-la-ciencia-alan-turing.html

[17] J. F. Kurose, (2017), *Computer networking: a top-down approach*, 7a edición, Pearson, New Jersey, USA.

[18] T. Limoncelli, (2017), *The practice of system and network administration*, 3a edición, Addison-Wesley, Boston, MA.

[19] Z. Cao, (2013), *New direction of modern cryptography*, Taylor & Francis group, Boca Raton, FL.

[20] P. R. Girard, (2007), *Quaternions, Clifford algebras and relativistic physics*, Birkhäuser, Basel, Boston.

[21] S. Epp, (2012), *Matemáticas discretas con aplicaciones*, 4ª edición, Cengage Learning, México, D.F.

[22] R. Espinosa, (2010), *Matemáticas discretas*, Alfaomega, México, Mariusz D., Roman R. *"Quaternion encryption method for image and video transmision".*

[23] K. H. Rosen, (1997), *Exploring Discrete Mathematics With MAPLE*, Mac Graw-Hill.

[24] R. Morant; A. Ribagorda Garnacho; J. Sancho Rodriguez, (1994), *Seguridad y protección de la información*, Centro de Estudios Ramón Areces.

[25] A. Bucio R., R. Castillo Perez, M. P. Ramirez T. (2011), *On the Numerical Construction of Formal Powers and their Application to the Electricak Impedance Equation*, 8th International Conference on Electrical Engineering, Computing Science and Automatic Control, IEEE Catalog Number: CFP11827-ART, ISBN: 978-1-4577-1013-1, pp.769-774.

[26] D. Poole, (2017), *Linear algebra: a modern introduction*, 4a edición, Cengaje Learning, México, D.F.

[27] https://csrc.nist.gov/csrc/media/publications/fips/46/3/archive/1999-10-25/documents/fips46-3.pdf

[28] https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf

[29] PKCS #1 v2.2: RSA Cryptography Standard

[30] R. L. Burden, J. D. Faires (2011), *Numerical Analysis*, 9th edition Brooks/Cole, Cengage Learning. ISBN: 978-0-538-73351-9, pp. 144-163.

[31] W. Stallings, (2017), *Network security essentials: applications and standars*, 6a edición, Pearson, Hoboken, New Jersey.

[32] C. A. Solis, (2017), *Propuesta de un algoritmo de cifrado híbrido basado en matrices de rotación cauterniónica y el estándar RSA*. Tesis, IPN, Ciudad de México.

[33] D. U. Contreras, (2017), *Cifrado de firma electrónica mediante el sistema criptográfico NTRU basado en cuaterniones*. Tesis, IPN, Ciudad de México.

[34] https://www.condusef.gob.mx/gbmx/?p=estadisticas

[35] https://web.itu.edu.tr/~orssi/dersler/cryptography/Chap2-1.pdf

[36] https://access.redhat.com/blogs/766093/posts/1976023

[37] https://www.fing.edu.uy/~jvieitez/cripto2001.pdf

[38] https://eltamiz.com/elcedazo/2018/06/23/explorando-el-algebra-geometrica-3-antecedentes-los-cuaterniones-i/

[39] http://neo.lcc.uma.es/evirtual/cdd/tutorial/presentacion/rsa.html

[40] http://www.ics-aragon.com/cursos/salud-publica/2014/pdf/M2T04.pdf

[41] http://www.dma.fi.upm.es/recursos/aplicaciones/matematica_discreta/web/aritmetica_modular/bibliografia.html

[42] https://criptografia.webnode.es/algoritmos-simetricos/

Constancias y publicaciones

CONCYS 2019
CONGRESO NACIONAL DE CIRCUITOS Y SISTEMAS
CONCYS

LA CRIPTOGRAFÍA EN LA SEGURIDAD INFORMÁTICA

Ávila-Castro H[a*], Oviedo-Galdeano H[b]

[a,b]Sección de Estudios de Posgrado e Investigación, Escuela Superior de Ingeniería Mecánica y Eléctrica unidad Zacatenco, Instituto Politécnico Nacional, Ciudad de México, México.
*delta5303@hotmail.com

RESUMEN:

La criptografía es una herramienta indispensable para poder realizar transmisiones seguras. Con la tecnología actual que nos permite estar conectados prácticamente en todo momento, existe la necesidad de algún método que asegure que la información que es transmitida no pueda ser utilizada o adquirida por las personas no autorizadas. La herramienta principal de la criptografía son las matemáticas, haciendo uso de distintos teoremas y algoritmos es posible desarrollar un buen sistema criptográfico.

PALABRAS CLAVE: Criptografía, algoritmo, seguridad.

I. INTRODUCCIÓN

Es imprescindible contar con experiencia y la mentalidad adecuada para resolver problemas relacionados con seguridad, pues es útil en el desarrollo de un buen sistema de encriptación, ya que éste depende de conocimientos científicos, así como del entendimiento de distintas herramientas matemáticas, por lo cual la criptografía es conocida como el arte y ciencia de la encriptación.

Actualmente la dependencia de las personas por las computadoras u otros dispositivos electrónicos que se pueden conectar a una red, para guardar información personal u otros datos de importancia, representan un problema para la seguridad. Aunque no son el único problema, generalmente cuando un dispositivo o una red tiene un sistema de seguridad que es considerado complejo, es más probable que los cibercriminales ataquen este sistema para obtener la información deseada.

Las redes son sistemas abiertos por naturaleza. Los sistemas conectados a la red pueden, por definición, interpretar todas las transmisiones, pero están programadas para ignorar las que no están destinadas para ellas. De esta forma la criptografía se utiliza para ocultar la información que se desea transmitir a través de un canal inseguro para que sólo sea utilizada y revisada por la persona indicada.

II. CRIPTOGRAFÍA MODERNA

La criptografía moderna se divide en dos ramas: la criptografía simétrica o de llave privada y la criptografía asimétrica o de llave pública. La de llave simétrica es la técnica más antigua y consiste en usar una misma llave tanto para encriptar como para desencriptar. La llave para encriptar puede ser un número, palabras o caracteres al azar. Se debe utilizar la misma llave para cifrar el texto plano así como para descifrarlo. Este es uno de los inconvenientes del cifrado simétrico ya la distribución de claves se tendría que hacer de forma personal o a través de un medio que se conozca como seguro, de otra manera se tendría que mandar la llave a través del

canal inseguro y el cifrado no tendría caso. Por otro lado, el cifrado por medio de llave pública necesita el uso de dos claves que están relacionadas matemáticamente. Una de las llaves puede ser distribuida a cualquier persona a través del canal inseguro. La segunda clave se mantiene en secreto.

Los principales objetivos que debe cumplir un algoritmo criptográfico o que hacen que un algoritmo criptográfico sea útil son:

- Confidencialidad: asegurar que la información solo esté disponible para las personas autorizadas.
- Integridad: certificar que la información que se recibe sea idéntica a la que fue enviada y que solo las personas autorizadas puedan modificarla.
- Disponibilidad: aseverar que la información pueda ser utilizada cada vez que se requiera por las personas autorizadas.

Distintos métodos numéricos pueden ser utilizados para desarrollar nuevos algoritmos de encriptación. Para esta investigación, se encuentra en desarrollo un algoritmo de encriptación de llave pública en el cual se emplearán las potencias formales para generar la llave pública. El uso de esta herramienta matemática permite que la encriptación sea más rápida debido a la cantidad de datos que puede encriptar a la vez. La investigación se encuentra en proceso, se pretende que el algoritmo sea más eficiente que los algoritmos de encriptación utilizados actualmente además de que se convierta en un algoritmo de encriptación que siga vigente en el futuro de la seguridad informática.

III. CONCLUSIONES

La criptografía ha sido utilizada por muchas civilizaciones como método para realizar comunicaciones seguras. Hoy en día la criptografía sigue siendo una herramienta extremadamente útil para certificar las comunicaciones seguras ya sea para proteger conversaciones meramente casuales entre dos personas, hasta para proteger información bancaria o información militar. Algunos gobiernos han dedicado e invertido grandes cantidades de tiempo y dinero al desarrollo de algoritmos de encriptación cada vez más complejos para poder mantener a salvo su privacidad.

Gracias al desarrollo de tecnología con capacidad de procesamiento cada vez mayor, la criptografía es una ciencia que seguirá evolucionando y que tiene un campo de acción que prácticamente nunca terminará. Existe una lucha constante entre las personas que desarrollan nuevos algoritmos para proteger la información y las que buscan romper con estos algoritmos por lo cual siempre será necesario desarrollar nuevos algoritmos de encriptación.

REFERENCIAS

[1] Contreras-Cortés D (2017). Cifrado de firma electrónica mediante el sistema criptográfico NTRU basado em cuaterniones. Tesis de Maestría. Instituto Politécnico Nacional, México.

[2] Ferguson N; Scheiner B; Khono T; (2010). Cryptography Engeering. Wiley Publishing, Inc. ISBN: 978-0-470-4742-2.

[3] Fine L (1997). Seguridad en centros de cómputo, políticas y procedimientos. 2ª edición. Editorial Trillas, México. ISBN: 968-24-4097-1.

[4] Loshin P; (1998). Personal Encryption. Academic Press. ISBN:0-12-455837-2.

UASLP
Universidad Autónoma de San Luis Potosí

COORDINACIÓN ACADÉMICA REGIÓN ALTIPLANO OESTE | FACULTAD DE CIENCIAS | FACULTAD DE INGENIERÍA

La Universidad Autónoma de San Luis Potosí, a través de la Coordinación Académica Región Altiplano Oeste, la Facultad de Ciencias y la Facultad de Ingeniería, otorga el presente

RECONOCIMIENTO

a **Ávila-Castro H.**

Por su **participación en cartel** con el trabajo:

LA CRIPTOGTAFÍA EN LA SEGURIDAD INFORMÁTICA

presentado en el **2º Congreso Nacional de Circuitos y Sistemas (CONCYS)**, que se llevó a cabo del 13 al 15 de marzo en la Unidad de Posgrados de la UASLP, como parte de las actividades de divulgación del Cuerpo Académico "Sistemas Dinámicos y Criptografía" de la institución.

San Luis Potosí, S.L.P., marzo de 2019.

ATENTAMENTE

"Siempre Autónoma. Por Mi Patria Educaré"

CONCYS

Dr. Luis Javier Ontañón García Pimentel
Líder del Cuerpo Académico
COARA

Dr. Marco Tulio Ramírez Torres
Miembro del Cuerpo Académico
COARA

Dr. Isaac Campos Cantón
Miembro del Cuerpo Académico
Facultad de Ciencias

Dr. Carlos Soubervielle Montalvo
Miembro del Cuerpo Académico
Facultad de Ingeniería

Propuesta de un algoritmo de cifrado híbrido empleando funciones pseudoanalíticas.

Ávila-Castro H[a], Oviedo-Galdeano H[b]
[a,b]Sección de Estudios de Posgrado e Investigación, Escuela Superior de Ingeniería Mecánica y Eléctrica unidad Zacatenco, Instituto Politécnico Nacional, Ciudad de México, México.
delta5303@hotmail.com

***Resumen*— La criptografía es una herramienta indispensable para poder realizar transmisiones seguras. Con la tecnología actual que nos permite estar conectados prácticamente en todo momento existe la necesidad de algún método que asegure que la información que es transmitida no pueda ser utilizada o adquirida por las personas no autorizadas. Aplicando la teoría de funciones pseudoanalíticas se propone un algoritmo de cifrado para la red de telecomunicaciones. Para dar solución a estas funciones no es posible emplear métodos matemáticos clásicos por lo que se utiliza el método de potencias formales de parámetro espectral para resolver el problema.**

***Palabras Clave* — Algoritmo, Criptografía, Funciones pseudoanalíticas, Potencias formales, Seguridad.**

I. Introducción

El desarrollo de un buen sistema de encriptación depende de conocimientos científicos; del entendimiento de distintas herramientas matemáticas, por lo cual la criptografía es conocida como el arte y ciencia de la encriptación. Los algoritmos de cifrado pueden estar basados en distintos métodos matemáticos siempre y cuando; estos demuestren tener alguna ventaja en seguridad o eficiencia en comparación con algoritmos criptográficos desarrollados actualmente con distintos métodos matemáticos. En la figura 1 se puede observar cómo funciona un sistema criptográfico:

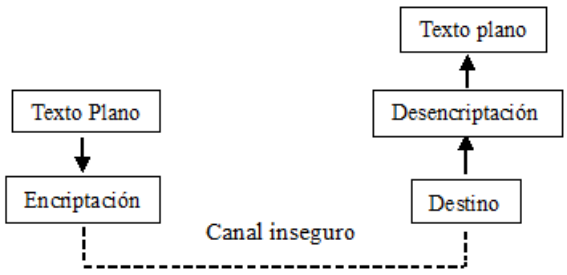


Figura 1.- Funcionamiento general de un sistema criptografico.

El algoritmo propuesto es desarrollado con base en la teoría de funciones pseudoanalíticas, la cual ha probado ser una herramienta muy útil para la Física teórica y las matemáticas aplicadas.

Las funciones pseudoanalíticas según L. Bers[1], [2], [3] son soluciones generalizadas a las ecuaciones de Cauchy-Riemann. La utilización de estas funciones ha hecho posible aproximarse analíticamente a soluciones generalizadas de distintos problemas matemáticos que no pueden ser solucionados con los métodos tradicionales. Esto ha permitido proponer nuevos sistemas criptográficos que cumplan con los requisitos de seguridad e integridad de la información.

En el presente trabajo se propone un algoritmo de cifrado utilizando funciones pseudoanalíticas y la aproximación de potencias formales de parámetro espectral. El objetivo principal es presentar un algortimo capaz de encriptar información para ser transmitida de forma segura y que sólo pueda ser descifrada por quienes posean la clave correcta.

II. Elementos de la teoría de funciones pseudoanalíticas y potencias formales

A. *Teoría de funciones pseudoanalíticas*

La propuesta del algoritmo se hace tomando en cuenta el estudio de las funciones pseudoanalíticas y su solución utilizando las potencias formales. Debido a que las funciones propuestas no pueden ser solucionadas empleando métodos matemáticos tradicionales, se puede concluir que el algoritmo tendrá un nivel de seguridad alto.

Considerando el trabajo de L. Bers[3] y de V. Kravchenko[4], se toman en cuenta un par de funciones complejas F y G que poseen en un dominio Ω derivadas parciales con respecto a las variables reales x y y, se dice que son una pareja generadora si satisfacen la condición:

$$\mathrm{Im}\,(\bar{F}G) > 0$$

En donde $\bar{F}$ es el complejo conjugado de F: $\bar{F} = \mathrm{Re}F - i\mathrm{Im}F$. Así cada función compleja W puede ser representada por los elementos de las funciones generadoras:

$$W = \Phi F + \Psi G$$

Donde Φ y Ψ son funciones valuadas en los reales. La pareja (F, G) generaliza la pareja (1, i) la cual corresponde a la usual función analítica compleja. L. Bers introduce la notación de la derivada de W como:

$$\partial_{(F,G)}W = (\partial_{(z)}\Phi)F + (\partial_{(z)}\Psi)G \quad (1)$$

Donde $\partial_{(z)} = \frac{\partial}{\partial_{(x)}} - i\frac{\partial}{\partial_{(y)}}$ e i denota la unidad imaginaria $i^2 = -1$. La derivada de la función W existe si y solo si

$$(\partial_{(\bar{z})}\Phi)F + (\partial_{(\bar{z})}\Psi)G = 0 \quad (2)$$

Donde $\partial_{(\bar{z})} = \frac{\partial}{\partial_{(x)}} + i\frac{\partial}{\partial_{(y)}}$.

Las siguientes expresiones son conocidas como coeficientes característicos de la pareja generadora (F, G):

$$a_{(F,G)} = -\frac{\bar{F}\partial_{(\bar{z})}G - \partial_{(\bar{z})}F\bar{G}}{F\bar{G} - \bar{F}G} \qquad b_{(F,G)} = \frac{F\partial_{(\bar{z})}G - \partial_{(\bar{z})}FG}{F\bar{G} - \bar{F}G}$$

$$A_{(F,G)} = -\frac{\bar{F}\partial_{(z)}G - \partial_{(z)}F\bar{G}}{F\bar{G} - \bar{F}G} \qquad B_{(F,G)} = -\frac{F\partial_{(z)}G - \partial_{(z)}FG}{F\bar{G} - \bar{F}G}$$

Con las notaciones anteriores se puede reescribir la ecuación (2) como:

$$\partial_{(\bar{z})}W = a_{(F,G)}W + b_{(F,G)}W$$

La cual es conocida como ecuación de Vekua. Las soluciones de esta ecuación son llamadas (F, G) pseudoanalíticas. La solución general de esta ecuación puede ser presentada en forma de una serie de Taylor de potencias formales:

$$W = \sum_n^{\infty} z^{(n)}(a_{n,}, z_0; z)$$

A. Potencias formales

La potencia formal $z_0^{(0)}(a_{0,}, z_0; z)$ con centro en z_0, donde el coeficiente es a_0 y el exponente es 0 se define como la combinación lineal de las generadoras F_m y G_m y se explica como:

$$z_0^{(0)}(a_{0,}, z_0; z) = \lambda F_0 + \mu G_0$$

Donde λ y μ son constantes reales tal que:

$$\lambda F_0(z_0) + \mu G_0(z_0) = a_0$$

Las potencias formales con exponentes mayores a 0 se definen con la fórmula recursiva:

$$Z_m^{(n)}(a_{n,}, z_0; z) = n\int_{z_0}^{z} Z_{m+1}^{(n-1)}(a_{n,}, z_0; z)d_{(F_m\, G_m)}z \quad (3)$$

Esta definición tiene distintas propiedades expresadas en [1] y [2] que son:

1. $\lim_{z\to z_0}(z)^n(a_{n,}, z_0; z) = a_n(z - z_0)^n$
2. Considerando $a_n = a' + a''$ y siendo a', a'' constantes reales entonces

$$Z_m^{(n)}(a' + a''_{,}, z_0; z) = a' Z_m^{(n)}(1, z_0; z) + a'' Z_m^{(n)}(i, z_0; z).$$

Estas propiedades son válidas para todas las potencias formales; lo que significa que cualquier potencia formal $z^{(n)}(a_{n,}, z_0; z)$ puede ser aproximada por la combinación lineal $z^{(n)}(1, z_0; z)$ y $z^{(n)}(i, z_0; z)$ así los cálculos numéricos serán realizados para aproximar estas dos clases de potencias formales.

Tomando en cuenta una pareja generadora con la forma:

$$F(x, y) = \frac{\sigma(x)}{\tau(y)} \qquad G(x, y) = i\frac{\tau(y)}{\sigma(x)} \quad (4)$$

En donde σ y τ son funciones valuadas en los reales de sus variables correspondientes y asumiendo que $z_0 = 0$ y $F(0) = 1$ para que los cálculos sean más simples; de esta manera las potencias formales están construidas de la siguiente forma:

$$X^{(0)}(x) = \tilde{X}^{(0)}(x) = Y^{(0)}(y) = \tilde{Y}^{(0)}(y) = 1 \quad (5)$$

y para potencias mayores a 0:

$$X^{(n)}(x) = \begin{cases} n\int_0^x X^{(n-1)}(\xi)\frac{d\xi}{\sigma^2(\xi)} & \text{para n impar} \\ n\int_0^x X^{(n-1)}(\xi)\sigma^2(\xi)d\xi & \text{para n par} \end{cases}$$

$$\tilde{X}^{(n)}(x) = \begin{cases} n\int_0^x \tilde{X}^{(n-1)}(\xi)\sigma^2(\xi)d\xi & \text{para n impar} \\ n\int_0^x \tilde{X}^{(n-1)}(\xi)\frac{d\xi}{\sigma^2(\xi)} & \text{para n par} \end{cases}$$

$$Y^{(n)}(y) = \begin{cases} n\int_0^y Y^{(n-1)}(\eta)\frac{d\eta}{\tau^2(\eta)} & \text{para n impar} \\ n\int_0^y Y^{(n-1)}(\eta)\tau^2(\eta)d\eta & \text{para n par} \end{cases}$$

$$\tilde{Y}^{(n)}(x) = \begin{cases} n\int_0^y \tilde{Y}^{(n-1)}(\eta)\tau^2(\eta)d\eta & \text{para n impar} \\ n\int_0^y \tilde{Y}^{(n-1)}(\eta)\frac{d\eta}{\tau^2(\eta)} & \text{para n par} \end{cases} \quad (6)$$

Donde ξ y η son variables de integración que representan a las funciones de x y y respectivamente.

Para $a_n = a' + a''$ se tiene:

$$z^{(n)}(a_n, 0; z) = \frac{\sigma(x)}{\tau(y)}Re\ {}_*z^{(n)}(a_n, 0; z) + i\frac{\tau(y)}{\sigma(x)}Im\ {}_*z^{(n)}(a_n, 0; z) \quad (7)$$

Donde:

$${}_*z^{(n)}(a_n, 0; z) = a'\sum_{j=0}^{n}\binom{n}{j}X^{(n-j)}i^jY^j + ia''\sum_{j=0}^{n}\binom{n}{j}\tilde{X}^{(n-j)}i^j\tilde{Y}^j \quad \text{para n impar}$$

$$ {}_{*}z^{(n)}(a_n, 0; z) = a\,'\sum_{j=0}^{n}\binom{n}{j}\tilde{X}^{(n-j)}i^{j}Y^{j} + ia\,'\sum_{j=0}^{n}\binom{n}{j}X^{(n-j)}i^{j}\tilde{Y}^{j} \quad \text{para n par} \tag{8} $$

Con estas fórmulas se obtendrán los valores necesarios para desarrollar el algoritmo de cifrado.

I. Desarrollo del algoritmo

Considerando un par de funciones generadoras (*F*, G) con la forma (4) y un circulo unitario con centro en $z_0 = 0$ y dado que las expresiones integrales de (3) son independientes, se tomarán en cuenta distintos radios *k* como se muestra en la figura 2.

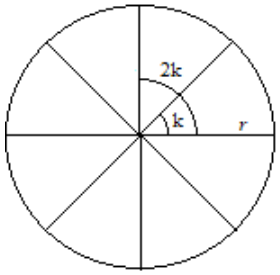


Fig. 2.- Circulo unitario dividido en k = 8 radios.

Considerando cada radio con su respectivo ángulo y tomando en cuenta lo que se muestra a continuación, es posible definir las funciones σ y τ.

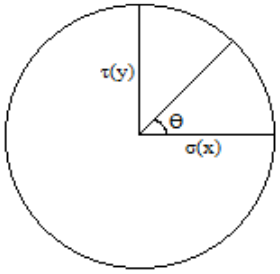


Fig. 3.- Definiendo τ y σ para un radio.

Cada radio sera divido en *q* puntos distribuidos equidistantemente a lo largo de cada radio como se observa en la figura 4.

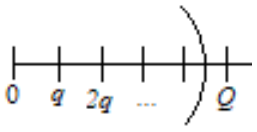


Fig. 4.- Un radio dividido en *q* puntos.

De acuerdo con las figuras 3 y 4 podemos definir nuestras funciones como:

$$ \sigma(x) = qcos(\Theta),\ \tau(y) = qsen(\Theta) $$

Una vez definidas las funciones, se utilizará la interpolación de splines cúbicos [5], para obtener los polinomios que existen entre cada punto *q* de un radio *k*.

Al haber obtenido los polinomios es posible utlizar cualquier método estándar de integración para resolver las expresiones en (6). En el presente trabajo se utiliza el método de integración Simpson 1/3 [6] para obtener el número de potencias *N*.

Después de obtener los valores de las expresiones en (6) para cada potencia *n* de cada punto *q* se utilizan las expresiones (7) y (8) para obtener los valores con los que se construirá la matriz que servirá como la llave del criptosistema.

$$ \{z^{(n)}[q,k]\}_{q=0,k=0}^{Q,K} $$

De esta manera se obtendrá un sistema lineal independiente, el cual podrá ser ortonormalizado al utilizar el estándar de ortonormalización Gram-Schmidt obteniendo una matriz $G_{[Q,K]}$ que contiene vectores ortonormalizados.

II. Funcionamiento del algoritmo

Se propone un algoritmo de llave privada, lo que significa que se utilizará la misma llave tanto para cifrar como para descifrar la información transmitida.

Considerando la matriz $G_{[Q,K]}$ que contiene los valores ortonormalizados de las potencias formales de distintos puntos de varios radios. Y considerando una matriz de *M x N* valores que se quieren encriptar; los datos a encriptar se organizan en una matriz $D_{[M,N]}$.

Teniendo en cuenta los elementos anteriormente mencionados, se describen los pasos de cifrado y descifrado que conforman el algoritmo propuesto.

A. Proceso de cifrado.

1) Se construye la matriz $z^{(n)}$ utilizando el proceso mencionado en la sección III.
2) Se ortnormaliza la matriz obtenida $z^{(n)}$ para obtener $G_{[Q,K]}$ que es la llave para cifrar la información.
3) Se cifra la información multiplicando los productos internos de la matriz $G_{[Q,K]}$ y la matriz de datos $D_{[M,N]}$. Lo que da como resultado una matriz con datos encriptados $C_{[M,N]}$.

B. Proceso de descifrado

1) Se obtiene la matriz ortonormalizada $G_{[Q,K]}$.
2) Se multiplica el producto interno de la matriz $G_{[Q,K]}$ y la matriz de datos encriptados $C_{[M,N]}$.

El proceso de cifrado de la información consiste en realizar la multiplicación de los productos internos de la matriz de vectores ortonormalizados $G_{[Q,K]}$ con la matriz de datos que se quieren ocultar $D_{[M,N]}$ obteniendo así la matriz de datos cifrados $C_{[M,N]}$.

Para descifrar el mensaje se realiza la misma operación de matrices ahora entre la matriz de datos encriptados $C_{[M,N]}$ y la matriz ortonormalizada $G_{[Q,K]}$.

I. CONCLUSIONES

La criptografía ha sido utilizada por muchas civilizaciones como método para establecer comunicaciones seguras. Algunos gobiernos han dedicado e invertido inmensas cantidades de tiempo y dinero al desarrollo de algoritmos de encriptación cada vez más complejos para poder mantener a salvo su privacidad. Es por esto que un sistema criptográfico debe tener como prioridad la confidencialidad y seguridad de la información que será transmitida.

Al utilizar funciones pseudoanalíticas la seguridad del sistema puede ser considerada como alta debido al proceso matemático utilizado para resolver dichas funciones. Si alguien no autorizado intenta robar o alterar la información es necesario resolver la función para poder obtener la llave con la que se han encriptado los datos y aun con esta información el intruso deberá descubrir cuántas potencias formales fueron utilizadas para la aproximación, ya que cualquier variación modificará los valores contenidos en las matrices y por lo tanto no será capaz de romper el sistema.

Gracias al desarrollo de tecnología con capacidad de procesamiento cada vez mayor, la criptografía es una ciencia que seguirá evolucionando y que tiene un campo de acción que sigue y seguirá desarrollándose. Hoy en día la criptografía sigue siendo una herramienta extremadamente útil para certificar las comunicaciones seguras ya sea para proteger conversaciones meramente casuales entre dos personas, hasta proteger información bancaria o incluso militar.

REFERENCIAS

[1] A. Bucio, A. Hernandez Becerril , C. M. A. Robles, M. P. Ramirez, A. Arista Jalife. (2013), *Construction of a New Cryptographic Method, Employing Pseudoanalytic Function Theory*, Proceedings of the Wolrd Congress on Engineering and Computer Science. ISBN: 978-988-19252-3-7.

[2] A. Bucio R., R. Castillo Perez, M. P. Ramirez T. (2011), *On the Numerical Cosntruction of Formal Powers and their Application to the Electricak Impedance Equation*, 8th International Conference on Electrical Engineering, Computing Science and Automatic Control, IEEE Catalog Number: CFP11827-ART, ISBN: 978-1-4577-1013-1, pp.769-774.

[3] L. Bers (1953), *Theory of Pseudoanalytic Functions*, IMM, Universidad de Nueva York.

[4] V. V. Kravchenko (2009), *Applied Pseudoanalytic Function Theory*, Serie: Frontiers in Mathematics, ISBN: 978-3-0346-0003-3.

[5] R. L. Burden, J. D. Faires (2011), *Numerical Analysis*, 9th edition Brooks/Cole, Cengage Learning. ISBN: 978-0-538-73351-9, pp. 144-163.

[6] S. Barnett (1998), *Discrete Mathematics: Numbers and Beyond*, Addison-Wesley.

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

Instituto Politécnico Nacional
"La Técnica al Servicio de la Patria"

XVIII CONGRESO NACIONAL DE INGENIERÍA ELECTROMECÁNICA Y DE SISTEMAS

Otorga el presente diploma a:

Héctor Ávila y Héctor Oviedo

Por su participación con el trabajo

Propuesta de un algoritmo de cifrado híbrido empleando funciones pseudoanalíticas

13 al 15 de Noviembre de 2019
Centro de Educación Continua "Ing. Eugenio Méndez Docurro"
Centro Histórico, Ciudad de México.

Dr. José Martínez Trinidad
Jefe de la Sección de Estudios de Posgrado
e Investigación de la ESIME
Unidad Zacatenco

M. en C. Hugo Quintana Espinosa
Director de la ESIME
Unidad Zacatenco