ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

ESCUELA DE INGENIERÍA ELECTRÓNICA EN TELECOMUNICACIONES Y REDES

"DESARROLLO DE UN MECANISMO DE CIFRADO BASADO EN EL ALGORITMO CRIPTOGRÁFICO SIMÉTRICO AES"

TRABAJO DE TITULACIÓN

Tipo: PROPUESTA TECNOLÓGICA

Para optar al Grado Académico de:

INGENIERA EN ELECTRÓNICA, TELECOMUNICACIONES Y REDES

AUTORA: CARLA ELIZABETH RAZA MONTALVÁN

TUTOR: ING. DIEGO VELOZ CH. MSc.

Riobamba – Ecuador

2019

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

ESCUELA DE INGENIERÍA ELECTRÓNICA EN TELECOMUNICACIONES Y REDES

El Tribunal del Trabajo de Titulación certifica que: El trabajo de Titulación Propuesta Tecnológica: "DESARROLLO DE UN MECANISMO DE CIFRADO BASADO EN EL ALGORITMO CRIPTOGRÁFICO SIMÉTRICO AES", de responsabilidad de la Srta. Carla Elizabeth Raza Montalván, ha sido minuciosamente revisado por los Miembros del Tribunal del Trabajo de Titulación, quedando autorizada su presentación.

NOMBRE	**FIRMA**	**FECHA**
Dr. Julio Santillán **VICEDECANO DE LA FACULTAD DE INFORMÁTICA Y ELECTRÓNICA**	_____	_____
Ing. Patricio Romero **DIRECTOR DE LA ESCUELA DE INGENIERÍA ELECTRÓNICA EN TELECOMUNICACIONES Y REDES**	_____	_____
Ing. Diego Veloz CH. **DIRECTOR DE TRABAJO DE TITULACIÓN**	_____	_____
Ing. Alberto Arellano **MIEMBRO DEL TRIBUNAL**	_____	_____

Yo, **CARLA ELIZABETH RAZA MONTALVÁN** soy responsable de las ideas, doctrinas y resultados expuestos en este Trabajo de Titulación y el patrimonio intelectual del mismo pertenecen a la Escuela Superior Politécnica De Chimborazo.

Carla Elizabeth Raza Montalván

DEDICATORIA

A Dios por darme la fortaleza y perseverancia necesaria para finalizar cada meta propuesta a lo largo de mi vida. A Wilmer Raza, mi padre quien con su amor, ejemplo, apoyo y constancia estuvo presente en cada paso, mostrándome que solo el esfuerzo y dedicación puede proveer satisfacción plena en el ejercicio laboral. A Roció Montalván, una mujer excepcional y a quien con orgullo puedo llamar mamá; tú me has demostrado que todo se puede lograr solo necesitas creer en ti misma, no podría haber tenido mejor madre que tú, los amo. A Diego, Edinson y Kevin Cando mis hermanos por estar presentes en mi vida, el amor que me entregan día con día y las porras que siempre escucho de su parte son las mejores, los amo. A Melissa y Dieguito Cando mis sobrinos a quienes adoro y amo como si fueran mis hijos deseo que crezcan sanos, fuertes y sean capaces de formarse como profesionales exitosos. A mi familia en general por su afecto a Diana y Víctor Montalván por su apoyo moral además de a David Miranda por el cariño que siempre me demostraste dios te mantenga siempre a su lado descansa en paz tío; a mi abuela Targelia por el cariño mostrado, a mi primo Luis por su apoyo. A Sergio Cando mi querido padrino un hombre a quien considero como un padre, el apoyo, cariño y respeto que siempre me ha mostrado lo hacen acreedor de mi admiración. A María Escaleras y Romina Gómez por su amistad y cariño, estoy feliz de que formen parte de mi familia. A Jessica Proaño y Mario Suarez, ustedes se ganaron mi cariño hemos pasado por muchas cosas durante nuestra vida universitaria siempre estuvieron ahí cuando los necesitaba por eso los considero parte de mi familia, mis hermanos los quiero. Mis amigos José Recalde, Jackeline Jiménez, Alex Rivera y demás quienes me han honrado con su amistad y cariño.

Carla

AGRADECIMIENTO

A Dios por todo lo bueno y malo que eh tenido que pasar durante mi vida, además de brindarme una familia, amigos y poder alcanzar poco a poco todo lo que me he propuesto. A mis adorados padres Roció y Wilmer, ustedes son los propulsores de todas mis metas cumplidas, me han inculcado valores que me han formado como una mujer responsable, fuerte, perseverante, humilde, luchadora y que siempre está buscando superarse. Hay tantas cosas que me gustaría agradecer, pero lo más importante es decir gracias por permitirme tenerlos en mi vida. A mis hermanos, sobrinos, tíos, tías, primos, cuñadas y familia en general por su ayuda, cariño y palabras de apoyo. También a Sergio Cando por estar pendiente de mi bienestar y avance en mis estudios. Gracias a todos. A mis amigos Jessica, Mario, José, Jackeline, Alex y demás por su amistad, ayuda y cariño gracias chicos. Al Sr. Milton Proaño y la Sra. Susana Vallejo quienes me abrieron las puertas de su hogar, me honraron con su amistad y cariño convirtiéndose así en mi segunda familia, gracias por todo. A la Escuela Superior Politécnica de Chimborazo, la Facultad de Informática y Electrónica, sus docentes quienes han compartido sus conocimientos y experiencias a lo largo de mi formación. Al Ing. Diego Veloz por su asesoramiento, ayuda, aporte, constancia y paciencia durante el desarrollo de este trabajo, así mismo al Ing. Alberto Arellano quien con su experiencia y conocimientos fue una guía en el progreso de este trabajo, incluyo al Ing. Edwin Altamirano por estar presente con su ayuda y compartiendo sus conocimientos durante toda mi formación académica. Por último, a quien inspiro este trabajo que además de ser una persona y profesional excepcional supo ganarse mi respeto y admiración, gracias Ing. Pablo Méndez.

Eli

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE ECUACIONES

ÍNDICE DE ANEXOS

ÍNDICE DE ABREVIATURAS

MECIB-AES: Mecanismo de cifrado basado en el algoritmo criptográfico simétrico AES

AES: Advanced Encryption Standard (Estándar de Encriptación Avanzada)

ENIAC: Computador e Integrador Numérico Electrónico

ASCII: Código Estándar Estadounidense para el Intercambio de Información

RSA: Sistema criptográfico de clave pública

MIT: Instituto de Tecnología de Massachusetts

SSL: Secure Sockets Layer (Protocolos criptográficos para comunicación)

VPN: Red privada virtual (Virtual Private Network)

DES: Data Encryption Standard (algoritmo de cifrado)

IDEA: International Data Encryption Algorithm

DSS: Digital Signature Standard

NIST: National Institute of Standards and Technology

RC2: Código de Ron diseño 2

ANSI: American National Standards Institute

JVM: Java Virtual Machine

API: Application Programming Interface

JCA: Java Cryptography Architecture

JCE: Java Cryptography Extension

IAIK: Institute of Applied Information Processing and Communications

ECB: Electronic Code Book Mode

CBC: Cipher Block Chaining Mode

CFB: Cipher Feedback Mode

RESUMEN

El presente trabajo se basó en el desarrollo del mecanismo de cifrado basado en el algoritmo criptográfico simétrico AES (MECIB-AES) para comparar la seguridad que brinda este a la información cifrada. El aporte se basa en analizar diferentes textos cifrados tomando en consideración la entropía, análisis de frecuencia, autocorrelación, fuerza bruta y análisis de tráfico; brindando una perspectiva del nivel de seguridad que presenta esta propuesta en comparación a la que está basada. La implementación de las modificaciones Mix-Shift, Mix-Key y Move-C, ayudó a realizar diferentes pruebas donde se aceptó la hipótesis nula la cual midió la entropía, con un nivel de confiabilidad del 95% y un error del 5%, el análisis de frecuencias presentó variaciones en cada prueba realizada, la autocorrelación dió como resultado una mayor similitud de secuencias a favor del MECIB-AES, aunque puede tomarse como desventaja que los valores no son grandes por lo cual se consideró viable el algoritmo. El análisis de fuerza bruta, donde se toma en consideración el tiempo, así como las posibles combinaciones de contraseñas, dió como resultado según sus medias Xa= 4,14E+25 y Xm= 4,24E+25 un aumento del 2,47% equivalente a 1,05E+24 años a favor del MECIB-AES. El análisis de tráfico realizado a los archivos dió como resultado una menor latencia y una mayor tasa de transferencia; notando una disminución del 43,19 % (latencia) y un aumento del 31,82 % (tasa de transferencia). Los resultados nos demuestran que es aceptable y viable la realización de esta modificación, por lo tanto, se impulsa la posibilidad de realizar mayores cambios en un futuro.

PALABRAS CLAVE: <SEGURIDAD DE REDES>, <CRIPTOGRAFÍA>, <ALGORITMO DE ENCRIPTACIÓN AVANZADA (AES)>, <SEGURIDAD CRIPTOGRÁFICA>, <ANÁLISIS DE TRÁFICO>, <CRIPTOGRAFÍA SIMÉTRICA>.

ABSTRACT

The present work was based on the development of the encryption mechanism based on the AES symmetric cryptographic algorithm (MECIB-AES) to compare the security that is provided with the encrypted information. The text is based on analyzing the different encrypted texts, taking into consideration, entropy, frequency analysis, autocorrelation, brute force, and traffic analysis; provide a perspective of the security level that presents this proposal compared to the one that is based. The implementation of the Mix-Shift, Mix-Key and Move-C modifications helped to perform different tests where the null hypothesis was accepted which measured the entropy, with a confidence level of 95% and an error of 5%, the analysis of frequency changes in each test performed, the autocorrelation gave rise to a greater similarity of sequences in favor of MECIB-AES, although it can be taken as a disadvantage that the values are not large, for which the algorithm was considered variable. The brute force analysis, where the time is taken into account, as well as the possible combinations of passwords, is given according to their means Xa= 4,14E+ 25 and Xm= 4,24E+25 an increase of 2, 47 % equivalent to 1,05E+24 years in favor of MECIB-AES. The analysis of the traffic made to the files resulted in lower latency and a higher transfer rate; observing a decrease of 43.19% (latency) and an increase of 31.82% (transfer rate). The results show us that this modification is acceptable and viable, therefore, it is possible to make major changes in the future.

KEYWORDS: <NETWORK SECURITY>, <CRYPTOGRAPHY> <ADVANCED ENCRYPTION ALGORITHM (AES)>, <CRYPTOGRAPHIC SECURITY>, <TRAFFIC ANALYSIS>, < SYMMETRIC CRYPTOGRAPHY>.

INTRODUCCIÓN

ANTECEDENTES

La necesidad de transmitir información manteniendo la confidencialidad hizo que los medios para ocultar información se desarrollen hace milenios ya sea por jeroglíficos, pergaminos, ruidos animales, pinturas en piedra, señales de humo, gestos, entre muchos más, debido a lo precario de estas formas de remitir información los gobiernos y ejércitos utilizaban formas de comunicación muy vulnerables. (La Criptografia y la Proteccion a la Informacion Digital, 2010 págs. 59-90)

Actualmente, debido a las altas tasas de intercambio de datos a través de redes de computadoras, internet, redes sociales, etc…, es necesario mayor seguridad porque existen personas que desean obtener la información con el fin de utilizarla para perjudicar a otras. (Jyoti, y otros, 2013)

Dado que la criptografía es una forma de brindar seguridad para guardar o enviar información los estudios relacionados a esta materia se remontan a tiempos antiguos, por lo tanto, la primera técnica criptográfica de la cual se tiene conocimiento es la de escitalo espartano, que data del siglo V a.c. (La Criptografia y la Proteccion a la Informacion Digital, 2010 págs. 59-90)

El escitalo es una vara en la que se coloca un pergamino de forma que lo enrolle, el emisor escribe el mensaje a través del escitalo, después de esto desenrolla el pergamino de tal forma que las letras del mensaje quedan traspuestas; luego, para descifrar el mensaje, se enrolla nuevamente en un escitalo en donde se verá claramente el mensaje enviado. (La Criptografia y la Proteccion a la Informacion Digital, 2010 págs. 59-90)

Luego aparece Julio César, quien utilizó técnicas esteganograficas, criptográficas y de codificación para asegurar la confidencialidad de sus mensajes, los cuales fueron vitales para la consecución de la expansión de su imperio. (La Criptografia y la Proteccion a la Informacion Digital, 2010 págs. 59-90)

Además, se ha encontrado que culturas mesoamericanas, como la azteca, cifraban su información a través de ideogramas. En Occidente, el mundo árabe y la antigua China se desarrollaban métodos de cifrado y ocultación: en China se ocultaban los mensajes escribiéndolos en seda, luego la tela era aplastada y se recubría de cera. (La Criptografia y la Proteccion a la Informacion Digital, 2010 págs. 59-90)

El mundo árabe, en lo que actualmente es Irak, se desarrolló el criptoanálisis de los métodos por sustitución. (La Criptografia y la Proteccion a la Informacion Digital, 2010 págs. 59-90)

En el siglo XVI, el aporte más importante lo realiza Blaise de Vigenère, quien diseñó criptosistemas polialfabéticos y, desde ese momento, se empiezan a usar los sistemas de alfabetos múltiples y los de transposición sencillas y múltiples, métodos que son la base de los actuales criptosistemas. (La Criptografia y la Proteccion a la Informacion Digital, 2010 págs. 59-90)

En 1839 con el sistema Wheatstone-Cooke se desarrolló el telégrafo, luego con los progresos de morse, el sistema cobró total popularidad y se adoptó la implementación del telégrafo trasmitiendo mensajes con el Código Morse. (La Criptografia y la Proteccion a la Informacion Digital, 2010 págs. 59-90)

Pero este avance tan importante presentaba una deficiente seguridad en lo que respecta a la confidencialidad, además con esta tecnología se empieza a dar un crecimiento notable en las comunicaciones, ya que la velocidad y la facilidad para transmitir mensajes hizo que las relaciones fueran más dinámicas. (La Criptografia y la Proteccion a la Informacion Digital, 2010 págs. 59-90)

Guglielmo Marconi a finales del siglo XIX, realizó progresos en el área de las telecomunicaciones, inventó la radiotransmisión. La comunicación por radio carecía de seguridad alguna ya que la señal una vez emitida podría ser interceptada por cualquiera que dispusiera de un receptor, lo que implicaba un desequilibrio entre seguridad y capacidad de comunicación. (La Criptografia y la Proteccion a la Informacion Digital, 2010 págs. 59-90)

Por otro lado, durante todas las guerras se ha buscado transmitir información de forma segura y para esto utilizar criptosistemas que permitan una comunicación confiable porque se realiza el mismo esfuerzo en atacar los sistemas de comunicación del oponente, interceptar las comunicaciones y, en caso de estar cifrada la información, descifrarla. (La Criptografia y la Proteccion a la Informacion Digital, 2010 págs. 59-90)

Ese esfuerzo por descifrar los mensajes ha hecho evolucionar los sistemas criptográficos y a la vez impulsar el desarrollo de la matemática y la informática. Después de la Segunda Guerra Mundial, no solamente aumentó el desarrollo y la investigación en criptografía, sino que además proliferó el interés en este tema dentro de la población en general. (La Criptografia y la Proteccion a la Informacion Digital, 2010 págs. 59-90)

Por esta razón es importante notar que muchos de los progresos que se han realizado en los ordenadores modernos han sido generados para tener una aplicación primigenia en la criptografía. Por el mismo camino, los primeros ordenadores creados alrededor de la Segunda Guerra Mundial fueron herramientas de cifrado y descifrado de información para las agencias de seguridad de Estados Unidos. (La Criptografia y la Proteccion a la Informacion Digital, 2010 págs. 59-90)

Desde la creación del primer computador moderno, el ENIAC, se empezaron a convertir los mensajes en códigos binarios, es decir en ceros y unos, utilizando protocolos como el ASCII. Cada letra de cada palabra para su transmisión se convierte en series de ceros y unos, a los cuales para cifrar el mensaje se les aplica el criptosistema. (La Criptografia y la Proteccion a la Informacion Digital, 2010 págs. 59-90)

Es así como los computadores dejaron de ser artículos exclusivos de agencias estatales para pasar a ser utilizados por empresas en actividades civiles de todo tipo: comerciales, estadísticas, comunicaciones, etc. (La Criptografia y la Proteccion a la Informacion Digital, 2010 págs. 59-90)

Con la llegada de sistemas computacionales cada vez más potentes y con el aumento del uso de estos equipos para procesar y transmitir información, se presentaron nuevas necesidades frente a la criptografía, surgiendo la necesidad de estandarizar el uso de un sistema criptográfico para hacer transmisiones sin problemas de compatibilidad. (La Criptografia y la Proteccion a la Informacion Digital, 2010 págs. 59-90)

Esta es la principal razón por la cual se propone el desarrollo de la siguiente propuesta tecnológica que consiste en modificar el proceso de cifrado del algoritmo criptográfico simétrico AES (Advanced Encryption Standard), lo cual permite medir el nivel de confiabilidad que presenta además de convertirse en un antecedente que podría ayudar a la evolución del mismo.

FORMULACIÓN DEL PROBLEMA

¿Cuál sería el nivel de confiabilidad que brinda el desarrollo de un mecanismo de cifrado basado en el algoritmo criptográfico simétrico AES?

SISTEMATIZACIÓN DEL PROBLEMA

¿Cuáles son las características, ventajas y desventajas del algoritmo criptográfico simétrico AES?

¿Cómo desarrollar un mecanismo de cifrado en el algoritmo criptográfico simétrico AES?

¿Podrá mejorar el rendimiento del algoritmo criptográfico simétrico AES el desarrollo de un mecanismo de cifrado?

JUSTIFICACIÓN TEÓRICA

La criptografía es una forma de almacenar o transmitir información valiosa, evitando que sea obtenida por terceros debido a que ésta no es transparente para todos los usuarios. Dado que hay un proceso de descifrado la información puede volver a su estado normal. (Méndez Naranjo, 2015), (Penazzi, 2006)

Tomando en cuenta que existen dos tipos de criptografía simétrica y asimétrica para el desarrollo de esta propuesta se eligió la criptografía simétrica y a su vez el algoritmo criptográfico simétrico Advanced Encryption Standard o AES, por: (Penazzi, 2006)

Si bien la criptografía de clave pública/privada es muy interesante y revolucionó la criptografía en los 1970, no es adecuada para encriptamiento de archivos grandes. El núcleo de las aplicaciones criptográficas han sido siempre los algoritmos de clave simétrica, en donde ambas partes deben conocer la clave para poder comunicarse. En particular los algoritmos de bloque siempre han sido muy populares. (Penazzi, 2006)

Quizás el más famoso de ellos fue el DES (Data Encryption Standard), que fue el estándar norteamericano (y de facto mundial) desde los 1970s. A fines de los 1990s, sin embargo, estaba claro que estaba llegando al fin de su vida útil, y el gobierno norteamericano lanzó un llamado internacional para un nuevo estándar. De entre 15 presentaciones, resulto elegido un algoritmo, Rijndael, que se convirtió en el AES (Advanced Encryption Standard). (Penazzi, 2006)

Las razones de la elección de Rijndael como AES son diversas. Entre ellas estuvo el hecho de su estructura simple y matemática, que no dejaba lugar a ninguna duda de que pudiera haber una "trapdoor" oculta; su gran rapidez y versatilidad de implementación, tanto en procesadores de 8 bits, como de 32 bits, así como en hardware; y su demostrable seguridad contra los dos ataques más importantes de la década de 1990: el criptoanálisis diferencial y el criptoanálisis lineal. (Penazzi, 2006)

El DES dejó de ser el cifrado estándar en el año 2000, cuando el NIST (Acrónimo de National Institute for Standards and Technology) declaró como nuevo estándar el sistema AES, también conocido como Rijndael, por las iniciales de sus creadores, los belgas Rijmen y Daemen. Este criptosistema trabaja con longitudes de clave variable entre 128 y 256 bits, el descifrado de este sistema se realiza mediante el mismo algoritmo de cifrado aplicado de forma inversa. Hasta el

momento este algoritmo ha pasado todas las pruebas de seguridad a las que se ha sometido. (La Criptografia y la Proteccion a la Informacion Digital, 2010)

El beneficio del uso de un algoritmo simétrico radica en el procesamiento rápido para encriptar y desencriptar un alto volumen de datos. Las principales desventajas de los métodos simétricos son la distribución y gestión de claves. (Demostracion de Cifrado Simétrico y Asimétrico, 2008)

JUSTIFICACIÓN APLICATIVA

Al verificar el funcionamiento de la clave y su comportamiento se realizará la adaptación de un mecanismo para medir el nivel de confiabilidad que brinde este a la información, cuyo procedimiento podría ayudar a evolucionar los algoritmos criptográficos simétricos en especial el AES (Advanced Encryption Standard).

Al finalizar la implementación de la modificación, se realizan pruebas para comparar el mecanismo implementado con el existente, cuyo propósito será medir el nivel de confiabilidad alcanzado.

OBJETIVOS

Objetivo General

- ✓ Desarrollar un mecanismo de cifrado basado en el algoritmo criptográfico simétrico AES.

Objetivos Específicos

- ✓ Analizar el funcionamiento del algoritmo criptográfico simétrico AES.
- ✓ Desarrollar el mecanismo de cifrado basado en el algoritmo criptográfico simétrico AES.
- ✓ Realizar pruebas para medir el nivel de confiabilidad del mecanismo desarrollado en comparación con el usado por el algoritmo criptográfico simétrico AES.

CAPITULO I

1. MARCO TEÒRICO

Este capítulo tiene la información correspondiente a criptografía y sus diferentes clasificaciones, así como también del algoritmo criptográfico simétrico AES (Advanced Encryption Standard) donde se analizará cada función para su posterior implementación.

1.1 CRIPTOGRAFÍA

1.1.1 Definición

Una técnica a la cual recurrir para cifrar información, criptografía proviene del griego *Kryptos* y *Graphein*, que significan "escondido" y "escritura", respectivamente, además designada como escritura secreta, porque el cifrado oculta información para evitar el descifrado por personas indiscretas. (Cushpa Guamán, 2018), (La Criptografia y la Proteccion a la Informacion Digital, 2010)

Otras definiciones para criptografía son:

Según el Diccionario de la Real Academia Española, este define a la palabra criptografía como: "Arte de escribir con clave secreta o de un modo enigmático"

De igual forma la RFC 2828 (Request For Comments) define la criptografía como: "Un conjunto de algoritmos criptográficos junto con los procesos de gestión de claves que apoyan el uso de los algoritmos en un contexto de aplicación."

1.1.2 Clasificación

La criptografía proviene de una rama de las matemáticas, que fue iniciada por el matemático Claude Shannon en 1948, denominada: "Teoría de la Información". Esta rama de las ciencias se divide en: Teoría de Códigos y Criptología. A su vez la Criptología se divide en Criptoanálisis y Criptografía. **Figura 1-1.** (Introducción a la Criptografía, 2006)

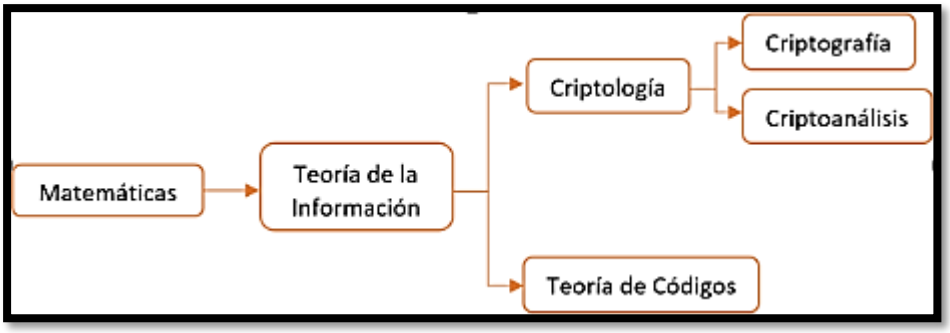


Figura 1-1. Origen de la Criptografía

Realizado por: Raza C., 2019

Fuente: (Introducción a la Criptografía, 2006)

Debido a esto se debe diferenciar entre proceso de cifrado y codificación: un criptosistema sirve para realizar el cifrado de cualquier mensaje dentro del alfabeto que esté construido; mientras que los códigos establecen una relación limitada dentro de cada asignación, así cada código está restringido al significado que se le haya establecido, funciona como un diccionario. (La Criptografia y la Proteccion a la Informacion Digital, 2010 pág. 61)

Según Cadavid J. A. la criptografía tiene relación con múltiples áreas del conocimiento como:

- ✓ La teoría de la información
- ✓ Teoría de conjuntos
- ✓ Informática
- ✓ La importancia de los números primos
- ✓ Las telecomunicaciones, hace necesario un estudio interdisciplinario para el desarrollo de criptosistemas robustos.

Como Medina Vargas Y., menciona la criptografía presenta 5 objetivos principales los cuales son:

- ✓ Autenticación: Comprobación de la identidad del receptor como emisor del mensaje. (Cushpa Guamán, 2018)
- ✓ Privacidad/Confidencialidad: Afirmar que el mensaje cifrado no sea leído por ninguna persona solo el receptor. (Cushpa Guamán, 2018)
- ✓ Integridad: La confianza de saber que el mensaje no posee ninguna alteración. (Cushpa Guamán, 2018)
- ✓ No repudio: Un mecanismo para probar que el remitente realmente envió este mensaje. (Cushpa Guamán, 2018)

- ✓ Servicio Fiabilidad y disponibilidad: Desde sistemas seguros generalmente atacados por indiscretos, lo cual afecta la disponibilidad y la asistencia a usuarios. Estos medios suministran una forma de otorgar a sus usuarios la calidad de servicio que esperan. (Cushpa Guamán, 2018)

1.1.3 Evolución de la Criptografía

La criptografía a su vez se clasifica en clásica y moderna.

1.1.3.1 Criptografía Clásica

La criptografía clásica se basa en algoritmos sencillos y claves muy largas para la seguridad. Actualmente, han perdido su eficacia puesto que resultan fácilmente criptoanalizables empleando cualquier ordenador doméstico. (Moya, 2013)

En este punto vamos a tratar brevemente un ejemplo del cifrado de César, siendo uno de los más antiguos que se conocen, debe su nombre al emperador Julio César, utilizado para establecer comunicaciones seguras con sus generales durante las Guerras Gálicas. (Huerta, 2003)

Ejemplo: (Huerta, 2003)

- ✓ Para el cifrado de la palabra **SEGURIDAD** utilizamos el alfabeto y valores enteros como se presenta en la **Tabla 1-1**.

Tabla 1-1. Alfabeto con su equivalencia numérica

A=1	B=2	C=3	D=4	E=5	F=6
G=7	H=8	I=9	J=10	K=11	L=12
M=13	N=14	O=15	P=16	Q=17	R=18
S=19	T=20	U=21	V=22	W=23	X=24
Y=25	Z=26				

Realizado por: Raza C., 2019

Fuente: (Huerta, 2003)

- ✓ Para trabajar matemáticamente utilizaremos la función f(x)=ax+b, siendo x el valor numérico asignado a cada letra y a, b la clave.
- ✓ Buscamos la equivalencia numérica de la palabra a cifrar y utilizando una clave (1, 5); traducido sencillamente que para cifrar una letra hemos de tomar su entero correspondiente y sumarle 5, como se presenta en la **Tabla 2-1**.

Tabla 2-1. Ejemplo del Cifrado de César

Palabra Original	**S**	**E**	**G**	**U**	**R**	**I**	**D**	**A**	**D**
Valor Numérico Original	19	5	7	21	18	9	4	1	4
A los valores anteriores le aplicaremos la función $f(x)= x+5$, obteniendo así:									
Palabra Cifrada	**X**	**J**	**L**	**Z**	**W**	**N**	**I**	**F**	**I**
Valor Numérico Cifrado	24	10	12	26	23	14	9	6	9

Realizado por: Raza C., 2019

- ✓ Llegando las siguientes letras al receptor **XJLZWNIFI**; una vez recibido el mensaje tendrá que ser descifrado utilizando la función inversa a la del cifrado $(f^{-1}(x)= ax-b)$ y la clave (debe ser establecida con anticipación) como se muestra en la **Tabla 3-1**:

Tabla 3-1. Ejemplo del Descifrado de César

Palabra Cifrada	**X**	**J**	**L**	**Z**	**W**	**N**	**I**	**F**	**I**
Valor Numérico Cifrado	24	10	12	26	23	14	9	6	9
A los valores anteriores le aplicaremos la función $f(x)=x-5$, obteniendo así:									
Palabra Descifrada	**S**	**E**	**G**	**U**	**R**	**I**	**D**	**A**	**D**
Valor Numérico Descifrado	19	5	7	21	18	9	4	1	4

Realizado por: Raza C., 2019

Cuyo resultado final sería el mensaje "**SEGURIDAD**"

Todos los algoritmos criptográficos clásicos son simétricos y, por tanto, emplean la misma clave para cifrar y descifrar. (Moya, 2013)

Clasificación de la Criptografía Clásica:

La clasificación la podemos observar en la **Figura 2-1.**

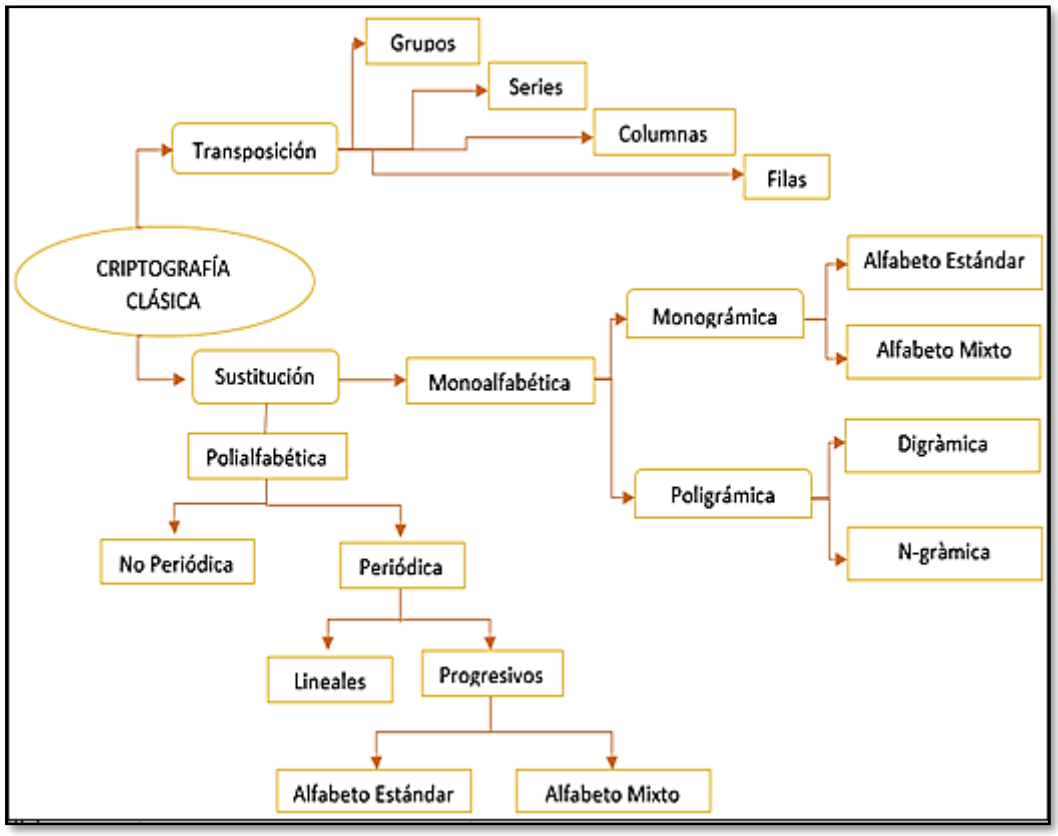


Figura 2-1. Clasificación de la Criptografía Clásica

Realizado por: Raza C., 2019

Fuente: (Introducción a la Criptografía, 2006)

Ejemplos:

✓ Método de Transposición de la Escitala

Se utilizó un aparato cilíndrico de madera al cual se le enrollaba una cinta de papiro y en el cual se escribía el mensaje a lo largo, era dificultoso entender el mensaje sin la escitala como se muestra en la **Figura 3-1**. (Moya, 2013)

Figura 3-1. La Escitala

Fuente: http://ojoscuriosos.com/wp-content/uploads/2013/12/images-13.jpg

✓ Método de Sustitución, Monoalfabético, Monográmico de César

Dado el nombre por Julio César, el tipo de alfabeto y la forma de ordenar las letras para realizar el cifrado. En el proceso se realiza un desplazamiento de un valor n y sustituir el carácter del mensaje en claro por otro n posiciones después como se observa en la **Figura 4-1**. (Moya, 2013)

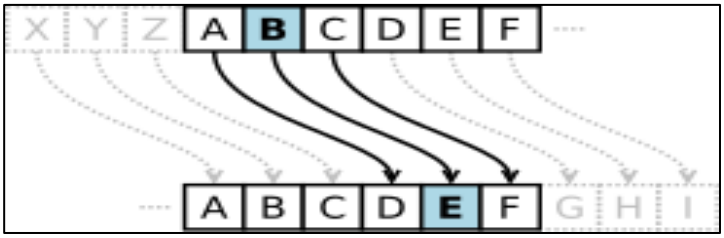


Figura 4-1. Cifrado César

Fuente: https://upload.wikimedia.org/wikipedia/commons/thumb/2/2b/Caesar3.svg/245px-Caesar3.svg.png

✓ Método de Sustitución, Monoalfabético, Poligrámica de Playfair

El método de cifrado trabaja con dos caracteres (bigrama) a la vez, por lo que el texto en claro se debe descomponer en parejas de dos caracteres. Cada una de las parejas de caracteres obtenidas después de la descomposición se sustituye por otra conforme a las siguientes reglas: (Martínez, 2009)

1. Si las dos letras se encuentran en el mismo renglón de la matriz antes construida, cada una de ellas se sustituye con la letra que esté a su derecha.
2. Si las dos letras se encuentran en la misma columna, cada una de las letras se sustituye por la letra que este debajo de ella.
3. La primera letra de la pareja se sustituye por la que este en la intersección de su misma fila y la columna de la segunda letra, la segunda letra se sustituye por la que este en la intersección de su misma fila y la columna de la primera letra.
4. Si la pareja está conformada por la misma letra, entonces se debe descomponer dicha pareja en dos nuevas parejas de la siguiente manera: suponiendo que la pareja es AA las nuevas dos parejas son AX y AX.
5. Si el número de caracteres del mensaje en claro es impar, se debe agregar una 'X' para poder formar todas las parejas.

Ejemplo:

✓ Encriptar la palabra **SEGURIDAD**

✓ Con la clave **CRIPTOSISTEMAPLAYFAIR**

Llenaremos de izquierda a derecha y arriba hacia abajo y omitiremos la W si una letra se repite no es necesario volver a escribirla y después se llena la tabla con las letras del alfabeto no utilizadas como se muestra en la **Tabla 4-1.**

Tabla 4-1. Ejemplo de Cifrado Playfair

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

Realizado por: Raza C., 2019

- ✓ Las letras **SE** son reemplazadas por **EM**
- ✓ Las letras **GU** son reemplazadas por **HQ**
- ✓ Las letras **RI** son reemplazadas por **IP**
- ✓ Las letras **DA** son reemplazadas por **ND**
- ✓ Las letras **DX** son reemplazadas por **BZ**

SEGURIDAD se encriptará como "**EMHQIPNDBZ**"; al aplicar las reglas en sentido contrario se obtendrá el mensaje original.

- ✓ Método de Sustitución, Polialfabético, Periódicos de Vigenère

Se utiliza varios alfabetos para cifrar el mensaje. Se considera periódico debido a la clave utilizada para cifrar el mensaje. Se asigna a cada letra un número entero que sumada con el mensaje daría origen al mensaje cifrado. (Moya, 2013)

Para cifrar se procede de la siguiente manera: (La Criptografía Clásica, 2004)

- ✓ Se busca una palabra clave fácil de recordar.
- ✓ Se escribe la palabra debajo del texto en claro, repitiéndose tantas veces como sea necesario.
- ✓ Cada letra del texto en claro se codifica con el alfabeto de la tabla marcado por la letra inferior, o sea, la letra de la clave que corresponde, como se muestra en la **Tabla 5-1.**

Ejemplo: (Gómez, 2010)

Clave = **SEGURIDAD**
Texto a remitir: **CIFRADOVIGENERE**
Ponemos la clave bajo el mensaje a cifrar repitiendo hasta que termine el mensaje.

Tabla 5-1. Ejemplo de Cifrado Vigénere

C	I	F	R	A	D	O	V	I	G	E	N	E	R	E
S	E	G	U	R	I	D	A	D	S	E	G	U	R	I

Realizado por: Raza C., 2019

Las letras que conforman el cifrado son obtenidas de la intersección en el tablero de Vigénere (**Figura 5-1**) de la clave y el texto a remitir.

Tablero de Vigenère

0	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
2	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
3	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
4	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
5	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
6	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
7	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
8	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
9	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
10	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
11	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
12	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
13	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
14	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
15	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
16	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
17	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
18	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
19	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
20	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
21	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
22	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
23	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
24	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
25	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Figura 5-1. Tablero de Vigénere

Fuente: (La Criptografía Clásica, 2004)

Así el texto cifrado es: **UMLLALR VLYITYIM**

Para el descifrado:

Se busca la letra del texto cifrado en la fila de la letra (clave) a la que corresponde, una vez encontrada se obtiene la letra de la columna a la cual pertenece y se forma el mensaje original.

Ejemplo:

En la fila de la letra **S** buscamos la letra **U** y así obtenemos la **C**.

1.1.3.2 Criptografía Moderna

La criptografía antigua se vio desplazada debido a la velocidad de cálculo, avance de las matemáticas y la necesidad de mayor seguridad además se unió el hecho de que en 1883 el criptógrafo militar holandés A. Kercknoffs propuso un nuevo paradigma de cifrado: el cifrado público. (Moya, 2013)

En efecto, hasta entonces la robustez del cifrado residía en dificultad para romper el algoritmo de encriptación. Sin embargo, en el enfoque de Kercknoffs, tanto el algoritmo como el criptosistema son públicos, siendo la clave el único elemento que debe permanecer secreto según la **Figura 6-1.** (Moya, 2013)

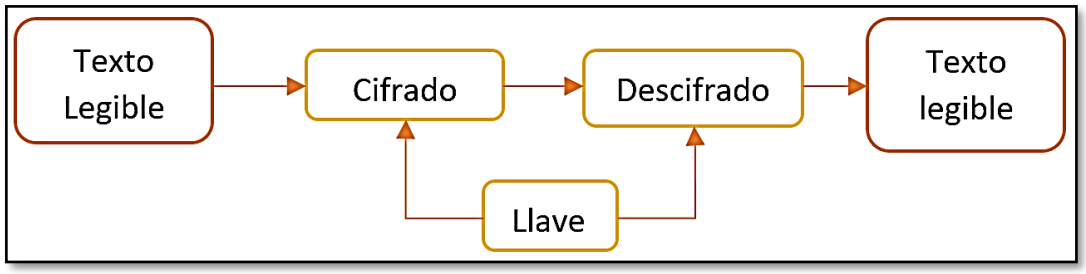


Figura 6-1. Esquema de un criptosistema.

Realizado por: Raza C., 2019

Fuente: (Moya, 2013)

Los estudios realizados por Claude Shannon "Teoría de la información y Criptología, 1948", Whitfield Diffie y Martin Hellman "Nuevo método de Cifrado, creando criptosistemas de clave pública, 1976" causaron un cambio en la criptografía y deja de ser considerada como un mero arte rodeado de un cierto aire de misterio y en algunos casos de escepticismo, para ser tratada como una rama más de las matemáticas. (Xifré, 2008-2009)

En el caso de los sistemas modernos hacen uso de algunas propiedades matemáticas como, por ejemplo, la dificultad del cálculo del logaritmo discreto o el problema de la factorización de grandes números, unido esto a la representación binaria de la información, por lo cual la criptografía moderna queda dividida en dos grandes grupos: criptografía de clave privada o criptografía de clave pública. (Xifré, 2008-2009)

Clasificación de la Criptografía Moderna

Según el tratamiento de la información

- ✓ Cifrado en flujo: cuando el mensaje se emite y se cifra a la vez, sin dividir el mensaje en partes. Esto es útil para sistemas de aplicación en telecomunicaciones, por ejemplo, en las comunicaciones móviles. (Moya, 2013)

- ✓ Cifrado en bloque: cuando antes de emitirse el mensaje se divide en bloques y se cifra cada uno de éstos por separado, utilizando la misma clave. Resulta útil para sistemas de protección de ficheros de ordenador. (Moya, 2013)

Según el tipo de clave:

- ✓ Sistemas de clave privada: en los cuales sólo hay involucrados un emisor y un receptor, que comparten una misma clave (secreta) para cifrar y para descifrar (simétrico), la cual debe permanecer en secreto. A estos sistemas se les denominan criptosistemas simétricos. (Moya, 2013)

- ✓ Sistemas de clave pública: en los cuales hay involucrados muchos usuarios que pueden comunicarse entre sí, cada uno con una clave privada (secreta) para poder leer los mensajes que van dirigidos a él, y una clave pública (conocida por todos los usuarios) para que cualquiera pueda enviarle un mensaje cifrado. Por tanto, estos criptosistemas también se llaman asimétricos. (Moya, 2013)

Una forma más sencilla de explicar la clasificación se muestra en la **Figura 7-1**:

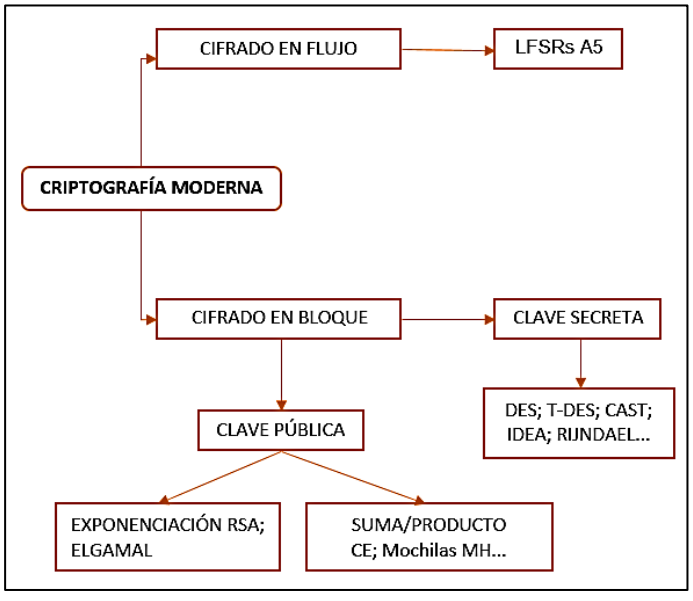


Figura 7-1. Clasificación de Criptografía Moderna

Realizado por: Raza C., 2019

Fuente: (Moya, 2013)

Los sistemas de clave privada son muy rápidos, frente a los de clave pública, pero carecen de firma digital, por lo que lo que se suele hacer es utilizar los primeros para el cifrado de la información y los segundos para el intercambio de las claves de cifrado y la firma de los documentos. (Moya, 2013)

1.1.4 Criptografía Asimétrica

El algoritmo de encriptación de Diffie-Hellman, creado en 1976, supuso una verdadera revolución en el campo de la criptografía, ya que fue el punto de partida para los sistemas asimétricos, basados en dos claves diferentes, la pública y la privada. (Moya, 2013)

Matemáticamente este algoritmo se basa en las potencias de los números y en la función *mod* (módulo discreto). Si bien el cálculo de potencias discretas es fácil, la obtención de su función inversa, el logaritmo discreto, no tiene una solución analítica para números suficientemente grandes. (Moya, 2013)

Su importancia se debe sobre todo al hecho de ser el inicio de los sistemas asimétricos (**Figura 8-1**), ya que en la práctica sólo es válido para el intercambio de claves simétricas, y con esta funcionalidad es muy usado en los diferentes sistemas seguros implementados en Internet, como SSL (Secure Socket Layer) y VPN (Virtual Private Network). (Moya, 2013)

Figura 8-1. Criptografía Asimétrica

Realizado por: Raza C., 2019

Fuente: (Introducción a la Criptografía, 2006)

Cuando se completa la generación de una clave asimétrica se define una clave de cifrado (clave pública) y una clave de descifrado (clave privada); la primera puede ser conocida por todo el mundo, pero, por otro lado, se debe tener mucho cuidado en ocultar la clave privada. Las claves asimétricas tienen la sorprendente propiedad de que lo que se está cifrando con una clave sólo se puede descifrar con la otra. (Demostracion de Cifrado Simétrico y Asimétrico, 2008 pág. 49)

Algoritmos de Clave Pública:

- ✓ RSA: Este sistema de clave pública fue diseñado en 1977 por los profesores del MIT (Massachusetts Institute of Technology) Ronald R. Rivest, Adi Shamir y Leonard M. Adleman, de ahí las siglas con las que es conocido. Desde entonces, este algoritmo de cifrado se ha convertido en el prototipo de los de clave pública. (Huerta, 2003 pág. 11)

La seguridad de RSA radica en la dificultad de la factorización de números grandes: es fácil saber si un número es primo, pero es extremadamente difícil obtener la factorización en números primos de un entero elevado, debido no a la dificultad de los algoritmos existentes, sino al consumo de recursos físicos (memoria, necesidades hardware... incluso tiempo de ejecución) de tales algoritmos. (Huerta, 2003 pág. 11)

Se ha demostrado que, si n es el número de dígitos binarios de la entrada de cualquier algoritmo de factorización, el coste del algoritmo es $q\ (2n)$, con un tiempo de ejecución perteneciente a la categoría de los llamados problemas intratables. (Huerta, 2003 pág. 11)

- ✓ ElGamal: Durante 1984 y 1985 ElGamal desarrolló un nuevo criptosistema de clave pública basado en la intratabilidad computacional del problema del logaritmo discreto: obtener el valor de x a partir de la expresión $y=a^x\ (mod\ p)$ es, como hemos visto para el caso de RSA, computacionalmente intratable por norma general. (Huerta, 2003 pág. 11)

Aunque generalmente no se utiliza de forma directa, ya que la velocidad de cifrado y autenticación es inferior a la obtenida con RSA, y además las firmas producidas son más largas (el doble de largo que el texto original), el algoritmo de ElGamal es de gran importancia en el desarrollo del DSS (Digital Signature Standard), del NIST (National Institute of Standards and Technology) estadounidense. (Huerta, 2003 pág. 11)

En la práctica, debido a que los algoritmos de clave pública requieren mucho tiempo para cifrar documentos largos, los protocolos de firma digital se implementan junto con funciones unidireccionales de resumen (funciones *hash*), de manera que, en vez de firmar un documento, se firma un resumen del mismo. (Moya, 2013)

Aplicaciones

- ✓ Una de las aplicaciones inmediatas de estos algoritmos es el cifrado de la información sin tener que transmitir la clave de decodificación, lo cual permite su uso en canales inseguros. (Moya, 2013)

- ✓ Autentificación de mensajes mediante la firma digital de los mismos. Genera una firma digital mucho más pequeña que el mensaje original de manera que resulte muy difícil (en el caso ideal, imposible) encontrar otro mensaje que dé lugar a la misma. (Moya, 2013)

1.1.5 Criptografía Simétrica

La criptografía simétrica utiliza la misma clave para cifrar y descifrar el mensaje, es decir se basa en un secreto compartido (**Figura 9-1**). Debido a esto la seguridad de este proceso depende de la posibilidad de que una persona no autorizada consiga la clave de sesión o clave secreta. (Demostracion de Cifrado Simétrico y Asimétrico, 2008 pág. 47)

Los algoritmos criptográficos simétricos tienen dos versiones: cifrador en bloque y cifrador en flujo. Los cifradores en bloque codifican datos en bloques pequeños de longitud fija de 64 bits de longitud. Hay muchos cifradores en bloque que incluyen DES, 3DES, RC2, RC5, RC6 y Rijndael (conocido como AES). (Demostracion de Cifrado Simétrico y Asimétrico, 2008 pág. 47)

Así pues, después del cifrado de cada carácter, el sistema evoluciona a un nuevo estado de acuerdo con una determinada regla. Como consecuencia de ello sucede que caracteres idénticos poseen por lo general cifrados diferentes, lo que contribuye a aumentar la seguridad del sistema. (Xifré, 2008-2009)

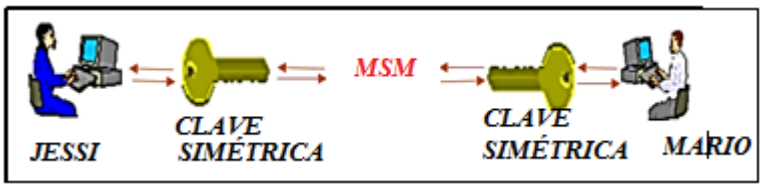


Figura 9-1. Criptografía Simétrica

Realizado por: Raza C., 2019

Fuente: (Xifré, 2008-2009)

Algunos de los principales algoritmos de clave simétrica, son:

- ✓ DES (Data Encryption Standard): Su arquitectura está basada en un sistema monoalfabético, donde un algoritmo de cifrado aplica sucesivas permutaciones y sustituciones al texto en claro.

- ✓ 3DES (Triple Data Encryption Standard): Se utiliza el algoritmo DES 3 veces, cuya clave tiene 128 bits. Al cifrar la misma información 2 veces con 2 llaves diferentes, la clave aumenta de tamaño. (Cushpa Guamán, 2018) (Méndez Naranjo, 2015), (Sergio, 2010)

El aumento de seguridad del algoritmo DES es significativo, pero demanda más recursos del computador. (Cushpa Guamán, 2018), (Méndez Naranjo, 2015), (Sergio, 2010)

- ✓ RC5 (Rivest Cipher 5): Se aplican operaciones XOR sobre los datos, siendo de 32, 64 o 128 bits. Consiente otras longitudes de clave e iteraciones variables, también funciona como un generador de número aleatorios, sumándoles a los bloques de texto rotados mediante la XOR. (Sergio, 2010)

- ✓ IDEA (International Data Encryption Algorithm): Con clave de 128 bits sin paridad a bloques de 64 bits utilizado para cifrar y descifrar. Se alteran los datos de entrada en una secuencia de iteraciones parametrizadas, con el objetivo de producir bloques de salida de texto cifrado de 64 bits. (Sergio, 2010)

IDEA combina operaciones matemáticas como XOR, sumas con acarreo de módulo 2^{16} y multiplicaciones de módulo $2^{16}+1$, sobre bloques de 16 bits. (Sergio, 2010)

- ✓ AES (Advanced Encryption Standard): Trabaja con bloques y claves de longitudes variable, como 128, 192 y 256 bits. Una matriz de 4x4, a la cual se le aplica varias funciones matemáticas. (Sergio, 2010)

1.2 ALGORITMO AES (ADVANCED ENCRYPTION STANDARD)

1.2.1 Historia

Proviene del desarrollo de un concurso convocado por el Instituto Nacional de Normas y Tecnologías (NIST), el mismo que expuso varios requisitos y factores de evaluación para que un algoritmo sea considerado como candidato los cuales a continuación detallamos: (Vázquez, 2007)
Requisitos: (Vázquez, 2007)

1. Ser público y estar: (Vázquez, 2007)
- ✓ Disponible gratuitamente.
- ✓ Disponible bajo términos consistentes con la política de patentes del Instituto Nacional Americano de Estándares (ANSI).
2. Algoritmo de cifrado en bloque simétrico, (Vázquez, 2007).
3. Diseñado para que se pueda aumentar la longitud de clave según las necesidades, (Vázquez, 2007).
4. Implementable en Hardware como Software, (Vázquez, 2007).

Factores: (Vázquez, 2007)

1. Seguridad.
2. Eficiencia computacional.
3. Requisitos de memoria.
4. Adecuación Hardware y Software.
5. Simplicidad de diseño.
6. Flexibilidad, (Vázquez, 2007)

También, soportar obligatoriamente una longitud de bloque de 128 bits como mínimo, y una longitud de clave de 128, 192 y 256 bits, al margen de cualesquiera otras longitudes posibles. (Vázquez, 2007)

El Instituto Nacional de Normas y Tecnologías (NIST) permitió que cualquier persona natural o jurídica pudiera participar en dicho concurso; para lo cual, desarrollo la competencia en dos rondas, la primera eligió los 5 algoritmos que cumplieron con los requisitos y factores (detallados anteriormente) de entre todos los concursantes y la segunda ronda elegiría al ganador como podemos observar en la **Figura 10-1**: (Vázquez, 2007)

Participantes que cumplieron los requisitos y factores a evaluar.	Resultados Ronda 1	Resultados Ronda 2
1. CAST-256 Entust Technologies, Inc. (C. Adams). 2. CRYPTION Future Systems, Inc. (Chae Hoon Lim). 3. DEAL L. Knudsen, R. Outerbridge. 4. DFC CNRS-Ecole Normale Superiere (S. Vaudenay). 5. E2 NTT Nippon Telegraph and Telephone Corporation (M. Kanda). 6. FROG TecApro International S.A. (D. Georgoudis, Leroux, Chaves). 7. HPC R. Schoeppel. 8. LOKI97 L. Brown, J. Pieprzyk, J. Seberry. 9. MAGENTA Deutshe Telekom AG (K. Huber). 10. MARS IBM (Nevenko Zunic). 11. RC6 RSA Laboratories (Rivest, M. Robshaw, Sidney, Yin). 12. Rijndael Joan Daemen, Vicent Rijmen. 13. SAFER+ Cylink Corporation (L. Chen). 14. SERPENT R. Anderson, E. Biham, L. Knudsen. 15. TWOFISH B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, N. Ferguson.	1. MARS. 2. RC6. 3. RIJNDAEL. 4. SERPENT. 5. TWOFISH.	2 de octubre de 2000, el NIST anunció el algoritmo ganador: **Rijndael** propuesto por los belgas Vicent Rijmen y Joan Daemen

Figura 10-1. Fases para la elección del Algoritmo de Encriptación Avanzada

Realizado por: Raza C., 2019

Fuente: (Pousa, 2011), (Vázquez, 2007)

Las razones de la elección de Rijndael como AES (Advanced Encryption Standard) son diversas entre ellas: (Penazzi, 2006)

- ✓ Su estructura simple y matemática, que no dejaba lugar a ninguna duda de que pudiera haber una "trapdoor" oculta.
- ✓ Su gran rapidez y versatilidad de implementación, tanto en procesadores de 8 bits, como de 32 bits, así como en hardware.
- ✓ Su demostrable seguridad contra los dos ataques más importantes de la década de 1990: el criptoanálisis diferencial y el criptoanálisis lineal.

1.2.2 Definición

Es un algoritmo de cifrado simétrico desarrollado por los estudiantes Vincent Rijmen y Joan Daemen de la Katholieke Universiteit Leuven en Bélgica, bajo el nombre "Rijndael" fue presentado en 1997 al concurso organizado por el Instituto Nacional de Normas y Tecnologías (NIST) ganando y transformándose en un estándar en el año 2002, con algunos cambios fue posteriormente renombrado AES (Advanced Encryption Standard). (Pousa, 2011), (Huerta, 2003 pág. 13)

1.2.3 Notación y Conversión

1.2.3.1 Entradas y Salidas

La entrada y salida para el algoritmo AES constan cada una de secuencias de 128 bits (dígitos con valores de 0 o 1). Estas secuencias a veces se denominarán bloques y se hará referencia a la cantidad de bits que contienen como su longitud. La clave de cifrado para el algoritmo AES (Advanced Encryption Standard) es una secuencia de 128, 192 o 256 bits. La norma no permite otras longitudes de entrada, salida y clave de cifrado. (Federal information processing standards publication 46, 1977)

1.2.3.2 Bytes

Una serie de ocho bits tratados como un ente. Las series de bits de entrada, salida y clave son procesadas como matrices de bytes que se constituyen al dividir estas series en grupos de ocho bits inmediatos para formar matrices. (Vega Paúl, 2002), (Federal information processing standards publication 46, 1977)

La entrada, salida o clave se formula como b, los bytes en la matriz resultante se referencian utilizando, *bs o b [s]*, donde los valores de *s* estarán en los rangos como se observa en la **Tabla 6-1.** (Vega Paúl, 2002), (Federal information processing standards publication 46, 1977)

Tabla 6-1. Rangos de *s* según la longitud de la clave

Longitud	Número de bits	Rango de s
Clave	128 bits	$0 \leq s < 16$
Clave	192 bits	$0 \leq s < 24$
Clave	256 bits	$0 \leq s < 32$

Realizado por: Raza Carla, 2019

Fuente: (Federal information processing standards publication 46, 1977)

1.2.4 Nociones Matemáticas

1.2.4.1 Campos Finitos GF (2^8)

Entidad formada con un número finito de elementos, son significativos en teoría de números, geometría algebraica, teoría de Galois y Criptografía. (Vázquez, 2007)

En el Algoritmo AES, los campos GF (2^n) son los coeficientes de los resultados del módulo 2, 0 y 1 dando una forma binaria. Así, cada elemento será simbolizado con *n* bits y la cifra de elementos será 2^n. (Vázquez, 2007)

Por ejemplo, para el campo GF (2^3) sus elementos son: (Vázquez, 2007)

$0, 1, p, p + 1, p^2 + 1, p^2 + p, p^2 + p + 1$, siendo los residuos de un polinomio de grado $n - 1 = 2$. (Vázquez, 2007)

En el algoritmo AES, se precisan procedimientos a nivel de byte, los mismos que se presentarán como la concatenación de su bit individual (0 o 1) como: *{k7, k6, k5, k4, k3, k2, k1, k0},* interpretados como elementos de campo finitos usando una representación polinomial: (Vega Paúl, 2002), (Vázquez, 2007)

$k_7 s^7 + k_6 s^6 + k_5 s^5 + k_4 s^4 + k_3 s^3 + k_2 s^2 + k_1 s + k_0 = \sum_{i=0}^{7} k_i s^i$, (Vega Paúl, 2002), (Vázquez, 2007)

Por ejemplo *{01010101}*, identificados como $x^6+x^4+x^2+1$.

1.2.4.2 Suma en GF (2^8)

En GF (g^n) se consideran funciones matemáticas sobre los coeficientes; se hacen en módulo g así GF (2^n) reduce los resultados utilizando para la suma y la resta un OR exclusivo, con lo cual los coeficientes iguales darán un 0 y coeficientes diferentes un 1. (Vázquez, 2007)

Como se observa en el siguiente proceso**.**

A= 45_{16}; 0100 0101
B= 64_{16}; 0110 0100

$A = 0100\ 0101$ ------ x^6+x^2+1

$B = 0110\ 0100$ ------ $x^6+x^5+x^2$

$A+B = (x^6+x^2+1) + (x^6+x^5+x^2+1)$ **mod** 2

$A+B = (2x^6 + x^5 + 2x^2 + 2)$ **mod** 2

$A+B = (x^5 + 1)$

$A+B = 0010\ 0001$

$\mathbf{A+B = 21_{16}}$

XOR

A= 0100 0101

B= 0110 0100

A+B = (0100 0101) $\oplus$ (0110 0100)

A+B = 0010 0001

$\mathbf{A+B = 21_{16}}$

Figura 11-1. Suma de dos polinomios en GF (2^8) y utilizando XOR:

Realizado por: Raza C., 2019

Fuente: (Vega Paúl, 2002)

Así concluimos que el resultado es el mismo utilizando cualquier método.

1.2.4.3 Multiplicación en GF (2^8)

En esta operación GF (2^n) podría resultar que haya elementos fuera del polinomio, por lo tanto, se debe disminuir los diferentes exponentes con un polinomio g*(x)* necesariamente irreducible y grado *m* como en la **Figura 12-1**. (Vázquez, 2007)

El polinomio irreducible es *n(x)*, donde $n(x) = x^8 + x^4 + x^3 + x + 1$. (Vázquez, 2007)

Ejemplo:

$A = 45_{16}$; 0100 0101
$B = 64_{16}$; 0110 0100
A= 0100 0101 ------ x^6+x^2+1
B= 0110 0100 ------ $x^6+x^5+x^2$

Sea $m(x) = x^8 + x^4 + x^3 + x + 1$ donde *m(x):* $x^8 = x^4 + x^3 + x + 1$

$A * B=(x^6+x^2+1) * (x^6+x^5+x^2)$ **mod** 2
$A * B=(x^{12}+x^8+x^6+x^{11}+x^7+x^5+x^8+x^4+x^2)$ **mod** 2
$A * B=(x^{12}+x^{11}+x^7+x^6+x^5+x^4+x^2)$

$x^{12} = x^4 * x^8 = x^4 * (x^4 + x^3 + x + 1)$
$x^{12} = x^8 + x^7 + x^5 + x^4$
$x^{12} = (x^4 + x^3 + x + 1) + x^7 + x^5 + x^4$
$x^{12} = x^7 + x^5 + 2x^4 + x^3 + x + 1$ **mod** 2
$\mathbf{x^{12} = x^7 + x^5 + x^3 + x + 1}$

$x^{11} = x^3 * x^8 = x^3 * (x^4 + x^3 + x + 1)$
$x^{11} = x^7 + x^6 + x^4 + x^3$ **mod** 2
$\mathbf{x^{11} = x^7 + x^6 + x^4 + x^3}$

Reemplazando los valores obtenemos:
$A * B=(x^{12}+x^{11}+x^7+x^6+x^5+x^4+x^2)$
$A * B= ((x^7 + x^5 + x^3+x+1) + (x^7 + x^6 + x^4 + x^3) + x^7+x^6+x^5+x^4+x^2)$
$A * B= (3x^7+2x^6+2x^5+2x^4+2x^3+x^2+x+1)$ **mod** 2
$A * B= \mathbf{x^7 + x^2+x+1}$ ----- **1000 0111**
$A * B= 87_{16}$

Figura 12-1. Multiplicación en GF (2^8)

Realizado por: Raza C., 2019

Fuente: (Vega Paúl, 2002)

1.2.4.4 Multiplicación de polinomios de 4 bytes en GF (2^8)

Para la multiplicación de 4 bytes *c(x)* será el resultado de multiplicar *a(x). b(x)*, cuyos términos de cada polinomio son 4 como muestra la **Figura 13-1**.

$$a(x) = a_3 x^3 + a_2 x^2 + a_1 x + a_0$$
$$b(x) = b_3 x^3 + b_2 x^2 + b_1 x + b_0$$
$$c(x) = a(x).b(x) = c_6 x^6 + c_5 x^5 + c_4 x^4 + c_3 x^3 + c_2 x^2 + c_1 x + c_0.$$

Figura 13-1. Polinomios de 4 términos

Fuente: (Vázquez, 2007)

En la **Figura 13-1,** *c(x)* no constituye una palabra de 4 bytes consecuentemente, que hay que reducir *c(x)* a un polinomio de módulo menor que 4; utilizando el polinomio x^4+1 como se muestra en la **Figura 14-1**. (Vázquez, 2007)

$$d(x) = a(x) \otimes b(x) = d_3\, x^3 \; + \; d_2\, x^2 \; + \; d_1\, x \; + \; d_0$$

Figura 14-1. Polinomio reducido

Fuente: (Vázquez, 2007)

Representando así la palabra de 4 bytes como se muestra en la **Figura 15-1.**

$$\begin{aligned} d_0 &= a_0.b_0 \oplus a_3.b_1 \oplus a_2.b_2 \oplus a_1.b_3 \\ d_1 &= a_1.b_0 \oplus a_0.b_1 \oplus a_3.b_2 \oplus a_2.b_3 \\ d_2 &= a_2.b_0 \oplus a_1.b_1 \oplus a_0.b_2 \oplus a_3.b_3 \\ d_3 &= a_3.b_0 \oplus a_2.b_1 \oplus a_1.b_2 \oplus a_0.b_3 \end{aligned}$$

Figura 15-1. Polinomio que representa una palabra de 4 bytes

Fuente: (Vázquez, 2007)

1.2.5 Especificación del Algoritmo

Constituido por un conjunto de rondas (procesos matemáticos invertibles). El algoritmo se basa en la aplicación de un conjunto de procesos matemáticos a un mensaje en texto plano, dando como resultado un mensaje cifrado. (Cushpa Guamán, 2018), (Vázquez, 2007), (Implementaciones criptográficas en FPGA, 2011)

Estructura del algoritmo

La longitud del bloque de entrada, bloque de salida y el estado es de 128 bits simbolizado como *Nb = 4*, esto representa palabras de 32 bits en el estado. (Vega Paúl, 2002) (Federal information processing standards publication 46, 1977)

Así podríamos decir que los datos a ser encriptados se dividen en segmentos de *16 bytes* (*128 bits*) (**Tabla 7-1**) y cada uno de estos se lo puede divisar como una matriz de *4x4* bytes al que se lo llama estado (**Tabla 8-1**), este se organiza de la siguiente forma: (Pousa, 2011), (Federal information processing standards publication 46, 1977)

Tabla 7-1. Bloque de 128 bits

KD	01	1A	AA	DW	RF	56	87	RG	23	KJ	IU	12	34	6G	9F

Realizado por: Raza C., 2019

Tabla 8-1. Estado formado por 128 bits

KD	DW	RG	12
01	RF	23	34
1A	56	KJ	6G
AA	87	IU	9F

Realizado por: Raza C., 2019

Por otro lado, la longitud de la Clave de cifrado representada por *K*, es de *128, 192 o 256 bits*, para obtener los valores de *Nk* utilizaremos la **Ecuación 1**.

Ecuación 1. Longitud de Clave del AES

$$N_K = K \,/\, 32$$

Expresando así el número de columnas (*4, 6, 8* respectivamente) en la clave de cifrado.

Además, el número de rondas representada por *Nr*, que se realizarán durante la ejecución del algoritmo depende del tamaño de la clave, ver **Tabla 9-1**. (Federal information processing standards publication 46, 1977)

Tabla 9-1. Correlación entre longitud de clave, tamaño de bloque y número de rondas.

	AES – 128	**AES – 192**	**AES – 256**
Longitud de Clave ***(Nk)***	4	6	8
Tamaño del bloque ***(Nb)***	4	4	4
Número de rondas ***(Nr)***	10	12	14

Realizado por: Raza C., 2019

Fuente: (Federal information processing standards publication 46, 1977)

1.2.6 Proceso de Cifrado

En el proceso de cifrado cada ronda está compuesta por cuatro operaciones matemáticas basadas en transformaciones uniformes e invertibles que han sido diseñadas para resistir a los criptoanálisis lineal y diferencial según la **Figura 16-1**. (Implementaciones criptográficas en FPGA, 2011)

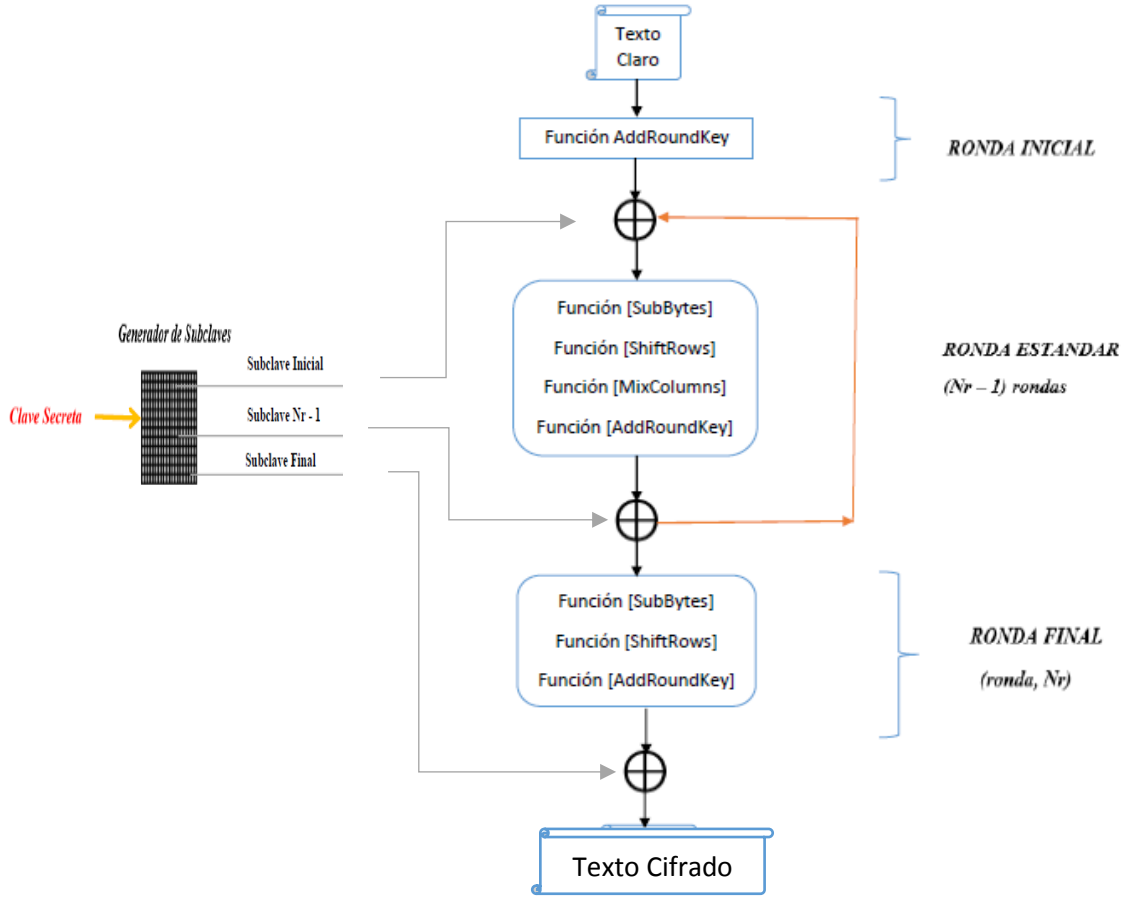


Figura 16-1. Cifrado del Algoritmo AES

Realizado por: Raza C., 2019

Fuente: (Vázquez, 2007)

El algoritmo utiliza los siguientes procesos: (Méndez Naranjo, 2015)

- ✓ AddRoundKey, Adición de clave.
- ✓ SubBytes, Sustituciones de bytes.
- ✓ ShiftRows, Desplazamiento de filas.
- ✓ MixColumns, Mezcla de columnas.
- ✓ KeySchedule, Expansión de clave.

1.2.6.1 Función AddRoundKey

Para esta fase según el número de ronda se ejecuta un XOR byte a byte entre matriz de estado y matriz de la clave o subclave como se muestra en la **Figura 17-1**. (Vázquez, 2007), (Pousa, 2011), (Méndez Naranjo, 2015)

Estado			
21	5B	23	3D
45	32	65	34
3F	8C	34	54
1A	20	48	2F

⊕

Subclave Inicial			
24	12	65	7D
8D	6D	3E	2A
2A	9A	6F	9E
5B	59	3D	6F

=

AddRoundKey			
5	49	46	40
C8	5F	5B	1E
15	16	5B	CA
41	79	75	40

Figura 17-1. Operación XOR entre matriz Estado y matriz Subclave Inicial

Realizado por: Raza C., 2019

Fuente: (Pousa, 2011)

Matemáticamente se puede expresar de la siguiente forma:

$$a_{ij} \oplus b_{ij} = c_{ij}$$

Finalizada la función MixColumns, se vuelve a realizar la función de AddRoundKey, creándose el estado intermedio 1 en la siguiente ronda, pero si es la última ronda se creará el bloque de salida. (Méndez Naranjo, 2015), (Pousa, 2011)

1.2.6.2 Función SubBytes

En esta fase se realiza un reemplazo no lineal a cada byte de la matriz de estado de manera independiente, formando un nuevo byte, así cada elemento de la matriz se le sustituye por otro byte que depende del primero. Este reemplazo se realiza utilizando la tabla S-Box, como se muestra en la **Figura 18-1**. (Méndez Naranjo, 2015), (Vázquez, 2007)

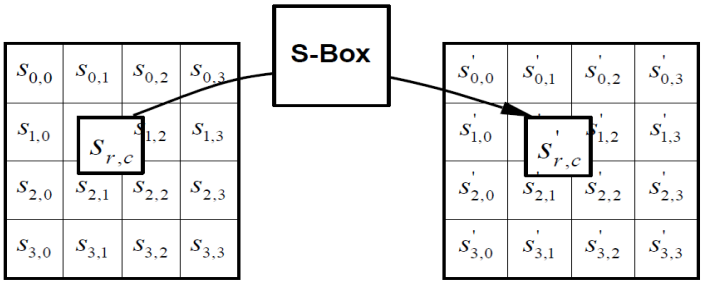


Figura 18-1. Aplicación de la tabla S-box a cada byte del Estado

Fuente: (Federal information processing standards publication 46, 1977)

Se construye componiendo:

- ✓ Byte es considerado como un elemento del G (2^8) formando el polinomio irreducible $n(x)= s^8 + s^4 + s^3 + s + 1$ y sustituido por su inversa multiplicativa. El valor cero en este caso no varía pues no tiene recíproco. (Méndez Naranjo, 2015), (Vázquez, 2007)

- ✓ Se aplica la siguiente transformación afín en G (2^8), siendo $s_0, s_1, ..., s_7$ los bits del byte correspondiente, y a su vez $r_0, r_1, ..., r_7$ los bits del byte correspondiente al resultado como por ejemplo la **Figura 19-1**. (Méndez Naranjo, 2015), (Vázquez, 2007)

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}.$$

Figura 19-1. Transformación afín al S-box

Fuente: (Vázquez, 2007)

El resultado final de las transformaciones anteriores se expresa en una tabla de sustitución denominada S-Box (**Figura 20-1**) la misma que se encuentra expresada en forma hexadecimal. (Vázquez, 2007)

Por ejemplo, si *S [i, j] = {73}*

7 corresponde a la fila (*x*), *3* corresponde a la columna (*y*)

		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

Figura 20-1. Tabla S-box en formato Hexadecimal

Fuente: (Federal information processing standards publication 46, 1977)

La intersección de fila y columna nos da como resultado **8F**.

1.2.6.3 Función ShiftRows

En esta fase se realiza un deslizamiento cíclico a la izquierda de las filas que conforman la matriz de estado actual Nb veces. (Méndez Naranjo, 2015), (Vázquez, 2007)

Por ejemplo, con un bloque de 128 bits:
La fila 1 desplaza 1 byte a la izquierda, la fila 2 desplaza 2 bytes a la izquierda y la fila 3 desplaza 3 bytes a la izquierda, como muestra la **Figura 21-1**.

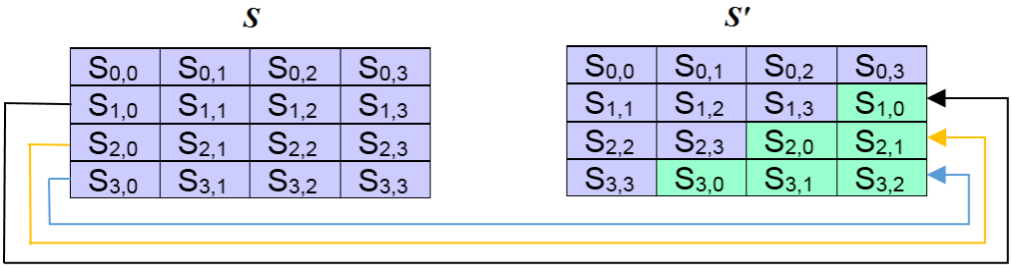


Figura 21-1. ShiftRows, desplazamiento cíclico a un bloque de 128 bits

Realizado por: Raza C., 2019

Fuente: (Federal information processing standards publication 46, 1977)

1.2.6.4 Función MixColumns

Se realiza sobre los bytes de una misma columna de la matriz de estado actual. La columna es vista como un polinomio, cuyos coeficientes pertenecen a G (2^8). La transformación consiste en multiplicar las columnas en módulo $s^4 + 1$ por el polinomio f(s) = {03} s^3 + {01} s^2 + {01} s + {02}, como ejemplo la **Figura 22-1**. (Cushpa Guamán, 2018), (Vázquez, 2007), (Méndez Naranjo, 2015)

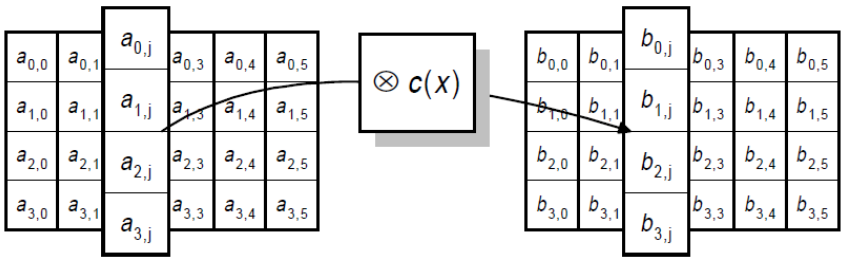


Figura 22-1. Proceso MixColumns

Fuente: (Daemen, y otros, 1999)

Matemáticamente puede ser expresado por:

$$R'(s) = f(s) \otimes R(s)$$

Así, R' (x) es la matriz de estado resultante y R(x) la matriz de entrada. Para una mejor comprensión un ejemplo en la **Figura 23-1**. (Vázquez, 2007), (Federal information processing standards publication 46, 1977)

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix} \quad \text{for } 0 \le c < \boldsymbol{Nb}$$

Figura 23-1. Proceso MixColumns representado en matriz

Fuente: (Federal information processing standards publication 46, 1977)

1.2.6.5 Key Schedule.

La seguridad de un algoritmo sólo debe depender de la clave, debido a esto, se usa otras subclaves L_i para el cifrado y descifrado, así el resultado dependerá de la clave del usuario. (Méndez Naranjo, 2015)

Las RoundKeys provienen de la clave principal *L* mediante el uso de dos funciones: Key Expansion y otra de selección. (Cushpa Guamán, 2018), (Vázquez, 2007), (Méndez Naranjo, 2015)

*Bits de subclaves = tamaño del bloque * (n + 1)*

✓ Key Expansion

La clave inicial de *16* bytes (*128* bits), que también se la puede ver como un bloque o matriz de *4x4* bytes, denotada por *M[i]*, se generan *10* claves, estas claves resultantes junto con la clave inicial son denominadas subclaves. A continuación, se explicará el proceso (Tabla **10-1**): (Pousa, 2011)

Clave usuario: 6t *ff yh gh ed fg td tt oo po fr sa aw aq qw wv*

Tabla 10-1. Clave de 128 bits expresado en matriz.

M[0]	*M[1]*	*M[2]*	*M[3]*
6t	ed	oo	aw
ff	fg	po	aq
yh	td	fr	qw
gh	tt	sa	wv

Realizado por: Raza C., 2019

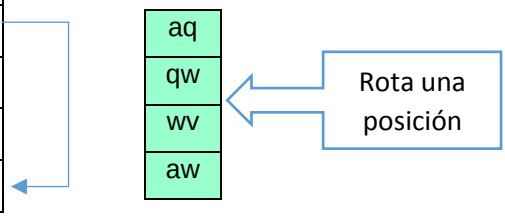
Fuente: (Pousa, 2011), (Vázquez, 2007)

El conjunto de subclaves puede verse como una matriz de *4* filas x *44* columnas, o sea una subclave a continuación de otra, debido a esto, cada vez que se encuentre la primera columna de la siguiente subclave se debe seguir el siguiente procedimiento. (Pousa, 2011)

1. Para calcular la primera columna de la siguiente subclave se toma la última columna *M [3]* de la subclave anterior (en este caso la clave inicial) y se aplica una operación llamada RotWord (**Tabla 11-1**) donde se realiza un giro del primer valor hacia el último lugar en la columna. (Pousa, 2011)

Tabla 11-1. Proceso RotWord

M[0]	*M[1]*	*M[2]*	*M[3]*
6t	ed	oo	aw
ff	fg	po	aq
yh	td	fr	qw
gh	tt	sa	wv

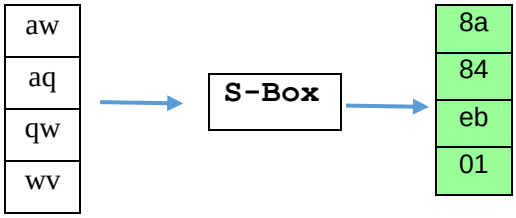


Realizado por: Raza C., 2019

Fuente: (Pousa, 2011), (Vázquez, 2007)

2. Continuado, a la columna resultante, se aplica una operación llamada SubBytes que consiste en reemplazar cada byte de la columna ya rotada por un byte almacenado en una tabla llamada S-Box, como ejemplo la **Tabla 12-1**. (Pousa, 2011)

Tabla 12-1. Proceso SubBytes



Realizado por: Raza C., 2019

Fuente: (Pousa, 2011), (Vázquez, 2007)

3. En seguida al resultado anterior se le aplica un *XOR* byte a byte con la columna *4* posiciones atrás (*M [0]*) y un *XOR* byte a byte con una columna de una tabla llamada RCON (**Tabla 13-1**) que mantiene en la primera fila constantes 2^i en el campo GF (2^8) y en las restantes filas 0. Por ser la primer subclave la que estamos calculando se toma para el cálculo la primera columna de la tabla RCON, para las siguientes subclaves se toma la próxima columna no utilizada de esta tabla: (Pousa, 2011)

Tabla 13-1. Tabla RCON para Nr = 10

01	02	04	08	10	20	40	80	1B	36
00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00

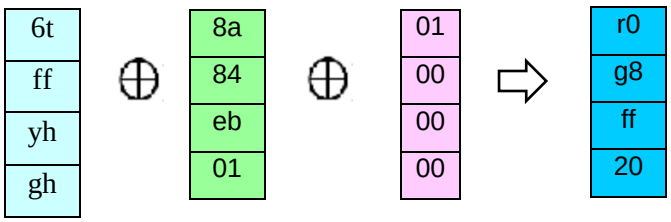
Realizado por: Raza C., 2019

Fuente: (Pousa, 2011), (Vázquez, 2007)

Proceso *XOR*

De la **Tabla 11-1** se toma la primera columna y se realiza un XOR con el resultado del proceso SubBytes luego otro XOR con la primera columna de la **Tabla 13-1** y por medio de este resultado se obtiene la primera columna de la matriz de expansión de claves (**Tablas 14-1 y 15-1**).

Tabla 14-1. Proceso XOR para cálculo de subclaves



Realizado por: Raza C., 2019

Fuente: (Pousa, 2011)

Encontrando después de este proceso la primera columna *M [4]* de la primera subclave por ejemplo la **Tabla 15-1**.

Tabla 15-1. Tabla de Expansión de Claves

M[0]	*M[1]*	*M[2]*	*M[3]*	*M[4]*	*M[5]*	*M[6]*	*M[7]*	*M[8]*	*M[9]*	*M[10]*	*M[11]*	*M[12]*
6t	ed	oo	6t	r0								
ff	fg	po	ff	g8		…						
yh	td	fr	yh	ff								
gh	tt	sa	gh	20								

Realizado por: Raza C., 2019

Fuente: (Pousa, 2011), (Vázquez, 2007)

Para calcular las tres columnas siguientes se hace un *XOR* entre la columna anterior y la columna de cuatro posiciones atrás *(M [5] = M [4]* **XOR** *M [1])*, como se muestra en la **Tabla 16-1**: (Pousa, 2011)

Tabla 16-1. Tabla de Expansión de Claves

M[0]	*M[1]*	*M[2]*	*M[3]*	*M[4]*	*M[5]*	*M[6]*	*M[7]*	*M[8]*	*M[9]*	*M[10]*	*M[11]*	*M[12]*
6t	ed	oo	6t	r0	41	29	aa					
ff	fg	po	ff	g8	5f	31	cc	…				
yh	td	fr	yh	ff	2e	03	c6					
gh	tt	sa	gh	20	c3	09	d5					

Realizado por: Raza C., 2019

Fuente: (Pousa, 2011), (Vázquez, 2007)

M [8], será encontrado aplicando el proceso anterior debido a que es múltiplo de 4.

Es sustancial recordar que la rutina de expansión de clave para claves de cifrado de 256 bits (*Nk = 8*) es levemente diferente que para las claves de cifrado de 128 y 192 bits porque, si *Nk = 8* e *i-4* es un múltiplo de *Nk*, entonces SubBytes () se aplica a *M [[i-1]]* antes del XOR. (Pousa, 2011)

1.2.7 Proceso de Descifrado

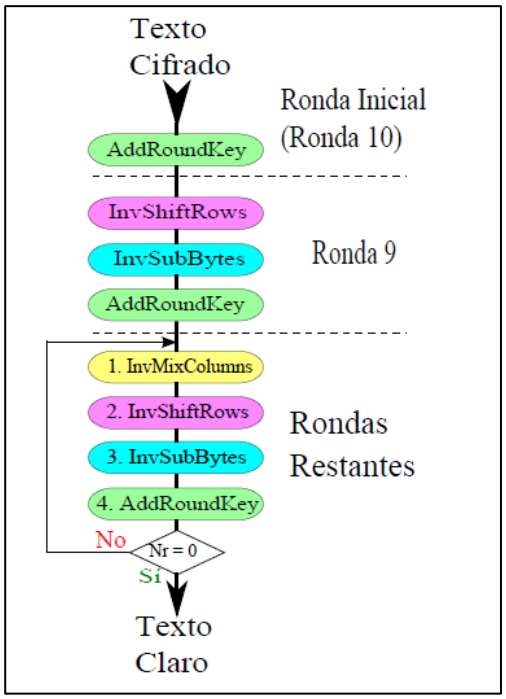


Figura 24-1. Descifrado del Algoritmo AES

Fuente: (Martínez, 2013)

En proceso de descifrado (**Figura 24-1**) se utilizará las funciones inversas de las operaciones realizadas al momento del cifrado. Se debe tomar en cuenta que, en el descifrado las subclaves serán utilizadas desde la última a la primera. (Vázquez, 2007), (Méndez Naranjo, 2015)

La función AddRoundKey desempeña el mismo proceso explicado anteriormente empezando por la última ronda y concluirá por la primera ronda. (Vázquez, 2007)

La función *Inv. SubBytes* se utiliza la caja S-box inversa a cada byte de la matriz de estado. Cuya caja se muestra en la (**Figura 25-1**). (Méndez Naranjo, 2015), (Vázquez, 2007)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	52	09	6 A	D5	30	36	A5	38	BF	40	A3	9 E	81	F3	D7	FB
1	7 C	E3	39	82	9 B	2 F	FF	87	34	8 E	43	44	C4	DE	E9	CB
2	54	7 B	94	32	A6	C2	23	3 D	EE	4 C	95	0 B	42	FA	C3	4 E
3	08	2 E	A1	66	28	D9	24	B2	76	5 B	A2	49	6 D	8 B	D1	25
4	72	F8	F6	64	86	68	98	16	D4	A4	5 C	CC	5 D	65	B6	92
5	6 C	70	48	50	FD	ED	B9	DA	5 E	15	46	57	A7	8 D	9 D	84
6	90	D8	AB	00	8 C	BC	D3	0 A	F7	E4	58	05	B8	B3	45	06
7	D0	2 C	1 E	8 F	CA	3 F	0 F	02	C1	AF	BD	03	01	13	8 A	6 B
8	3 A	91	11	41	4 F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1 C	75	DF	6 E
A	47	F1	1 A	71	1 D	29	C5	89	6 F	B7	62	0 E	AA	18	BE	1 B
B	FC	56	3 E	4 B	C6	D2	79	20	9 A	DB	C0	FE	78	CD	5 A	F4
C	1 F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5 F
D	60	51	7 F	A9	19	B5	4 A	0 D	2 D	E5	7 A	9 F	93	C9	9 C	EF
E	A0	E0	3 B	4 D	AE	2 A	F5	B0	C8	EB	BB	3 C	83	53	99	61
F	17	2 B	04	7 E	BA	77	D6	26	E1	69	14	63	55	21	0 C	7 D

Figura 25-1. Tabla S-Box Inversa

Fuente: (Vázquez, 2007)

La función *Inv. ShiftRows* movimiento de las filas de la matriz actual a la derecha, Nb veces que se desplazaron en el proceso de cifrado. (Méndez Naranjo, 2015), (Vázquez, 2007)

La función *Inv. MixColumns* cuya operación se realiza sobre los bytes de una misma columna, la columna será tomada como polinomio con coeficientes en GF (2^8), multiplicados por $y(s) = 0\ K\ s^3 + 0\ L\ s^2 + 09\ s + 0\ M$, siendo el inverso de f(s). (Cushpa Guamán, 2018), (Méndez Naranjo, 2015), (Vázquez, 2007)

Matemáticamente expresada por:

$$R(s) = y(xs) \otimes R'(s),$$

R(s) representa la matriz de estado resultante de esta etapa, y $R'(s)$ representa la matriz de entrada. (Vázquez, 2007)

CAPITULO II

En este capítulo se desarrollará la implementación del AES (Advanced Encryption Standard) y la propuesta tecnológica MECIB-AES, además de explicar las diferentes funciones utilizadas para programar el nuevo proceso de cifrado. La implementación de estos ayudara a medir el nivel de confiabilidad.

2.1 IMPLEMENTACIÓN DEL ALGORITMO AES

Antes de la implementación, es necesario tener en cuenta diferentes aspectos básicos que aportaran diferentes puntos de vista al momento de realizar las aplicaciones.

Rijndael y AES difieren solo en el rango de valores admitidos para la longitud del bloque y la longitud de la clave de cifrado. Para Rijndael, la longitud del bloque y la longitud de la clave pueden especificarse independientemente a cualquier múltiplo de 32 bits, siendo 128, 192 y 256 bits. AES arregla la longitud del bloque a 128 bits y solo admite longitudes de clave de 128, 192 o 256 bits. (Daemen, y otros, 1999)

✓ Requisitos de longitud de clave

Una implementación del algoritmo AES soportará al menos una de las tres longitudes de clave especificadas anteriormente (**Tabla 10-1**.)

Las implementaciones pueden admitir opcionalmente dos o tres longitudes de clave, lo que puede promover la interoperabilidad de las implementaciones de algoritmo. (Standards, 2001)

✓ Restricciones de claves

No se han identificado claves débiles o semi-débiles para el algoritmo AES, y no hay restricciones en la selección de teclas. (Standards, 2001)

✓ Sugerencias de implementación con respecto a varias plataformas.

Son posibles variaciones de implementación que, en muchos casos, pueden ofrecer rendimiento u otras ventajas. Dada la misma clave de entrada y datos (texto plano o texto cifrado), cualquier implementación que produzca la misma salida (texto cifrado o texto plano) como el algoritmo especificado en este estándar es una implementación aceptable del AES. (Standards, 2001)

Por lo tanto, el cifrado AES (Advanced Encryption Standard) es adecuado para ser implementado eficientemente en una amplia gama de procesadores y en hardware dedicado. Por ejemplo, en procesadores de 8 bits, típicos para tarjetas inteligentes actuales y en procesadores de 32 bits, típicos para PC. (Daemen, y otros, 1999)

✓ Lenguajes de programación.

El índice Tiobe (**Figura 1-2**) es uno de los datos más confiables que existen en cuanto a las estadísticas de uso y evolución de los diferentes lenguajes de programación. (Cipsa, 2018) En primer lugar, se encuentra Java, que continúa siendo el lenguaje programación más utilizado por los desarrolladores. (Cipsa, 2018)

Resumen de datos proporcionados por Tiobe

Jan 2018	Jan 2017	Change	Programming Language	Ratings	Change
1	1		Java	14.215%	-3.06%
2	2		C	11.037%	+1.69%
3	3		C++	5.603%	-0.70%
4	5	^	Python	4.678%	+1.21%
5	4	v	C#	3.754%	-0.29%

Figura 1-2. El índice de programación Tiobe

Fuente: (Cipsa, 2018)

Realizaremos una comparación de los tres (3) primeros lenguajes de programación según la **Figura 1-2**; estas comparaciones se encuentran en las **Tablas 1-2,2-2.**

Tabla 1-2. Tabla comparativa entre C y C++

Parámetro	C	C++
La extensión del archivo	Se guardan con extensión c, ejemplo práctica.	Se guardan con extensión cpp (c plus plus), ejemplo practica.cpp
Operadores lógicos	Únicamente se utilizan los operadores &&, || y !.	AND, OR y NOT pueden ser utilizados como palabras.
Los flujos de entrada y salida.	Printf y scanf.	Cin y cout.
Uso de clases	Programación orientada a objetos.	No posee programación orientada a objetos.
Las bibliotecas	Terminan con h (son headers) ejemplo: math.h	Mismas bibliotecas sirven para C++, pero por convención, se elimina la terminación '.h' y mejor se agrega 'c' al principio. Ejemplo: cmath.

Realizado por: Raza, C. 2019

Fuente: (Torres, 2013)

De la comparación anterior podemos elegir al lenguaje C++ como mejor opción y el cual será comparado con java para obtener el que se utilizará.

Tabla 2-2. Tabla comparativa entre Java y C++

Java	C++
Es tanto un lenguaje de programación como una plataforma de software.	Es sólo un lenguaje de programación.
Es un lenguaje puramente orientado a objetos.	Este lenguaje da soporte tanto a la programación estructurada como a la programación orientada a objetos.
Todas las declaraciones de variables y métodos deben estar dentro de la estructura.	Las declaraciones de variables y funciones pueden estar presentes fuera de la estructura. No es necesario para un programa en C++ tener estructura.
El lenguaje es independiente del tipo de plataforma. El código Java, una vez escrito, puede ser ejecutado en cualquier plataforma.	El código C++, una vez escrito para una plataforma, necesita ser compilado de nuevo, y el código objeto reenlazado para ser ejecutado en otra plataforma diferente.
Maneja la memoria automáticamente.	Los programadores tienen que hacerse cargo de liberar la memoria no utilizada.
No soporta características como sobrecarga de operadores y conversiones automáticas en ambos sentidos.	Da soporte a características como sobrecarga de operadores y conversiones automáticas en ambos sentidos.
Una estructura no puede heredar directamente de otra estructura. Se da soporte a la herencia múltiple usando interfaces.	Una estructura puede heredar directamente de otra estructura.
Las librerías se pueden extender.	Las librerías no se pueden extender.
La programación de redes es más fácil. Los objetos pueden ser accedidos a través de la red usando URLS.	Esta programación para redes es compleja, a menos que se usen APIS de terceros. C++, es un lenguaje, que no brinda soporte incorporado para programación de redes.
Los programadores no pueden usar apuntadores. Los apuntadores se usan internamente.	Los programadores pueden usar apuntadores.
Implementa arreglos verdaderos.	Se implementan los arreglos con aritmética de apuntadores.

Realizado por: Raza, C. 2019

Fuente: (Aguirre, 2016)

Por la facilidad de utilizar diferentes plataformas optamos por elegir al lenguaje de programación Java para la realización de las aplicaciones, por lo tanto:

✓ Java

El lenguaje de programación Java se ha vuelto bastante popular en los últimos años. Esto se debe en parte al hecho de que los programas de Java son independientes de la plataforma (es decir, del procesador y del sistema operativo), tanto en su fuente como en su forma binaria. (Sterbenz, y otros, 2000)

Esto es posible mediante el empleo de un modelo de compilación diferente al de la mayoría de los otros idiomas. En lugar de compilar el código fuente en el código de máquina para una familia

de procesadores en particular, el compilador produce un código de máquina (llamado "bytecode") para una máquina virtual Java (JVM) imaginaria. (Sterbenz, y otros, 2000)

En tiempo de ejecución, este bytecode se traduce en código de máquina mediante una implementación de JVM para la plataforma en particular. Por supuesto, hay varias opciones para la traducción de bytecode a código de máquina. El más simple y más obvio es usar un intérprete: tome una instrucción JVM a la vez y ejecute las instrucciones de código máquina correspondientes. (Sterbenz, y otros, 2000)

- ✓ Java en aplicaciones criptográficas:

El punto es que, Java difícilmente será el idioma elegido para los servidores de alta carga, bien puede ser la opción para los servidores de carga media y especialmente para los clientes. Una ventaja particular de Java es que existe una API criptográfica estándar bien establecida, la arquitectura JCA y JCE de Javasoft. El éxito de las bibliotecas de criptografía en Java, incluidas las bibliotecas del IAIK, confirma esta posición. (Sterbenz, y otros, 2000)

- ✓ Tipo de Cifrado a utilizar

Además, dado que el algoritmo AES (Advanced Encryption Standard) pertenece a un cifrado en bloque es importante definir que cifrado se utiliza: (Pousa, 2011)

- ✓ Modo ECB (Electronic Codebook): El texto se divide en bloques y cada bloque es cifrado en forma independiente utilizando la clave; por lo cual una entrada siempre da la misma salida. (Pousa, 2011)
- ✓ Modo CBC (Cipher-block Chaining): El texto se divide en bloques y cada bloque es mezclado con la cifra del bloque previo, luego es cifrado utilizando la clave lo cual permite que generen diferentes salidas. (Pousa, 2011)
- ✓ Modos CFB (Cipher FeedBack) y OFB (Output FeedBack) su funcionamiento es similar, utilizado para generar un stream de bits que realice XOR con el texto plano. (Pousa, 2011)

Explicadas las diferentes formas de cifrado elegiremos el modo CBC (Cipher-block Chaining) porque así podremos elegir libremente la longitud del texto a cifrar.

2.1.1 Herramienta para la Implementación:

Para encontrar el IDE de java a utilizar utilizaremos la escala de Likert con 5 Ítems como se muestra en la **Tabla 3-2**.

5: Muy Importante; **4**: Importante; **3**: Modestamente Importante; **2**: De Poca Importancia; **1**: Sin Importancia.

Tabla 3-2. Escala de Likert.

IDE	Ítems	5	4	3	2	1
Netbeans	IDE Gratuito y de Código Abierto	x				
	Facilidad con la que se pueden crear aplicaciones con interfaz gráfica.	x				
	Identifica y soluciona problemas de código comunes.	x				
	IDE ideal para quien recién empieza a programar en java.	x				
	Mayor consumo de Recursos					x
Eclipse	IDE Gratuito y de Código Abierto		x			
	Facilidad con la que se pueden crear aplicaciones con interfaz gráfica.				x	
	Identifica y soluciona problemas de código comunes.			x		
	IDE ideal para quien recién empieza a programar en java.				x	
	Mayor consumo de Recursos					x
IntelliJ IDEA	IDE Gratuito y de Código Abierto				x	
	Facilidad con la que se pueden crear aplicaciones con interfaz gráfica.				x	
	Identifica y soluciona problemas de código comunes.	x				
	IDE ideal para quien recién empieza a programar en java.				x	
	Mayor consumo de Recursos			x		
JCreator	IDE Gratuito y de Código Abierto					x
	Facilidad con la que se pueden crear aplicaciones con interfaz gráfica.			x		
	Identifica y soluciona problemas de código comunes.			x		
	IDE ideal para quien recién empieza a programar en java.				x	
	Mayor consumo de Recursos			x		
jGRASP	IDE Gratuito y de Código Abierto	x				
	Facilidad con la que se pueden crear aplicaciones con interfaz gráfica.	x				
	Identifica y soluciona problemas de código comunes.					x
	IDE ideal para quien recién empieza a programar en java.	x				
	Mayor consumo de Recursos			x		

Realizado por: Raza, C. 2019

Fuente: (Palomino, 2016)

Por lo tanto, el IDE a utilizar es Netbeans debido a un mayor peso en la escala de Likert.

2.1.2 Programación

Una vez revisado todas las funciones utilizadas por el algoritmo criptográfico AES, así como el entorno de desarrollo a utilizar se procede a la programación utilizando el IDE de desarrollo Netbeans.

El algoritmo programado tiene el nombre de "Algoritmo AES Simétrico" cuya interfaz con el usuario es como se presenta en la **Figura 2-2**.

Figura 2-2. Interfaz de la Primera Aplicación

Realizado por: Raza C., 2019

La programación está dividida en paquetes y clases (**Figura 3-2**) que ayudan a una mejor organización del mismo, por lo cual tenemos:

Paquete de fuentes

1. algoritmoaes
2. frm
3. imágenes; estos a su vez contienen a las clases:

El paquete algoritmoaes*;* contiene programado el algoritmo cada función y transformación utilizada en el cifrado y descifrado además de su proceso para la clave principal y subclaves.

- ✓ AES.java
- ✓ AlgoritmoAES.java
- ✓ Utilidades.java
- ✓ clsHexa.java

El paquete frm contiene las ventanas principales, cifrado y descifrado.

- ✓ frmPrincipal.java
- ✓ ifrmCifrar.java
- ✓ ifrmDesCifrar.java

El paquete imágenes contiene los siguientes archivos utilizados en la interfaz con el usuario.

- ✓ archivo.png
- ✓ cancelar.png
- ✓ clave.png
- ✓ cpu.png
- ✓ disquetes.png
- ✓ documento.png
- ✓ pagina-en-blanco.png
- ✓ revisar.png

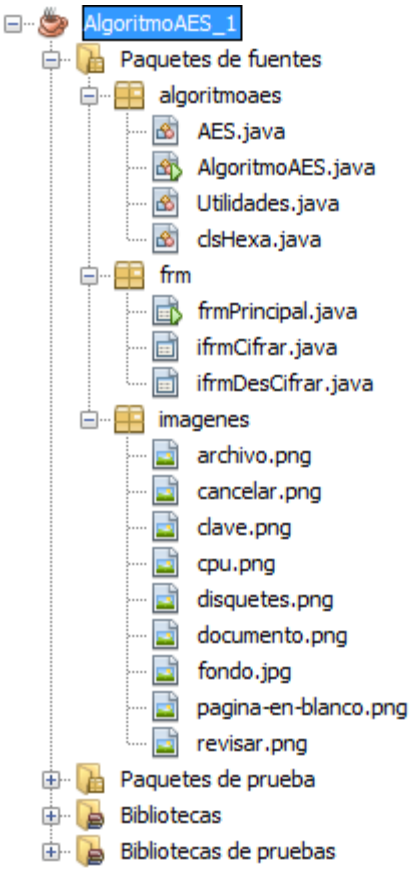


Figura 3-2. Paquetes utilizados para la Aplicación

Realizado por: Raza C., 2019

Ventana de cifrado **Figura 4-2** y descifrado **Figura 5-2**.

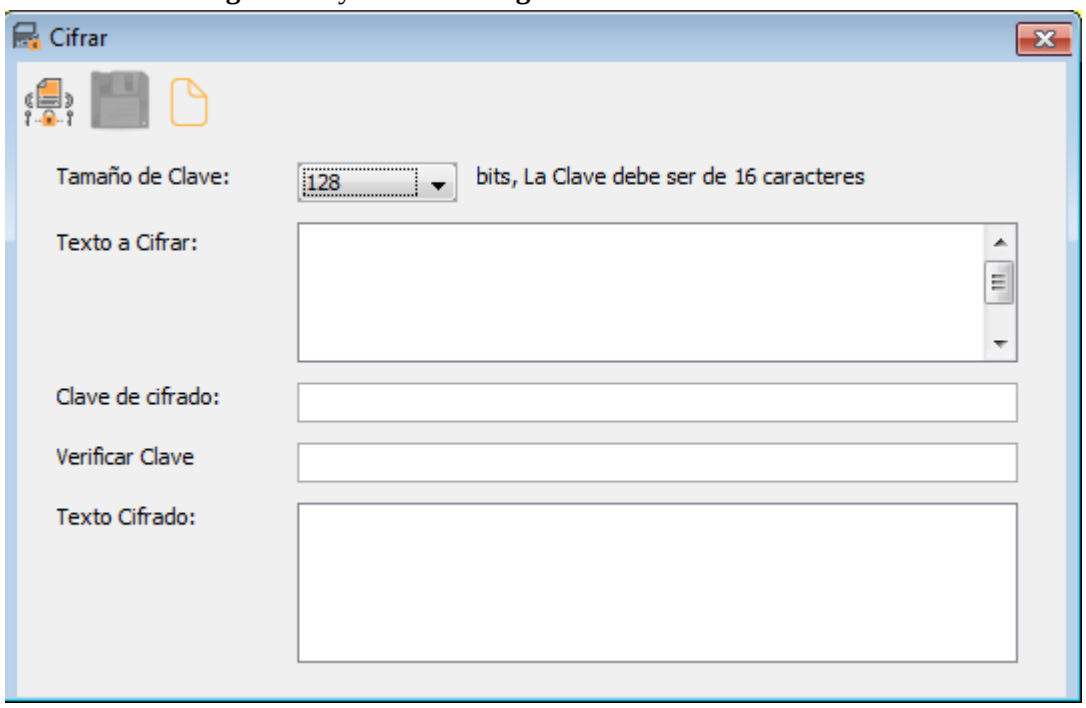


Figura 4-2. Ventana de Cifrado del Algoritmo AES.

Realizado por: Raza C., 2019

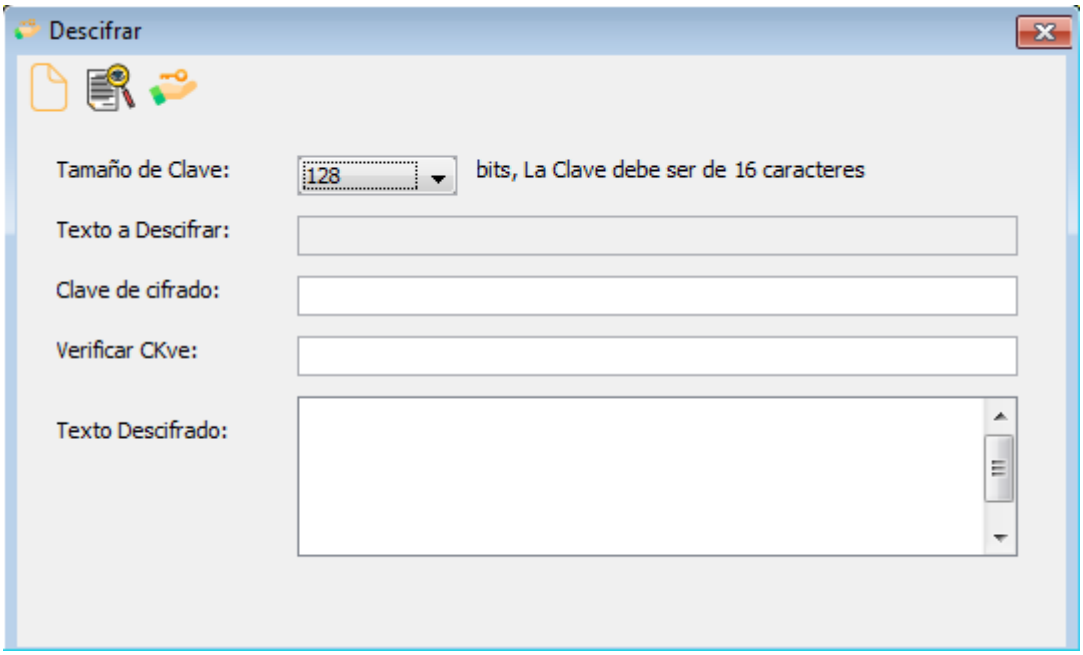


Figura 5-2. Ventana de Descifrado del Algoritmo AES.

Realizado por: Raza C., 2019

Para la clave de cifrado se utilizó la ocultación del texto ingresado para prevenir cualquier tipo de robo, pero su autenticación está realizada al ingresar de nuevo la misma clave para poder realizar el cifrado y descifrado como se muestra en las **Figuras 6-2; 7-2; 8-2; 9-2**.

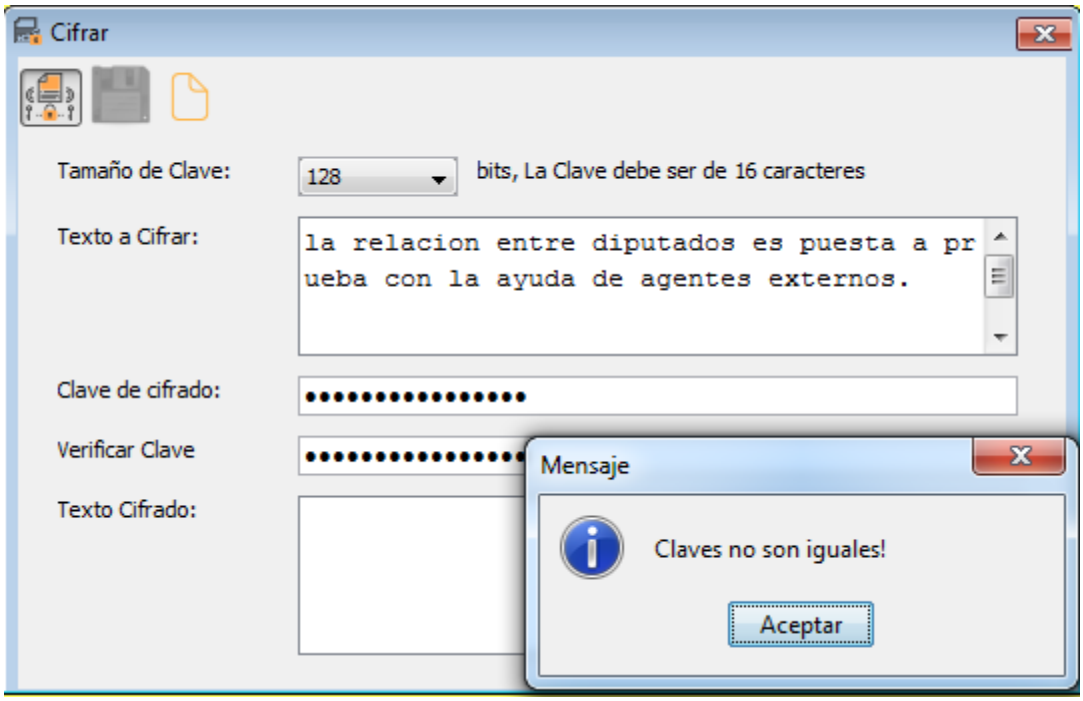


Figura 6-2. Autenticación fallida de la clave de cifrado.

Realizado por: Raza C., 2019

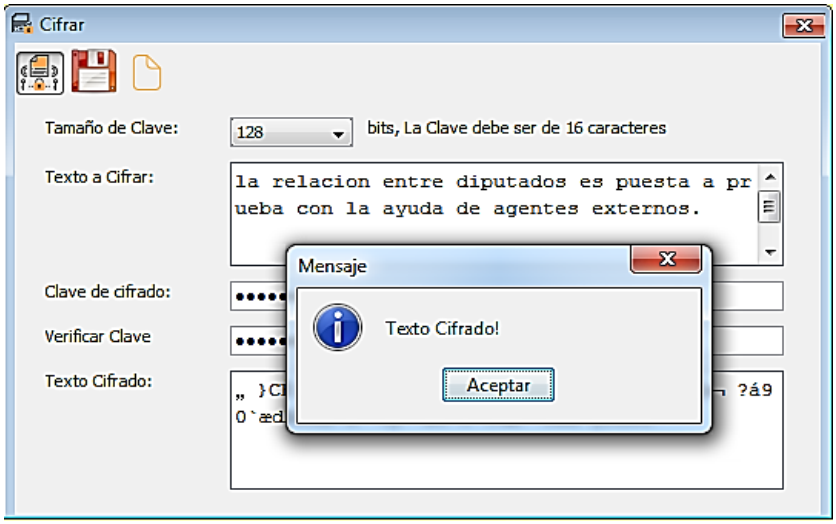


Figura 7-2. Autenticación exitosa de la clave de cifrado.

Realizado por: Raza C., 2019

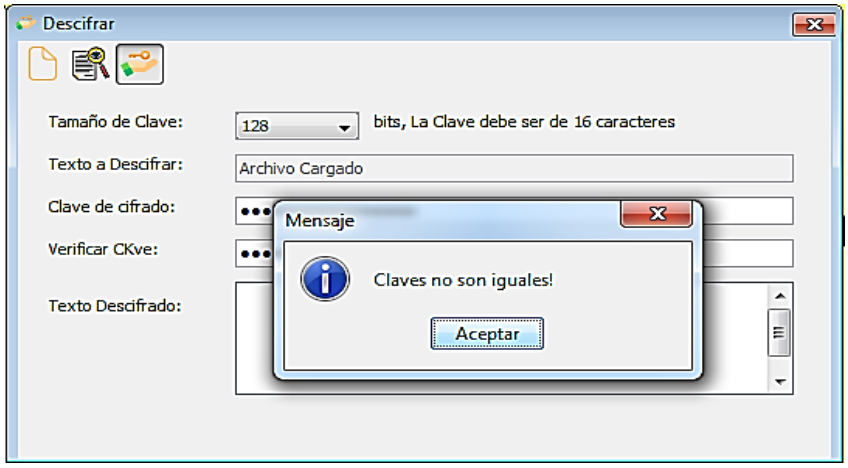


Figura 8-2. Autenticación fallida de la clave de descifrado.

Realizado por: Raza C., 2019

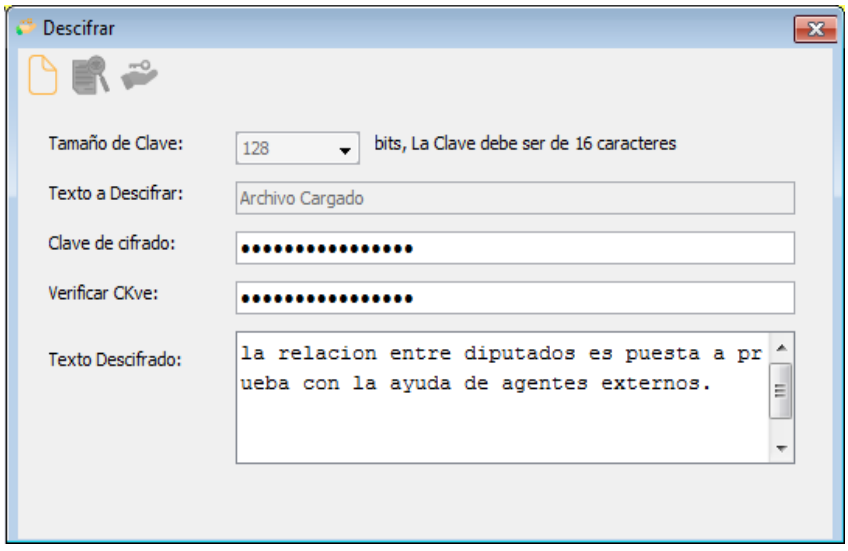


Figura 9-2. Autenticación exitosa de la clave de descifrado.

Realizado por: Raza C., 2019

Para probar la funcionalidad de la aplicación se procedió a cifrar y descifrar como se presenta a continuación.

✓ Cifrado (**Figuras 10-2; 11-2**) y Descifrado (**Figuras 12-2**) con clave de 128 bits.

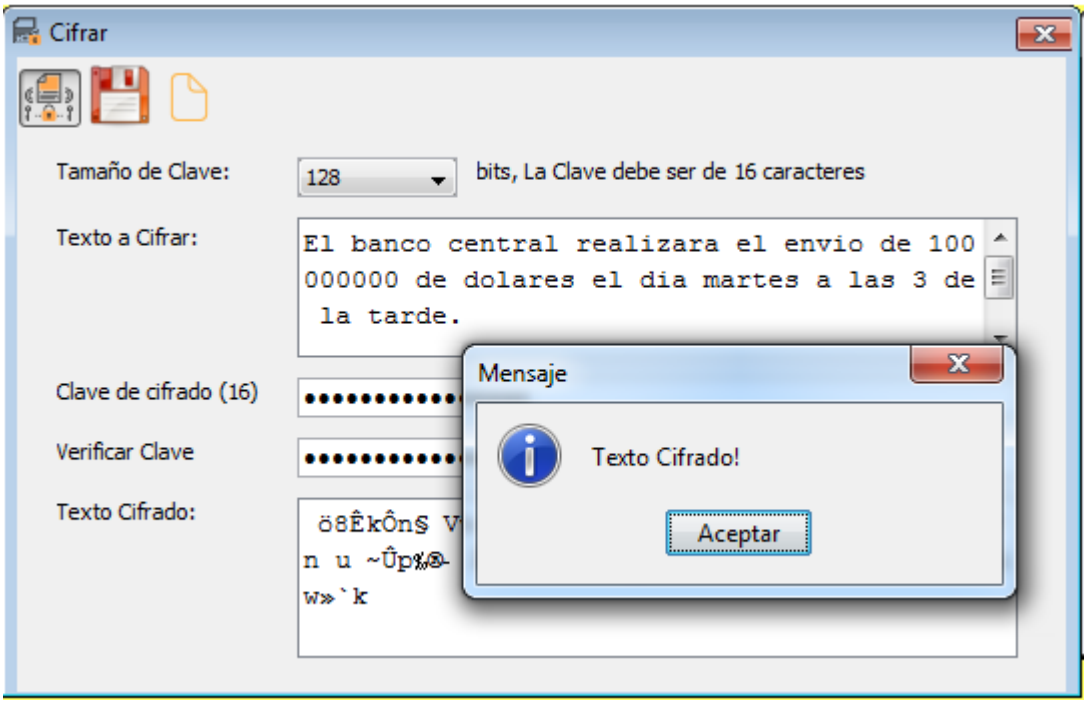


Figura 10-2. Cifrado con el algoritmo AES simétrico y clave 128 bits.

Realizado por: Raza C., 2019

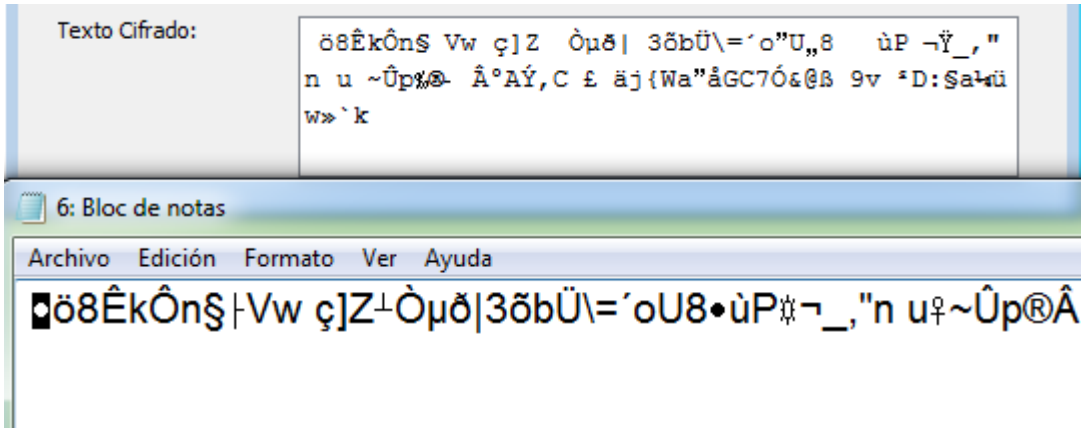


Figura 11-2. Texto cifrado guardado en .txt, clave 128 bits

Realizado por: Raza C., 2019

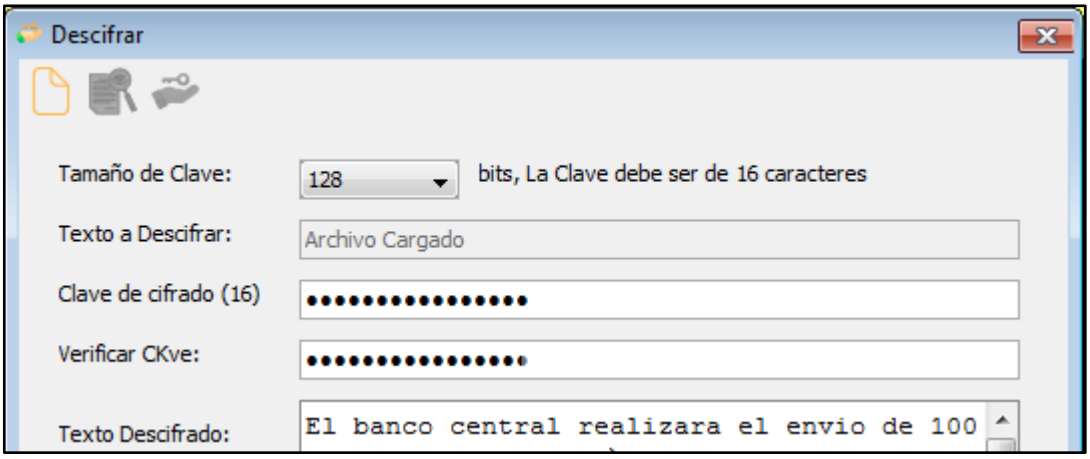


Figura 12-2. Texto descifrado con clave 128 bits.

Realizado por: Raza C., 2019

✓ Cifrado (**Figuras 13-2; 14-2**) y Descifrado (**Figuras 15-2**) con clave de 192 bits.

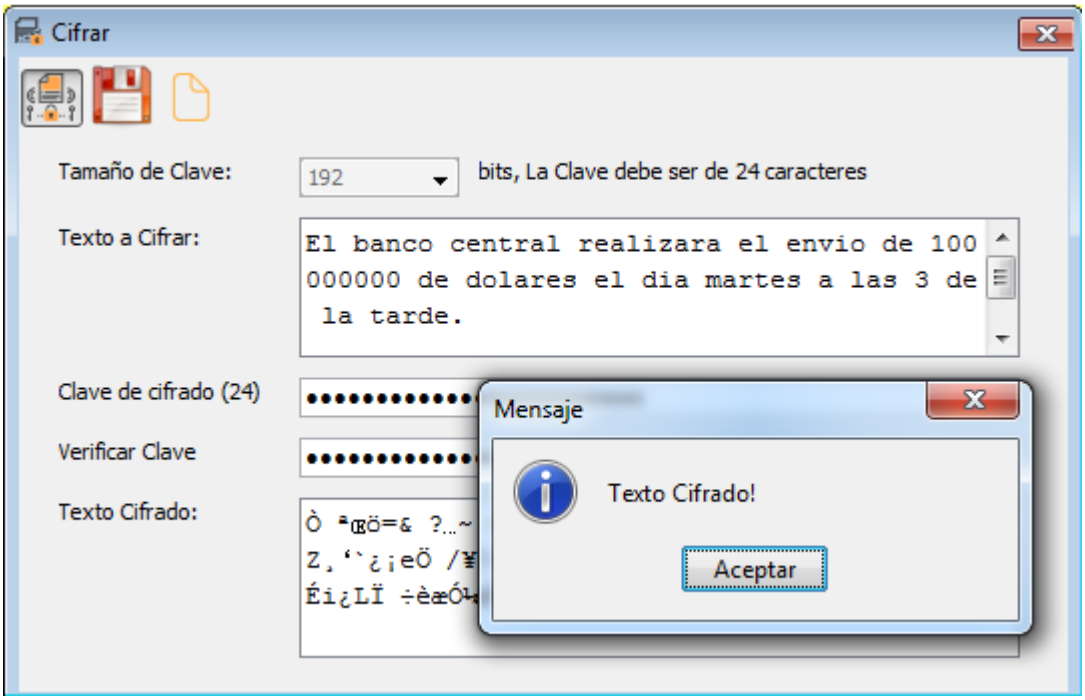


Figura 13-2. Texto cifrado con clave 192 bits.

Realizado por: Raza C., 2019

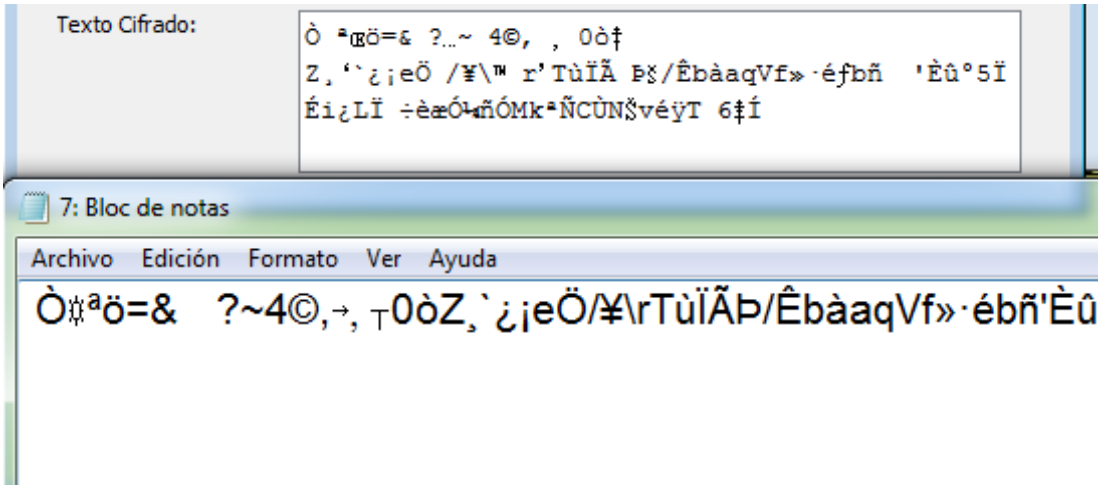


Figura 14-2. Texto cifrado guardado en .txt, clave 192 bits

Realizado por: Raza C., 2019

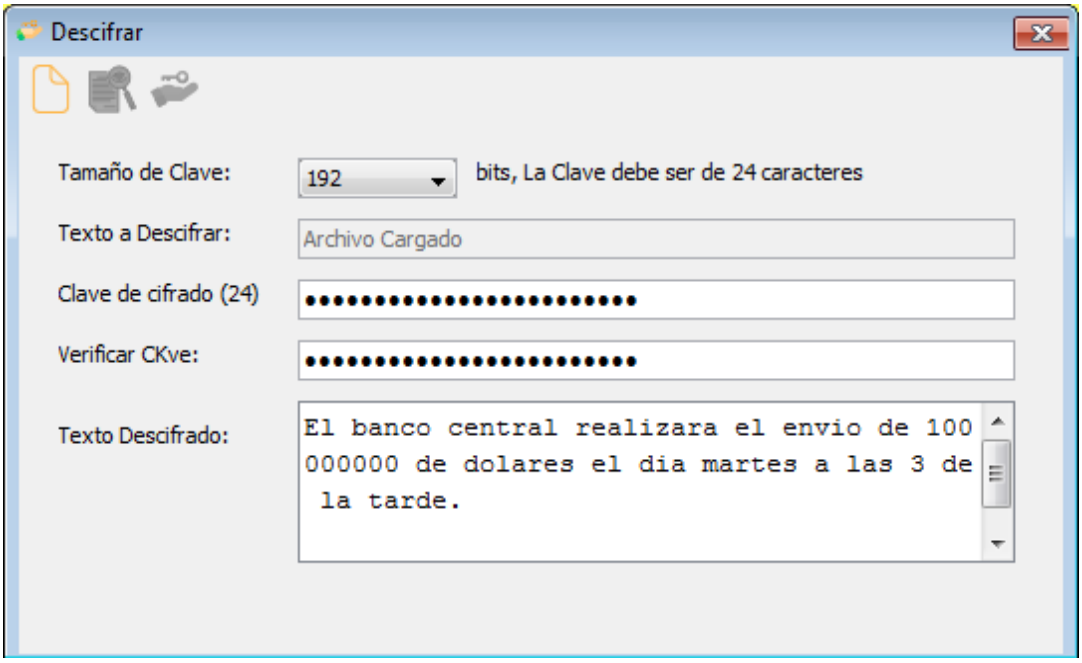


Figura 15-2. Texto descifrado con clave 192 bits.

Realizado por: Raza C., 2019

✓ Cifrado (**Figuras 16-2; 17-2**) y Descifrado (**Figuras 18-2**) con clave de 256 bits.

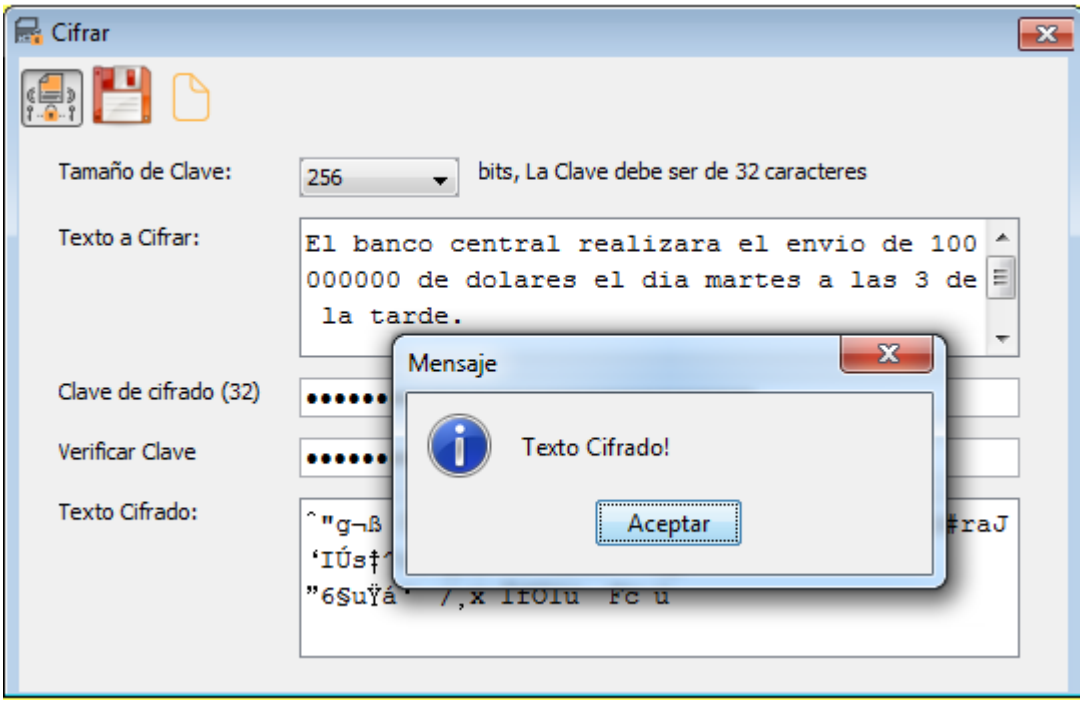


Figura 16-2. Texto cifrado con clave 256 bits.

Realizado por: Raza C., 2019

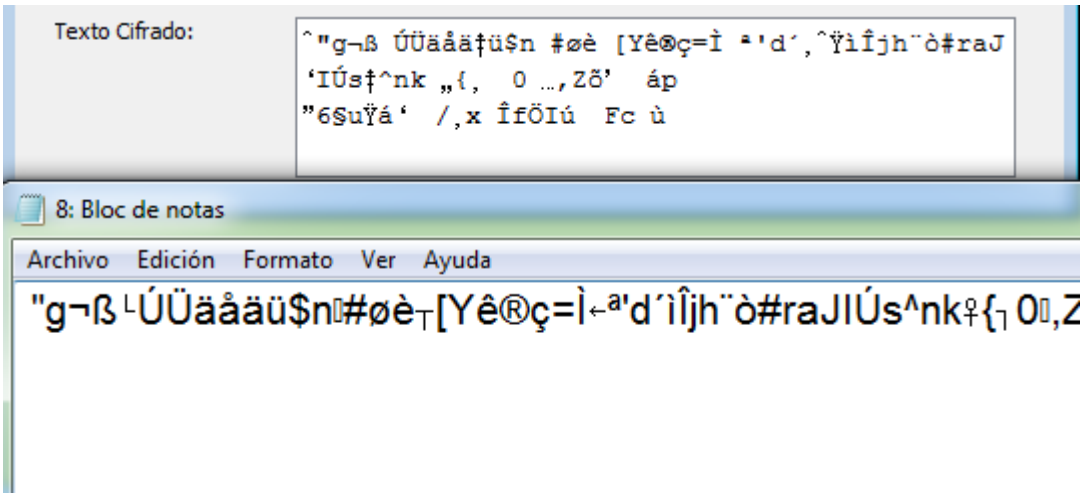


Figura 17-2. Texto cifrado guardado en .txt, clave 256 bits.

Realizado por: Raza C., 2019

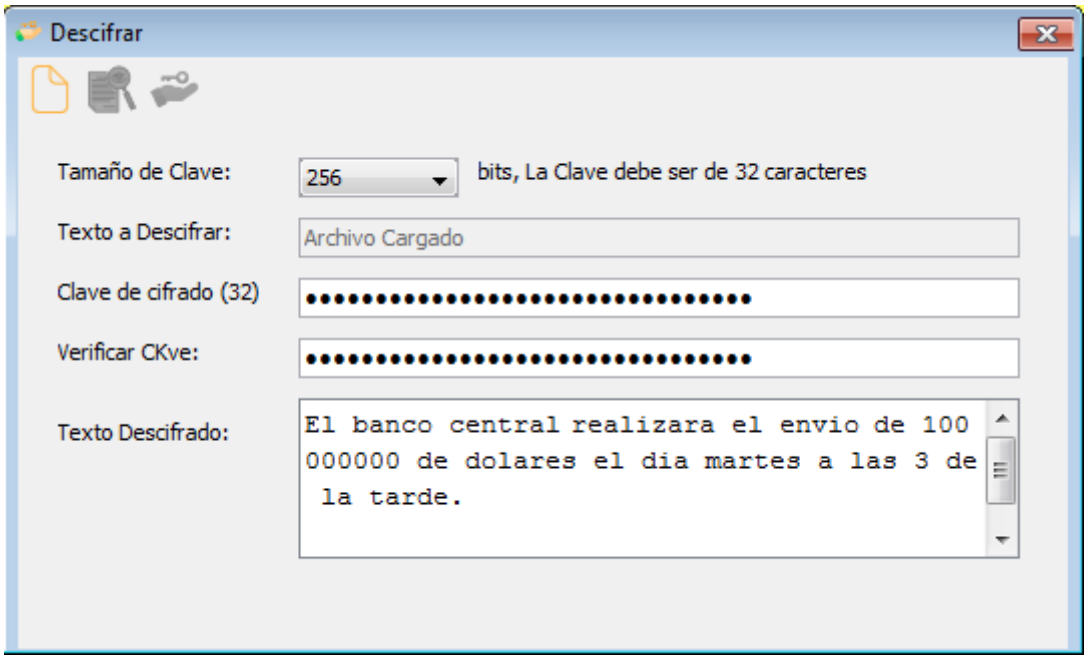


Figura 18-2. Texto descifrado con clave 256 bits.

Realizado por: Raza C., 2019

2.2 DESARROLLO DEL NUEVO MECANISMO DE CIFRADO

El nuevo proceso de cifrado fue llamado **MECIB-AES** debido a que es una modificación del algoritmo criptográfico simétrico AES (Advanced Encryption Standard).

Una vez analizadas todas las funciones que actúan en el proceso de cifrado del algoritmo criptográfico simétrico AES (Advanced Encryption Standard) se procedió a realizar cambios en sus diferentes etapas con el fin de medir el nivel de confiabilidad que ofrecerá.

Al realizar las diferentes modificaciones se les asigno un nuevo nombre para el presente trabajo.

- ✓ **Mix-Shift**, modificado de ShiftRows
- ✓ **Move-C**, modificado de MixColumns
- ✓ **Mix-Key**, modificado del Bloque de Clave

Estas funciones serán incorporadas al proceso de cifrado normal y así obtendremos la modificación deseada.

2.2.1 Función Mix-Shift

Para la modificación la matriz del estado actual (**Tabla 4-2**) se transforma en vectores los cuales están en forma lineal (**Tabla 5-2**), además se crea un vector con valores estáticos ingresados por el programador (**Tabla 6-2**), a continuación, la función Mix-Shift se ejecuta 10, 12,14 rondas lo que produce que las posiciones se mantengan fijas pero el valor contenido cambiara dependiendo de los siguientes procesos a ejecutarse (**Tabla 7-2**).

Tabla 4-2. Ejemplo, matriz de estado actual

A	E	I	M
B	F	J	N
C	G	K	O
D	H	L	P

Realizado por: Raza C., 2019

Tabla 5-2. Vectores realizados con la matriz de estado actual.

Vector	V1[i]				V2[i]				V3[i]				V4[i]			
Posiciones	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Valores	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P

Realizado por: Raza C., 2019

Tabla 6-2. Posiciones de la matriz de estado actual.

Vector definido[4]	8	12	4	0
Columna que utiliza la modificación.	0	1	2	3

Realizado por: Raza C., 2019

Tabla 7-2. Formación de la nueva matriz de estado actual

0	1	2	3
I	M	E	A
J	N	F	B
K	O	G	C
L	P	H	D

Realizado por: Raza C., 2019

Una vez realizado el proceso la matriz esta lista para realizar la siguiente función.

2.2.2 Función Move-C

Este proceso está relacionado a una alteración al orden en el cual se desarrollan las etapas (**Figura 20-2**) dentro de las 10, 12, 14 rondas realizadas por el algoritmo AES (**Figura 19-2**).

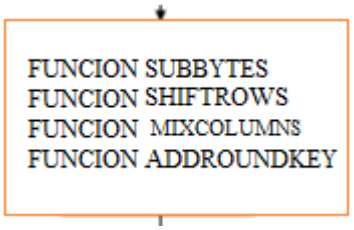


Figura 19-2. Proceso Normal del Algoritmo AES.

Realizado por: Raza C., 2019

Analizando todas las posibles alteraciones se optó por cambiar de lugar la función MixColumns y la función ShiftRows, cuyo resultado es (**Figura 20-2**) y (**Figura 21-2**):

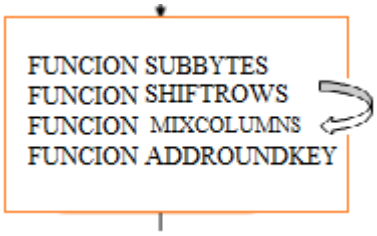


Figura 20-2. Cambio realizado a las funciones del AES

Realizado por: Raza C., 2019

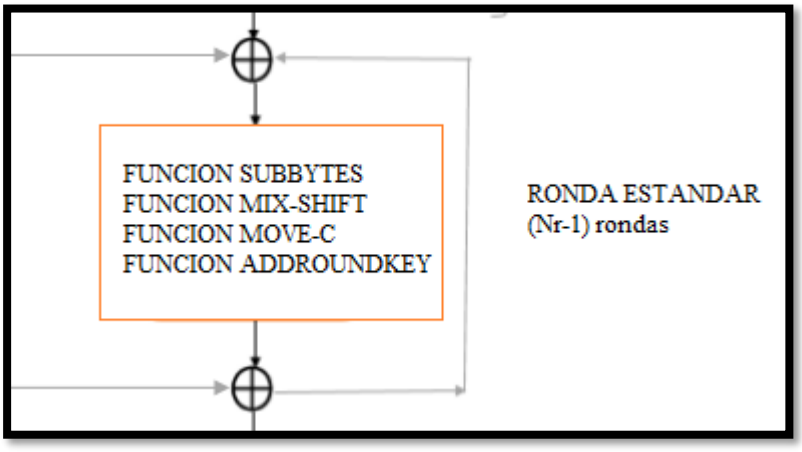


Figura 21-2. Modificación al proceso del algoritmo AES

Realizado por: Raza C., 2019

2.2.3 Mix-Key

En esta función la modificación al bloque de la clave, produce que la clave que ingresa al proceso de cifrado sea alterada al cambiar sus posiciones por medio de la utilización de un vector, el cual es definido por el programador, al momento de ingresar al proceso de expansión de clave, así como el proceso de cifrado será una clave diferente.

Para tal proceso se tiene en cuenta el tamaño del bloque de la clave, ya sea 128 bits, 192 bits, 256 bits representados por K. Los mismos que pasan por el siguiente proceso matemático (**Ecuación 1-1**) para encontrar el Nk este a su vez define el número de columnas.

$$N_K = K \,/\, 32$$

$N_K = K \,/\, 32$	$N_K = K \,/\, 32$	$N_K = K \,/\, 32$
$N_K = 128/32$	$N_K = 192/32$	$N_K = 256/32$
$N_K = 4$	$N_K = 6$	$N_K = 8$

Utilizaremos un ejemplo para un ingreso de una clave de 256 bits como se observa en las siguientes **Tablas 8-2; 9-2; 10-2**.

Tabla 8-2. Matriz 4x8, Clave de 256 bits.

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
A	E	I	M	Q	U	Y	CC
B	F	J	N	R	V	Z	DD
C	G	K	O	S	W	AA	EE
D	H	L	P	T	X	BB	FF

Realizado por: Raza C., 2019

Tabla 9-2. Vector y posiciones de la matriz

Posiciones Matriz Normal	Vector con valores definidos
0	**1**
1	**3**
2	**0**
3	**2**
4	**5**
5	**4**
6	**7**
7	**6**

Realizado por: Raza C., 2019

El cambio realizado a la matriz se realiza por columnas.

Tabla 10-2. Nueva Matriz 4x8, Clave de 256 bits.

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
I	A	M	E	U	Q	CC	Y
J	B	N	F	V	R	DD	Z
K	C	O	G	W	S	EE	AA
L	D	P	H	X	T	FF	BB

Realizado por: Raza C., 2019

Una vez realizado los cambios se modificó el diagrama del algoritmo AES para adaptarle las diferentes funciones modificadas.

Dando como resultado:

El diagrama de cifrado del MECIB-AES (**Figura 22-2**).

El diagrama de descifrado del MECIB-AES (**Figura 23-2**).

2.2.4 Diagrama del MECIB_AES

2.2.4.1 *Proceso de Cifrado del MECIB-AES*

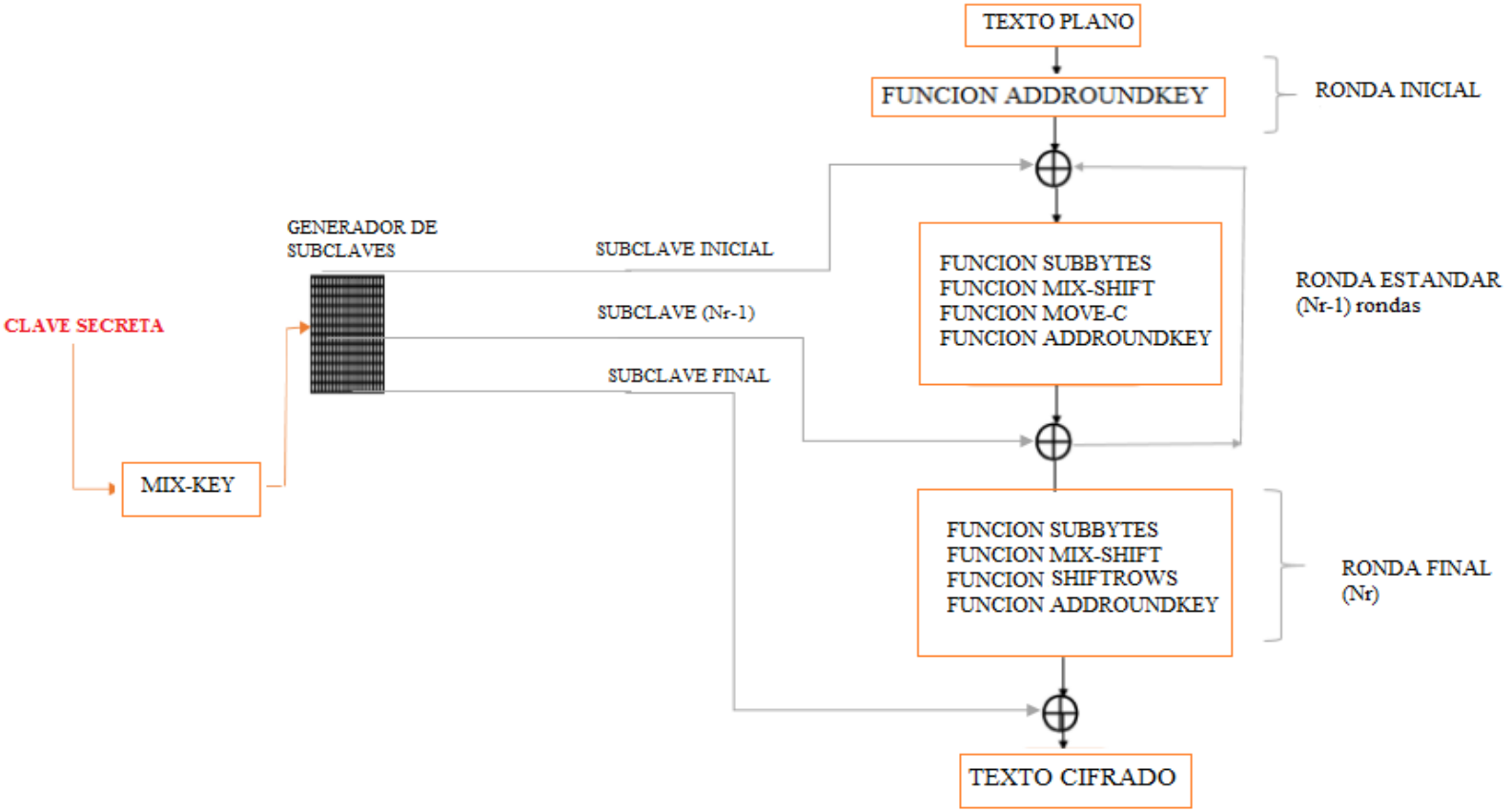


Figura 22-2. Proceso de Cifrado MECIB-AES

Realizado por: Raza C., 2019

Fuente: (Vázquez, 2007)

2.2.4.2 Proceso de Descifrado MECIB-AES

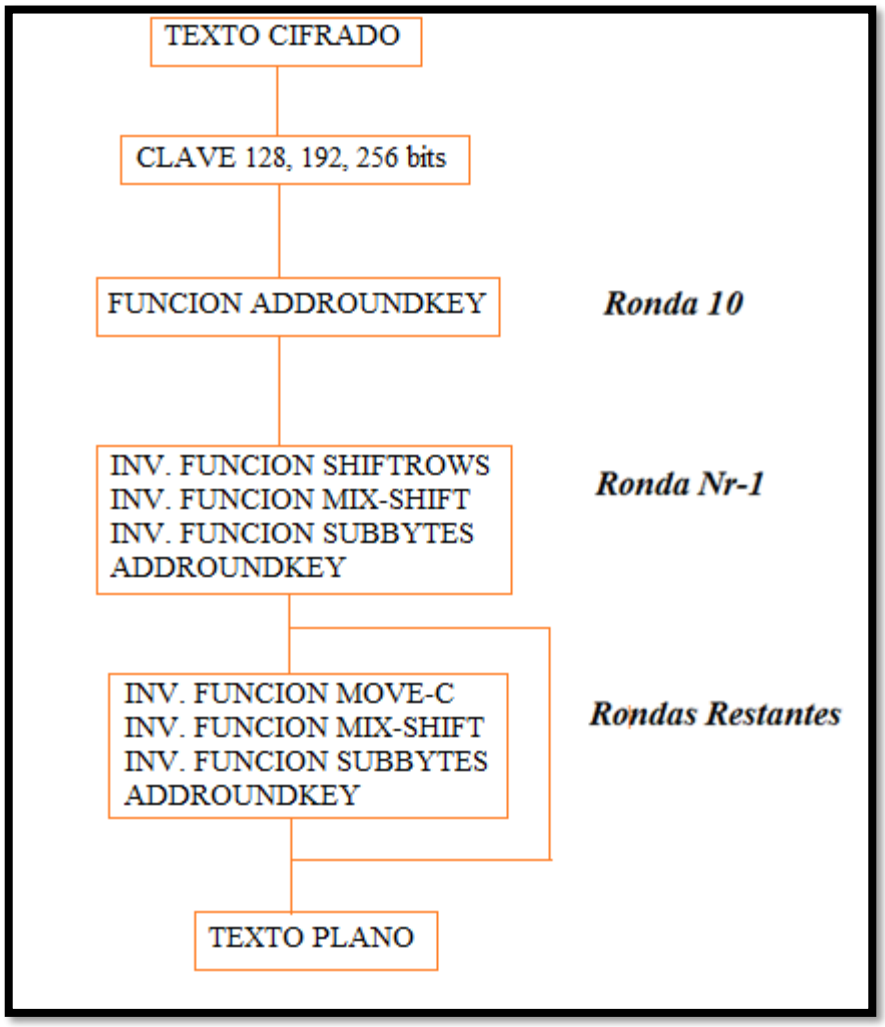


Figura 23-2. Proceso de Descifrado MECIB-AES

Realizado por: Raza C., 2019

2.2.5 Implementación del Mecanismo de Cifrado Basado en el AES

Al obtener los diagramas tanto para cifrado y descifrado se procedió a la implementación del MECIB_AES.

El algoritmo programado tiene el nombre de "Algoritmo AES Simétrico - Modificado" cuya interfaz con el usuario es como se presenta a continuación en la **Figura 24-2**:

Figura 24-2. Interfaz de la Segunda Aplicación, MECIB_AES.

Realizado por: Raza C., 2019

La programación está dividida en paquetes y clases (**Figura 25-2**) que ayudan a una mejor organización del mismo, por lo cual tenemos:

Paquete de fuentes

3 algoritmoaes

4 frm

5 imágenes; estos a su vez contienen a las clases:

El paquete algoritmoaes*;* contiene programado el algoritmo cada función y transformación utilizada en el cifrado y descifrado además de su proceso para la clave principal y subclaves.

- ✓ AES.java
- ✓ AlgoritmoAES.java
- ✓ Utilidades.java

- ✓ clsHexa.java

El paquete frm contiene las ventanas principales, cifrado y descifrado.

- ✓ frmPrincipal.java
- ✓ ifrmCifrar.java
- ✓ ifrmDesCifrar.java

El paquete imágenes contiene los siguientes archivos utilizados en la interfaz con el usuario.

- ✓ archivo.png
- ✓ cancelar.png
- ✓ clave.png
- ✓ cpu.png
- ✓ disquetes.png
- ✓ documento.png
- ✓ pagina-en-blanco.png
- ✓ revisar.png

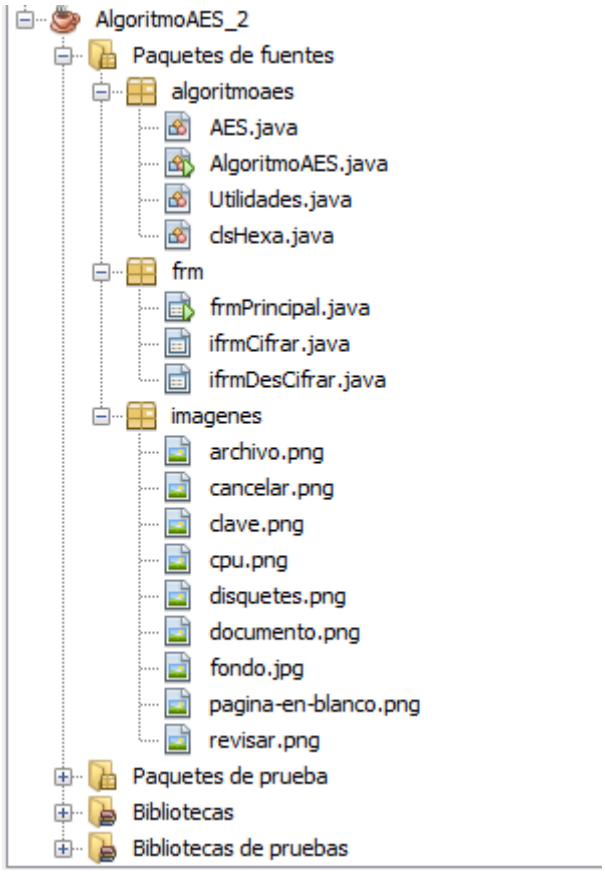


Figura 25-2. Paquetes utilizados para la aplicación MECIB-AES

Realizado por: Raza C., 2019

✓ Ventana de cifrado **Figura 26-2** y descifrado **Figura 27-2**.

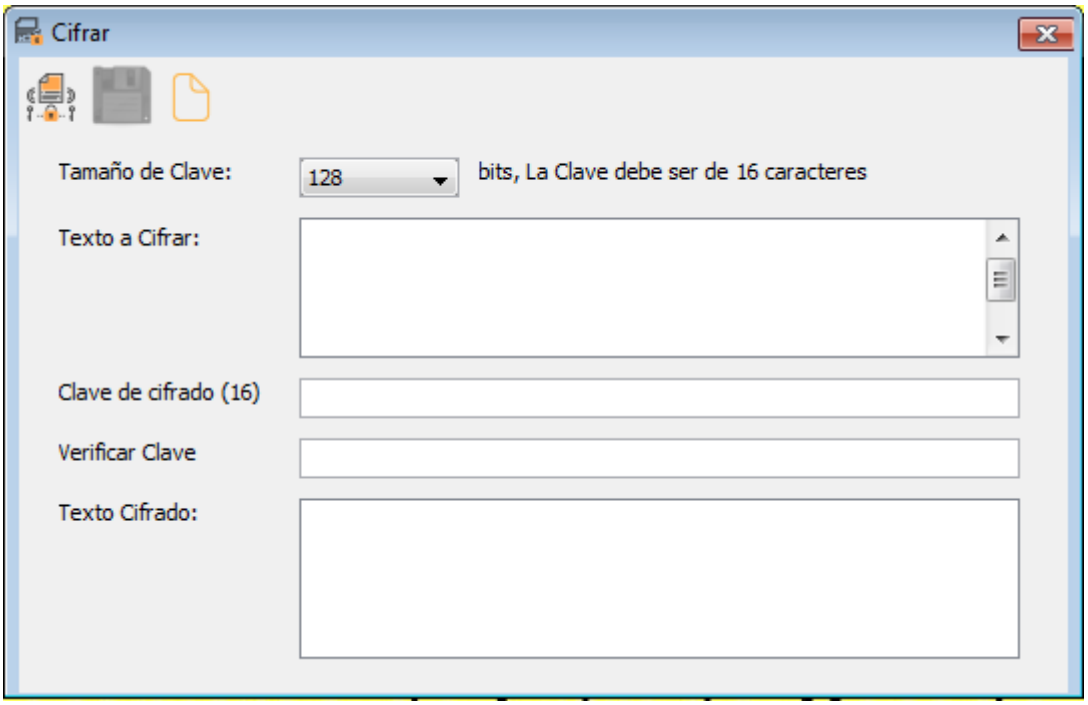


Figura 26-2. Interfaz de Cifrado MECIB-AES.

Realizado por: Raza C., 2019

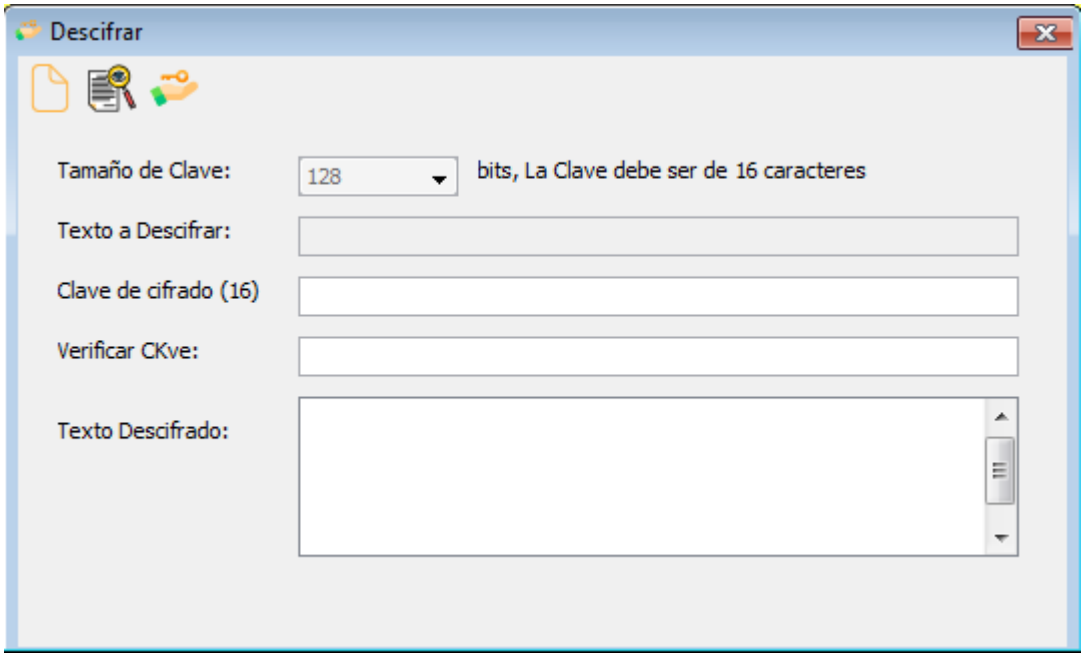


Figura 27-2. Interfaz de Descifrado MECIB-AES.

Realizado por: Raza C., 2019

✓ Verificación de clave

Primero hace una comprobacion de la longitud de la clave, ya sea 16, 24, 32 bits caracteres respectivamente, ademas de comprobar que las dos claves sean las mismas como se muestra en las **Figuras 28-2; 29-2.**

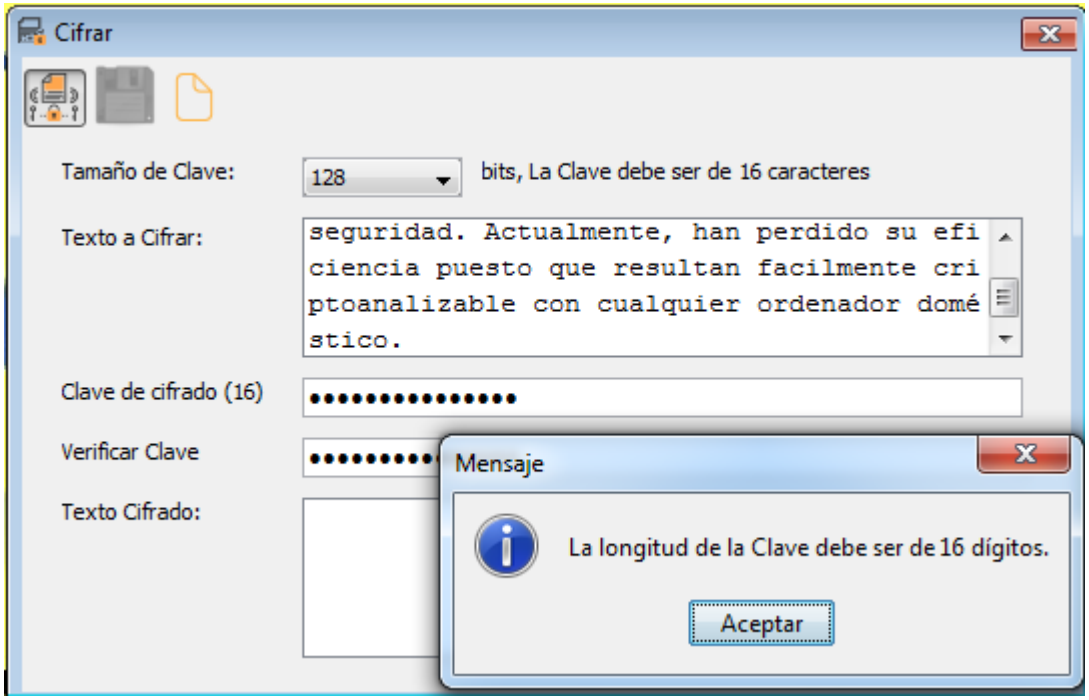


Figura 28-2. Longitud incorrecta de la clave en MECIB-AES.

Realizado por: Raza C., 2019

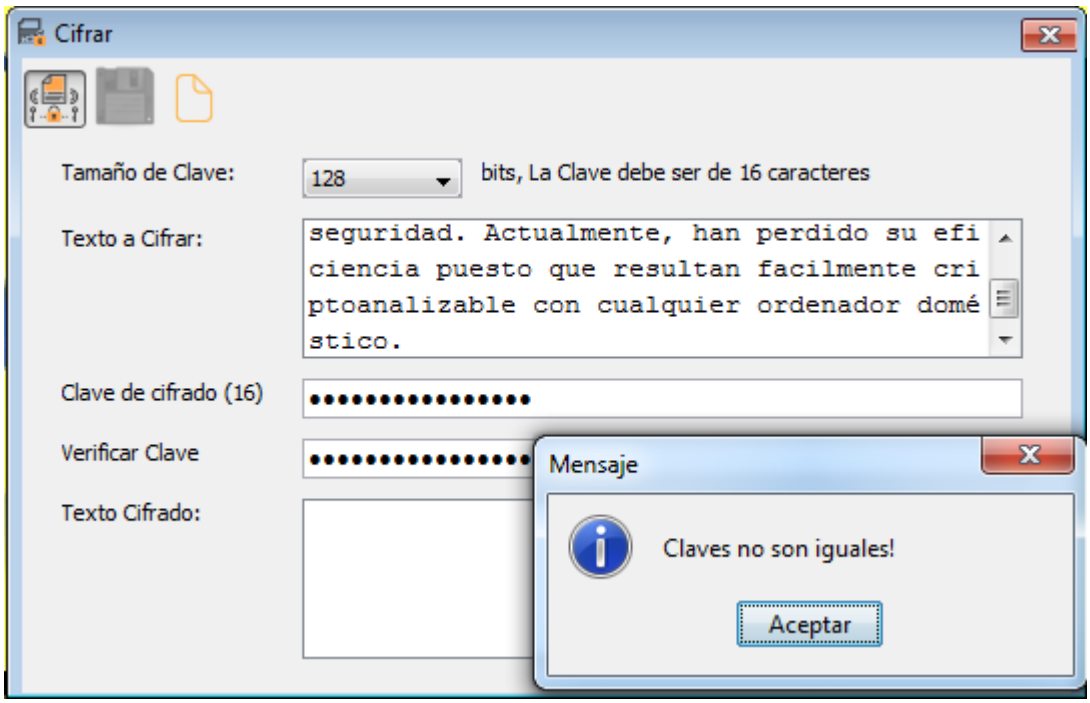


Figura 29-2. Verificación de Clave en MECIB-AES.

Realizado por: Raza C., 2019

Por consiguiente, la funcionalidad del algoritmo se puso a prueba al cifrar el siguiente párrafo. La criptografía clásica se basa en algoritmos sencillos y claves muy largas para la seguridad. Actualmente, han perdido su eficacia puesto que resultan fácilmente criptoanalizables empleando cualquier ordenador doméstico. (Huidobro, 2013)

- ✓ Cifrado (**Figuras 30-2; 31-2**) y Descifrado (**Figuras 32-2**) con clave de 128 bits, MECIB-AES.

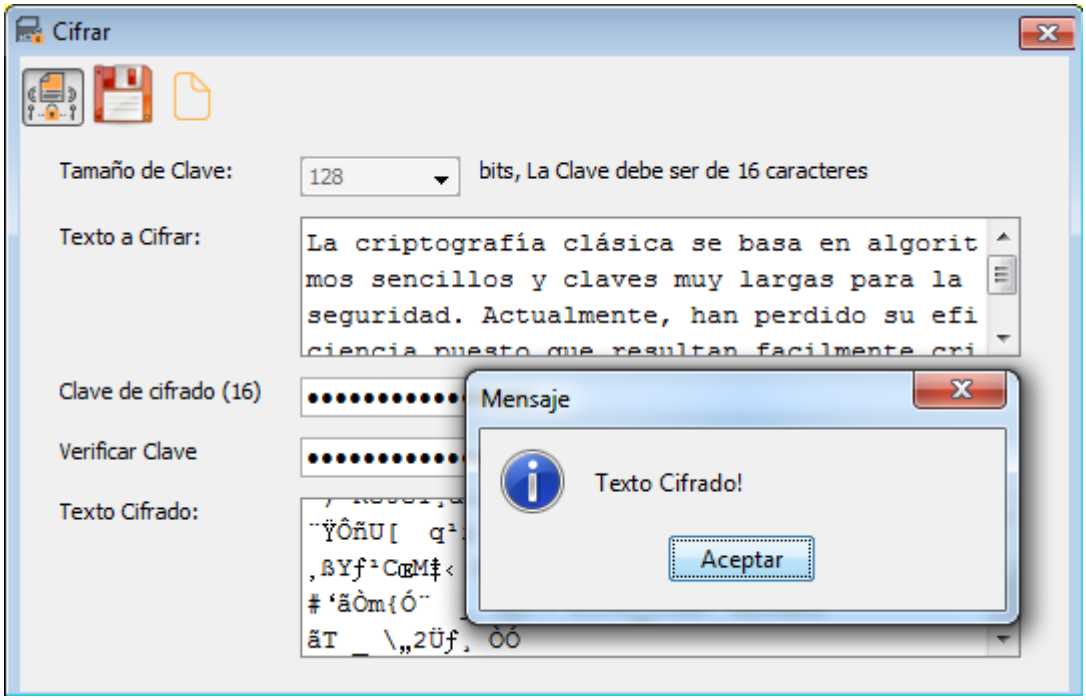


Figura 30-2. Texto cifrado con clave de 128 bits en MECIB-AES

Realizado por: Raza C., 2019

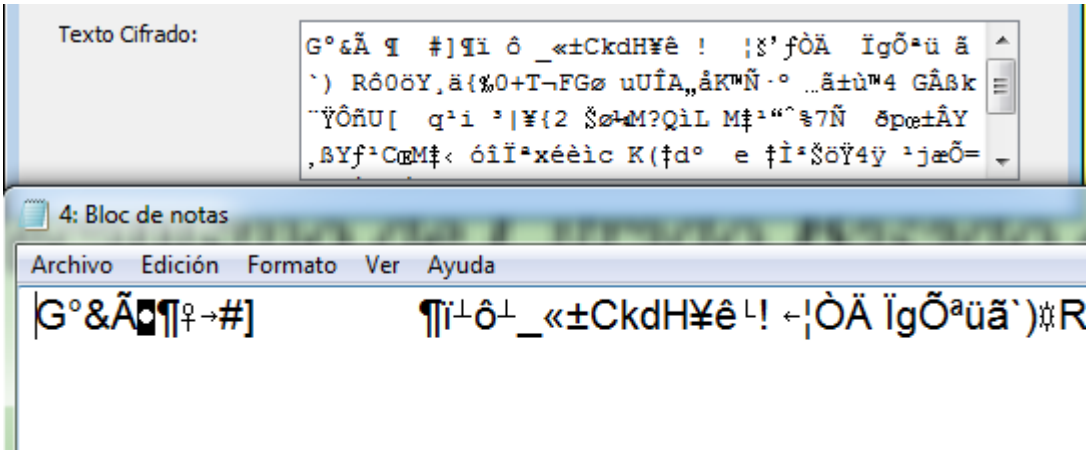


Figura 31-2. Texto cifrado guardado en .txt, clave 128 bits en MECIB-AES.

Realizado por: Raza C., 2019

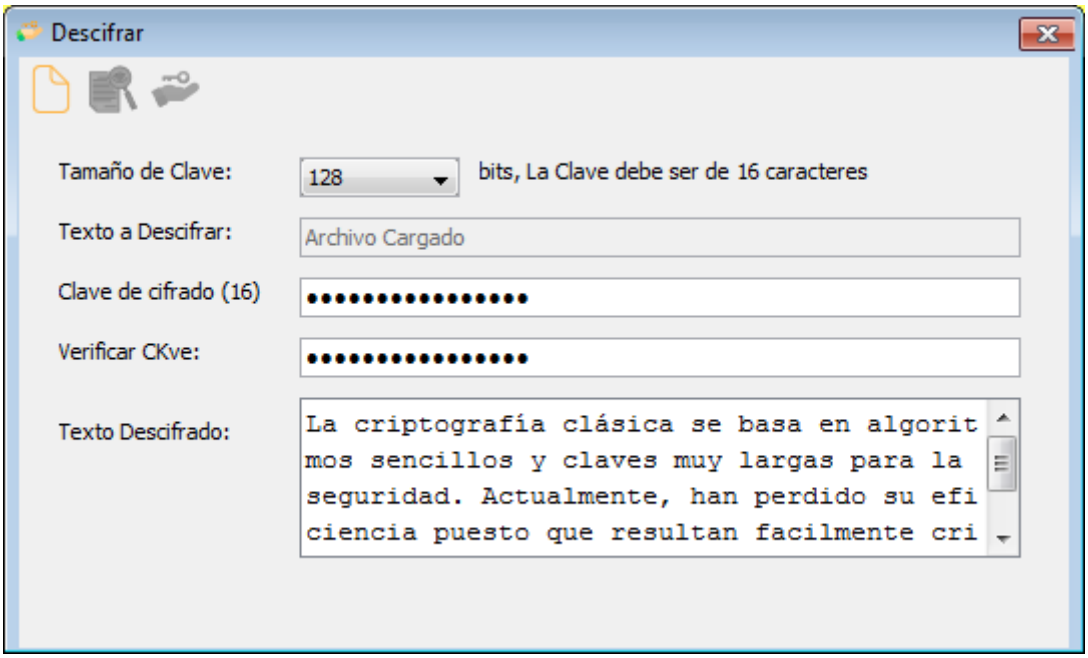


Figura 32-2. Texto descifrado con clave 128 bits en MECIB-AES.

Realizado por: Raza C., 2019

✓ Cifrado (**Figuras 33-2; 34-2**) y Descifrado (**Figuras 35-2**) con clave de 192 bits, MECIB-AES.

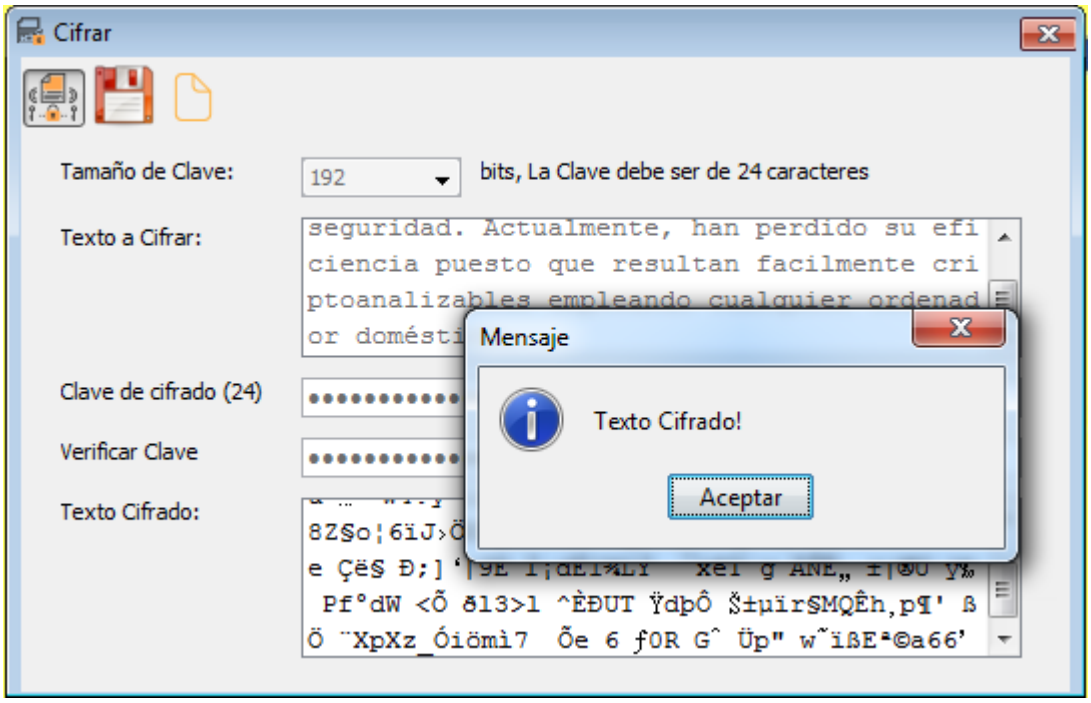


Figura 33-2. Texto cifrado con clave 192 bits en MECIB-AES.

Realizado por: Raza C., 2019

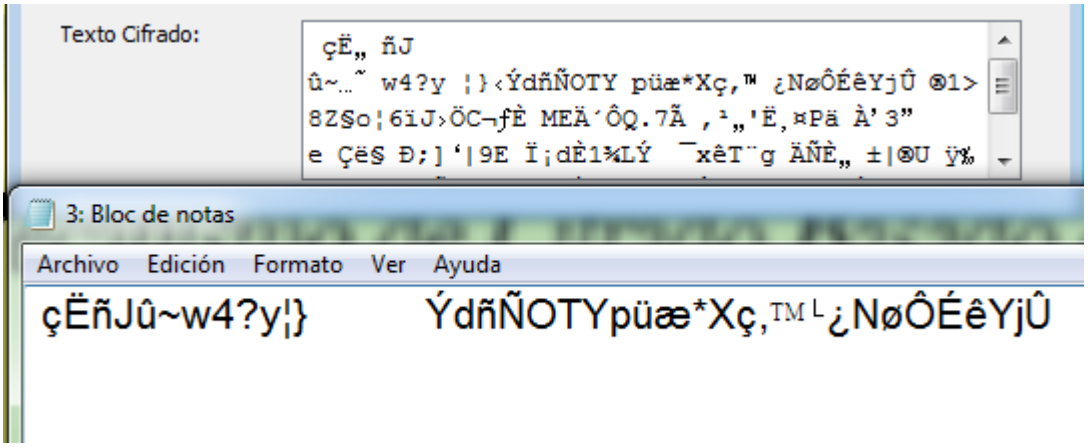


Figura 34-2. Texto cifrado guardado en .txt, clave 192 bits en MECIB-AES.

Realizado por: Raza C., 2019

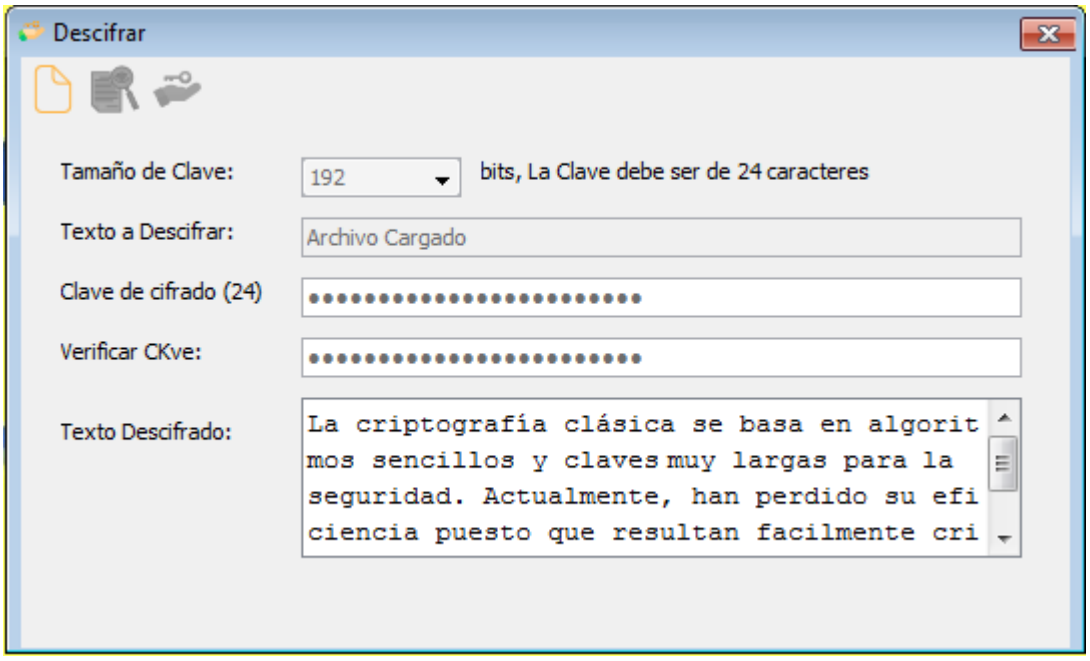


Figura 35-2. Texto descifrado con clave 192 bits en MECIB-AES.

Realizado por: Raza C., 2019

✓ Cifrado (**Figuras 36-2; 37-2**) y Descifrado (**Figuras 38-2**) con clave de 256 bits, MECIB-AES.

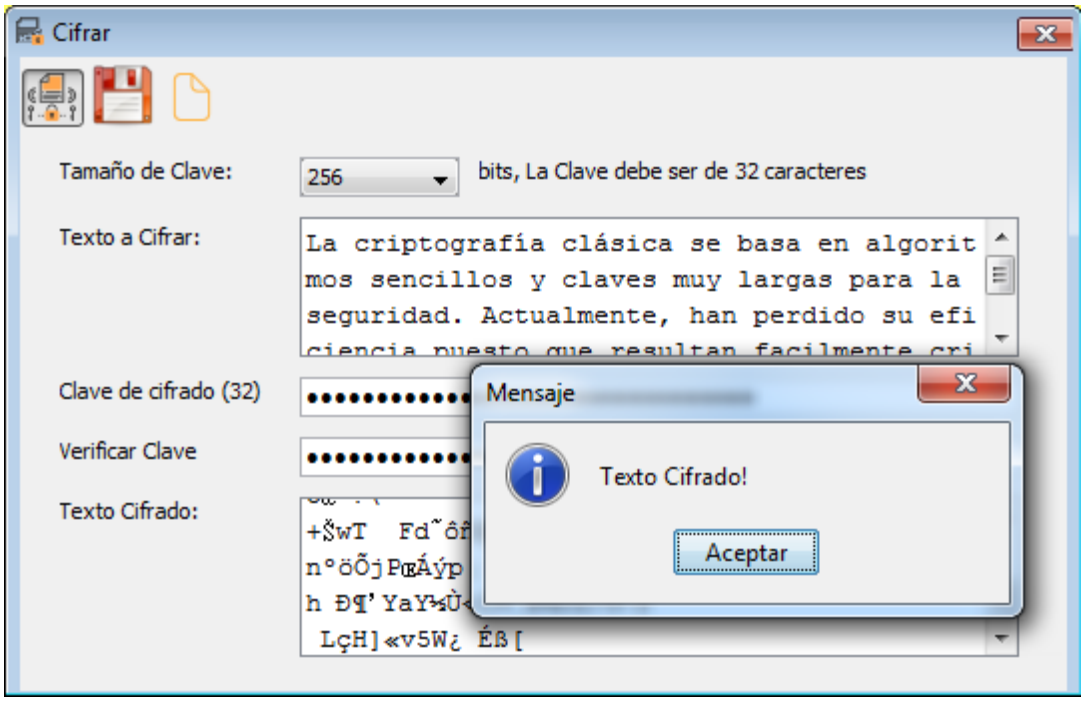


Figura 36-2. Texto cifrado con clave 256 bits en MECIB-AES.

Realizado por: Raza C., 2019

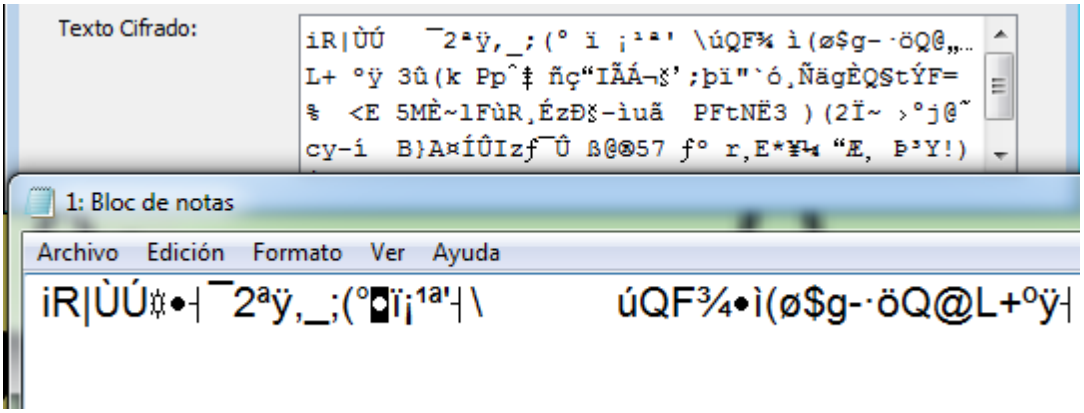


Figura 37-2. Texto cifrado guardado en .txt, clave 256 bits en MECIB-AES.

Realizado por: Raza C., 2019

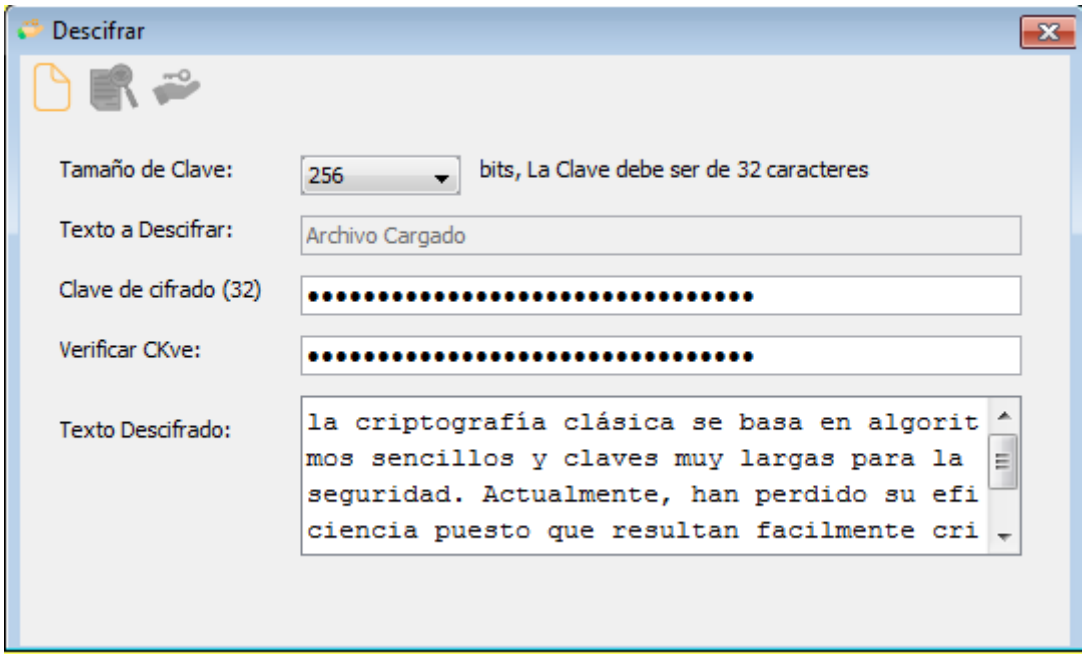


Figura 38-2. Texto descifrado con clave 256 bits en MECIB-AES.

Realizado por: Raza C., 2019

2.3 COMPLEJIDAD DE LOS ALGORITMOS AES Y MECIB-AES

La complejidad (o costo) de un algoritmo es una medida de la cantidad de recursos (tiempo, memoria) que el algoritmo necesita. La complejidad de un algoritmo se expresa en función del tamaño (o talla) del problema. La función de complejidad tiene como variable independiente el tamaño del problema y sirve para medir la complejidad (espacial o temporal). Mide el tiempo/espacio relativo en función del tamaño del problema. (Dávila, 2017)

El comportamiento de la función determina la eficiencia. No es única para un algoritmo: depende de los datos. Para un mismo tamaño del problema, las distintas presentaciones iniciales de los datos dan lugar a distintas funciones de complejidad. Es el caso de una ordenación si los datos están todos inicialmente desordenados, parcialmente ordenados o en orden inverso. (Dávila, 2017)

2.3.1 Notación Asintótica

La notación asintótica se describe por medio de una función cuyo dominio son los números naturales (N) estimados a partir de tiempo de ejecución o de espacio de memoria de algoritmos en base a la longitud de la entrada. Se consideran las funciones asintóticamente no negativas. La notación asintótica captura el comportamiento de la función para valores grandes de *N*. (Dávila, 2017)

2.3.2 La O Mayúscula

La notación O se utiliza para comparar funciones. Resulta particularmente útil cuando se quiere analizar la complejidad de un algoritmo, en otras palabras, la cantidad de tiempo que le toma a un computador ejecutar un programa. (Dávila, 2017)

Definición: Sean f y g funciones con dominio en **R** ≤ 0 o **N** es imagen en R. Si existen constantes C y k tales que: (Dávila, 2017)

$$\forall\ x > k, |f\ (x)| \leq C\ |\ g\ (x)\ |$$

Es decir, que para x > k, f es menor o igual a un múltiplo de g, decimos que: (Dávila, 2017)

$$f\ (x) = O\ (g\ (x))$$

La definición formal es:

$$f\ (x) = O\ (g\ (x)) \Leftrightarrow \exists\ k, N\ |\ \forall\ x > N, |\ f\ (x)\ | \leq k |\ g\ (x)\ |$$

Por lo tanto, se obtiene como resultado que una función es menor a otra al despreciar los valores constantes y pequeños.

Así, $3N^3 + 5N^2 - 9 = O\ (N^3)$, no significa que existe una función $O(N^3)$ que es igual a: $3N^3 + 5N^2 - 9$. (Dávila, 2017)

Debe leerse como: (Dávila, 2017)

"$3N^3+5N^2-9$ es O-Grande de N^3"

Que significa: (Dávila, 2017)

"$3N^3+5N^2-9$

Está asintóticamente dominada por N^3".

2.4 POBLACIÓN

Dado que estamos cifrando diferentes caracteres y debido a que no hay un límite para las combinaciones que puedan formar se utiliza una población infinita. Es por este motivo que se limitó la población utilizando conglomerados y sectorización (**Figura 39-2**). Además, basándonos en procesos estocásticos y las clases de equivalencia podemos utilizar una población definida debido a que la matriz estocástica al sumar su fila o columna da como resultado 1.

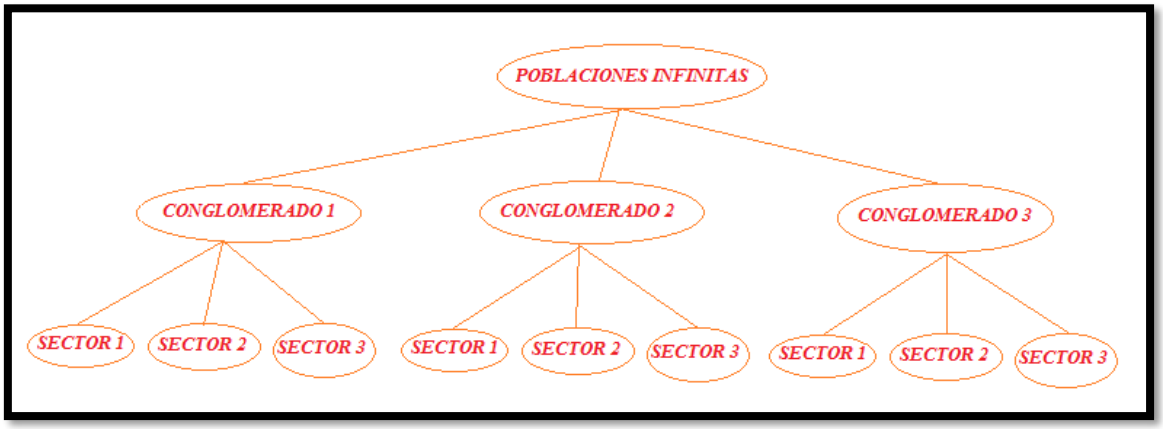


Figura 39-2. Conglomerados y Sectores en estadística

Realizado por: Raza C., 2019

Debido a esto se tomó como población cien (100) párrafos de diferentes tamaños y contenido, los mismos que una vez realizado el cifrado poseen las siguientes características como se muestra en la Tabla 11-2; cabe recalcar que la información cifrada se guarda en Bloc de Notas (.txt).

Tabla 11-2. Archivos Cifrados con el algoritmo AES y MECIB-AES.

Algoritmo AES	Algoritmo MECIB-AES	Tamaño AES (Bytes)	Tamaño MECIB-AES (Bytes)
Prueba_1	Prueba_m_1	722	719
Prueba_2	Prueba_m_2	410	407
Prueba_3	Prueba_m_3	754	744
Prueba_4	Prueba_m_4	1960	1937
Prueba_5	Prueba_m_5	648	630
Prueba_6	Prueba_m_6	576	578
Prueba_7	Prueba_m_7	292	293
Prueba_8	Prueba_m_8	345	357
Prueba_9	Prueba_m_9	269	262

Prueba_10	Prueba_m_10	527	525
Prueba_11	Prueba_m_11	402	398
Prueba_12	Prueba_m_12	745	743
Prueba_13	Prueba_m_13	474	474
Prueba_14	Prueba_m_14	218	206
Prueba_15	Prueba_m_15	312	323
Prueba_16	Prueba_m_16	332	332
Prueba_17	Prueba_m_17	210	219
Prueba_18	Prueba_m_18	298	298
Prueba_19	Prueba_m_19	460	477
Prueba_20	Prueba_m_20	204	161
Prueba_21	Prueba_m_21	466	450
Prueba_22	Prueba_m_22	147	144
Prueba_23	Prueba_m_23	377	374
Prueba_24	Prueba_m_24	119	126
Prueba_25	Prueba_m_25	217	220
Prueba_26	Prueba_m_26	372	392
Prueba_27	Prueba_m_27	183	185
Prueba_28	Prueba_m_28	316	304
Prueba_29	Prueba_m_29	484	482
Prueba_30	Prueba_m_30	282	298
Prueba_31	Prueba_m_31	996	1024
Prueba_32	Prueba_m_32	422	414
Prueba_33	Prueba_m_33	146	142
Prueba_34	Prueba_m_34	442	435
Prueba_35	Prueba_m_35	645	662
Prueba_36	Prueba_m_36	599	587
Prueba_37	Prueba_m_37	515	484
Prueba_38	Prueba_m_38	626	633
Prueba_39	Prueba_m_39	2020	2005
Prueba_40	Prueba_m_40	2507	2539
Prueba_41	Prueba_m_41	870	882
Prueba_42	Prueba_m_42	278	291
Prueba_43	Prueba_m_43	213	224
Prueba_44	Prueba_m_44	260	270

Prueba_45	Prueba_m_45	244	237
Prueba_46	Prueba_m_46	1488	1498
Prueba_47	Prueba_m_47	856	847
Prueba_48	Prueba_m_48	168	172
Prueba_49	Prueba_m_49	546	553
Prueba_50	Prueba_m_50	799	826
Prueba_51	Prueba_m_51	876	850
Prueba_52	Prueba_m_52	505	494
Prueba_53	Prueba_m_53	501	504
Prueba_54	Prueba_m_54	428	423
Prueba_55	Prueba_m_55	164	167
Prueba_56	Prueba_m_56	288	291
Prueba_57	Prueba_m_57	368	388
Prueba_58	Prueba_m_58	1272	1284
Prueba_59	Prueba_m_59	149	147
Prueba_60	Prueba_m_60	245	235
Prueba_61	Prueba_m_61	647	634
Prueba_62	Prueba_m_62	230	254
Prueba_63	Prueba_m_63	367	362
Prueba_64	Prueba_m_64	387	389
Prueba_65	Prueba_m_65	550	545
Prueba_66	Prueba_m_66	494	509
Prueba_67	Prueba_m_67	377	374
Prueba_68	Prueba_m_68	279	288
Prueba_69	Prueba_m_69	269	263
Prueba_70	Prueba_m_70	391	387
Prueba_71	Prueba_m_71	1813	1773
Prueba_72	Prueba_m_72	497	503
Prueba_73	Prueba_m_73	141	136
Prueba_74	Prueba_m_74	292	288
Prueba_75	Prueba_m_75	229	239
Prueba_76	Prueba_m_76	487	482
Prueba_77	Prueba_m_77	104	96
Prueba_78	Prueba_m_78	286	284
Prueba_79	Prueba_m_79	303	315

Prueba_80	Prueba_m_80	311	301
Prueba_81	Prueba_m_81	335	329
Prueba_82	Prueba_m_82	193	196
Prueba_83	Prueba_m_83	611	593
Prueba_84	Prueba_m_84	219	214
Prueba_85	Prueba_m_85	282	288
Prueba_86	Prueba_m_86	592	605
Prueba_87	Prueba_m_87	303	317
Prueba_88	Prueba_m_88	427	441
Prueba_89	Prueba_m_89	278	299
Prueba_90	Prueba_m_90	170	170
Prueba_91	Prueba_m_91	491	493
Prueba_92	Prueba_m_92	622	632
Prueba_93	Prueba_m_93	611	822
Prueba_94	Prueba_m_94	196	187
Prueba_95	Prueba_m_95	554	555
Prueba_96	Prueba_m_96	451	452
Prueba_97	Prueba_m_97	448	462
Prueba_98	Prueba_m_98	321	302
Prueba_99	Prueba_m_99	365	352
Prueba_100	Prueba_m_100	341	331

Realizado por: Raza C., 2019

2.4.1 Muestra

Estará dada por los diferentes archivos y claves que se utilizó para cifrar los párrafos y su tamaño dependerá del resultado de la siguiente formula:

Ecuación 2-2. Tamaño de la Muestra

$$n = \frac{N \cdot Z\alpha^2 \cdot p \cdot q}{d^2 \cdot (N-1) + Z_{\alpha}^2 \cdot p \cdot q}$$

Donde: (Herrera, 2011)

- ✓ N = 100
- ✓ Zα= 1.96, equivalente al 95%
- ✓ p = 5% = 0.05; proporción esperada
- ✓ q = 0.95
- ✓ d = 5%; precisión

$$n = \frac{100 \cdot 1{,}96^2 \cdot 0{,}05 \cdot 0{,}95}{0{,}05^2 \cdot (99) + 1{,}96^2 \cdot 0{,}05 \cdot 0{,}95}$$

$$n = 42.43$$

$$n = 42$$

2.4.2 Operacionalización de variables

Variables utilizadas para realizar el análisis y comparación de los algoritmos (**Tabla 12-2**).

Tabla 12-2. Operacionalización de variables

Variable	Dimensiones	Indicadores
Variable Independiente Análisis del Algoritmo AES (Advanced Encryption Advanced).	Complejidad Temporal.	$O_1(n)$
Variable Dependiente Diseño e implementación de las modificaciones realizadas al AES (Advanced Encryption Advanced).	Complejidad Temporal.	$O_2(n)$
Análisis y Comparación de los algoritmos.	Complejidad Temporal. Entropía Análisis de Frecuencia Autocorrelación Fuerza Bruta Análisis de Tráfico	$A_1(n)$ vs $A_2(n)$

Realizado por: Raza C., 2019

Fuente: (Dávila, 2017)

2.5 PRUEBAS A EFECTUAR

Las pruebas a continuación ayudarán a realizar un criptoanálisis al texto cifrado con las dos diferentes aplicaciones desarrolladas anteriormente y a su vez permitirán medir el nivel de confiabilidad en el siguiente capítulo.

2.5.1 Entropía

La entropía de un documento es un índice de su contenido de información. La entropía se mide en bits por carácter. (Esslinger, 1998 - 2018)

✓ Contenido de la información de una fuente

Desde el punto de vista de la teoría de la información, los datos en la ventana actual se pueden ver como un origen de mensaje. Para calcular el contenido de información, uno examina la distribución de probabilidad de esta fuente. Aquí se supone que los mensajes individuales (caracteres en el documento / archivo) son estocásticamente independientes entre sí y son transmitidos por la fuente con una probabilidad uniforme. (Esslinger, 1998 - 2018), (Areitio, y otros, 2009)

El contenido de información de un mensaje M [i] está definido por:

Ecuación 3-2. Ecuación de Entropía

$$(M [i]): = \log (1 / p [i]) = -\log (p [i]),$$

Donde p [i] es la probabilidad de que el mensaje M [i] sea transmitido por el origen del mensaje y log denota logaritmos a base 2. (Esslinger, 1998 - 2018)

Esto significa que el contenido de la información depende exclusivamente de la distribución de probabilidad con la que la fuente genera los mensajes. El contenido semántico del mensaje no entra en el cálculo. Como el contenido de información de un mensaje inusual es más alto que el de un mensaje común, el valor inverso de la probabilidad se usa en la definición. (Esslinger, 1998 - 2018)

La entropía de una fuente indica su distribución característica. Mide la cantidad promedio de información que uno puede obtener a través de la observación de la fuente o, por el contrario, la indeterminación que prevalece sobre los mensajes generados cuando uno no puede observar la fuente. (Esslinger, 1998 - 2018), (Areitio, y otros, 2009)

$$(p\,[1], p\,[2], ..., p\,[r]) := -\,[p\,[1] * \log(p\,[1]) + p\,[2] * \log(p\,[2]) + ... + p\,[r] * \log(p\,[r])]$$

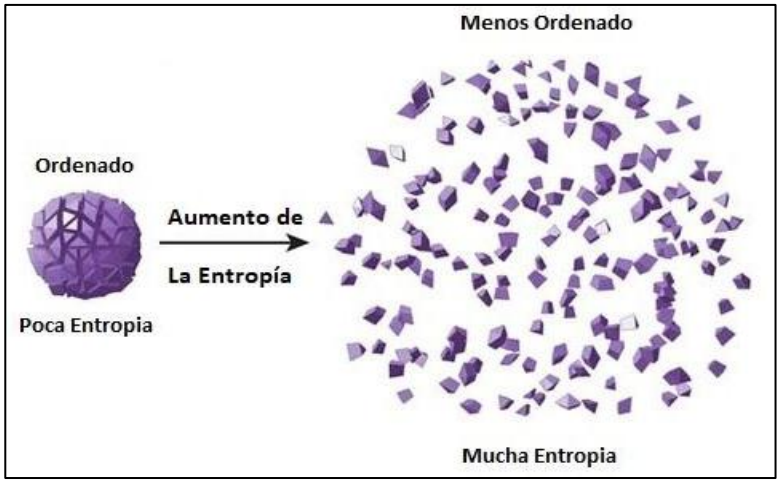


Figura 40-2. Mayor o menor Entropía.

Realizado por: Raza C., 2019

Fuente: http://www.areaciencias.com/fisica/entropia.html

✓ Valores extremos de entropía

Para documentos que contienen solo letras mayúsculas, la entropía se encuentra entre 0 bit / char (en un documento que consta de un solo carácter) y log (26) bit / char = 4.700440 bit / char (en un documento en el que los 26 caracteres aparecen con la misma frecuencia). (Gómez, 2010), (Esslinger, 1998 - 2018)

Para los documentos que pueden contener todos los caracteres del conjunto de caracteres (0 a 255), la entropía se encuentra entre 0 bit / char (en un documento que consta de un solo carácter) y log (256) bit / char = 8 bit / char (en un documento en el que los 256 caracteres aparecen con la misma frecuencia). (Gómez, 2010), (Esslinger, 1998 - 2018)

Ejemplo:

✓ Mensaje

X = {a, d, s, f, w, e, f, g, h, w, w, w, s, d, f, g, d, a, a, s, d, g, h, e, e, s, d, f}

✓ Alfabeto

S = {a, s, d, f, w, e, g, h,}

✓ Análisis frecuencial:

fa = 3/28

fs = 4/28

fd = 5/28

ff = 4/28

fw = 4/28

fe = 3/28

fg = 3/28

fh = 2/28

Entropía: Calculo de la entropía con cálculos matemáticos (**Ecuación 3-2)** y por medio del software CrypTool (**Figuras 41-2; 42-2**).

$$H(x) = \frac{3}{28} * \log_2 \frac{1}{3/28} + \frac{4}{28} * \log_2 \frac{1}{4/28} + \frac{5}{28} * \log_2 \frac{1}{5/28} + \frac{4}{28} * \log_2 \frac{1}{4/28} + \frac{4}{28} * \log_2 \frac{1}{4/28} + \frac{3}{28} * \log_2 \frac{1}{3/28} + \frac{3}{28} * \log_2 \frac{1}{3/28} + \frac{2}{28} * \log_2 \frac{1}{2/28}$$

$$= \mathbf{2.9547}$$

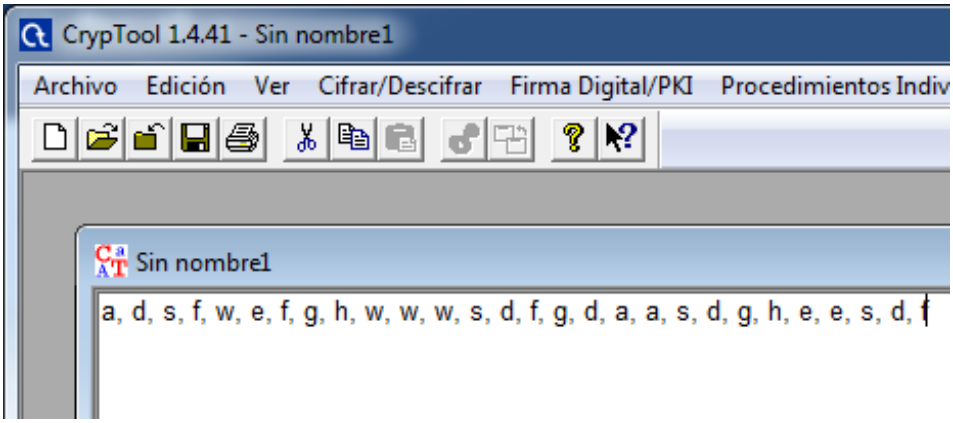


Figura 41-2. Mensaje ingresado en CrypTool.

Realizado por: Raza C., 2019

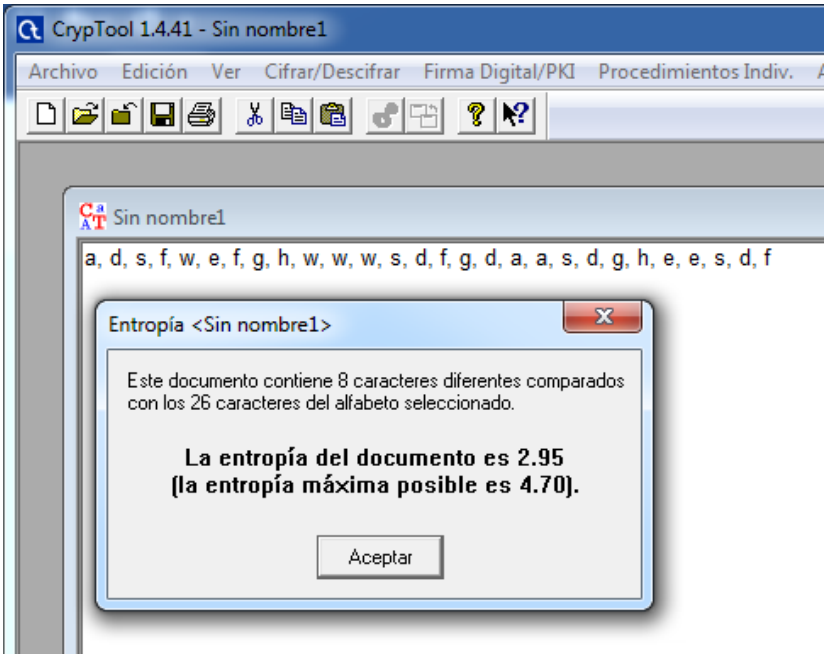


Figura 42-2. Calculo de entropía a Sin nombre1

Realizado por: Raza C., 2019

El resultado obtenido de forma teórica y por medio del programa CrypTool es el mismo así, podemos constatar que la entropía medida por el software CrypTool es correcta (**Figura 42-2**).

2.5.2 Análisis de Frecuencias

La situación ideal es que todos los caracteres aparezcan con la misma frecuencia, siendo así más complicado el descifrado del documento. Por el contrario, si el histograma (**Figura 43-2**) es desigual y las frecuencias presentan grandes variaciones entre símbolos, el cifrado será débil y fácilmente recuperado el documento original a través de un estudio de frecuencias del idioma en que se escribió. (Gómez, 2010)

Para obtener información más directa de cada uno de las frecuencias, se estudiará también el N-grama, que proporciona una tabla indicando la frecuencia de todos o parte de los caracteres del histograma. Esta herramienta también permite obtener las frecuencias de los bigramas, trigramas, etc., datos que pueden ser de gran utilidad en ciertos casos. (Gómez, 2010)

Ejemplo:

Se presenta la siguiente imagen donde encontramos la visualización del histograma CrypTool.tx.

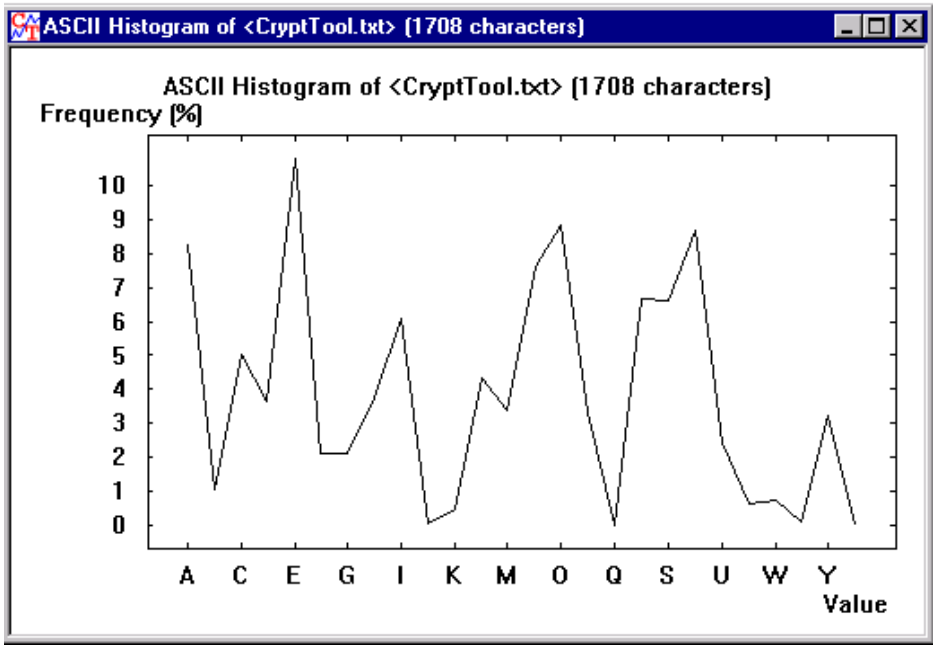


Figura 43-2. Histograma de CryptTool.txt

Realizado por: Raza C., 2019

Fuente: (Esslinger, 1998 - 2018)

En el siguiente histograma podemos comprobar que la letra que más aparece es la E y la que menos aparece es la letra Q, lo que nos lleva a tener una mejor selección de letras que el mensaje contiene.

Ejemplo:

El siguiente mensaje X = a, d, d, f, g, a, d, d, e, e, f, g, r; será útil para revisar los N-gramas además de brindarnos una forma detallada de la frecuencia real de cada letra (**Figura 44-2**).

- ✓ a = 2 f = 2 e = 2
- ✓ d = 4 g = 2 r = 1

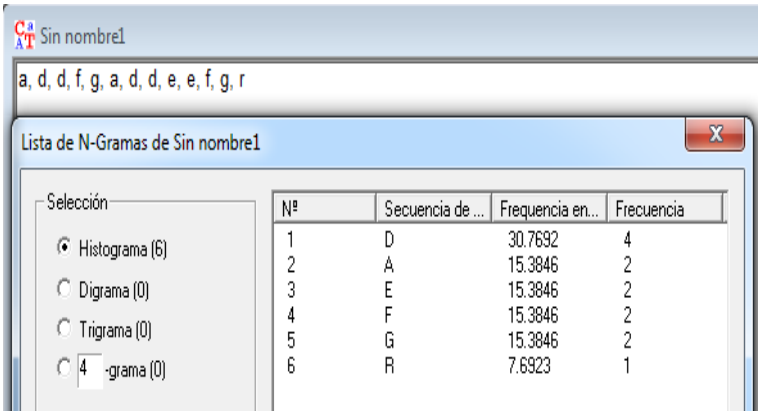


Figura 44-2. Lista de N-Gramas de Sin nombre1

Realizado por: Raza C., 2019

Podemos ver la lista de las secuencias en el texto, así como también sus respectivas frecuencias y podemos constatar mediante el cálculo hecho teóricamente esta realizado correctamente; dando como resultado la frecuencia más alta de la letra D y la más baja la letra R.

2.5.3 Autocorrelación

La autocorrelación de un documento es un índice de la similitud de diferentes secciones del documento. A veces es posible calcular la longitud de la clave de un documento cifrado a partir de su autocorrelación. (Gómez, 2010), (Esslinger, 1998 - 2018)

La función de autocorrelación C (t) mide la similitud de la secuencia (s [i]) = s [1] s [2] ... a la secuencia (s [i + t]) = s [t] s [t + 1] .... Que se desplaza por t lugares. Se examina una secuencia de longitud n y se definen los siguientes parámetros: (Gómez, 2010), (Esslinger, 1998 - 2018)

- ✓ A (t): = número de miembros de secuencias (s [i]) y (s [i + t]) en el segmento bajo consideración que concuerdan. (Gómez, 2010), (Esslinger, 1998 - 2018)
- ✓ D (t): = número de miembros de secuencias (s [i]) y (s [i + t]) en el segmento bajo consideración que no concuerdan. (Gómez, 2010), (Esslinger, 1998 - 2018)

$$C(t) = (A(t) - D(t)) / n.$$

En el caso de secuencias finitas, la secuencia s se desplaza (cíclicamente) por posiciones t, de modo que la secuencia (s [i + t]) tiene solo miembros de secuencia n-t (donde n es la longitud de la secuencia (s [i]). calcule la autocorrelación, ahora considera las secuencias s [1] s [2] ... s [n-t + 1] y s [t] s [t + 1] ... s [n] y calcula su correlación. (Gómez, 2010), (Esslinger, 1998 - 2018)

Ejemplo:

El propósito de esta prueba empírica de independencia es verificar las correlaciones entre los resultados sucesivos del generador de números pseudoaleatorios y / o entre las secuencias binarias y una versión alternativa de s desplazada por las posiciones t. (Schiestl, 1999)

T debe estar en el rango, 1 <= t <= (n / 2) y ser fijo. El número de posiciones de bit en s que no concuerdan con la versión de s que ha sido desplazada por t posiciones está determinado por: (Schiestl, 1999)

D (t) = suma [i = 0; i = n - t - 1] si XOR s (i + t)

Las estadísticas de prueba utilizadas están dadas por: (Schiestl, 1999)

X5 = 2 * [(D (t) - [(n - t) / 2]) / (n - t) ^ (1/2)];

Por lo que X5 se aproxima a una distribución N (0, 1), siempre que n - t> = 10.

s = (Schiestl, 1999)

00010111 01101101 01111101 11110011 00101111

00001111 10100100 11001111 11000011 11010001

11010001 00101110 11010100 11000011 01010001

11010110 00110010 10001111 00000111 01000111

Cuya longitud n = 160 (20 bytes). El nivel de significancia se establece con alpha = 0.05.

Elegimos t = 8. La secuencia s parte de la copia de s desplazada por 8 posiciones precisamente en D (8) = 68 posiciones. La prueba de autocorrelación produce una estadística de prueba de: (Schiestl, 1999)

*X5 = 2 * (68 - [(160 - 8) / 2) / (160 - 8) ^ (1/2) = -1.29777*

El límite para una estadística de prueba distribuida normalmente X5 es 1.6449 con un nivel de significancia de alfa = 0.05. Con X5 = -1.29777 <= 1.6449, la secuencia s pasa la prueba de autocorrelación. (Schiestl, 1999)

2.5.4 Fuerza Bruta

Un ataque de fuerza bruta es un ataque en el que se intentan todas las claves posibles a su vez en un intento de descubrir la clave con la que se ha cifrado el documento. El tiempo de CPU requerido aumenta muy drásticamente a medida que aumenta la longitud de la clave. (Esslinger, 1998 - 2018)

Ejemplo:

✓ Ataque de fuerza bruta a un archivo con clave de 128 bits. **Figura 46-2**

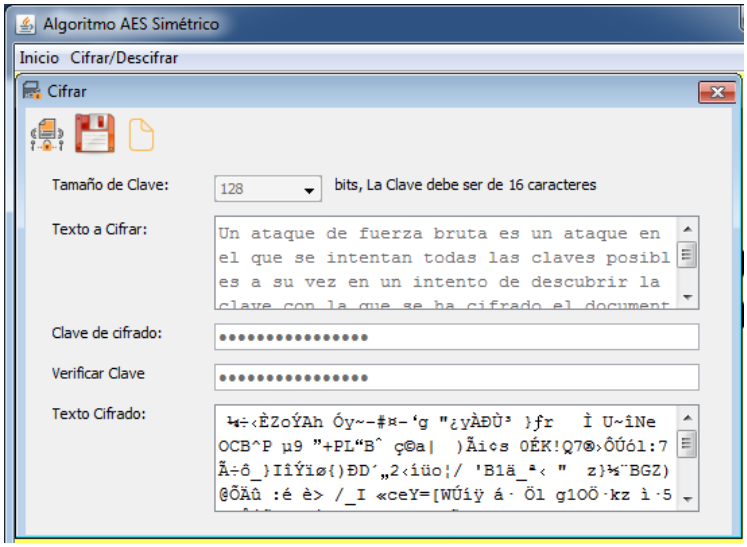


Figura 45-2. Texto cifrado: TEXTO_128BITS

Realizado por: Raza C., 2019

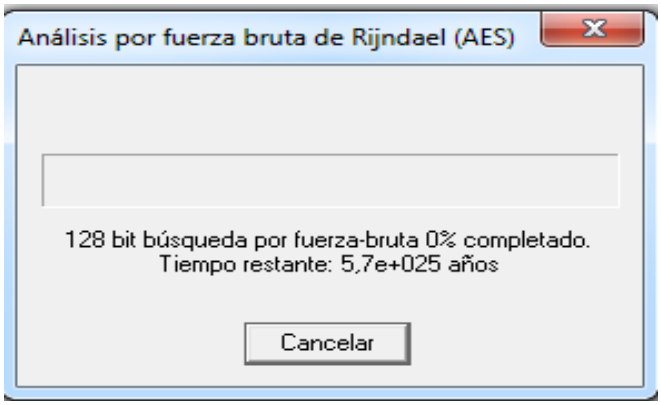


Figura 46-2. Resultado de la prueba de fuerza bruta 1 a TEXTO_128BITS

Realizado por: Raza C., 2019

El resultado mostrado a partir de la realización de la prueba de entropía a un texto cifrado con una clave de longitud 128 bits es la que nos muestra en la **Figura 46-2** donde podemos observar el tiempo en años que tardaría en encontrar la clave.

CAPITULO III

3 ANÁLISIS DE RESULTADOS

En este capítulo se recopilo los resultados obtenidos al realizar las diferentes pruebas relacionadas con criptoanálisis analizadas en el capítulo II, estas serán aplicadas a los archivos .txt obtenidos al cifrar información con la utilización de las dos aplicaciones el AES normal y el MCIB-AES cuyos resultados serán analizados para una mejor comprensión.

3.1 COMPLEJIDAD DE LOS ALGORITMOS AES Y MECIB-AES

3.1.1 Complejidad del AES (Advanced Encryption Standard)

La complejidad además de la notación asintótica utilizada para realizar dicho análisis se explicó en el capítulo II; una vez implementado el AES (Advanced Encryption Standard) y siguiendo el algoritmo publicado por el National Institute of Standards and Technology (NITS), utilizaremos la notación O mayúscula para el análisis asintótico del algoritmo como se muestra en la **Figura 1-3.** (Dávila, 2017)

```
Cipher(){                                   
state = input;                              
KeyExpansion(k);                            O(KeyExpansion(k))
AddRoundKey (0);                            O(AddRoundKey(0))
for (round=1; round<Nr; round++) {          O(Nr (
SubBytes ();                                O(SubBytes())
ShiftRows ();                               O(ShiftRows())
MixColumns ();                              O(MixColumns())
AddRoundKey(round);                         O(AddRoundKey(round))
} ))                                        
SubBytes();                                 O(SubBytes())
ShiftRows();                                O(ShiftRows())
AddRoundKey(Nr);                            O(AddRoundKey(Nr))
Out = state;                                
}
```

Figura 1-3. Análisis Asintótico del Algoritmo AES

Fuente: (Dávila, 2017)

El AES (Advanced Encryption Standard) posee su complejidad computacional O ($n*k$), donde k obedece al tamaño de la clave, tabla SBox, operación XOR, operación Rcon, el ShiftRow y la tabla xTime de MixColumns y n el tamaño del texto a cifrar. (Dávila, 2017)

3.1.2 Complejidad del MECIB-AES

Dado que el algoritmo MECIB-AES está basado en el AES (Advanced Encryption Standard) la complejidad será la misma salvo en las funciones modificadas por lo cual se encontrará la complejidad de las funciones Mix-Shift, Move-C, Mix-Key.

Cipher(){ (Dávila, 2017)	
state = input;	
Mix-Key(k);	O(**Mix-Key(k)**)
AddRoundKey(0);	O(AddRoundKey(0))
for (round=1; round<Nr; round++) {	O(Nr (
SubBytes ();	O(SubBytes())
Mix-Shift ();	O(**Mix-Shift**())
Move-C ();	O(**Move-C**())
AddRoundKey(round);	O(AddRoundKey(round))
}))	
SubBytes();	O(SubBytes())
Mix-Shift ();	O(**Mix-Shift**())
ShiftRows();	O(ShiftRows())
AddRoundKey(Nr);	O(AddRoundKey(Nr))
Out = state;	
}	

Así obtenemos la función:

$$O(\text{Mix-Key}) + O(AddRoundkey) + O(Nr(O(SubBytes) + O(\text{Mix-Shift}) + O(Move\text{-C}) + O(AddRoundKey))) + O(\text{SubBytes}) + O(\text{Mix-Shift}) + O(\text{ShiftRows}) + O(\text{AddRoundKey}).$$

- ✓ Función Mix-Shift: esta función esta explicada paso a paso en el **Capítulo II** (**2.2.1 Función Mix-Shift**), por este motivo vamos directamente a el análisis asintótico.

```
int[] mod_shift = {8, 12, 4, 8};
int md = 0;
int var = 0;
for (i = 0; i < BLOCK_SIZE; i++) {          O(BLOCK_SIZE)
if ((i % COL_SIZE) == 0)                    O(COL_SIZE)
{                                           (
var = mod_shift[md];                        O(1)
md = md+1;                                  O(1)
}                                           )
k = (i+var)% BLOCK_SIZE;                    O(2)
tb[i] = ta[k];                              O(1)
}
```

$$O(\text{BLOCK\_SIZE}(O(\text{COL\_SIZE}(O(1) + O(1)) + (O(2) + O(1)))$$

Donde, BLOCK_SIZE, COL_SIZE son constantes y valores pequeños por lo cual se desprecian además serán representados por la letra k, obteniendo así:

$$O(\text{k}(O(\text{k})))$$

Al reducir quedara:

$$\text{O (k)}$$

- ✓ Función Move-C: esta función esta explicada paso a paso en el **Capítulo II** (**2.5.1 Función Move-C**), por este motivo vamos directamente a el análisis asintótico, pero según la hemos implementado la función en si no cambia lo que cambia es su posición dentro del desarrollo del algoritmo, por tanto, la complejidad será la misma que la MixColumns original.

Su resultado final es:

$O(k)$, donde k es un valor muy pequeño.

- ✓ Función Mix-Key: esta función esta explicada paso a paso en el **Capítulo II** (**2.5.1 Función Mix-Key**), por este motivo vamos directamente al análisis asintótico.

```
public void setKey(byte[] key) {
 // constantes
final int BC = BLOCK_SIZE / 4; //
final int Klen = key.length;   //tamaño de clave 16, 24, 32
final int Nk = KEY_SIZE / 4;       //número de palabras de 32 bits - 4,6,8 bytes
int i, j, r;
// copia la clave en el arreglo
for (i=0, j=0; i < Nk; i++) {                    O(Nk(
w0[i] = key[j++];                                O(1)
w1[i] = key[j++];                                O(1)
w2[i] = key[j++];                                O(1)
w3[i] = key[j++];                                O(1))
}
////////////////////////////////////////////////////////////
//          bloque modificado
byte[] v0 = new byte[Nk];                        O(1)
byte[] v1 = new byte[Nk];                        O(1)
byte[] v2 = new byte[Nk];                        O(1)
byte[] v3 = new byte[Nk];                        O(1)
int[] md = {1, 3, 0, 2, 5, 4, 7, 6};
for (i=0, j=0; i < Nk; i++) {                    O(Nk(
v0[md[j]] = w0[i];                               O(1)
v1[md[j]] = w1[i];                               O(1)
v2[md[j]] = w2[i];                               O(1)
v3[md[j]] = w3[i];                               O(1))
j++;}
for (i=0; i < Nk; i++) {                         O(Nk(
w0[i] = v0[i];                                   O(1)
```

```
w1[i] = v1[i];                                  O(1)
w2[i] = v2[i];                                  O(1)
w3[i] = v3[i];                                  O(1))
}
////////////////////////////////////////////////////////////
// algoritmo de expansión
byte t0, t1, t2, t3, old0;      // variables temporales
for (i = Nk; i < ROUND_KEY_COUNT; i++) {        O (ROUND_KEY_COUNT(
t0 = w0[i-1];                                   O(1)
t1 = w1[i-1];                                   O(1)
t2 = w2[i-1];                                   O(1)
t3 = w3[i-1];   // temporal = w[i-1]            O(1)
if (i % Nk == 0) {                                              O(Nk(
old0 = t0;        // intercambia variable                       O(1)
t0 = (byte)(S[t1 & 0xFF] ^ rcon[i/Nk]); // XOR a la constante   O(1)
t1 = (byte)(S[t2 & 0xFF]);                                      O(1)
t2 = (byte)(S[t3 & 0xFF]); // RotWord ordenamiento de los bytes O(1)
t3 = (byte)(S[old0 & 0xFF]);                                    O(1))
}
else if ((Nk > 6) && (i % Nk == 4)) {                           O(1(
// temporal = SubWord(temp)
t0 = S[t0 & 0xFF]; t1 = S[t1 & 0xFF]; t2 = S[t2 & 0xFF]; t3 = S[t3 & 0xFF];     O(1)
}                                                                               )
// w[i] = w[i-Nk] ^(XOR) temporal
w0[i] = (byte)(w0[i-Nk] ^ t0);                                  O(1)
w1[i] = (byte)(w1[i-Nk] ^ t1);                                  O(1)
w2[i] = (byte)(w2[i-Nk] ^ t2);                                  O(1)
w3[i] = (byte)(w3[i-Nk] ^ t3);                                  O(1)
}                                                               )
// copia al arreglo en/decrypt
for (r = 0, i = 0; r < numRounds + 1; r++) {    // ronda       O(numRounds +1(
for (j = 0; j < BC; j++) {        // palabra en ronda          O(BC(
Ke[r][4*j] = w0[i];                                           O(1)
Ke[r][4*j+1] =  w1[i];                                        O(1)
```

Ke[r][4*j+2] = w2[i];	O(1)
Ke[r][4*j+3] = w3[i];	O(1)
Kd[numRounds - r][4*j] = w0[i];	O(1)
Kd[numRounds - r][4*j+1] = w1[i];	O(1)
Kd[numRounds - r][4*j+2] = w2[i];	O(1)
Kd[numRounds - r][4*j+3] = w3[i];	O(1)
i++;	
}	)
}	)

Ecuación 1-3: Análisis asintótico del MECIB-AES

$$O(Nk) + O(4) + O(Nk) + O(Nk) + O\left(\left(\left(\frac{O(5)+O(4)+ROUND\_KEY\_COUNT}{Nk}\right) + O\left(\frac{2}{Nk}\right) + O(4)\right) + O\left(numRounds\left(O\left(BC(O(4) + O(4))\right)\right)\right)\right)$$

$$3 * O(k) + O(4) + O\left(\left(\left(\frac{O(5+4)+ROUND\_KEY\_COUNT}{k}\right) + O\left(\frac{2}{k}\right) + O(4)\right) + O\left(numRounds\left(O(BC(O(4 + 4)))\right)\right)\right)$$

$$3 * O(k) + O\left(\left(\left(\frac{ROUND\_KEY\_COUNT}{k}\right) + O\left(\frac{2}{k}\right)\right) + O(numRounds)\right)$$

$$O\left(\left(\left(\frac{ROUND\_KEY\_COUNT}{k}\right)\right) + O(numRounds)\right)$$

Donde ROUND_KEY_COUNT y numRounds son valores constantes y pequeños por lo tanto se desprecian quedando así:

$$O\left((k) + O(k)\right) = O(k)$$

Una vez realizados los cálculos necesarios para encontrar la complejidad de las modificaciones realizadas e implementadas al algoritmo MECIB-AES podemos reducir la función principal:

O(*Mix-Key*) + O($AddRoundkey$) + O(Nr(O($SubBytes$) + O(*Mix-Shift*) + O($Move\text{-}C$) + O($AddRoundKey$))) + O(*SubBytes*) + O(*Mix-Shift*) + O(*ShiftRows*) + O(*AddRoundKey*).

O(k) + O(k) + O(Nr(O(k) + O(k) + O(k) + O(k))) + O(k) + O(k) + O(k) + O(k) = O(k)

El resultado es $O(n*k)$, el análisis asintótico correspondiente a cada algoritmo es igual, por lo tanto, se comprobó los recursos utilizados por el algoritmo AES y MECIB-AES.

Tabla 1-3. Uso de memoria por parte del algoritmo AES y MECIB-AES.

Archivo	Tamaño AES (Bytes)	Tamaño MECIB-AES (Bytes)	[AES] - [MECIB-AES]
1	722	719	3
2	410	407	3
3	754	744	10
4	1960	1937	23
5	648	630	18
6	576	578	-2
7	292	293	-1
8	345	357	-12
9	269	262	7
10	527	525	2
11	402	398	4
12	745	743	2
13	474	474	0
14	218	206	12
15	312	323	-11
16	332	332	0
17	210	219	-9
18	298	298	0
19	460	477	-17
20	204	161	43
21	466	450	16
22	147	144	3
23	377	374	3
24	119	126	-7
25	217	220	-3
26	372	392	-20

27	183	185	-2
28	316	304	12
29	484	482	2
30	282	298	-16
31	996	1024	-28
32	422	414	8
33	146	142	4
34	442	435	7
35	645	662	-17
36	599	587	12
37	515	484	31
38	626	633	-7
39	2020	2005	15
40	2507	2539	-32
41	870	882	-12
42	278	291	-13

Realizado por: Raza C., 2019

Cuya media del algoritmo AES es: 552.07 y del MECIB-AES es: 551.33.

Ecuación 2-3: Regla de 3 simple

$$X= (a*b) / c$$

X= (551.33*100) / 552.07

X= 99.87 %

Considerándose así que el uso de memoria es igual para los dos algoritmos en un porcentaje del 99.18% y diferente en 0,13%.

Para comprobar el uso del tiempo utilizaremos la complejidad encontrada ponderando valores.

Ecuación 3-3. Complejidad del Algoritmo AES y MECIB-AES

$$O\ (n*k)$$

Valores (128; 3,7, 9, 13, 15, 19)

AES	MECIB-AES
O (128*3) = 384	O (128*3) = 384
O (128*7) = 896	O (128*7) = 896
O (128*9) = 1152	O (128*9) = 1152

O (128*13) = 1664	O (128*13) = 1664
O (128*15) = 1920	O (128*15) = 1920
O (128*19) = 2432	O (128*19) = 2432

Reemplazando en la fórmula de la complejidad da como resultado obteniendo así un valor igual para los dos algoritmos.

Con estos cálculos realizados se realiza la comprobación de las igualdades en las complejidades del algoritmo siendo la implementación viable para ser utilizada.

3.2 MEDICIÓN DE ENTROPÍA

La investigación necesaria para la medición y desarrollo de esta prueba se la puede encontrar en el **Capítulo II** (**2.5.1 Entropía**) y también se utiliza el software CrypTool. Una vez cifrado todos los archivos con los dos algoritmos diferentes se procederá a medir la entropía en cada uno de ellos, por los tanto, definimos el alfabeto a utilizar como se ve en la **Figura 2-3**.

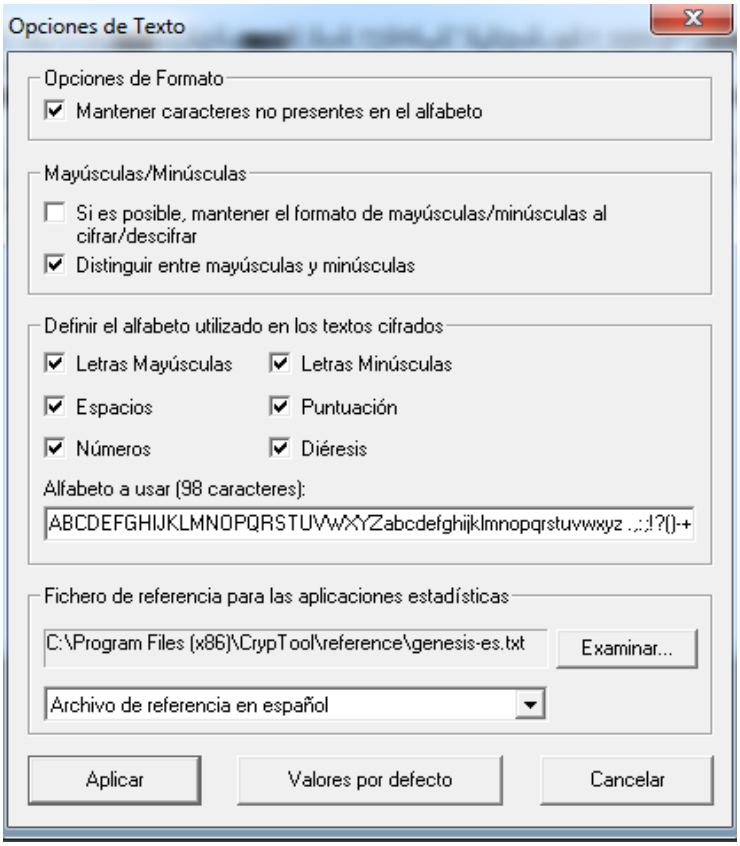


Figura 2-3. Definición del alfabeto de CrypTool.

Realizado por: Raza C., 2019

Con los datos encontrados realizamos la **Figura 3-3** que nos permite realizar el análisis estadístico de diferencias de medias.

Cifrado AES Nombre del Archivo	Cifrado MECIB-AES Nombre del Archivo	ENTROPIA	
		AES (Advanced Encryption Standard) (bits/carácter)	MECIB-AES (Modificación del AES) (bits/carácter)
Prueba_1	Prueba_m_1	6,08	6,07
Prueba_2	Prueba_m_2	5,65	5,79
Prueba_3	Prueba_m_3	5,99	6,11
Prueba_4	Prueba_m_4	6,38	6,32
Prueba_5	Prueba_m_5	6,02	6,10
Prueba_6	Prueba_m_6	6,04	5,98
Prueba_7	Prueba_m_7	5,44	5,45
Prueba_8	Prueba_m_8	5,46	5,30
Prueba_9	Prueba_m_9	5,33	5,27
Prueba_10	Prueba_m_10	5,92	6,02
Prueba_11	Prueba_m_11	5,95	5,94
Prueba_12	Prueba_m_12	6,11	6,15
Prueba_13	Prueba_m_13	5,75	5,83
Prueba_14	Prueba_m_14	5,20	5,39
Prueba_15	Prueba_m_15	5,42	5,43
Prueba_16	Prueba_m_16	5,55	5,54
Prueba_17	Prueba_m_17	5,17	5,44
Prueba_18	Prueba_m_18	5,41	5,21
Prueba_19	Prueba_m_19	6,04	5,78
Prueba_20	Prueba_m_20	5,16	5,22
Prueba_21	Prueba_m_21	5,75	5,67
Prueba_22	Prueba_m_22	4,75	4,73
Prueba_23	Prueba_m_23	5,54	5,32
Prueba_24	Prueba_m_24	4.62	4,70
Prueba_25	Prueba_m_25	5,09	5,16
Prueba_26	Prueba_m_26	5,78	5,83
Prueba_27	Prueba_m_27	5,12	5,28
Prueba_28	Prueba_m_28	5,28	5,40
Prueba_29	Prueba_m_29	5,69	5,87
Prueba_30	Prueba_m_30	5,47	5,45
Prueba_31	Prueba_m_31	6,27	6,17
Prueba_32	Prueba_m_32	5,72	5,61
Prueba_33	Prueba_m_33	4,59	4,46
Prueba_34	Prueba_m_34	5,73	5,71
Prueba_35	Prueba_m_35	6,07	6,01
Prueba_36	Prueba_m_36	5,84	5,91
Prueba_37	Prueba_m_37	5,83	5,90
Prueba_38	Prueba_m_38	6,04	5,95
Prueba_39	Prueba_m_39	6,35	6,36
Prueba_40	Prueba_m_40	6,39	6,42
Prueba_41	Prueba_m_41	6,23	5,97
Prueba_42	Prueba_m_42	5,49	5,46

Figura 3-3 Medición de Entropía al AES y MECIB-AES.

Realizado por: Raza C., 2019

Para comprobar que las mediciones realizadas y su discrepancia no son casualidades utilizaremos una distribución normal de diferencia de medias para saber si los datos obtenidos en la medición de la entropía son correctos se utiliza la **Ecuación 1-2.**

$$Z = \frac{(\bar{x} - \bar{y}) - (\mu_x - \mu_y)}{\sqrt{\frac{\sigma_x^2}{nx} \frac{+\sigma_y^2}{n_y}}}$$

$\bar{x} = 5{,}65976$ $\sigma_x^2 = 0{,}45858$

$\bar{y} = 5{,}65905$ $\sigma_y^2 = 0{,}44824$

$$Z = \frac{(5{,}65976 - 5{,}65905) - (50 - 50)}{\sqrt{\frac{0{,}45858_x^2}{100} \frac{+0{,}44824_y^2}{100}}}$$

$$Z = 0{,}01107$$

Utilizando un nivel de confianza del 95%, error del 5% y un nivel de significación del 5% en la tabla de la distribución normal nos da un valor de $Z_{(0,05)} = 1{,}645$ y nuestro $Z = 0{,}01107$.

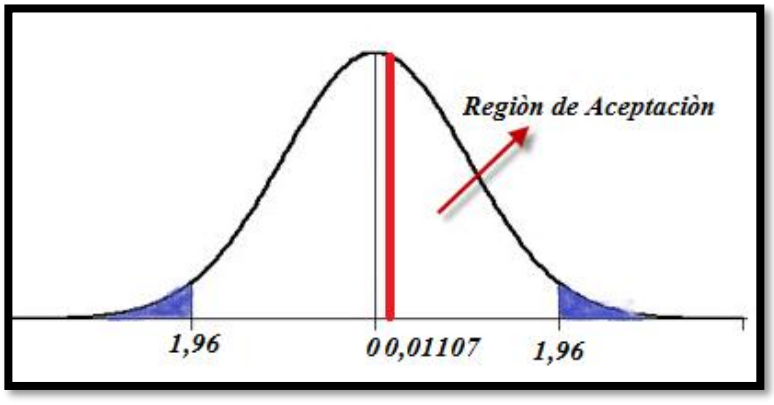


Figura 4-3. Z calculado versus Z de la tabla de distribución normal.

Realizado por: Raza C., 2019

Dando como resultado que la entropía del MECIB-AES es mayor a la del algoritmo AES.

3.3 ANÁLISIS DE FRECUENCIAS

En el análisis de frecuencias dada la diversidad de pruebas que se pueden realizar se optó por: Histograma, Diagrama, Trigrama y 4-Grama, los mismos que se encuentran dentro del análisis de N-gramas realizado en el programa CrypTool para profundizar en el tema se aconseja revisar el **Capítulo II** (**2.5.2 Análisis de Frecuencias**).

Concluida la medición se realizó una tabla donde se observan la cantidad de caracteres que forman parte de los distintos análisis; se utilizó el alfabeto de la **Figura 2-3.,** cabe recalcar que en el análisis del histograma se tomó como resultado final la cantidad de caracteres que mayor frecuencia poseían en el texto cifrado tal y como se observa en la **Figura 5-3**.

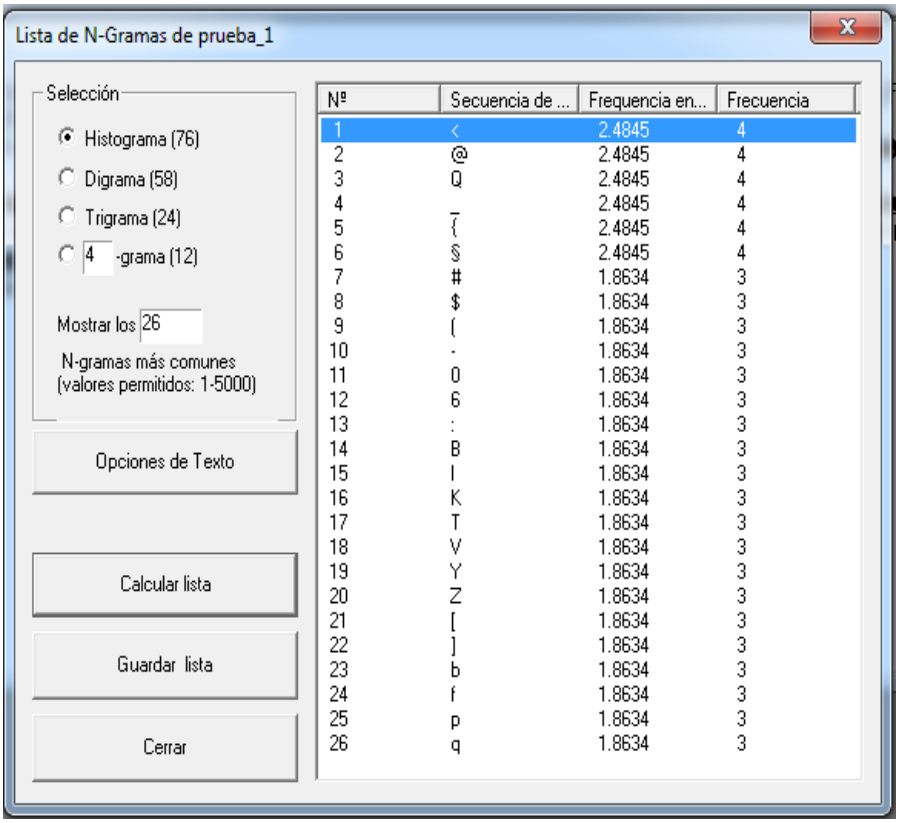


Figura 5-3. Lista de N-Gramas de prueba_1.

Realizado por: Raza C., 2019

Obteniendo así la **Figura 6-3**.

FRECUENCIAS DEL ALGORITMO AES					FRECUENCIAS DEL ALGORITMO MECIB-AES				
AR.	**HIST.**	**DIAG.**	**TRIG.**	**4-G.**	**AR.**	**HIST.**	**DIAG.**	**TRIG.**	**4-G.**
1	76	58	24	12	1	80	64	31	17
2	56	37	16	9	2	62	42	15	6
3	76	49	19	7	3	79	70	29	11
4	91	155	54	18	4	90	173	65	22
5	74	43	15	5	5	79	73	32	16
6	75	52	19	8	6	73	53	17	3
7	50	26	12	4	7	48	17	4	1
8	49	24	11	5	8	43	15	3	0
9	44	24	9	4	9	43	18	7	2
10	67	36	7	1	10	75	47	15	4
11	68	39	9	2	11	68	39	16	7
12	78	55	23	10	12	81	58	15	4
13	62	37	9	2	13	66	38	10	1
14	40	13	3	2	14	45	19	7	1
15	48	25	13	6	15	48	24	8	1
16	52	26	10	0	16	53	32	8	3
17	40	21	8	2	17	46	23	12	6
18	46	17	3	0	18	41	17	3	0
19	75	51	23	12	19	64	41	15	5
20	37	12	3	1	20	40	26	14	8
21	61	39	13	4	21	59	39	14	1
22	28	10	3	1	22	28	14	4	1
23	51	21	11	4	23	45	18	3	0
24	26	9	2	0	24	27	7	3	1
25	37	15	5	1	25	38	21	10	5
26	62	43	21	10	26	63	30	8	3
27	39	22	11	6	27	42	20	6	3
28	45	20	8	3	28	47	21	6	1
29	60	42	17	8	29	65	30	9	1
30	51	28	8	0	30	47	17	6	3
31	85	96	35	14	31	85	77	27	10
32	58	28	10	1	32	57	31	13	6
33	25	5	1	0	33	25	12	7	5
34	59	30	10	4	34	61	40	22	10
35	77	58	22	7	35	70	55	24	8
36	66	35	6	1	36	70	55	24	8
37	64	31	8	2	37	68	52	23	8
38	77	58	24	9	38	72	49	15	5
39	92	149	36	7	39	92	161	59	23
40	92	216	89	32	40	92	211	75	27
41	86	69	23	8	41	75	50	15	1
42	50	34	13	5	42	48	25	6	1

Figura 6-3. Análisis de frecuencias en AES y MECIB-AES.

Realizado por: Raza C., 2019

Para el análisis de los datos obtenidos se realizó gráficos de barras con los cuales compararemos los diferentes resultados.

✓ Histograma

Histograma del AES, **Figura 7-3**.

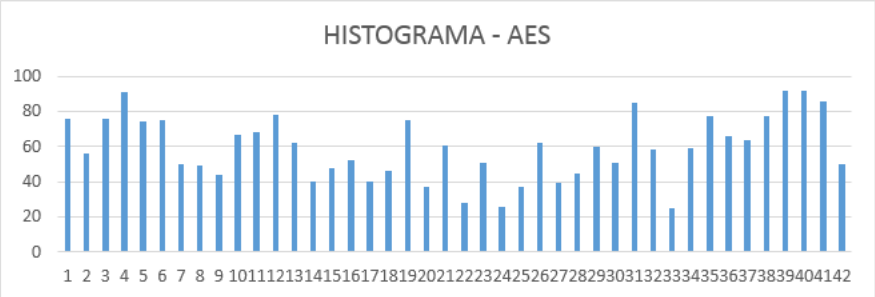


Figura 7-3. Histograma del algoritmo AES.

Realizado por: Raza C., 2019

Histograma del MECIB-AES, **Figura 8-3**.

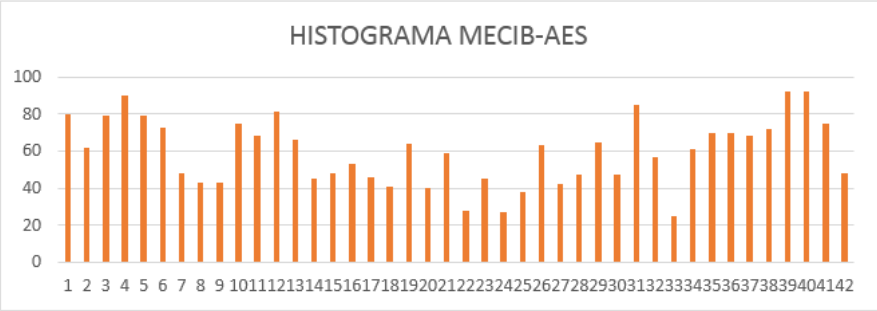


Figura 8-3. Histograma del algoritmo MECIB-AES.

Realizado por: Raza C., 2019

Histograma AES vs MECIB-AES, **Figura 9-3**.

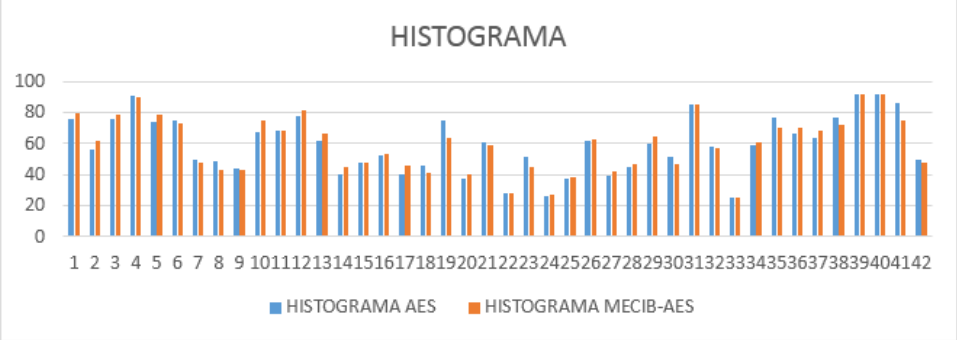


Figura 9-3. Histograma del algoritmo AES vs MECIB-AES.

Realizado por: Raza C., 2019

Como resultado final podemos observar los dos histogramas los mismos que presentan variaciones no significativas, por lo tanto, el análisis realizado por medio del histograma no presenta valores máximos o mínimos alternadamente que ayuden a realizar un ataque, debido a que menos caracteres y su frecuencia ayudaran a tener una idea clara para descifrar el mensaje, se puede comprobar que el tamaño de los caracteres no posee relación directa entre archivos, obteniendo así una distribución normal, por ejemplo, tomando en consideración los 8 primeros valores obtenidos se observa una relación con una distribución teórica normal debido a la forma acampanada de los datos.

✓ Diagrama

Diagrama del AES, **Figura 10-3**.

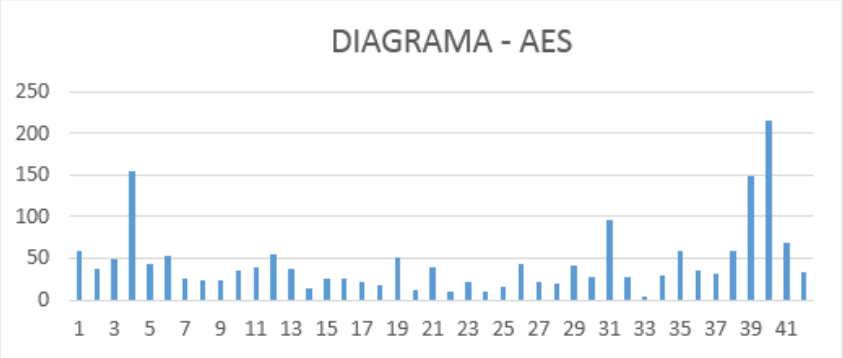


Figura 10-3. Diagrama del algoritmo AES

Realizado por: Raza C., 2019

Diagrama del MECIB-AES, **Figura 11-3**.

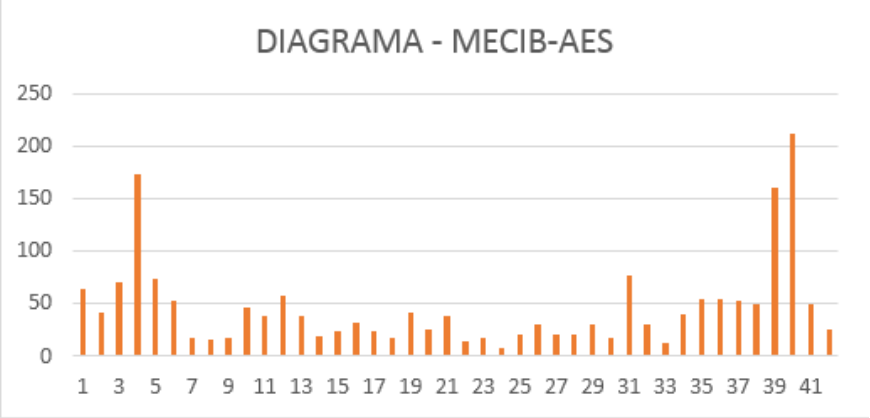


Figura 11-3. Diagrama del algoritmo MECIB-AES

Realizado por: Raza C., 2019

Diagrama del AES vs MECIB-AES, **Figura 12-3**.

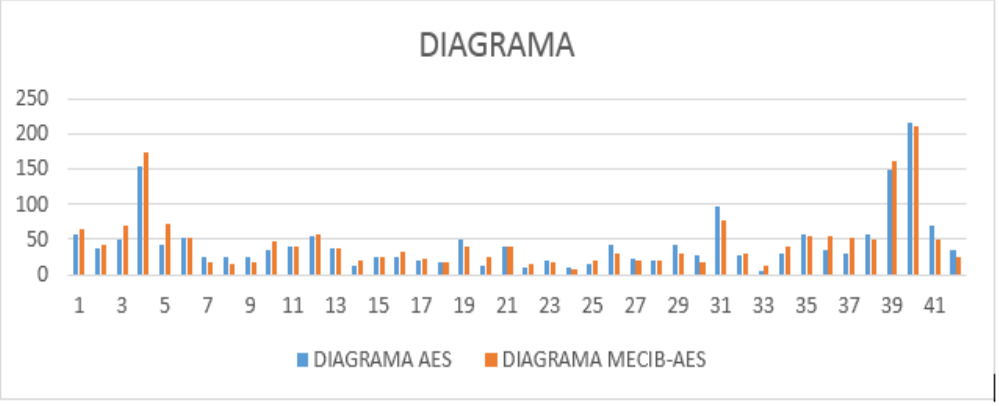


Figura 12-3. Diagrama del algoritmo AES vs MECIB-AES

Realizado por: Raza C., 2019

El análisis de los distintos diagramas presentes en cada archivo cifrado nos da como resultado que poseen en su mayoría menos de 50 diagramas, además encontramos valores muy altos en comparación a otros archivos lo cual puede deberse a el tamaño del archivo cifrado, pero al comparar los dos algoritmos la relación de diferencia entre el MECIB-AES y AES no es grande, pero se debe tomar en cuenta que más del 50 % de los archivos poseen menos o igual diagramas y estos pertenecen al MECIB-AES.

✓ Trigrama

Trigrama del AES, **Figura 13-3**.

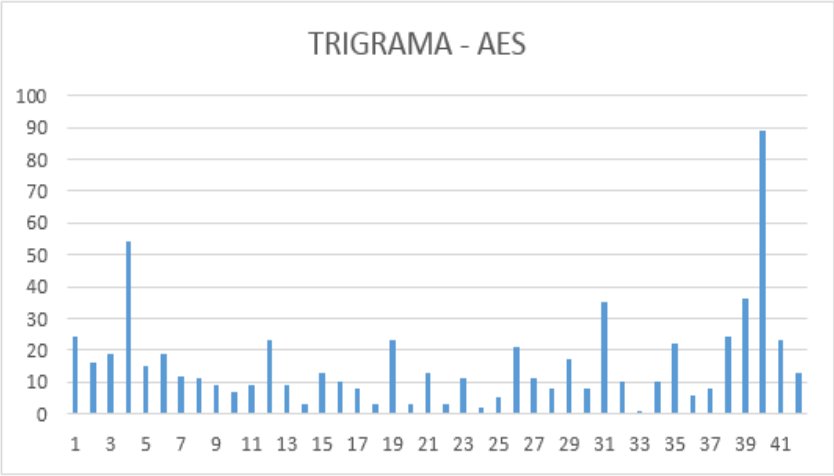


Figura 13-3. Trigrama del algoritmo AES

Realizado por: Raza C., 2019

Trigrama del MECIB-AES, **Figura 14-3**.

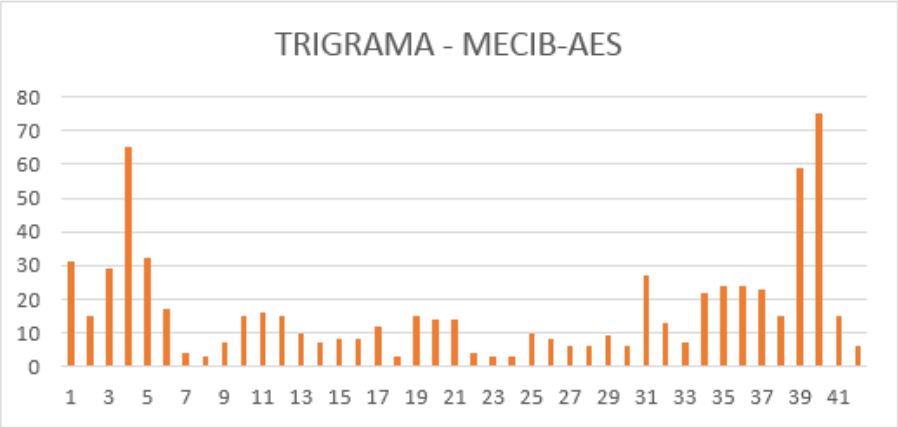


Figura 14-3. Trigrama del algoritmo MECIB-AES

Realizado por: Raza C., 2019

Trigrama del AES vs MECIB-AES, **Figura 15-3**.

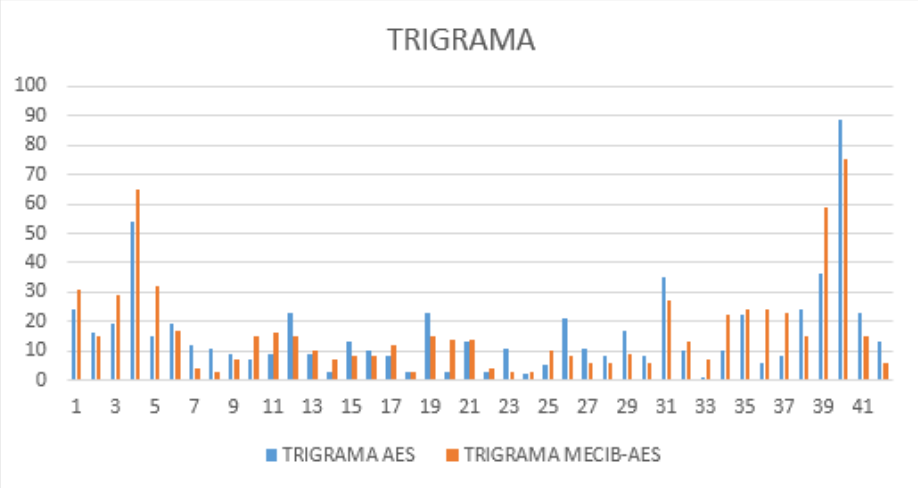


Figura 15-3. Trigrama del algoritmo AES vs MECIB-AES

Realizado por: Raza C., 2019

La cantidad de trigramas presentes en los archivos posee valores aleatorios los cuales se encuentran concentrados mayormente por debajo de los 20 aunque si existe picos altos lo cual nos permite tomar en consideración el tamaño del texto cifrado, la diferencia de valores en los algoritmos es notoria a favor del MECIB-AES lo cual no quiere decir que pueda ser descifrado el mensaje, pero podemos tomar como antecedente durante futuros análisis.

✓ 4-Grama

4-Grama del AES, **Figura 16-3**.

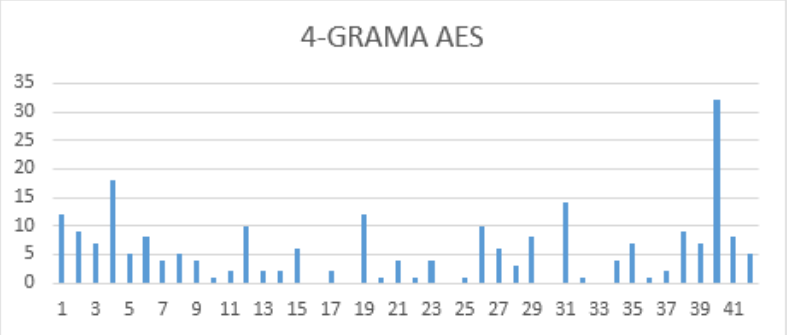


Figura 16-3. 4-Grama del algoritmo AES

Realizado por: Raza C., 2019

4-Grama del MECIB-AES, **Figura 17-3**.

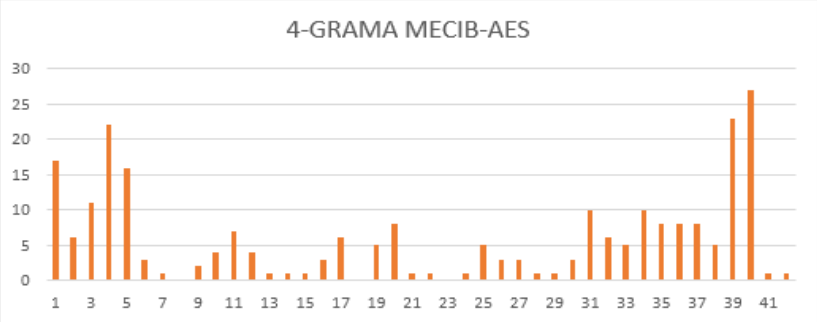


Figura 17-3. 4-Grama del algoritmo MECIB-AES

Realizado por: Raza C., 2019

4-Grama del AES vs MECIB-AES, **Figura 18-3**.

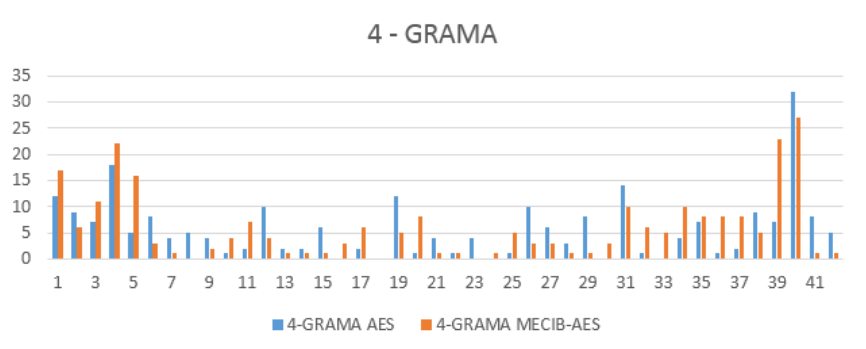


Figura 18-3. 4-Grama del algoritmo AES vs MECIB-AES

Realizado por: Raza C., 2019

Los 4-Gramas están por debajo de los 5 por párrafo en la mayor parte de los archivos, pero la relación de diferencias entre los algoritmos no es muy grande para considerar uno de los algoritmos como superior.

3.4 MEDICIÓN DE AUTOCORRELACIÓN

Tomando en cuenta la explicación en el **Capítulo II (2.5.3 Autocorrelación)**, y basándonos en la similitud de secciones se registró los valores máximos como se puede observar en la **Figura 19-3**

ARCHIVO	CONCORDANCIA MÁXIMA DE SECCIONES EN EL AES (u)	CONCORDANCIA MÁXIMA DE SECCIONES EN EL MECIB-AES (u)
1	4	6
2	3	4
3	5	5
4	10	10
5	6	5
6	4	4
7	3	3
8	3	2
9	2	2
10	3	5
11	3	3
12	5	6
13	4	4
14	2	3
15	3	2
16	3	4
17	2	1
18	2	5
19	4	4
20	1	2
21	4	4
22	1	1
23	2	5
24	3	1
25	1	3
26	3	2
27	2	2
28	3	3
29	4	3
30	3	2
31	6	8
32	3	4
33	1	4
34	4	4
35	4	5
36	4	4
37	3	4
38	5	4
39	11	10
40	14	12
41	6	7
42	3	3

Figura 19-3. Autocorrelación de AES y MECIB-AES

Realizado por: Raza C., 2019

Autocorrelación del AES, **Figura 20-3**.

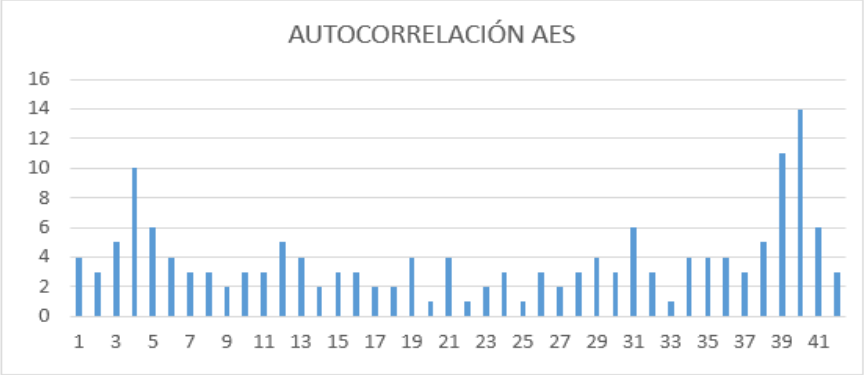


Figura 20-3. Autocorrelación del algoritmo AES

Realizado por: Raza C., 2019

Autocorrelación del MECIB-AES, **Figura 21-3**.

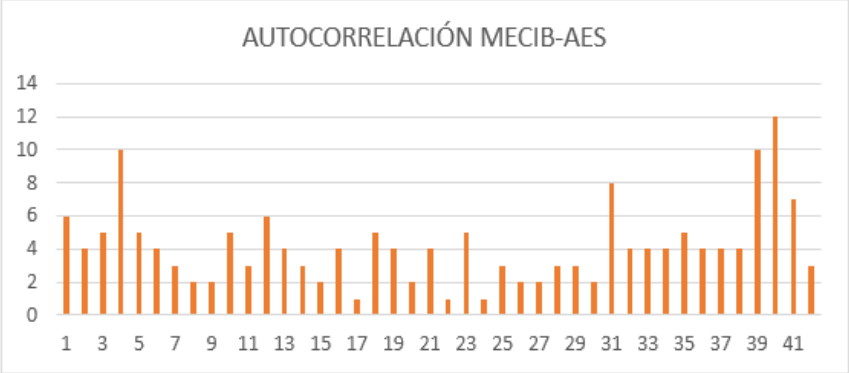


Figura 21-3. Autocorrelación del algoritmo MECIB-AES

Realizado por: Raza C., 2019

Autocorrelación del AES vs MECIB-AES, **Figura 22-3**.

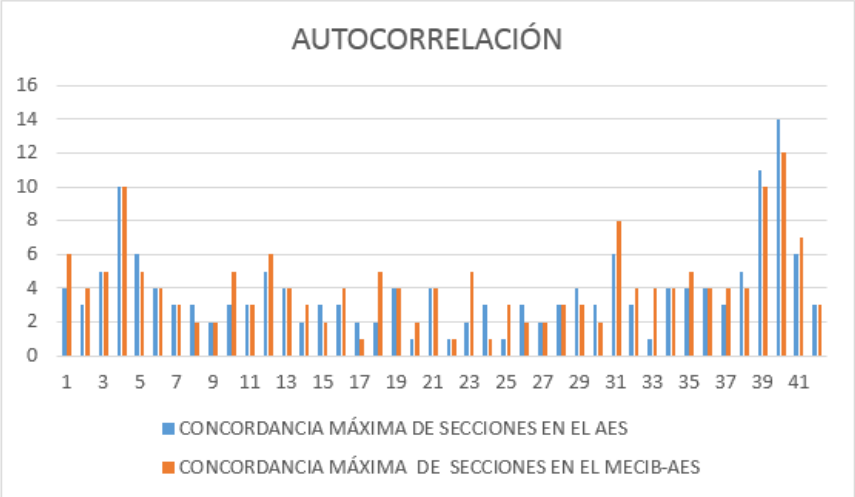


Figura 22-3. Autocorrelación del algoritmo AES vs MECIB-AES

Realizado por: Raza C., 2019

La concordancia de secciones es variable, pero en la mayor parte de los archivos se mantiene bajo el valor de 6 aunque también se encuentran valores muy pequeños y picos grandes, se puede observar que la diferencia entre los algoritmos es notoria para el MECIB-AES poseyendo mayor similitud.

3.5 MEDICIÓN DEL ANÁLISIS DE FUERZA BRUTA

La medición realizada a los archivos cifrados con la utilización del programa CrypTool se observa en la **Figura 24-3,** la cual nos permite comparar el tiempo en años que sería necesario para romper la seguridad que brindan los dos algoritmos utilizados, si se desea profundizar sobre cómo actúa y funciona un taque de fuerza bruta revise el **CAPITULO II (2.5.4 Fuerza Bruta),** el diccionario que se utilizó se observa en la **Figura 23-3**.

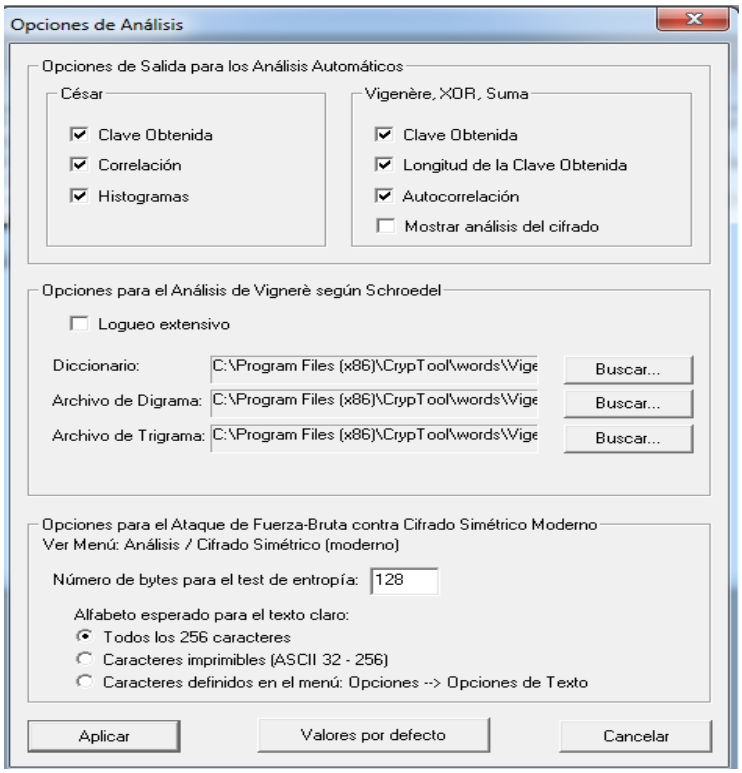


Figura 23-3. Definición del alfabeto para análisis de fuerza bruta.

Realizado por: Raza C., 2019

FUERZA BRUTA		
ARCHIVO	**AES (años)**	**MECIB-AES (años)**
1	3,40E+25	3,60E+25
2	3,80E+25	4,00E+25
3	3,50E+25	4,10E+25
4	4,90E+25	5,00E+25
5	3,50E+25	4,90E+25
6	4,50E+25	4,40E+25
7	3,70E+25	3,70E+25
8	3,90E+25	3,50E+25
9	3,40E+25	3,50E+25
10	3,90E+25	3,50E+25
11	3,60E+25	3,80E+25
12	4,30E+25	4,10E+25
13	3,70E+25	4,00E+25
14	4,40E+25	4,40E+25
15	4,10E+25	4,20E+25
16	5,20E+25	5,50E+25
17	3,80E+25	4,00E+25
18	3,60E+25	4,00E+25
19	3,50E+25	3,70E+25
20	4,60E+25	4,40E+25
21	3,60E+25	4,10E+25
22	4,70E+25	4,70E+25
23	3,90E+25	4,20E+25
24	4,20E+25	4,20E+25
25	4,20E+25	4,20E+25
26	3,50E+25	3,60E+25
27	4,70E+25	4,70E+25
28	4,50E+25	4,60E+25
29	4,60E+25	4,60E+25
30	3,70E+25	4,40E+25
31	4,10E+25	3,80E+25
32	4,00E+25	4,10E+25
33	4,30E+25	4,20E+25
34	4,60E+25	4,20E+25
35	4,10E+25	4,10E+25
36	4,30E+25	4,90E+25
37	4,90E+25	4,50E+25
38	3,70E+25	4,50E+25
39	5,40E+25	5,30E+25
40	4,90E+25	4,40E+25
41	4,40E+25	4,30E+25
42	4,10E+25	4,20E+25

Figura 24-3. Fuerza Bruta de AES y MECIB-AES.

Realizado por: Raza C., 2019

Fuerza Bruta del AES, **Figura 25-3**.

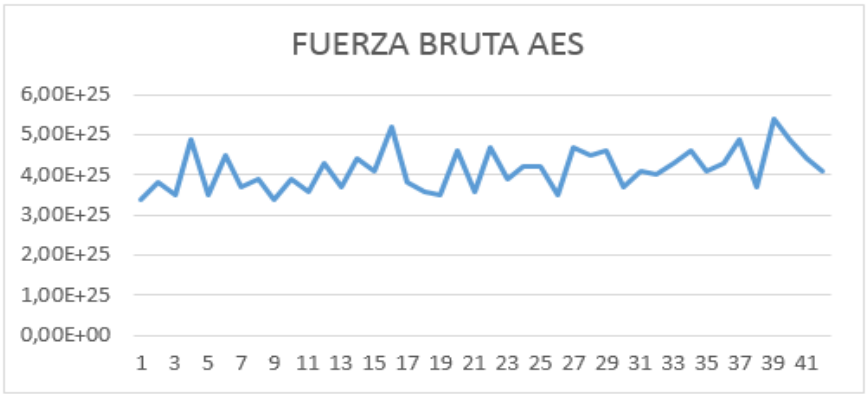


Figura 25-3. Fuerza Bruta del algoritmo AES

Realizado por: Raza C., 2019

Fuerza Bruta del MECIB-AES, **Figura 26-3**.

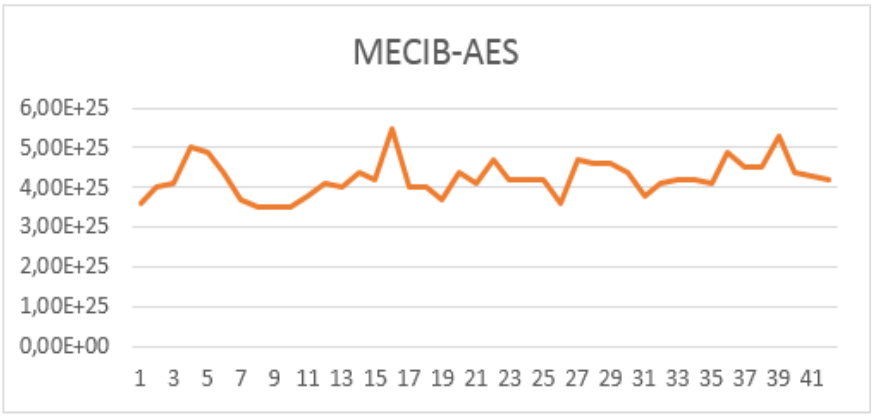


Figura 26-3. Fuerza Bruta del algoritmo MECIB-AES

Realizado por: Raza C., 2019

Fuerza Bruta del AES vs MECIB-AES, **Figura 27-3**.

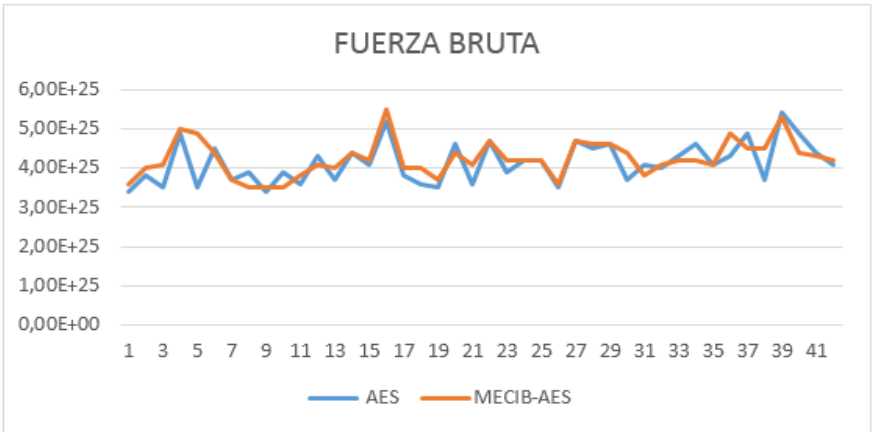


Figura 27-3. Fuerza Bruta del algoritmo AES vs MECIB-AES

Realizado por: Raza C., 2019

El algoritmo AES tarda un tiempo exuberante para poder ser descifrado, por lo tanto, el MECIB-AES debe ser igual o superar ese tiempo y así lograr ser considerado un algoritmo viable; el análisis de la fuerza bruta, donde se toma en consideración el tiempo, así como las posibles combinaciones de contraseñas, dio como resultado según sus medias Xa= 4,14E+25 y Xm= 4,24E+25 respectivamente, un aumento del 2,47% equivalente a 1,05E+24 años a favor del MECIB-AES.

3.6 ANÁLISIS DE TRÁFICO

Se realizó un enlace punto a punto, mediante el cual se envió información cifrada con las dos aplicaciones (AES y MECIB_AES) y con la ayuda del software Wireshark se capturo el tráfico y se procedió a su análisis.

Escenario: Dos computadoras conectadas por medio de un cable UTP con tasa de transferencia de 100 Mbps.

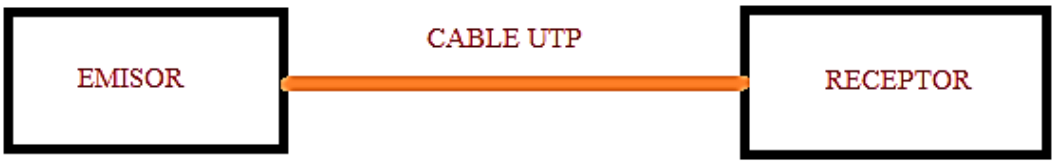


Figura 28-3. Escenario para análisis de Tráfico

Realizado por: Raza, C. 2019

Los archivos de igual texto fueron cifrados con diferente algoritmo como se muestra en la **Tabla 11-2**; se enviaron 42 archivos por medio del enlace punto a punto obteniendo así la tasa de transferencia y latencia para ser comparada como se observa a continuación.

Display

Display filter:	smb2
Ignored packets:	0 (0.000%)

Traffic	Captured	Displayed	Displayed %	Marked	Marked %
Packets	681	646	94.860%	0	0.000%
Between first and last packet	6.966 sec	5.099 sec			
Avg. packets/sec	97.761	126.697			
Avg. packet size	266 bytes	261 bytes			
Bytes	181412	168798	93.047%	0	0.000%
Avg. bytes/sec	26042.605	33105.671			
Avg. MBit/sec	0.208	0.265			

Figura 29-3. Resumen de tráfico capturado usando el algoritmo AES

Realizado por: Raza, C. 2019

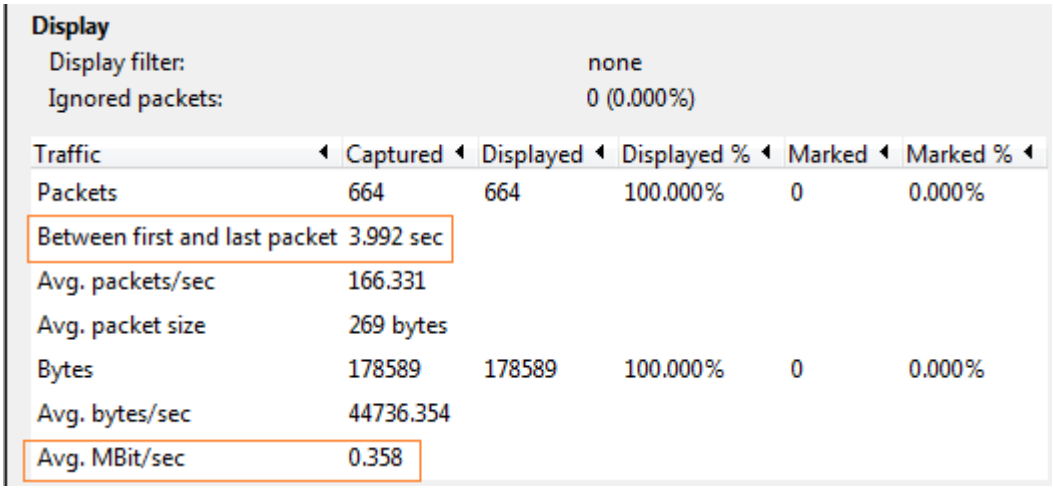


Figura 30-3. Resumen de tráfico capturado usando el algoritmo MECIB-AES

Realizado por: Raza, C. 2019

Dando así:

Tasa de transferencia (AES): 6,966

Tasa de transferencia (MECIB-AES): 3,992

Latencia (AES): 0,208

Latencia (MECIB-AES): 0,358

Utilizando la **Ecuación 2-3:**

X= (a*b) / c X= (a*b) / c

Xl= 57.31% Xts= 58.1%

Por lo tanto, notamos un aumento en la tasa de transferencia y una disminución de la latencia en más del 50% al enviar 42 archivos conjuntamente, siendo esta la razón por la cual se ha capturado el tráfico del envío de un solo archivo para corroborar la información obtenida como se encuentra en la **Tabla 2-3**.

Tabla 2-3. Características de los archivos Prueba_3 y Prueba_m_3.

Nombre	**Cifrado**	**Tamaño**
Prueba_3	AES	754 bytes
Prueba_m_3	MECIB-AES	744 bytes

Realizado por: Raza, C. 2019

Al capturar el tráfico se observó la presencia del protocolo SMB2, dado que es el protocolo utilizado para la transferencia de archivos entre Windows como se muestra en la **Figura 31-3** y **Figura 32-3.**

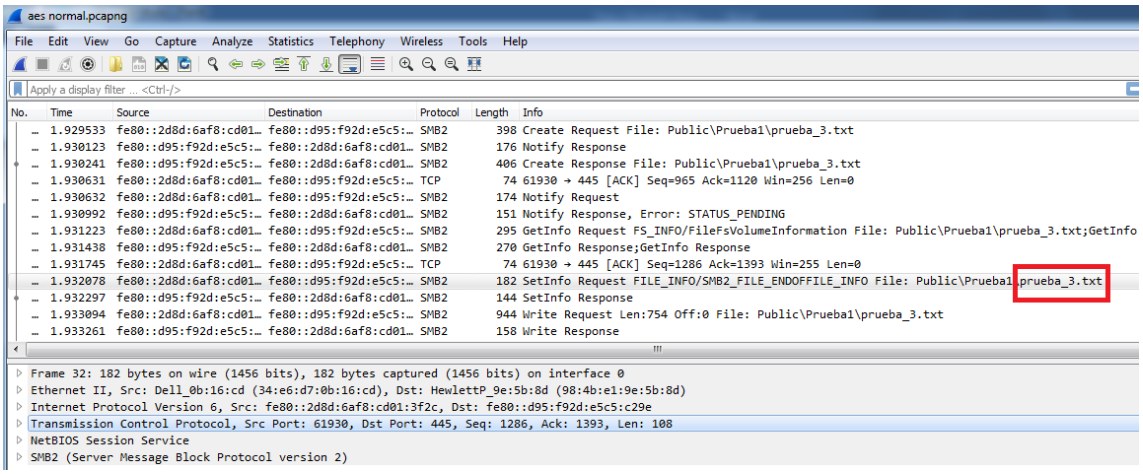


Figura 31-3. Tráfico Capturado de Prueba_3

Realizado por: Raza, C. 2019

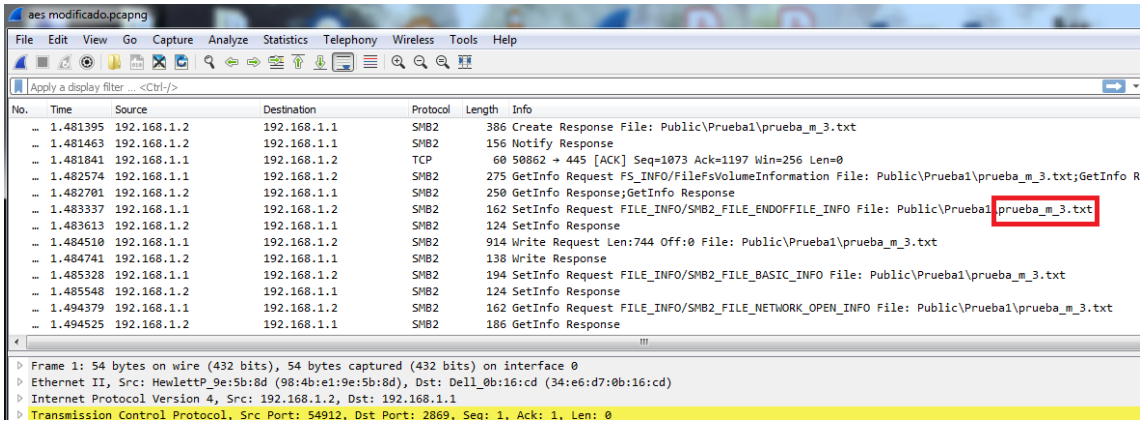


Figura 32-3. Tráfico Capturado de Prueba_m_3

Realizado por: Raza, C. 2019

Se procedió a realizar un resumen de todo el tráfico capturado obteniendo así lo visualizado en la **Figura 33-3** y **Figura 34-3**.

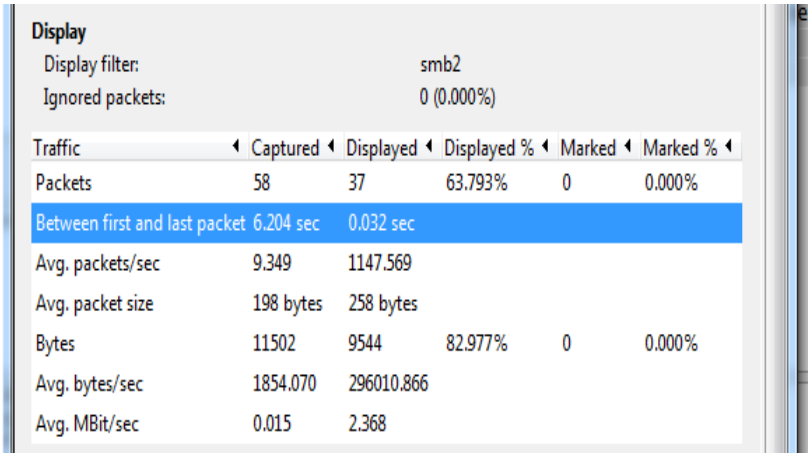


Figura 33-3. Resumen de Tráfico de Prueba_3

Realizado por: Raza, C. 2019

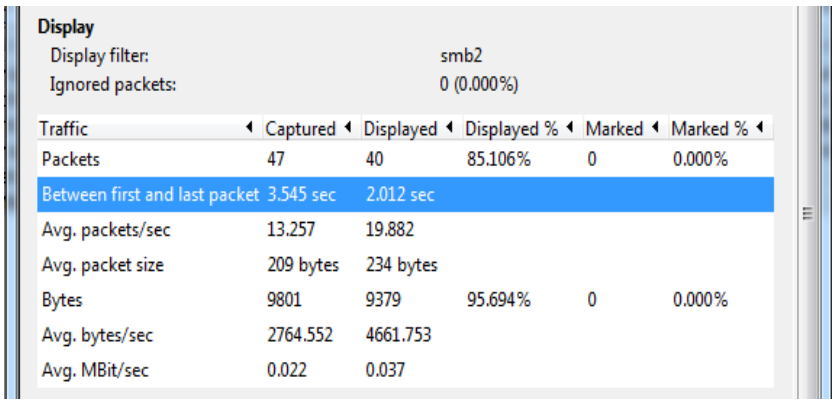


Figura 34-3. Resumen de Tráfico de Prueba_m_3

Realizado por: Raza, C. 2019

Consecuentemente a lo observado se realizó la **Tabla 3-3**, donde se constata claramente la tasa de transferencia y latencia.

Tabla 3-3. Tasa de Transferencia y Latencia.

Archivo	Tamaño	Latencia	Tasa de Transferencia	Promedio Paquetes/sec.
Prueba_3	754 bytes	6,204 sec.	0,015 Mbit/sec.	9,349
Prueba_m_3	744 bytes	3,545 sec.	0,022 Mbit/sec.	13,257

Realizado por: Raza, C. 2019

Utilizando la **Ecuación 2-3,** tenemos los siguientes resultados:

$$X = (a*b) / c$$

$X = (3,545*100) / 6,204$

$X = 43,19\%$

$X = (0,015*100) / 0,022$

$X = 31,82\%$

Haciendo uso de la ecuación 2-3, se nota una disminución del 43,19 % en Latencia y un aumento del 31,82 % Tasa de Transferencia. Como se pudo constatar el aumento en la tasa de transferencia y la disminución de latencia corroboran los datos anteriores.

DISCUSIÓN DE RESULTADOS

Tomando como base al algoritmo Advanced Encryption Standard o AES por sus siglas, las modificaciones realizadas a este algoritmo nos ayudan a medir la seguridad que brindan a la información cifrada. Dado el análisis realizado a todo el algoritmo se filtraron diferentes modificaciones como se explica a continuación en la **Tabla 3-3**:

Tabla 3-3. Posibles Modificaciones al algoritmo AES

Función	Modificación	Resultado	Valor
AddRoundKey	Cambiar posiciones de las filas	Cambio de variables al descifrar.	0
SubBytes	Utilizar una tabla S-Box diferente	Dificultad al encontrar la S-Box inversa.	0
ShiftRows	Utilización de un vector y modificación de columnas.	Se obtuvo un aumento en el análisis de entropía.	1
MixColumns	Se cambió la posición en la cual se ejecutó la operación	Encontró variaciones en la entropía e Histograma.	1
Key Expansion	Utilización de un vector y modificación de columnas.	Mayor Entropía	1
Key Expansion	Alterar aleatoriamente las posiciones de la matriz.	Igual Entropía	0
Rondas Estándar	Aumento del número de rondas.	Bajo Entropía	0
4 Funciones en Rondas Estándar	Se aumentó una función.	Aumento la entropía.	1
3 Funciones en la Ronda final	Se aumentó una función.	Aumento el tiempo en años a ser descifrado.	1

Realizado por: Raza C., 2019

Tomando solo aquellas cuyo valor fue 1 y combinándolas se obtuvo la función Mix-Key, Move-C y Mix-Shift; proporcionando un análisis más extensivo de su localización, funcionamiento e implementación en el algoritmo base como se expuso anteriormente.

Las dos funciones añadidas producen que el algoritmo aumente su proceso matemático en un 25% tomando en consideración que el AES posee 8 procesos y ahora el MECIB-AES cuenta con 10 procesos y por tal motivo es posible que el tiempo de cifrado aumente, por lo tanto, se realizó un análisis asintótico el mismo que dio como resultado la no alteración en el consumo de recursos del computador quedando como resultado $O(n*k)$; donde n y k dependerán de los valores ingresados además de las constantes encontradas durante el cifrado del algoritmo.

Una vez implementadas las modificaciones se realizó diferentes pruebas para comparar los dos algoritmos cuyos resultados fueron satisfactorios pues según esta se aceptó la hipótesis nula donde se midió la entropía, con un nivel de confiabilidad del 95% así como de un error del 5%, el análisis de frecuencias varió, dependiendo así de la aplicación de cada prueba realizada. La autocorrelación dio como resultado una mayor similitud de secuencias a favor del MECIB-AES, aunque puede tomarse como desventaja los valores no son grandes por lo cual se consideró viable el algoritmo.

El análisis de la fuerza bruta, donde se toma en consideración el tiempo, así como las posibles combinaciones de contraseñas, dio como resultado según sus medias Xa= 4,14E+25 y Xm= 4,24E+25 un aumento del 2,47% equivalente a 1,05E+24 años a favor del MECIB-AES.

El resultado del análisis de trafico realizado a los archivos dio como resultado una menor Latencia y una mayor Tasa de Transferencia; haciendo uso de una regla de 3 simple se nota una disminución del 43,19 % (Latencia) y un aumento del 31,82 % (Tasa de Transferencia) a favor del MECIB-AES.

Tomando en consideración todo lo desarrollado en este trabajo nuestro aporte se basa en analizar diferentes textos cifrados tomando en consideración la Entropía, Análisis de Frecuencia, Autocorrelación, Fuerza Bruta y Análisis de Tráfico brindando una perspectiva del nivel de seguridad que presenta nuestra propuesta en comparación a la que está basada; los resultados nos demuestran que es aceptable y viable la realización de esta modificación, por lo tanto, se impulsa la posibilidad de realizar mayores cambios en un futuro.

CONCLUSIONES

- ✓ Las modificaciones planteadas pasaron por un proceso empírico de selección que nos llevó a probar diferentes combinaciones las mismas que poco a poco fueron filtrando hasta encontrar las funciones Mix-Shift, Move-C, Mix-Key estas fueron añadidas al algoritmo AES modificando así su composición y funcionamiento, aumentando su proceso matemático en un 25% cumpliendo su propósito al permitir el desarrollo de un proceso de cifrado basado en el algoritmo simétrico AES.

- ✓ Para el análisis de la utilización de tiempo y recursos del computador se tomó un análisis asintótico, obteniendo una igualdad del 99.18% y diferencia del 0,13% por lo tanto las complejidades son iguales para los dos algoritmos.

- ✓ Para el análisis de la entropía se relacionado con la diferencia de medias muéstrales, por lo tanto, al realizar dichos cálculos con una confiablidad del 95% se obtuvo una mayor entropía para el MECIB-AES

- ✓ Los datos obtenidos del algoritmo AES y MECIB-AES presentan variaciones que, según las pruebas realizadas de histograma, diagrama, trigrama, 4-grama; se encontró una distribución normal, una reducción en diagramas en un 50 % de los archivos comparados a favor del MECIB-AES, también hay que señalar que en los trigramas y 4-gramas la diferencia encontrada no es muy significativa, por lo tanto, la consideramos igual.

- ✓ La comparación entre los dos algoritmos de forma empírica y el gráficamente realizado se concluyó que la autocorrelación es mayor en el MECIB-AES, pero a su vez tampoco implica que su seguridad sea vulnerada debido a que el valor de similitud no es significativo.

- ✓ El análisis de la fuerza bruta, dio como resultado según sus medias Xa= 4,14E+25 (AES) y Xm= 4,24E+25 (MECIB-AES), un aumento del 2,47% equivalente a 1,05E+24 años a favor del MECIB-AES.

- ✓ El análisis de tráfico dio como resultado menor Latencia y una mayor Tasa de Transferencia a favor del MECIB-AES, una disminución del 43,19 % (Latencia) y un aumento del 31,82 % (Tasa de Transferencia).

RECOMENDACIONES

- ✓ El estudio de diferentes fuentes ayuda al desarrollo lógico y matemático del algoritmo, por lo cual es necesario reconocer las bases matemáticas aplicadas en cada función.

- ✓ Para futuros trabajos se podría aplicar el algoritmo modificado a otro tipo de información ya sea voz o video.

- ✓ La utilización del algoritmo en diferentes archivos nos brinda una mejor perspectiva al momento de realizar las pruebas pues los resultados serán más concisos.

BIBLIOGRAFÍA

Aguirre, Juan David.. *C++ vs JAVA* [En línea]. Sistemas con el Niggi. 25 de 04 de 2016. [Citado el: 15 de 09 de 2018.] Disponible en: https://sites.google.com/site/sistemasconelniggi/home/c-vs-java.

Areitio, Gloria y Areitio, Ana. *Información, Informática e Internet: del ordenador personal a la Empresa 2.0* [En línea]. España : Vision Libros Calle San Benito 21 Local, 2009. 978-84-9886-525-7. Disponible en: https://books.google.com.ec/books?id=mnFTzjdoczIC&printsec=frontcover&hl=es&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false

Clavijo, Fabián y Forero, Javier Fernando. "Implementaciones criptográficas en FPGA". *Visión Electrónica* [En línea]**.** 06 de 2011, Vol. 5, págs. 26-37. [Citado el: 12 de 03 de 2018.]. Disponible en: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=2ahUKEwi_lc-F8t_fAhVQx1kKHW0CDjwQFjAAegQIChAC&url=https%3A%2F%2Fdialnet.unirioja.es%2Fdescarga%2Farticulo%2F4016340.pdf&usg=AOvVaw2lKKaaI3ZRaB1m0wEf7X3R

Cipsa, Centro de Informática Profesional. *Lenguajes Programación más populares 2018 (Ranking Tiobe)* [En línea]. Cipsa.net, 2018. [Citado el: 19 de 05 de 2018.]. Disponible en: https://cipsa.net/lenguajes-programacion-mas-populares-2018-ranking-tiobe/.

CUSHPA GUAMÁN, Ana Lucia. 2018. Diseño e Implementación de un Nuevo Algoritmo Criptográfico Simétrico para Mensajería Instantánea en un Entorno Web. (Tesis)(Maestría). [En línea]. Escuela Superior Politécnica del Chimborazo, Posgrado y Educación Continua. Riobamba. 2018. págs. 35-40, Citado el: 04 de 11 de 2018.]. Disponible en: http://dspace.espoch.edu.ec/bitstream/123456789/8166/1/20T00989.pdf

DAEMEN, Joan y RIJMEN, Vincent. *AES Proposal: Rijndael* [En línea]. Proton World Int.l; Katholieke Universiteit Leuven, ESAT-COSIC. Belgium, 1999. págs. 3-72. [Citado el: 14 de 04 de 2018.]. Disponible en: http://www.cs.miami.edu/home/burt/learning/Csc688.012/rijndael/rijndael_doc_V2.pdf

DÁVILA TORRES, Juan Pablo. Análisis y optimización del algoritmo de encriptación rijndael en el que se basa el estándar de encriptación avanzada aes (advanced encryption standard). (Tesis) (Pregrado). [En línea]. Universidad Nacional del Altiplano,Facultad de Ingeniería Mecánica Eléctrica, Electrónica y Sistemas. Puno - Peru. 2017. págs. 83-88. [Citado el: 12 de 06 de 2018.]. Disponible en: http://repositorio.unap.edu.pe/bitstream/handle/UNAP/4795/Davila_Torres_Juan_Carlos.pdf?sequence=1&isAllowed=y

Esslinger, Bernhard. *Cryptography for Everybody* [En línea]. CRYPTOOL PORTAL.CrypTool Contributors, 1998 - 2018. [Citado el: 10 de 10 de 2018.] Disponible en: https://www.cryptool.org/en/.

Fernandez, Santiago. "La Criptografía Clásica". *Sigma: Revista de Matemáticas* [En línea]. 04 de 2004, Bilbao : 2004ko Apirila, Vol. 24, págs. 119-141. [Citado el: 12 de 04 de 2018.]. Disponible en: https://s3.amazonaws.com/academia.edu.documents/40562076/9_Criptografia_clasica.pdf?AWSAccessKeyId=AKIAIWOWYYGZ2Y53UL3A&Expires=1547000448&Signature=hHF4mviwQ8lPArlf%2FhRiO2TM8sE%3D&response-content-disposition=inline%3B%20filename%3DCriptografia_clasica.

Granados, Paredes Gibrán. "Introducción a la Criptografía". *Revista Digital Universitaria* [En línea]. 10 de 07 de 2006, Coordinación de Publicaciones Digitales. DGSCA-UNAM, Vol. 7, págs. 3-17. 1067-6079. [Citado el: 22 de 07 de 2018.]. Disponible en: http://www.revista.unam.mx/vol.7/num7/art55/int55.htm

GÓMEZ HERNÁNDEZ, Sara. Análisis de textos cifrados de los siglos XVI y XVII. (Tesis) (Pregrado) [En línea]. Universidad Carlos III de Madrid, Ingeniería Técnica de Telecomunicaciones, Sistemas de Telecomunicación. Madrid, 2010. págs. 35-80. [Citado el: 16 de 09 de 2018.]. Disponible en: https://e-archivo.uc3m.es/bitstream/handle/10016/11110/PFC_Sara_Gomez_Hernandez.pdf?sequence=1&isAllowed=y

Herrera, Castellanos Mario. *Formula para Cálculo de la Muestra Poblaciones Finitas.* Hospital Roosevelt. 2011. [Citado el: 05 de 05 de 2018.]. Disponible en:

https://investigacionpediahr.files.wordpress.com/2011/01/formula-para-cc3a1lculo-de-la-muestra-poblaciones-finitas-var-categorica.pdf

HUERTA VILLALÓN, Antonio. Criptografía: Historia, Teoría de números, Sistemas de clave pública. Sistemas de clave privada (Paper) [En línea]. Colegio Oficial de Ingenieros Industriales de la Comunidad Valenciana. Valencia.2003. Vol. 55 pág. 11. [Citado el: 12 de 02 de 2018.]. Disponible en: http://shutdown.es/alfa55.pdf

JYOTI, Gaba y KUMAR, Mukesh. Implementation of Steganography Using CES Technique. (Conference Paper) [En línea]. The Technological Institute of Textile & Sciences, Department of Computer Engineering, IEEE Second International Conference on Image Information Processing (ICIIP), 2013. págs. 395-399. 10.1109/ICIIP.2013.6707622. [Citado el: 16 de 09 de 2017.]. Disponible en: https://www.researchgate.net/publication/261458593_Implementation_of_steganography_using_CES_technique

Martínez, Aguillón Erika. *Criptosistema PlayFair*. [En línea]. TextosCientificos.com, 23 de 03 de 2009. [Citado el: 27 de 12 de 2017.] Disponible en: https://www.textoscientificos.com/criptografia/playfair.

MARTÍNEZ DE LA TORRE, Javier. Implementación del algoritmo de cifrado AES para bajo consumo sobre FPGA (Tesis) (Pregrado) [En línea]. Escuela Politécnica Superior "Universidad Carlos III De Madrid, Tecnología Electrónica. Madrid, 2013. págs. 35-40. [Citado el: 13 de 01 de 2018.] Disponible en: https://e-archivo.uc3m.es/handle/10016/17607

MÉNDEZ NARANJO, Pablo Martí. 2015. Nuevo Algoritmo Criptográfico con la Incorporación de la Esteganografía en Imágenes. (Tesis)(Maestría). [En línea]. Escuela Superior Politécnica de Chimborazo, Posgrado y Educación Continua. Riobamba. 2015. págs. 13-50. [Citado el: 13 de 04 de 2018.] Disponible en: http://dspace.espoch.edu.ec/bitstream/123456789/4374/1/20T00628.pdf

Mendoza, Julio César. "Demostracion de Cifrado Simétrico y Asimétrico*". INGENIUS: Revista de Ciencias Exactas y Tecnologias* [En línea], 2008, Quito : Universidad Politécnica Salesiana,

Vol. 3, págs. 46- 53. [Citado el 20 de 03 de 2018]. Disponible en: https://dialnet.unirioja.es/servlet/articulo?codigo=5972695

MOYA, Jóse Manuel. Introducción a la Protección de la Información. "Criptografía" (Paper). [En línea]. Colegio Oficial de Ingenieros de Telecomunicación, Telecomunicaciones, In Escrituras silenciadas: paisaje como histografía, 2013. págs. 495-513, 978-84-15595-84-7. [Citado el: 12 de 12 de 2017.] Disponible en: http://www.academia.edu/21679789/INTRODUCCION_A_LA_PROTECCIÓN_DE_LA_INFORMACIÓN._CRIPTOGRAFÍA_

Pabón Cadavid, Jhonny Antonio. "La Criptografia y la Proteccion a la Informacion Digital". *Revista de la Propiedad Inmaterial* [En línea]**.** 2010, Colombia, Vol. 14, págs. 59-90. [Citado el: 12 de 03 de 2018.]. Disponible en: https://dialnet.unirioja.es/servlet/articulo?codigo=3647623

Palomino, Diego. *Escala de Likert* [En línea]. Diarlu, 27 de 12 de 2016. [Citado el: 10 de 12 de 2018.]. Disponible en: https://www.diarlu.com/mejores-ide-programar-java/.

PENAZZI, Daniel. *CRIPTOGRAFIA DE CLAVE SIMETRICA: AES (*Notas de curso para las Jornadas de Criptografía y Códigos Autocorrectores.*)* [En línea]. Universidad Nacional de Córdoba, Facultad de Matemática, Astronomía y Física. Mar de Plata, 2006. págs. 3-10. [Citado el: 03 de 02 de 2018.] Disponible en: http://www.famaf.unc.edu.ar/~penazzi/17NovAESMardePlata.pdf

POUSA, Adrián. Algoritmo de cifrado simétrico AES. Aceleración de tiempo cómputo sobre arquitecturas multicore (Tesis) (Pregrado) [En línea]. Universidad Nacional de La Plata,Facultad de Informática, Buenos Aires, 2011. págs. 9-17. [Citado el: 03 de 02 de 2018.] Disponible en: http://sedici.unlp.edu.ar/bitstream/handle/10915/4210/Documento_completo.pdf?sequence=1&isAllowed=y

Sergio, De Luz. *Criptografía : Algoritmos de Cifrado de Clave Simétrica* [En línea]. REDES@ZONE, 4 de 11 de 2010. [Citado el: 07 de 06 de 2018.]. Disponible en: https://www.redeszone.net/2010/11/04/criptografia-algoritmos-de-cifrado-de-clave-simetrica/.

Schiestl, Christian. *Pseudozufallszahlen in der Kryptographie,* Hamburg, Diplomica Verlag GmbH, 1999. 9783832441494. [Citado el: 03 de 12 de 2018.]. Disponible en: https://www.diplom.de/document/219773

Standards Federal Information Processing. *Specification for the ADVANCED ENCRYPTION STANDARD (AES)* [En línea]. National Institute of Standards and Technology, Department of Commerce, Information Technology Laboratory (ITL), 2001, págs. 5-12, (FIPS) 197. [Citado el: 12 de 03 de 2018.]. Disponible en: https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.197.pdf

Standard Data Encryption and others. *Federal information processing standards publication 46.* National Bureau of Standards [En línea]**.** 1977, US Department of Commerce, Vol. 23, págs. 20-30. [Citado el: 14 de 03 de 2018.]. Disponible en: https://pdfs.semanticscholar.org/453a/595acc14b04fe7f8093cb73e5da5cb245949.pdf?_ga=2.4863411.1013710420.1547008161-437990252.1547008161

STERBENZ, Andreas y LIPP, Peter. Performance of the AES Candidate Algorithms in Java [En línea]. Institute for Applied Information Processing and Communications. Austria : AES Candidate Conference, 2000. págs. 161-165. [Citado el: 14 de 07 de 2018.]. Disponible en: https://www.researchgate.net/publication/221005726_Performance_of_the_AES_Candidate_Algorithms_in_Java

Torres, Michelle. *Blog de desarrollo web, tecnologia, software libre y otras cosas.* [En línea]. michelletorres.mx, 2013. [Citado el: 14 de 09 de 2018.]. Disponible en: http://michelletorres.mx/diferencia-entre-c-y-cpp/.

UNID, Universidad Interamericana para el Desarrollo. *Distribucion_tStudent* [En línea]. Mi Materia en Linea, 11 de 09 de 2013. [Citado el: 09 de 08 de 2018.]. Disponible en: http://moodle2.unid.edu.mx/dts_cursos_mdl/lic/AE/E/AM/12/Distribucion_tStudent.pdf.

VÁZQUEZ FERNÁNDEZ, Miguel D. El Algoritmo Criptográfico AES para Protección de Datos (Tesis) (Pregrado) [En línea]. Universidad Pontificia Comillas, Escuela Técnica Superior de Ingeniería (ICAI). Madrid, 2007. págs. 27-58. [Citado el: 14 de 02 de 2018.]. Disponible en:

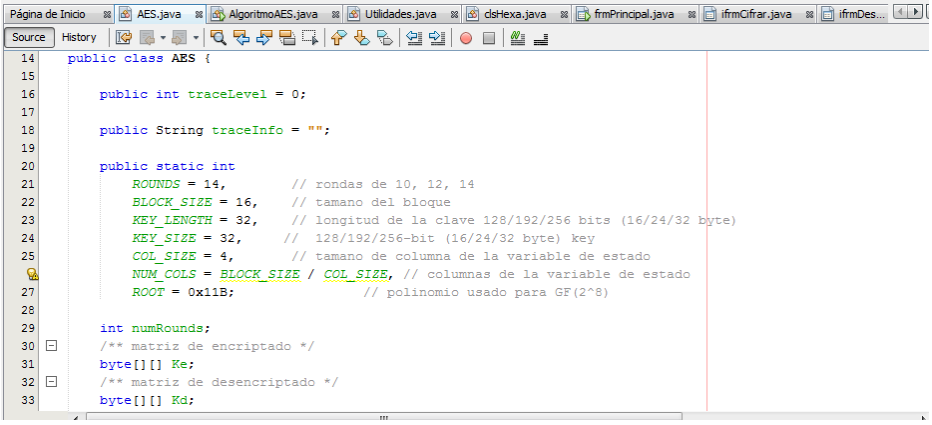
https://www.iit.comillas.edu/pfc/resumenes/46ea7511774d8.pdf / https://vdocuments.site/algoritmo-aes-565b49a845df8.html

VEGA PAÚL, Dino Marcelo. 2002. Tecnologías para la seguridad digital: Privacidad y seguridad en la transmisión de datos de un proceso electoral. (Tesis)(Pregrado). [En línea]. Universidad Nacional de Ingeniería, Facultad de Ingeniería Industrial y de Sistemas. Lima. 2002. págs. 27-34. [Citado el: 12 de 11 de 2018.]. Disponible en: file:///C:/Users/HP/Downloads/vega_pd.pdf

XIFRÉ SOLANA, Patricia. Antecedentes y Perspectivas de Estudio en Historia de la Criptografia (Tesis) (Pregrado) [En línea]. Universidad Carlos III. Madrid. 2008-2009. págs. 129-148**.** [Citado el: 23 de 01 de 2018.]. Disponible en: https://e-archivo.uc3m.es/bitstream/handle/10016/6173/PFC_Patricia_Xifre_Solana.pdf

ANEXOS

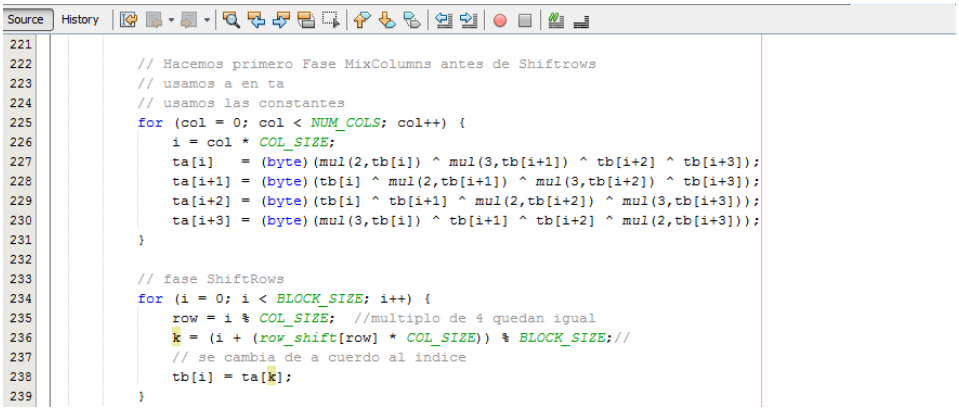
Anexo A: Declaración de las variables del MECIB-AES



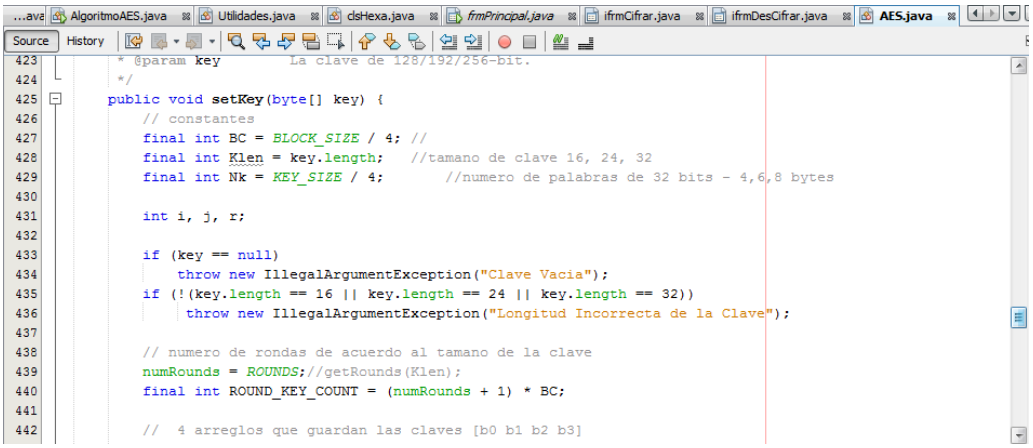
Anexo B: Bloque Mix-Shift

```
////////////////////////////////////////////////////////////////
//          bloque mdificado shift row
int[] mod_shift = {8, 12, 4, 8};
 int md = 0;
 int var = 0;
 for (i = 0; i < BLOCK_SIZE; i++) {
     if ((i % COL_SIZE) == 0)
     {
         var = mod_shift[md];
         md = md+1;
     }
     k = (i+var)% BLOCK_SIZE;
     tb[i] = ta[k];
 }
```

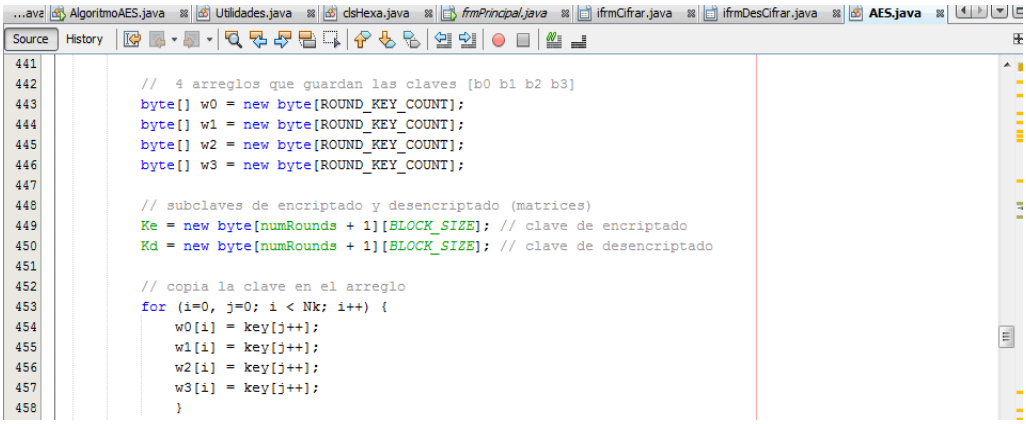
Anexo C: Proceso Move-C



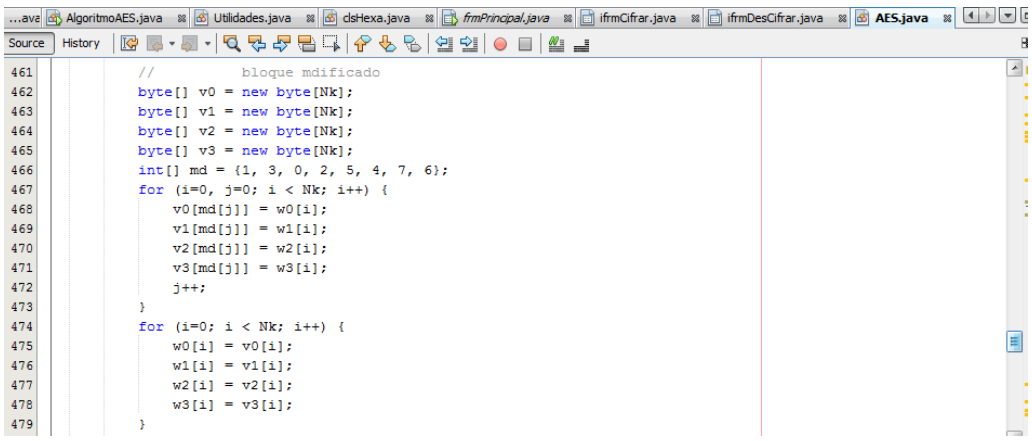
Anexo D: Proceso Mix-Key



```
423      * @param key        La clave de 128/192/256-bit.
424      */
425     public void setKey(byte[] key) {
426         // constantes
427         final int BC = BLOCK_SIZE / 4; //
428         final int Klen = key.length;   //tamano de clave 16, 24, 32
429         final int Nk = KEY_SIZE / 4;       //numero de palabras de 32 bits - 4,6,8 bytes
430 
431         int i, j, r;
432 
433         if (key == null)
434             throw new IllegalArgumentException("Clave Vacia");
435         if (!(key.length == 16 || key.length == 24 || key.length == 32))
436             throw new IllegalArgumentException("Longitud Incorrecta de la Clave");
437 
438         // numero de rondas de acuerdo al tamano de la clave
439         numRounds = ROUNDS;//getRounds(Klen);
440         final int ROUND_KEY_COUNT = (numRounds + 1) * BC;
441 
442         //  4 arreglos que guardan las claves [b0 b1 b2 b3]
```



```
441 
442         //  4 arreglos que guardan las claves [b0 b1 b2 b3]
443         byte[] w0 = new byte[ROUND_KEY_COUNT];
444         byte[] w1 = new byte[ROUND_KEY_COUNT];
445         byte[] w2 = new byte[ROUND_KEY_COUNT];
446         byte[] w3 = new byte[ROUND_KEY_COUNT];
447 
448         // subclaves de encriptado y desencriptado (matrices)
449         Ke = new byte[numRounds + 1][BLOCK_SIZE]; // clave de encriptado
450         Kd = new byte[numRounds + 1][BLOCK_SIZE]; // clave de desencriptado
451 
452         // copia la clave en el arreglo
453         for (i=0, j=0; i < Nk; i++) {
454             w0[i] = key[j++];
455             w1[i] = key[j++];
456             w2[i] = key[j++];
457             w3[i] = key[j++];
458             }
```



```
461         //          bloque mdificado
462         byte[] v0 = new byte[Nk];
463         byte[] v1 = new byte[Nk];
464         byte[] v2 = new byte[Nk];
465         byte[] v3 = new byte[Nk];
466         int[] md = {1, 3, 0, 2, 5, 4, 7, 6};
467         for (i=0, j=0; i < Nk; i++) {
468             v0[md[j]] = w0[i];
469             v1[md[j]] = w1[i];
470             v2[md[j]] = w2[i];
471             v3[md[j]] = w3[i];
472             j++;
473         }
474         for (i=0; i < Nk; i++) {
475             w0[i] = v0[i];
476             w1[i] = v1[i];
477             w2[i] = v2[i];
478             w3[i] = v3[i];
479         }
```