UNIVERSIDAD NACIONAL JORGE BASADRE GROHMANN - TACNA

Facultad de Ingeniería

Escuela Profesional de Ingeniería en Informática y Sistemas

ESTUDIO COMPARATIVO DE LOS ALGORITMOS DE ENCRIPTACIÓN ADVANCED ENCRYPTION STANDARD (AES) Y RIVEST, SHAMIR & ADLEMAN (RSA)

TESIS

Presentada por:

Bach. Heber Jhonatan Cabrera Jara

Para optar el Título Profesional de:

INGENIERO EN INFORMÁTICA Y SISTEMAS

TACNA - PERÚ

2018

UNIVERSIDAD NACIONAL JORGE BASADRE GROHMANN – TACNA

Facultad de Ingeniería

JURADO CALIFICADOR Y CALIFICACIÓN DE LA SUSTENTACIÓN DE TESIS

TESIS N°:_____ **TITULO PROFESIONAL DE:**
Ingeniero en Informática y Sistemas

La secretaría académica de la Facultad de Ingeniería, por resolución de Facultad N°05227-2018-FAIN/UNJBG, designó jurado para la sustentación oral de la Tesis titulada: "Estudio comparativo de los algoritmos de encriptación Advanced Encryption Standard (AES) y Rivest, Shamir & Adleman (RSA)".

El mismo que está conformado por:
Presidente: MSc. Edgardo Teófilo Valdez Cortijo
Secretario: MSc. Edgar Aurelio Taya Acosta
Vocal: Mgr. Gianfranco Alexey Málaga Tejada

Para calificar la sustentación de la Tesis en acto público el día 19 de octubre del 2018. Presentado por el Bachiller Heber Jhonatan Cabrera Jara, de la Escuela Profesional de Ingeniería en Informática y Sistemas.
El jurado calificador en forma secreta e individual emitió su opinión sobre el tema de la tesis expuesta y procedió a obtener el promedio que arrojó el calificativo de aprobado con la nota de catorce (14).
Para ratificar lo detallado firman:

MSc. Edgardo Teófilo Valdez Cortijo
Presidente

MSc. Edgar Aurelio Taya Acosta
Secretario

Mgr. Gianfranco Alexey Málaga Tejada
Vocal

UNIVERSIDAD NACIONAL JORGE BASADRE GROHMANN – TACNA

FACULTAD DE INGENIERÍA

ESCUELA PROFESIONAL DE INGENIERÍA EN INFORMÁTICA Y SISTEMAS

"ESTUDIO COMPARATIVO DE LOS ALGORITMOS DE ENCRIPTACIÓN ADVANCED ENCRYPTION STANDARD (AES) Y RIVEST, SHAMIR & ADLEMAN (RSA)"

TESIS SUSTENTADA Y APROBADA EL 19 DE OCTUBRE DEL 2018
ESTANDO EL JURADO CALIFICADOR INTEGRADO POR

Presidente

MSc. Edgardo Teófilo Valdez Cortijo

Secretario

MSc. Edgar Aurelio Taya Acosta

Vocal

Mgr. Gianfranco Alexey Málaga Tejada

Asesor

Dr. Edwin Antonio Hinojosa Ramos

DEDICATORIA

Porque de él, y por él, y para él, son todas las cosas. A él sea la gloria por los siglos. Amén.

Romanos 11:36

AGRADECIMIENTO

A mis profesores, quienes me guiaron en este camino

para lograr ser profesional.

A mi familia, quienes me apoyaron en todo momento

a pesar de las adversidades.

Y principalmente a Dios, quien por su gracia,

me ha permitido llegar hasta aquí.

CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

La seguridad de la información es un tema importante en la comunicación de datos. Actualmente existen muchos algoritmos de encriptación, cada cual se usa según la necesidad, debido a que, como todo algoritmo, tienen fortalezas y debilidades; es por ello que la presente tesis tiene como objetivo realizar un estudio comparativo de los algoritmos Advanced Encryption Standard (AES) y Rivest, Shamir & Adleman (RSA) para establecer la diferencia que existe entre ambos.

El diseño de la investigación es no experimental comparativa, donde de forma descriptiva se darán a conocer las diferencias que existen entre los algoritmos de encriptación con respecto al tiempo de ejecución y seguridad, para lo cual se someterán a pruebas donde los algoritmos tendrán que encriptar y resistir ataques.

Los resultados obtenidos muestran que los algoritmos AES y RSA llegan a complementarse, debido a que cada uno es óptimo según la necesidad que se tenga. Esto es porque AES trabaja mejor en el cifrado y descifrado que RSA, mientras que en cuanto a la seguridad, el intercambio de claves resulta más seguro RSA, debido a que no comparte la misma clave para el cifrado y el descifrado.

INTRODUCCIÓN

Todos de alguna manera queremos tener privacidad en nuestros datos, la criptografía surge para ayudar a lograr mantener segura la información. Ningún algoritmo de encriptación garantiza la total seguridad de los datos, pero si podrá hacer que terceras personas no le resulte tan fácil lograr vulnerar y tener acceso a la información.

La presente tesis se ha estructurado de la siguiente manera: En el Capítulo I trata la descripción del problema, donde se da a conocer la problemática de la investigación, alcances y limitaciones del mismo, y se trazan los objetivos; en el Capítulo II se desarrolla los fundamentos teóricos, que nos servirá de base en la tesis; en el Capítulo III se ve el tipo y diseño de la investigación, la operacionalización de variables, así como la técnica e instrumento para la recolección de datos; en el Capítulo IV se desarrollan las pruebas de cifrado y descifrado, así como las pruebas de ataques de vulnerabilidad; en el Capítulo V se da a conocer el análisis y la interpretación de resultados, y se discuten los resultados obtenidos teniendo en cuenta los antecedentes que se han considerado en el presente estudio. Para ir finalizando se da a conocer las conclusiones a las que se ha llegado, se sugieren algunas recomendaciones que se considera luego de haber realizado este estudio, y se citan las referencias bibliográficas que sirvió de ayuda.

CAPÍTULO I

PLANTEAMIENTO DEL PROBLEMA

1.1 Descripción del problema

1.1.1 Antecedentes del problema

A nivel mundial Chacón Zárate (2009) en su trabajo de "Comparativa de seguridad de algoritmos para resúmenes criptográficos" trazó como objetivo el realizar un estudio de las funciones hash por medio del análisis de su estructura, las operaciones matemáticas que integran su procesamiento interno. Las diferentes funciones evaluadas fueron en general basadas en el modelo de Merkle Damgard por el que se puede inferir que puede ser una base adecuada para los algoritmos hash que se utilicen con fines criptográficos, a pesar de la existencia de otros modelos.

Desde un punto de vista matemático Paguay Cuvi (2015) con "Análisis de algoritmos matemáticos de criptografía pública para mejorar el aprendizaje de la materia de criptografía en la carrera de ingeniería en sistemas de la ESPOCH" partiendo del estudio de las teorías matemáticas y definiendo los parámetros para realizar el análisis de los algoritmos de criptografía pública, se seleccionó el algoritmo más adecuado para mejorar el aprendizaje de la materia de criptografía.

Se desarrolló ambientes de prueba sobre RSA, Diffie-Hellman, El Rabin y El Gamal, siendo RSA el mejor algoritmo para la enseñanza con un total de 100% en su valoración técnica; mientras que en conceptos matemáticos Diffie-Hellman resulta ser el de mejor comprensión en los estudiantes con un 50,40%.

También Vasquez Fernández (2007) con "El algoritmo criptográfico AES para protección de datos" cuyo principal objetivo consiste en desarrollar una descripción de los fundamentos matemáticos y los métodos numéricos que cumplan con unas condiciones especificadas utilizadas por el algoritmo. Este proyecto también sirve para demostrar y comprobar cómo mediante el procesamiento matemático, numérico y simbólico, la importancia que este campo tiene en una disciplina tan importante como es el de la seguridad y la protección de datos, tan significativo en nuestros días, donde hay muchísimas posibilidades de acceso a una misma información.

A nivel nacional Samaniego Zanabria (2018) en "Evaluación de Algoritmos Criptográficos para mejorar la Seguridad en la Comunicación y Almacenamiento de la Información" se sometió a pruebas, herramientas, enfoques y ataques a los algoritmos criptográficos globalmente conocidos: AES, International Data Encryption Algorithm (IDEA) y Rivest Cipher (RC5), con el propósito de obtener resultados del grado de seguridad, fortaleza de clave, diseño – modo de cifrado, rendimiento y resistencia de cada uno de ellos, con el objetivo de evidenciar cuál

de ellos proporciona mayor seguridad en la comunicación y almacenamiento de la información.

Otro aporte de Mamani Ttito (2014) titulado “Modelo de sistema criptográfico de seguridad para las redes de comunicaciones en la región Puno” trata la implementación del modelo de seguridad, el cual se realizó a través de la aplicación del sistema de seguridad criptográfico asimétrica estándar corporativo determinando mecanismos y niveles de protección de datos, para las transacciones en redes, utilizando protocolo Secure Sockets Layer (SSL) con autenticación certificada, con la cual se cifran los datos para intercambiar entre el servidor y el cliente mediante el algoritmo de criptografía asimétrica. Se terminó demostrando que la aplicación de sistema seguridad con SSL VeriSign con tecnología web mejora de manera continua en seguridad e integridad de datos, asegurando el tráfico de HTTP, que lo convierte en HTTPS que garantiza que las transacciones sean seguras entre terminales distintas en redes, de esta manera disminuye vulnerabilidades y riesgos.

López Egusquiza (2013) en su tesis “Propuesta algorítmica para el ocultamiento de fotografías usando criptografía y esteganografía” busca lograr el ocultamiento total de la fotografía sin perder la calidad de la imagen ni la dela que lo va a ocultar, de manera que sea imperceptible por cualquier persona que tenga acceso al dispositivo; logrando implementar el algoritmo.

A nivel local no se ha encontrado trabajos de investigación que tengan que ver con algoritmos de encriptación.

1.1.2 Problemática de la investigación

Debido a la importancia de la información, es necesario tener un mecanismo que permita protegerla, para así evitar que alguien ajeno pueda manipularla. El desarrollo de la criptografía ha sido siempre consecuencia de la lucha entre quienes diseñan códigos para ocultar información y quienes ingenian estrategias para romperlos.

Existen diversos algoritmos matemáticos que intentan cubrir características básicas de seguridad. El nivel de cumplimiento es difícil de evaluar, ya que diversos algoritmos pueden ser vulnerables ante técnicas de ataque diferentes, además la mayoría de los algoritmos pueden trabajar con claves de distinta longitud lo cual afecta directamente a la robustez.

A nivel mundial, la criptografía ya dejó de ser exclusiva de gobiernos y mandos militares, ya que estos eran los únicos que necesitaban proteger sus comunicaciones. Esto provocó que la criptografía solo la conocieran unos pocos, sin embargo este aspecto ha cambiado, ya que ahora toda persona posee un ordenador en el cual tiene información y ahora busca la protección de la misma, lo que ha despertado el interés de empresas y universidades por esta ciencia. Según

Zimmermann (2004) con la introducción de los sistemas informáticos en las comunicaciones, el auge del internet y la globalización económica y social, la criptografía se ha hecho indispensable en las comunicaciones. Las cuentas de correo electrónico, las redes sociales, las transacciones bancarias por cajero electrónico o a través de internet, entre muchos otros, están protegidos por codificación electrónica. La criptografía está presente, algunos desconocen que sus datos en realidad viajan protegidos, mientras que otros sí saben cómo es que funciona y buscan como interceptar.

Morillo (1998) señala que los algoritmos criptográficos pueden ser analizados de diversos ámbitos: la informática, la matemática, se lo puede analizar por la propia teoría de algoritmos e incluso por la dificultad para su criptoanálisis, qué algoritmo es el más adecuado. Mientras que en los criptosistemas de clave pública la seguridad está basada en la dificultad computacional de problemas matemáticos como, por ejemplo, la descomposición en factores primos de un número entero de gran tamaño, la seguridad de los esquemas para compartir secretos es incondicional, es decir, no depende de los recursos computacionales del usuario. Otra diferencia importante son las herramientas matemáticas utilizadas: mientras que en los sistemas de clave pública la Teoría de Números, los Cuerpos Finitos y la Teoría de la complejidad juegan un papel fundamental, en la compartición de secretos intervienen principalmente la Combinatoria y el Álgebra Lineal.

Por ello, Xifre Solana (2009) en su trabajo señala que con más precisión, cuando se habla de esta área de conocimiento como ciencia se debería hablar de criptología, que engloba tanto las técnicas de cifrado, la criptografía propiamente dicha, como sus técnicas complementarias, el criptoanálisis, que estudia los métodos que se utilizan para romper textos cifrados con objeto de recuperar la información original en ausencia de la clave.

A nivel nacional Chuco Güere (2013) hizo un aporte en la sede Campo Armiño de Electroperu S.A., para incrementar la seguridad informática que se viene dando en la empresa, reduciendo así las vulnerabilidades frente a las amenazas de origen de software malicioso y personas donde la información se enfrenta a riesgos de daño o pérdida. La implementación del Sistema de Encriptación RSA con la funcionalidad de encriptación de 1024 - 2048 bits bajo una Metodología experimental tiene un impacto de fiabilidad alta e incrementa la fiabilidad en la transmisión de archivos de texto en la sede.

Es importante también tener en cuenta algunas páginas web, ver si poseen o no una conexión cifrada. En el caso por ejemplo de la página del Registro Nacional de Identificación y Estado Civil (RENIEC), que tiene a disposición datos muy importantes de todos los peruanos, podemos estar de alguna manera tranquilos al verificar que efectivamente en la información de la página, cuenta con conexión cifrada, lo cual dificulta que personas no autorizadas vean la información que viaja entre sistemas. Sin embargo, en otras páginas web que he

podido observar, en el caso de las universidades peruanas, la mayoría de sus páginas web, no están con conexión cifrada; incluso, algo tal vez más preocupante, es que la página del mismo banco de la nación, hasta la fecha, las conexiones de sus páginas tampoco se encuentran cifradas, lo cual significa un gran riesgo.

1.2 Formulación del problema

1.2.1 Problema general

¿Cuál es la diferencia de eficiencia del algoritmo de encriptación simétrico AES con respecto al algoritmo de encriptación asimétrico RSA?

1.2.2 Problemas específicos

- ¿Cuál es la diferencia de la encriptación AES y RSA con respecto al tiempo de ejecución?
- ¿Cuál es la diferencia de la encriptación AES y RSA con respecto a la seguridad?

1.3 Justificación e importancia

En la actualidad es difícil pensar en una sociedad sin acceso a la información. El auge en las telecomunicaciones ha hecho posible que diversas actividades comunes se puedan realizar con una computadora. La criptografía busca establecer nuevos y mejores mecanismos mediante los cuales sea posible el

intercambio de información de una manera segura, es decir, que llegue a las personas autorizadas y sin que esta información pueda ser alterada durante su transmisión. (Buchmann, 2001)

El desarrollo de las comunicaciones electrónicas, unido al uso de los computadores, hace posible la transmisión y almacenamiento de grandes flujos de información que es necesario proteger. (Merkle, 1978)

Mantener secreta la información es una prioridad para la mayor parte de las personas y prácticamente para todas las entidades. Es por ello que la criptografía adquiere gran importancia ya que permite la transmisión de datos de manera confidencial. Teniendo en cuenta la existencia de los algoritmos de encriptación, y sabiendo que es necesario contar con herramientas eficientes para proteger la información, se ha tomado en cuenta analizar, partiendo de los tipos de algoritmos de encriptación, para poder llegar a saber cuál es el algoritmo más eficiente.

La tarea no es crear un algoritmo de encriptación, pero sí emplear los que se tiene de forma adecuada, pero qué conceptos previos se debe de tener, qué algoritmo es el más adecuado para su aplicabilidad, son preguntas que no se pueden resolver de forma concreta ya que cada algoritmo tiene una creación y conceptualización distinta, es por ello que en esta tesis se someterá a estudio dos algoritmos muy conocidos de tipo simétrico y asimétrico.

1.4 Alcances y limitaciones

1.4.1 Alcances

La criptografía se encarga de crear y mejorar los métodos que permiten el acceso y la transmisión de información de tal manera que puedan otorgar privacidad, autenticidad e integridad. Se desarrollarán pruebas que permitan definir la eficiencia de un algoritmo de encriptación asimétrico (RSA) con respecto a uno simétrico (AES) con el propósito de demostrar qué algoritmo es mejor utilizar y cuándo, para ello tomaremos el tiempo que toma para el cifrado y descifrado, así como la resistencia a los ataques a los cuales serán sometidos.

En cualquier sistema o dispositivo, la seguridad del cifrado depende de diversos factores que van directamente relacionados con la capacidad del dispositivo que los aloja.

1.4.2 Limitaciones

Entre las limitaciones que se presentó para llevar a cabo la presente investigación fue la recopilación de antecedentes a nivel local y nacional, debido a que sólo se han realizado trabajos que describían o utilizaban un algoritmo de encriptación, mas no de ninguna comparación. Afortunadamente, a nivel internacional si han habido contribuciones de comparaciones de algoritmos de encriptación.

De igual manera una limitación para la investigación, debido a tratarse de algoritmos de encriptación, fue la computadora con la que se contó, ya que es fundamental para las operaciones que va realizar, el tiempo que se va a tardar, por poner un ejemplo, en tratar de romper una clave y poder así saber qué tan vulnerable puede ser. Para este estudio se utilizó una PC Intel® Core™ i5-2500 CPU 3.30GHz con 4,00 GB de RAM. Sistema operativo de 32 bits.

1.5 Objetivos

1.5.1 Objetivo general

Realizar un estudio comparativo de los algoritmos AES y RSA para establecer la diferencia que existe entre ambos algoritmos de encriptación.

1.5.2 Objetivo específico

•Analizar y comparar los algoritmos AES y RSA con respecto al tiempo de ejecución

•Analizar y comparar los algoritmos AES y RSA con respecto a la seguridad

1.6 Hipótesis

1.6.1 Hipótesis global

H_0: El algoritmo RSA no es más eficiente que el algoritmo AES

H_1: El algoritmo RSA es más eficiente que el algoritmo AES

1.6.2 Sub hipótesis

H_0: El tiempo de ejecución del algoritmo RSA no es mayor que el algoritmo AES

H_1: El tiempo de ejecución del algoritmo RSA es mayor que el algoritmo AES

H_0: El algoritmo RSA no proporciona mayor seguridad que el algoritmo AES

H_1: El algoritmo RSA proporciona mayor seguridad que el algoritmo AES

CAPÍTULO II

MARCO TEÓRICO

2.1 Antecedentes del estudio

Dado que, se busca siempre tener lo más seguro posible la información, es que surge la necesidad de poder encontrar el mejor cifrado posible, lo que conlleva a identificar que algoritmo más óptimo es el indicado para poder utilizarlo; es por ello que podremos encontrar estudios comparativos entre diferentes algoritmos de encriptación.

Al-Tamimi (2008) Proporciona una comparación de rendimiento entre cuatro algoritmos más comunes: Data Encryption Standard (DES), 3DES, AES, Blowfish. La comparación se ha llevado a cabo mediante la ejecución de varios ajustes diferentes para procesar diferentes tamaños de bloques de datos para evaluar la velocidad de cifrado/descifrado del algoritmo. La configuración de la simulación era en lenguaje C# de programación. Los resultados de este trabajo muestran que AES tiene un mejor rendimiento que otros algoritmos de cifrado común.

Mahajan & Sachdeva (2013) implementan tres técnicas de encriptación como algoritmos AES, DES y RSA y comparan el rendimiento de las técnicas de cifrado

basadas en el análisis de su tiempo estimulado en el momento del cifrado y descifrado. Los resultados de los experimentos se dan para analizar la efectividad de cada algoritmo.

Hercigonja (2016) proporciona un estudio analítico sobre varios algoritmos de encriptación simétrica como DES, 3DES, Carlisle Adams y Stafford Tavares (CAST-128), BLOWFISH, IDEA, AES, RC6 y Algoritmo RSA asimétrico. El análisis se basa en la arquitectura de los algoritmos, los aspectos de seguridad y las limitaciones que tienen. La comparación establece claramente que, aunque los algoritmos asimétricos son superiores en seguridad, toman más tiempo para procesar y requieren más memoria. Prácticamente, algoritmos asimétricos como RSA se utilizan para el intercambio de claves y algoritmos simétricos se utilizan para el cifrado/descifrado.

Muhammad Inaam ul haq, Asfand Yar Galani, & Zafar, (2018) seleccionan AES y RSA para encontrar cual es la mejor para proporcionar seguridad de datos en el entorno de computación en la nube. El rendimiento de todos estos algoritmos analizados mediante el uso de diferentes parámetros como la capacidad de cifrado de datos, la fuerza en función de la clave, el cifrado de datos y el tiempo de descifrado. El entorno utilizado para este fin es Windows Azure. El proceso de análisis después de comparar todos estos parámetros que se analizan a través del simulador desarrollado con Windows Azure SDk concluye que AES es la mejor opción porque es rápido y su capacidad de cifrado de datos es alta. RSA es más

seguro pero requiere muchos recursos y poder de cómputo. Su capacidad de cifrado de datos es muy baja en comparación con AES.

2.2 Fundamentos teóricos

2.2.1 Seguridad de la información

Una de las clasificaciones que más se utilizan es la que se recoge en el estándar internacional "ISO 7498-2, Arquitectura de Seguridad", cuyas definiciones y términos se han extendido ampliamente. Desde el punto de vista de esta norma, los servicios de seguridad son (Menezes, van Oorschot, & Vanstone, 1996) :

Autenticación. Es la identificación ante un sistema, subsistema, red o aplicación, mediante algún mecanismo o combinación de mecanismos. Una vez que una entidad se ha autenticado puede que necesite volver a autenticarse para otros fines.

Control de acceso. Protege contra el uso no autorizado de recursos. Generalmente, hay un orden implícito, según el cual una entidad primero se identifica y autentifica y a continuación se le proporciona el acceso o se le niega, basándose en mecanismos de control de acceso asociados a las credenciales de la entidad. Cada entidad tiene sus permisos de acceso a cada recurso especificado.

Confidencialidad. Consiste en proteger los datos trasmitidos o almacenados de difusiones no autorizadas. Este aspecto de la seguridad se ha vuelto cada vez

más importante puesto que cada vez mayor cantidad de información se trasmite sobre redes inseguras y cada vez mayor volumen de información sensible llega a los equipos menos protegidos de una red. Se debe asumir que la red no es fiable y que la información trasmitida puede ser interceptada. En la actualidad el cifrado de la información es la principal herramienta utilizada para conseguir la confidencialidad.

Integridad de Datos. Consiste en detectar cuando los datos almacenados o trasmitidos han sido modificados, borrados o reproducidos.

No rechazo. Consiste en asociar la identidad de un individuo con su participación en un proceso. Los mecanismos de no rechazo proporcionan pruebas de un intercambio digital significativo de algún tipo.

Gestión. Consiste en la administración y gestión de los mecanismos asociados con las categorías de seguridad.

2.2.2 Criptosistema

Un Criptosistema se define como la quíntupla (m, C, K, E, D), donde:

- m representa el conjunto de todos los mensajes sin cifrar (texto plano) que pueden ser enviados.
- C Representa el conjunto de todos los posibles mensajes cifrados, o criptogramas.
- K representa el conjunto de claves que se pueden emplear en el Criptosistema.

- E es el conjunto de transformaciones de cifrado o familia de funciones que se aplica a cada elemento de m para obtener un elemento de C. Existe una transformación diferente E_k para cada valor posible de la clave k.
- D es el conjunto de transformaciones de descifrado, análogo a E.

Todo criptosistema cumple la condición $\mathbf{D_k(E_k(m))=m}$ es decir, que si se tiene un mensaje **m**, se cifra empleando la clave **K** y luego se descifra empleando la misma clave, se obtiene el mensaje original **m**. (Stinson, 2006)

Existen dos tipos fundamentales de criptosistemas utilizados para cifrar datos e información digital y ser enviados posteriormente después por medios de transmisión libre. (Stinson, 2006)

Simétricos o de clave privada: Se emplea la misma clave K para cifrar y descifrar, por lo tanto el emisor y el receptor deben poseer la clave. El mayor inconveniente que presentan es que se debe contar con un canal seguro para la transmisión de dicha clave.

Asimétricos o de llave pública: Se emplea una doble clave conocidas como Kp (clave privada) y KP (clave Pública). Una de ellas es utilizada para la transformación E de cifrado y la otra para el descifrado D.

En muchos de los sistemas existentes estas clave son intercambiables, es decir que si empleamos una para cifrar se utiliza la otra para descifrar y viceversa. (Ramió, 1999)

2.2.3 Criptología

La criptología (del griego cryptos = oculto y logos = tratado, ciencia) es el nombre con el que se designan a la disciplina que se dedica al estudio de la escritura secreta, la cual se divide en otras dos disciplinas opuestas y a la vez complementarias: Criptografía y criptoanálisis. (Salomon, 2005)

En su clasificación dentro de las ciencias, la criptología proviene de una rama de las matemáticas, que fue iniciada por el matemático Claude Elwood Shannon en 1948, denominada: "Teoría de la Información". Esta rama de las ciencias se divide en: "Teoría de Códigos" y en "Criptología". Y a su vez la Criptología se divide en Criptoanálisis y Criptografía, como se muestra en la siguiente figura: (Caballero Gil, 2002)

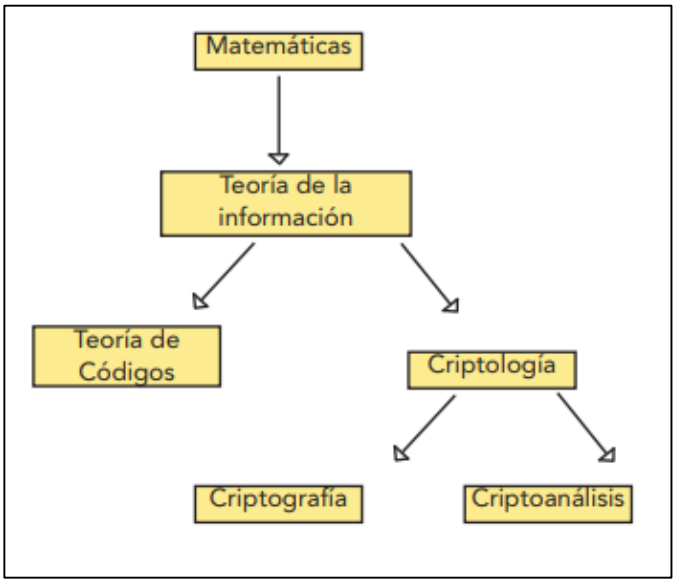


Figura 1. Origen de la Criptología
Fuente: Caballero Gil (2002)

2.2.4 Criptografía

Rama inicial de las Matemáticas y en la actualidad de la Informática y la Telemática, que hace uso de métodos y técnicas con el objeto principal de cifrar y/o proteger un mensaje o archivo por medio de un algoritmo, usando una o más claves.

Esto da lugar a diferentes tipos de sistemas de cifra que permiten asegurar estos cuatro aspectos de la seguridad informática: la confidencialidad, la integridad, la disponibilidad y el no repudio de emisor y receptor. (Ramió Aguirre, 2006)

Entre las disciplinas que engloba cabe destacar a la Teoría de la Información, la Teoría de Números y la complejidad algorítmica. (Ramió, 1999)

En general los algoritmos criptográficos se pueden clasificar en tres grandes familias (Menezes, van Oorschot, & Vanstone, 1996):

- Criptografía de clave secreta o criptografía simétrica.
- Criptografía de clave pública o criptografía asimétrica.
- Algoritmos HASH o de resumen.

El esquema fundamental de un proceso criptográfico (cifrado/descifrado) puede resumirse del modo en que se muestra en la Figura 2. (Fúster Sabater, de la Guía Martinez, Hernández Encinas, Montoya Vitini, & Muñoz, 2000)

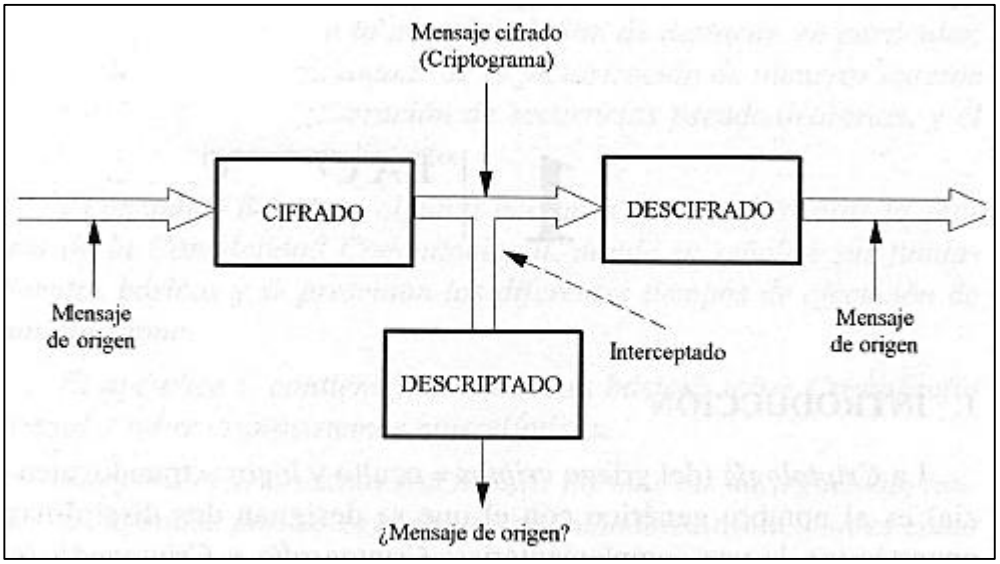


Figura 2. Proceso general cifrado/descifrado
Fuente: Fúster Sabater, de la Guía Martinez, Hernández Encinas, Montoya Vitini, & Muñoz (2000)

La criptografía ha sido usada desde tiempos antiguos para mantener la confidencialidad de la información muchos de estos usos que se le ha dado viene originado por las guerras, para comunicar las tácticas y técnicas a usarse en el campo de batalla desde los mandos superiores a sus tropas, el éxito de una batalla se reduce por lo tanto a interceptar y descifrar los mensajes enemigos. (Singh, 1999)

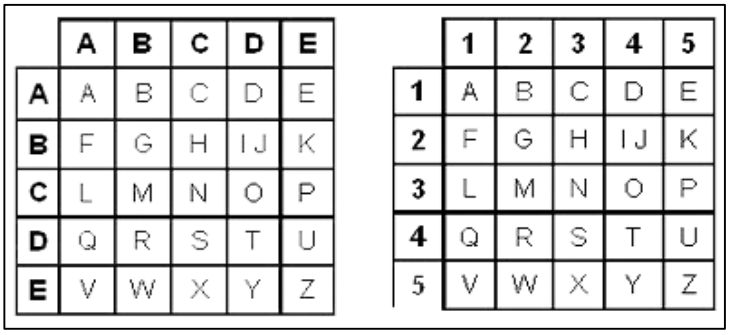
Los primeros usos de la criptografía se remonta la época de los espartanos en donde se usaba una vara de madera de un diámetro especifico, el cual tenían tanto emisor como receptor, el mensaje se escribía en una cinta de cuero que se enrollaría después en la vara especificando su inicio para que el mensaje coincidiera. La cinta por si sola posee un mensaje incoherente más al combinarla

con los códigos de columna de inicio tendría sentido para el remitente. (Galende Díaz, 1995)

Figura 3. Criptografía Espartana
Fuente: Galende Díaz (1995)

A mediados del siglo II antes de J.C., encontramos el cifrador por sustitución de caracteres más antiguo que se conoce. Atribuido al historiador griego Polybios, el sistema de cifra consistía en hacer corresponder a cada letra del alfabeto un par de letras que indicaban la fila y la columna en la cual aquella se encontraba, en un recuadro de 5 x 5 = 25 caracteres, transmitiéndose por tanto en este caso el mensaje como un criptograma. (Galende Díaz, 1995)

En la Figura 4 se muestra una tabla de cifrar de Polybios, con un alfabeto de cifrado consistente en el conjunto de letras A, B, C, D y E aunque algunos autores representan el alfabeto de cifrado como los números 1, 2, 3, 4 y 5.



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

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

Figura 4. Cifrado de Poliybio
Fuente: Galende Díaz (1995)

En la antigua Roma los ejércitos de Julio César necesitaron cifrar los mensajes del emperador para ser llevados a las líneas del ejército con órdenes para los combates, esta técnica de cifrado se conoció como el cifrado cesar, era un método en el cual cada letra del mensaje original consistía en una letra del alfabeto pero sumadas tres (3) veces su posición, con ello el mensaje original quedaba oculto y seguro.

Tomando en cuenta que ya se hace uso de una matemática básica para soporte, juegos de coordenadas, posiciones. (Kahn, 1995)

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

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

En el siglo XV se dan los primeros pasos hacia la criptografía moderna introduciendo los conceptos de confusión y difusión de Claude Shannon, el disco de Alberti era un cifrado de clave secreta que consistía en dos discos, estos discos proveían un mensaje cifrado el cual no poseía correspondencia con el método usado, el anillo eterno poseía veinte (20) letras más 4 números el interno tenía las

mismas letras y números pero en minúsculas pero en orden inverso añadiendo el carácter &. (Galende Díaz, 1995)

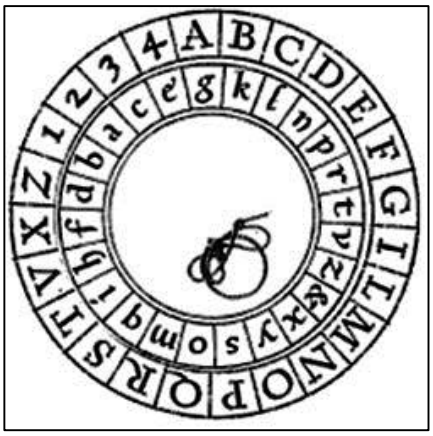


Figura 5. Discos de Alberti
Fuente: Galende Díaz (1995)

El tablero de Vigenere es una variación de Alberti, que usa una matriz originada por el mensaje a cifrar y una palabra clave reiterativa, la letra del mensaje más la letra de la palabra clave representan una letra del alfabeto, y este alfabeto es originado de los veinte y seis (26) alfabetos de cesar en orden. En el Siglo XVIII Beaufort crea un cifrado inverso del anterior en donde la letra de la palabra clave más la letra del alfabeto de cesar genera la letra del mensaje cifrado. (Singh, 1999)

En 1917 se crea el primer sistema criptográfico seguro ideal según la teoría de SHANNON, era el sistema de Vernan que consistía en tener una clave de cifrado al menos tan larga como el mensaje original, transcrito a un alfabeto binario de cinco (5) dígitos por cada carácter a cifrar esto se realiza con probabilidad la

operación módulo veinte y seis (26) así como una operación OR entre las cadenas para obtener el mensaje final, su uso fue mayormente militar durante la segunda guerra mundial, pero otro método surgió en este periodo del tiempo con el auge de la guerra la máquina ENIGMA dejando atrás los cifrados imprecisos a mano, aunque en Estados Unidos ya existían patentes de equipos criptográficos como SIGABA, en Japón Purple y de esta era criptográfica no solo se benefició el ejército sino también bancos e industrias por medio de la creación de la compañía Aktiebolget Cryptograph. (Solana, 2009)

Hay dos categorías principales de la criptografía en función del tipo de claves de seguridad utilizadas para cifrar / descifrar los datos. Estas dos categorías son: técnicas de cifrado simétricas y asimétricas.

2.2.4.1 Cifrado simétrico

También se conoce como la criptografía de clave única. Se utiliza una sola clave. En este proceso de cifrado del receptor y el emisor tiene que ponerse de acuerdo sobre una sola clave secreta (compartido).

Dado un mensaje (denominado texto plano) y la clave, cifrado produce datos ininteligibles, que es aproximadamente la misma longitud que el texto en claro. El descifrado es la inversa de la codificación, y utiliza la misma clave de cifrado. (Pastor Franco, Sarasa López, & Salazar Riaño, 1998)

En la Figura 6 se observa el proceso de clave para criptografía simétrica

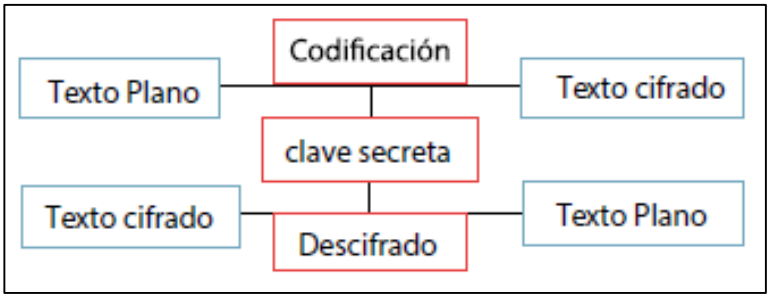


Figura 6. Proceso de clave para criptografía simétrica
Fuente: Pastor Franco, Sarasa López, & Salazar Riaño (1998)

2.2.4.2 Cifrado asimétrico

También se conoce como la criptografía de clave pública. Se utiliza dos claves: clave pública, que es conocido por el público, que se utiliza para el cifrado y la clave privada, que es conocida sólo por el usuario de esa clave, que se utiliza para el descifrado. Las claves públicas y la privada están relacionadas entre sí por cualquier medio matemático.

En otras palabras, los datos cifrados por una clave pública pueden ser encriptados sólo por su clave privada correspondiente. (Pastor Franco, Sarasa López, & Salazar Riaño, 1998)

En la Figura 7 se observa el proceso de clave para criptografía asimétrica

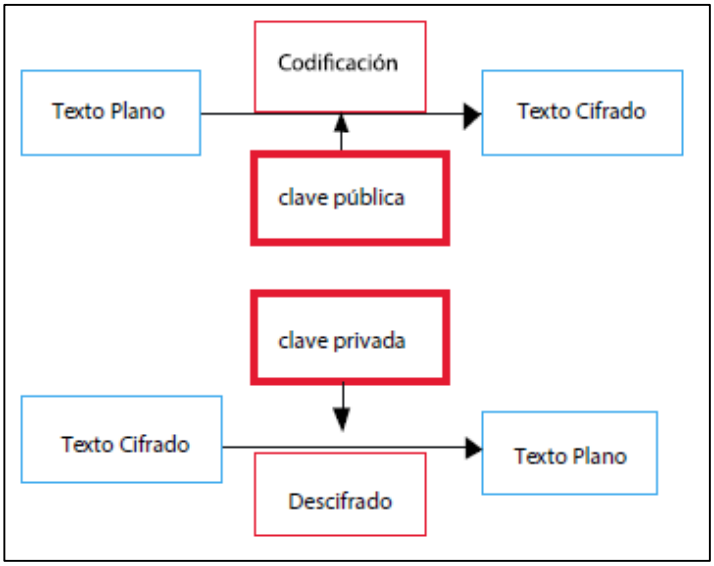


Figura 7. Proceso de clave para criptografía asimétrica.
Fuente: Pastor Franco, Sarasa López, & Salazar Riaño (1998)

Según Diffie y Hellman, todo algoritmo de clave pública debe cumplir las siguientes propiedades de complejidad computacional (Fernandez Domingo, 2006):

1. Cualquier usuario puede calcular sus propias claves pública y privada en tiempo polinomial.

2. El emisor puede cifrar su mensaje con la clave pública del receptor en tiempo polinomial.

3. El receptor puede descifrar el criptograma con la clave privada en tiempo polinomial.

4. El criptoanalista que intente averiguar la clave privada mediante la pública se encontrará con un problema intratable.

5. El criptoanalista que intente descifrar un criptograma teniendo la clave pública se encontrará con un problema intratable.

2.2.5 Criptoanálisis

El criptoanálisis consiste en comprometer la seguridad de un criptosistema. Esto se puede hacer descifrando un mensaje sin conocer la llave, o bien obteniendo a partir de uno o más criptogramas la clave que ha sido empleada en su codificación. No se considera criptoanálisis el descubrimiento de un algoritmo secreto de cifrado. (Ramió, 1999)

El mecanismo que se emplee para obtenerlos es indiferente. En general, todas las técnicas que buscan exhaustivamente por el espacio de claves K se denominan de fuerza bruta, y no suelen considerarse como auténticas técnicas de criptoanálisis, reservándose este término para aquellos mecanismos que explotan posibles debilidades intrínsecas en el algoritmo de cifrado.

Se denomina ataque a cualquier técnica que permita recuperar un mensaje cifrado empleando menos esfuerzo computacional que el que se usaría por la fuerza bruta. Se da por supuesto que el espacio de claves para cualquier criptosistema digno de interés ha de ser suficientemente grande como para que los métodos basados en la fuerza bruta sean inviables. Un par de métodos de

criptoanálisis que han dado interesantes resultados son el análisis diferencial y el análisis lineal. (Lucena López, 2009)

2.2.6 Encriptación RSA

En criptografía, RSA es un algoritmo para encriptación con clave pública. Es utilizado ampliamente en protocolos de comercio electrónico. Sujeto a múltiples controversias, desde su nacimiento nadie ha conseguido probar o rebatir su seguridad, y se le tiene como uno de los algoritmos asimétricos más seguros (Lucena López, 2009)

Debe su nombre a sus tres inventores: Ronald Rivest, Adi Shamir y Leonard Adleman, y desde 1983 estuvo bajo patente de los RSA Laboratories hasta el 20 de setiembre del 2000, por lo que su uso comercial estuvo restringido hasta esa fecha.

RSA se basa en la dificultad para factorizar grandes números. Las claves pública y privada se calculan a partir de un número que se obtiene como producto de dos primos grandes. El atacante se enfrentará, si quiere recuperar un texto claro a partir del criptograma y la clave pública, a un problema de factorización o tendrá que resolver un logaritmo discreto. (Lucena López, 2009)

Para generar un par de claves (KP, Kp), en primer lugar se eligen aleatoriamente dos números primos grandes, p y q. Después se calcula el producto

$$n = pq \quad [1]$$

Escogeremos ahora un número e primo relativo con (p – 1)(q – 1). (e, n) será la clave pública. Nótese que e debe tener inversa módulo (p – 1)(q – 1), por lo que existirá un número d tal que

$$de \equiv 1 \ (\text{mód}\ (p-1)(q-1)) \quad [2]$$

Es decir, que *d* es la inversa de *e* módulo (p–1)(q–1). (*d*; n) será la clave privada. Esta inversa puede calcularse fácilmente empleando el Algoritmo Extendido de Euclides. Nótese que si desconocemos los factores de n, este cálculo resulta prácticamente imposible.

La operación de cifrado se lleva a cabo según la expresión:

$$c = m^e \ (\text{mód}\ n) \quad [3]$$

Mientras que el descifrado se hará de la siguiente forma:

$$m = c^d \ (\text{mód}\ n) \quad [4]$$

Ya que

$$c^d = (m^e)^d = m^{ed} = m^{k(p-1)(q-1)+1} = (m^k)^{(p-1)(q-1)}m \quad [5]$$

Recordemos que $\phi(n) = (p-1)(q-1)$, por lo que, según la ecuación $a^{\phi(n)} \equiv 1(\text{mód}\ n)$

$$(m^k)^{(p-1)(q-1)} = 1 \quad [6]$$

Lo cual nos lleva de nuevo a m, siempre y cuando m y n sean primos relativos.

Ya que en nuestro caso n es compuesto, puede ocurrir que no sea primo relativo con m. Para ver lo que ocurre, podemos llevar a cabo el siguiente razonamiento: Buscamos un número *a* tal que

$$m^a \equiv 1 \text{ (mód n)} \qquad [7]$$

Tiene que cumplirse que $m^a \equiv 1$ (mód p) y $m^a \equiv 1$ (mód q), ya que p y q dividen a n. Aplicando el Teorema de Fermat (expresión $a^{p-1} \equiv 1$ (mód p)), tenemos que *a* debe ser múltiplo de (p – 1) y de (q – 1), por lo que *a* = mcm (p – 1, q – 1). Ya que el mínimo común múltiplo de (p–1) y (q–1) divide a (p–1)(q–1), el razonamiento dado inicialmente para demostrar el buen funcionamiento del algoritmo sigue siendo válido.

Por esta razón, en muchos lugares se propone obtener *d* de forma que:

$$de \equiv 1 \text{ (mód mcm}(p - 1; q - 1)) \qquad [8]$$

Con lo que obtendremos valores más pequeños, y por lo tanto más manejables, para la clave de descifrado. (Lucena López, 2009)

En muchos casos, se suele utilizar el Teorema Chino del Resto para facilitar los cálculos a la hora de descifrar un mensaje. Para ello se incluyen p y q en la clave privada, se calcula p1 = p-1 (mód q), y cuando se desea descifrar un mensaje c, se plantea el siguiente sistema de congruencias:

$$[c \text{ mód } p]^{[d \text{ mód } (p-1)]} \equiv m_1 \text{ (mód p)} \qquad [9]$$

$$[c \text{ mód } q]^{[d \text{ mód } (q-1)]} \equiv m_2 \text{ (mód } q) \qquad [10]$$

Como ya se vio, estas ecuaciones tienen una solución única m módulo n. Para recuperar el mensaje original m, en lugar de usar la fórmula que dimos en la demostración del teorema, emplearemos otra ligeramente distinta:

$$m = m_1 + p[(m_2 - m_1)\, p_1 \text{ (mód } q)] \qquad [11]$$

Es inmediato comprobar que esta expresión es igual a m1 módulo p. Si por el contrario tomamos módulo q, vemos que el segundo sumando es igual a $m_2 - m_1$, por lo que nos quedará m_2. Con ello conseguimos que el módulo a la hora de hacer las exponenciaciones sea sensiblemente menor, y, en consecuencia, los cálculos más rápidos. Nótese que los valores [d mód (p - 1)], [d mód (q - 1)] y p1 pueden tenerse calculados de antemano y, de hecho, se suelen incluir en la clave privada.

En la práctica, cuando queramos generar un par de claves RSA, escogeremos p y q con un número grande de bits, por ejemplo 512, con lo que n tendrá 1024 bits. A la hora de cifrar, subdividiremos el mensaje en bloques de, por ejemplo, 1016 bits, de esta forma queda garantizado que el valor de cada bloque sea menor que n, y efectuaremos la codificación de cada uno. Obtendremos un mensaje cifrado ligeramente más grande que el original, puesto que cada bloque tendrá ahora 1024 bits. Para decodificar, partiremos el mensaje cifrado en trozos de 1024

bits, y nos quedaremos con los primeros 1016 de cada uno tras aplicar el algoritmo. (Lucena López, 2009)

El atacante, si quiere recuperar la clave privada a partir de la pública, debe conocer los factores p y q de n, y esto representa un problema computacionalmente intratable, siempre que p y q (y por lo tanto n) sean lo suficientemente grandes.

Matemática del RSA

• Aritmética Modular

El RSA utiliza aritmética modular. Es similar a la aritmética convencional pero solo utiliza enteros positivos menores a un valor determinado llamado el módulo o modulus. Operaciones como la suma, resta y multiplicación trabajan como en las matemáticas convencionales pero no existe la división. Se puede utilizar cualquier valor como módulo. (Johnston, 2005)

También es conocida como la aritmética del reloj. Por ejemplo: asumiendo que los días de la semana se numeran del 0 al 6, éstos utilizan módulo 7 pues al llegar al valor 6 el siguiente día vuelve a tomar el valor 0. El horario utiliza un módulo 60 para los segundos y minutos y un módulo 12 para las horas.

• Prueba de Primalidad de Miller-Rabin

Existen diversos métodos para la comprobación entre métodos determinísticos y probabilísticos. El método más sencillo entre los

determinísticos, es el comprobar si el número en cuestión es divisible por alguno de los primeros números enteros empezando desde el 2 hasta la raíz cuadrada del mismo número.

Una mejora que se puede aplicar a este algoritmo es considerar sólo los números primos existentes dentro de ese rango. El algoritmo computacional más difundido: “Miller-Rabin Primality Test”

• Algoritmo de Euclides

El algoritmo de Euclides es el método más eficiente y simple para encontrar el valor del máximo común divisor de dos números (m.c.d), mediante operaciones modulo en los números enteros (Aguilera, 2013), procedimiento:

Siendo a y b dos números enteros, diferente de 0 y no iguales:

1.- Se divide el mayor número para el menor.

2.- Se determina el residuo de la división.

3.- Si la división es exacta el último divisor generado en el m.c.d.

4.- Si la división no es exacta se divide al divisor para el resto de la división hasta obtener una división exacta.

• Algoritmo Extendido de Euclides

El Algoritmo Extendido de Euclides también puede ser empleado para calcular inversas. Es una ampliación del de Euclides, que posee el mismo orden de

complejidad, y que se obtiene simplemente al tener en cuenta los cocientes además de los restos en cada paso.

El invariante que mantiene es el siguiente, suponiendo que se le pasen como parámetros n y a:

$$gi = nui + avi \quad [12]$$

El último valor de g_i será el máximo común divisor entre a y n, que valdrá 1 si estos números son primos relativos, por lo que tendremos:

$$1 = nu_i + av_i \quad [13]$$

o sea,

$$av_i \equiv 1 \text{ (mód n)} \quad [14]$$

luego (v_i mód n) será la inversa de a módulo n.

2.2.7 Encriptación AES

En 1996, el Instituto Nacional de Estándares y Tecnología (NIST) dio los primeros pasos para la creación de un Estándar de Cifrado Avanzado (Advanced Encryption Standard) que de forma abreviada se conoce como AES.

Su objetivo fue desarrollar una especificación para encontrar un algoritmo de cifrado que sustituyera al anticuado DES, de tal forma que el nuevo algoritmo fuese capaz de proteger la información sensible de los ciudadanos y del gobierno hasta bien entrado el siglo XXI.

En 1997, el NIST decidió realizar un concurso para escoger un nuevo algoritmo de cifrado capaz de proteger información sensible.

También conocido con el nombre "Rijndael", ya que fue desarrollado por dos criptólogos belgas, Joan Daemen y Vincent Rijmen, ambos estudiantes de la Katholieke Universiteit Leuven.

AES ha definido cada ronda como una composición de cuatro funciones invertibles diferentes, formando tres capas, diseñadas para proporcionar resistencia frente a criptoanálisis lineal y diferencial. (Lucena López, 2009)

Cada una de las funciones tiene un propósito preciso:

- La *capa de mezcla lineal* (funciones DesplazarFila y MezclarColumnas) permite obtener un alto nivel de difusión a lo largo de varias rondas.
- La *capa no lineal* (función SubBytes) consiste en la aplicación paralela de s-cajas con propiedades óptimas de no linealidad.
- La *capa de adición de clave* es un simple or-exclusivo entre el estado intermedio y la subclave correspondiente a cada ronda.

AES es un algoritmo que se basa en aplicar un número determinado de rondas a un valor intermedio que se denomina estado. Dicho estado puede representarse mediante una matriz rectangular de bytes, que posee cuatro filas, y N_b columnas. Así, por ejemplo, si nuestro bloque tiene 160 bits (Figura 7), N_b será igual a 5.

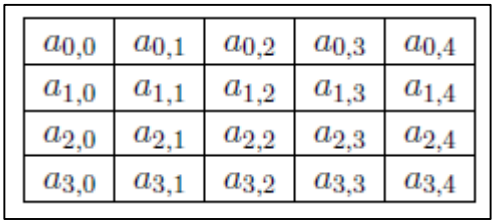


Figura 8. Ejemplo de matriz de estado con Nb=5 (160 bits).
Fuente: Lucena López (2009)

La clave tiene una estructura análoga a la del estado, y se representará mediante una tabla con cuatro filas y N_k columnas.

Si nuestra clave tiene, por ejemplo, 128 bits, N_k será igual a 4 (Figura 8).

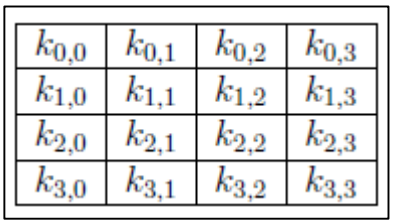


Figura 9. Ejemplo de clave con Nk=4 (128 bits).
Fuente: Lucena López (2009)

En algunos casos, tanto el estado como la clave se consideran como vectores de registros de 32 bits, estando cada registro constituido por los bytes de la columna correspondiente, ordenados de arriba a abajo.

El bloque que se pretende cifrar o descifrar se traslada directamente byte a byte sobre la matriz de estado, siguiendo la secuencia a0,0; a1,0; a2,0; a3,0; a0,1…, y análogamente, los bytes de la clave se copian sobre la matriz de clave en el mismo orden, a saber, k0,0; k1,0; k2,0; k3,0; k0,1… (Lucena López, 2009)

Siendo B el bloque que queremos cifrar, y S la matriz de estado, el algoritmo AES con n rondas queda como sigue:

1. Calcular K_0, K_1,..,K_n subclaves a partir de la clave K.

2. $S \leftarrow B \oplus K_0$

3. Para i = 1 hasta n hacer

4. Aplicar ronda i–ésima del algoritmo con la subclave K_i.

Puesto que cada ronda es una sucesión de funciones invertibles, el algoritmo de descifrado consistirá en aplicar las inversas de cada una de las funciones en el orden contrario, y utilizar los mismos Ki que en el cifrado, sólo que comenzando por el último. (Lucena López, 2009)

2.2.7.1 Las Rondas de AES

Puesto que AES permite emplear diferentes longitudes tanto de bloque como de clave, el número de rondas requerido en cada caso es variable. En la tabla 1 se especifica cuántas rondas son necesarias en función de Nb y Nk.

Tabla 1

Número de rondas para AES en función de los tamaños de clave y bloque

	$N_b = 4$ (128 bits)	$N_b = 6$ (192 bits)	$N_b = 8$ (256 bits)
$N_k = 4$ (128 bits)	10	12	14
$N_k = 6$ (192 bits)	12	12	14
$N_k = 8$ (256 bits)	14	14	14

Fuente: Lucena López (2009)

Siendo S la matriz de estado, y K_i la subclave correspondiente a la ronda i–ésima, cada una de las rondas posee la siguiente estructura:

1. S ← SubBytes(S)

2. S ← DesplazarFila(S)

3. S ← MezclarColumnas(S)

4. S ← $K_i \oplus S$

La última ronda es igual a las anteriores, pero eliminando el paso 3.

Función SubBytes

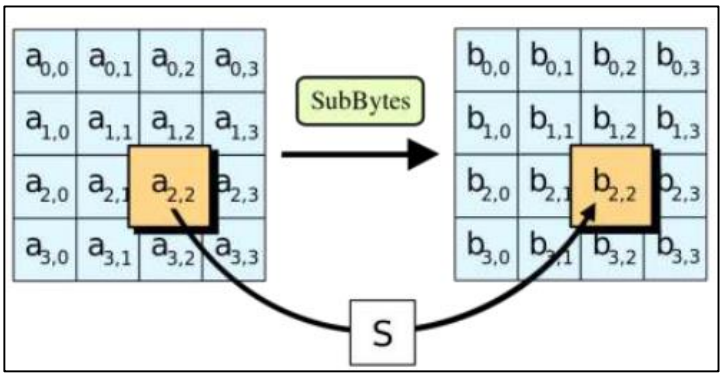


Figura 10. Función SubBytes
Fuente: Lucena López (2009)

La transformación SubBytes es una sustitución no lineal que se aplica a cada byte de la matriz de estado, mediante una s-caja 8x8 invertible, que se obtiene componiendo dos transformaciones (Ramió Aguirre, 2006):

1. Cada byte es considerado como un elemento del GF(2^8) que genera el polinomio irreducible $m(x) = x^8 + x^4 + x^3 + x + 1$, y sustituido por su inversa multiplicativa. El valor cero queda inalterado.

2. El siguiente paso consiste en aplicar la siguiente transformación afín en GF(2), siendo $x_0, x_1, \ldots, x_7$ los bits del byte correspondiente, e $y_0, y_1, \ldots, y_7$ los del resultado:

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

La función inversa de SubBytes sería la aplicación de la inversa de la s-caja correspondiente a cada byte de la matriz de estado.

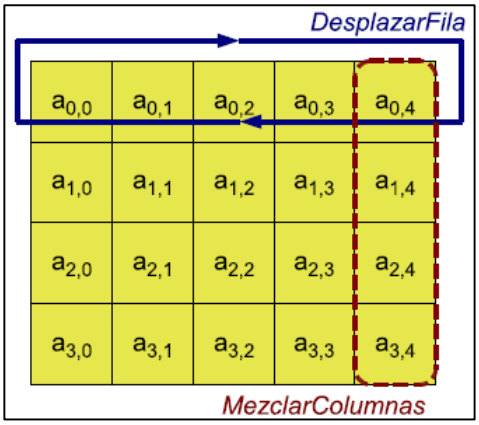


Figura 11. Esquema de las funciones DesplazarFila y MezclarColumnas de AES.
Fuente: Lucena López (2009)

Función ShiftRows (DesplazarFila)

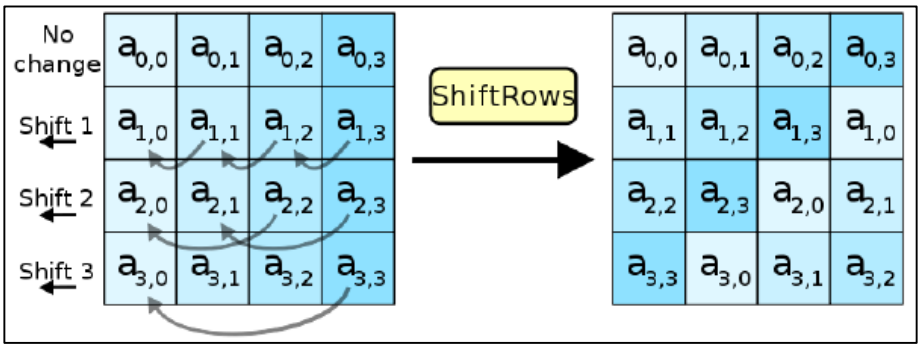


Figura 12. Función ShiftRows (DesplazarFila)
Fuente: Lucena López (2009)

Esta transformación, en la Figura 11, consiste en desplazar a la izquierda cíclicamente las filas de la matriz de estado.

Cada fila f_i se desplaza un número de posiciones c_i diferente. Mientras que c_0 siempre es igual a cero (esta fila siempre permanece inalterada), el resto de valores viene en función de N_b y se refleja en la Figura 13.

N_b	c_1	c_2	c_3
4	1	2	3
6	1	2	3
8	1	3	4

Figura 13. Valores de Ci según el tamaño de bloque Nb
Fuente: Lucena López (2009)

La función inversa de DesplazarFila será, obviamente, un desplazamiento de las filas de la matriz de estado el mismo número de posiciones que en la Figura 11, pero a la derecha. (Lucena López, 2009)

Función MixColumns (MezclarColumnas)

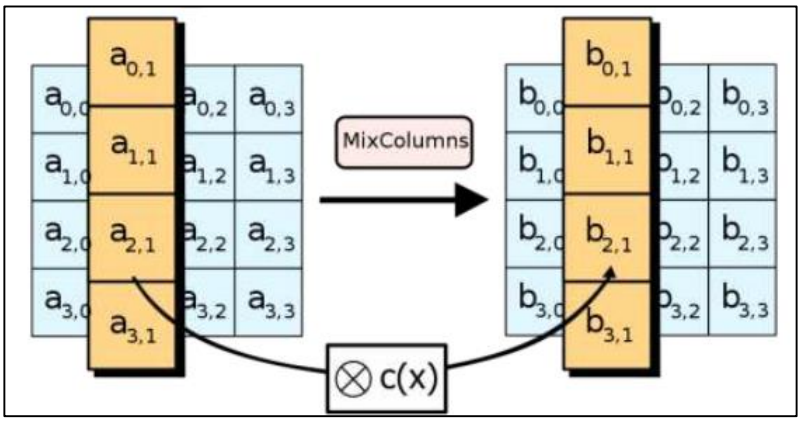


Figura 14. Función MixColumns (MezclarColumnas)
Fuente: Lucena López (2009)

Para aplicar esta función, ver figura 11, cada columna del vector de estado se considera un polinomio cuyos coeficientes pertenecen a GF(2^8), es decir, son también polinomios, y se multiplica módulo x4 + 1 por:

$c(x) = 03x^3 + 01x^2 + 01x + 02$ [15]

Donde 03 es el valor hexadecimal que se obtiene concatenando los coeficientes binarios del polinomio correspondiente en GF(2^8), en este caso 00000011, o sea, x+1, y así sucesivamente. (Lucena López, 2009)

La inversa de MezclarColumnas se obtiene multiplicando cada columna de la matriz de estado por el polinomio:

$d(x) = 0Bx^3 + 0Dx^2 + 09x + 0E$ [16]

2.2.7.2 Cálculo de las Subclaves

Las diferentes subclaves Ki se derivan de la clave principal K mediante el uso de dos funciones: una de expansión y otra de selección. Siendo n el número de rondas que se van a aplicar, la función de expansión permite obtener, a partir del valor de K, una secuencia de 4·(n+1)·Nb bytes. La selección simplemente toma consecutivamente de la secuencia obtenida bloques del mismo tamaño que la matriz de estado, y los va asignando a cada Ki. (Lucena López, 2009)

Sea K(i) un vector de bytes de tamaño 4·Nk, conteniendo la clave, y sea W(i) un vector de Nb·(n + 1) registros de 4 bytes, siendo n el número de rondas. La función de expansión tiene dos versiones, según el valor de Nk:

a) Si $N_k \leq 6$:

1. Para i desde 0 hasta N_k - 1 hacer
2. $W(i) \leftarrow K_i \quad (K(4 \cdot i), K(4 \cdot i + 1), K(4 \cdot i + 2), K(4 \cdot i + 3))$ [17]
3. Para i desde N_k hasta $N_b \cdot (n + 1)$ hacer
4. $tmp \leftarrow W(i - 1)$ [18]
5. Si i mód $N_k = 0$
6. $tmp \leftarrow Sub(Rot(tmp)) \oplus Rc(i/N_k)$ [19]
7. $W(i) \leftarrow W(i - N_k) \oplus tmp$ [20]

b) Si $N_k > 6$:

1. Para i desde 0 hasta N_k - 1 hacer

2. $W(i) \leftarrow K_i \quad (K(4 \cdot i), K(4 \cdot i + 1), K(4 \cdot i + 2), K(4 \cdot i + 3))$ [21]

3. Para i desde N_k hasta $N_b \cdot (n + 1)$ hacer

4. $tmp \leftarrow W(i - 1)$ [22]

5. Si i mód $N_k = 0$

6. $tmp \leftarrow Sub(Rot(tmp)) \oplus Rc(i/N_k)$ [23]

7. Si i mód $N_k = 4$

8. $tmp \leftarrow Sub(tmp)$ [24]

9. $W(i) \leftarrow W(i - N_k) \oplus tmp$ [25]

En los algoritmos anteriores, la función Sub devuelve el resultado de aplicar las-caja de AES a cada uno de los bytes del registro de cuatro que se le pasa como parámetro. La función Rot desplaza a la izquierda una posición los bytes del registro, de tal forma que si le pasamos como parámetro el valor (a, b, c, d) nos devuelve (b, c, d, a).

Finalmente, Rc(j) es una constante definida de la siguiente forma:

$Rc(j) = (R(j), 0, 0, 0)$

Cada R(i) es el elemento de $GF(2^8)$ correspondiente al valor x(i-1), módulo x8 + x4 + x3 + x + 1.

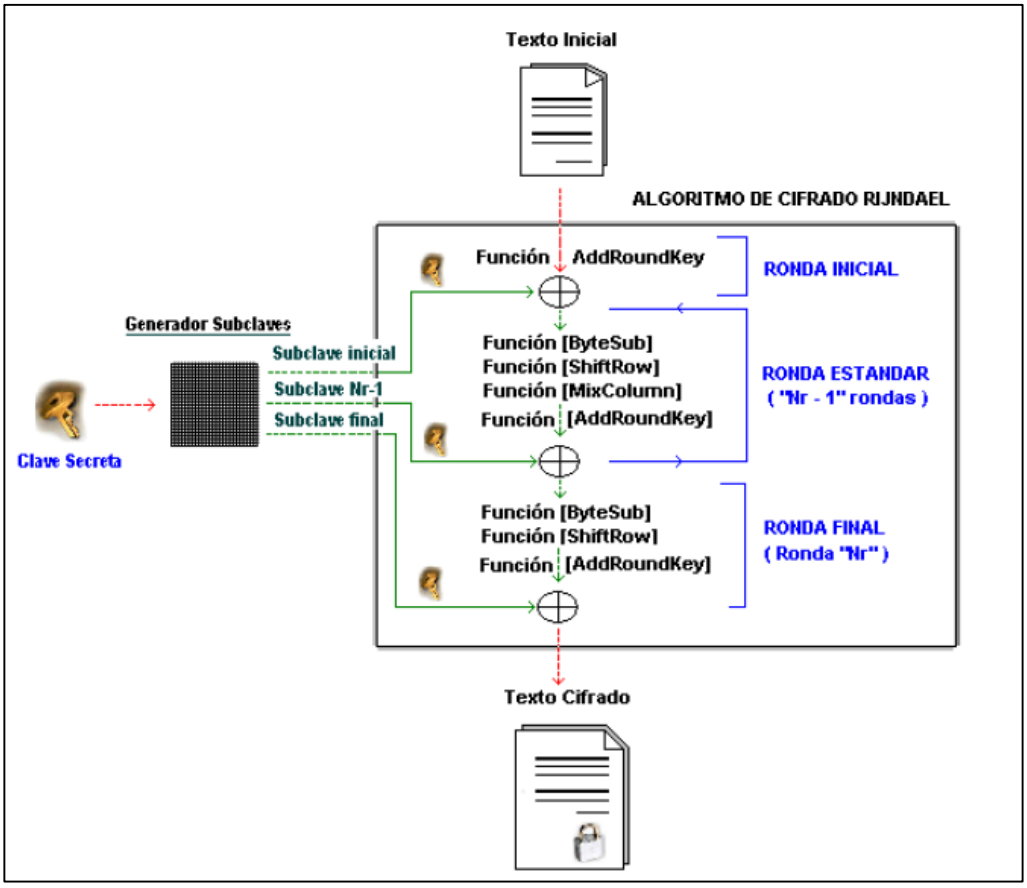


Figura 15. Algoritmo Rijndael utilizado para AES
Fuente: Daemen & Rijmen (2002)

CAPÍTULO III

MARCO METODOLÓGICO

3.1 Tipo y diseño de la investigación

Tipo

En una investigación descriptiva, debido a que se trata de un estudio donde no se manipulan intencionadamente las variables para ver su consecuencia sobre las otras variables. Es decir se observa fenómenos tal como se dan en su contexto natural, para posteriormente analizarlos. (Hernandez Sampieri, Fernández Collado, & Baptista Lucio, 2016)

Diseño

En un trabajo no experimental el investigador recaba datos sin introducir tratamiento o cambio alguno. Es decir, se trata de estudios donde no se varía en forma intencional las variables independientes para ver su efecto sobre otras variables. (Hernandez Sampieri, Fernández Collado, & Baptista Lucio, 2016)

El presente proyecto a investigar es un no experimental comparativo. No experimental debido a que se va a obtener información pero no se va a alterar las variables ni se van a dar cambios, y posteriormente se analizarán los resultados obtenidos. Comparativo porque examina diferencias en variables en dos o más

grupos que ocurren naturalmente en un escenario. En este caso las variables a comparar son el algoritmo de encriptación (que se desarrolla con un tiempo de ejecución definido) y la seguridad (que además de depender del algoritmo de encriptación, también dependerá del ataque que reciba).

3.2 Población y muestra

Población (N) se define como el conjunto general o universo de todas las personas que intervienen en la realización de la investigación, y la muestra se define como la elección sistemática de elementos representativos de la población. (Hernandez Sampieri, Fernández Collado, & Baptista Lucio, 2016)

Como estudio descriptivo, cuyo fin es analizar los algoritmos de encriptación y ver qué tan vulnerables son, se someterá a pruebas con archivos, mensajes de diferente tipo. Se ha considerado mensajes cortos, y también un total de 20 archivos, es decir, la población es N = 20 archivos

La *muestra (n)* es, en esencia, un subgrupo de la población. Es un subconjunto de elementos que, desde luego, se pretende que éste sea un reflejo fiel del conjunto de la población. (Hernandez Sampieri, Fernández Collado, & Baptista Lucio, 2016).

La muestra para el proyecto de investigación es igual a la población en estudio, es decir, n = N = 20 archivos. En consecuencia, dado que la muestra es igual a la población, el muestreo vendría a ser del total de la población considerada.

3.3 Operacionalización de variables

3.3.1 Identificación de variables

a. Variable Independiente

Algoritmo de encriptación

b. Variable Dependiente

Seguridad de la transmisión de información

3.3.2 Definición de las variables

a. Algoritmo de encriptación

Por la función que cumplen: Variable independiente

Por su naturaleza: Atributiva

Por el método de estudio: Cualitativo

b. Seguridad de la transmisión de la información

Por la función que cumplen: Variable dependiente

Por su naturaleza: Activa

Por el método de estudio: Cualitativo

Tabla 2

Operacionalización de variables

Variable	Definición Conceptual	Definición Operacional	Indicadores
Independiente Algoritmo de encriptación	Algoritmo que modifica los datos de un documento con el objetivo de alcanzar algunas características de seguridad, los mismos que constan de datos de entrada, algoritmos de cifrado, algoritmo de des-cifrado y conceptos matemáticos para su comprensión.	Eficiencia del Algoritmo, que se medirá de acuerdo al tiempo de ejecución y la cantidad de recursos que utiliza la computadora para realizarlo.	Tiempo de ejecución.
Dependiente Seguridad de la transmisión de información	Nivel de seguridad con el que la información es transmitida como privacidad, integridad, autenticidad.	Se medirá teniendo en cuenta los indicadores que se usan para proteger la información. Teniendo en cuenta las herramientas a utilizar en los ataques, se podrá medir si es vulnerable.	Privacidad, integridad.

Fuente: Elaboración propia

3.4 Técnicas e instrumentos para la recolección de datos

Técnica

Debido a que la presente tesis es descriptiva, y tomando en cuenta que analizaremos el cifrado, descifrado y ataques, se utilizará la técnica de observación.

La observación es una técnica que consiste en visualizar, en forma sistemática, cualquier hecho, fenómeno o situación que se produzca en la naturaleza o en la sociedad, en función de unos objetivos de investigación preestablecidos. (Arias, 2012)

Instrumento

Un instrumento de recolección de datos es cualquier recurso, dispositivo o formato (en papel o digital), que se utiliza para obtener, registrar o almacenar información; como una computadora por ejemplo. (Arias, 2012)

Por lo tanto, para recabar la información necesaria el instrumento a utilizar será una computadora la cual mediante programas se procesarán los archivos para posteriormente utilizar tablas donde se colocarán los tiempos que demore una actividad en específica, como puede ser el cifrado, y también los ataques. Para un mejor análisis, los gráficos de dichas tablas permitirán observar de una forma más clara.

3.5 Procesamiento y análisis de datos

Se utilizará herramientas de prueba de cifrado, que nos permitirá cifrar, descifrar, y atacar. AESphere y genRSA son los programas a utilizar. También se hará uso de OpenSSL. Consiste en un robusto paquete de herramientas de administración y bibliotecas relacionadas con la criptografía, que suministran funciones criptográficas a otros paquetes como OpenSSH y navegadores web (para acceso seguro a sitios HTTPS). Se utilizará OpenSSL en Ubuntu, para ello se utilizará una máquina virtual. Para las tablas y gráficos se hará uso de las hojas de cálculo de Microsoft Excel.

CAPÍTULO IV

DESARROLLO METODOLÓGICO DE LA INVESTIGACIÓN

4.1 Encriptación

El algoritmo AES tiene un tamaño fijo de bloque de 128 bits a la hora de cifrar. El tamaño de la clave, sin embargo, puede ser de 128, 192 ó 256 bits.

El proceso de cifrado consta de un número variable de rondas que depende exclusivamente del tamaño de la clave, 10 rondas en caso de que la clave sea de 128 bits, 12 rondas para claves de 192 y por último 14 rondas para las de 256.

AES opera sobre una matriz de 4x4 bytes, denominada state sobre la que va realizando las modificaciones pertinentes.

El esquema del proceso es el siguiente:

1 - Expansión de la clave usando el esquema de claves de Rijndael.

2 - Etapa inicial: AddRoundKey

3 - Rondas:

SubBytes - Se realiza una sustitución no lineal donde cada byte es reemplazado con otro de acuerdo a una tabla de búsqueda.

ShiftRows - En este paso se realiza una transposición donde cada fila del state es rotada de manera cíclica un número determinado de veces.

MixColumns - operación de mezclado que opera en las columnas del state, combinando los cuatro bytes en cada columna usando una transformación lineal.

AddRoundKey - cada byte del state es combinado con la clave de ronda; cada clave de ronda se deriva de la clave de cifrado usando una iteración de la clave.

4 - Etapa final, en la que se realizan las siguientes operaciones:

SubBytes

ShiftRows

AddRoundKey

El bloque 3 correspondiente a las rondas es el que se repite dependiendo del tamaño de la clave. Para claves de 128 bits, donde el número de rondas es 10, este bloque se repitiría un total de 9 veces que más la ronda final (en la que no se realiza el proceso de MixColumns) darían las 10 rondas de las que hablábamos. Para tamaños de clave superiores, el recorrido sería análogo.

A modo de prueba, cifraremos el mensaje: «Texto plano a cifrar.»

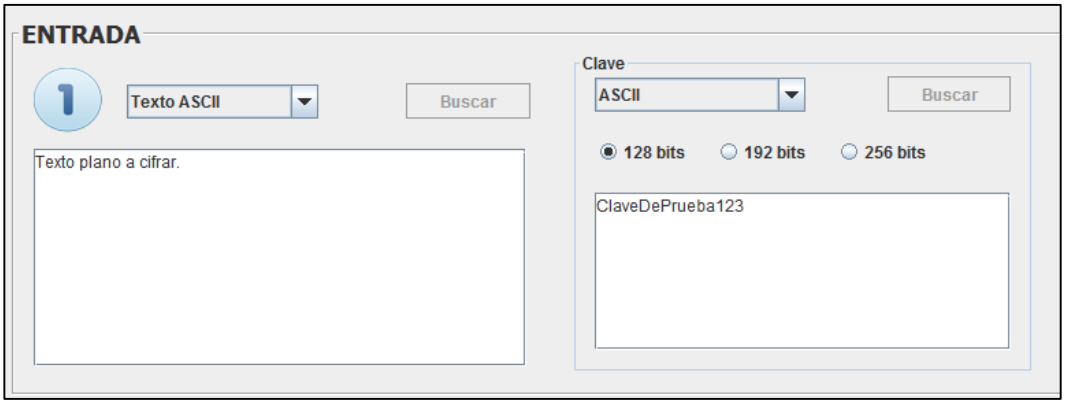


Figura 16. Cifrado AES 128 bits
Fuente: Elaboración propia

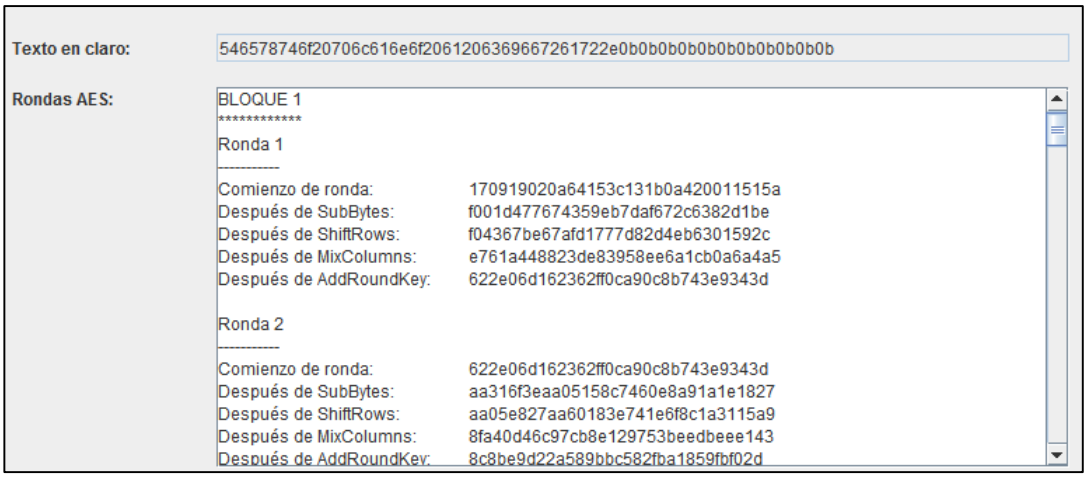


Figura 17. Ejecutándose Rondas AES
Fuente: Elaboración propia

Todo el proceso de las Rondas AES para este ejemplo se podrá ver en el Anexo 02. AES no solamente nos permite cifrar mensajes, sino también archivos tales como pdf, videos, etc. En el Anexo 03 hay ejemplos de cifrado de estos archivos utilizando OpenSSL.

4.2 Ataques

AES usa la tabla de sustitución de la caja S, que se genera al determinar el inverso multiplicativo para un número dado en el campo finito de Galois y tiene la capacidad de resistir el criptoanálisis diferencial de líneas. Recientemente, Warren D. Smith definió un método analítico y afirmó que existe la posibilidad de que exista un algoritmo de craqueo que pueda extraer la clave AES de 256 bits de cualquier par de texto cifrado texto randomplain. Sin embargo, fue solo como una adición y no ha proporcionado ningún algoritmo de cracking. Como resultado, hasta ahora, el algoritmo AES no tiene ninguna propiedad matemática que pueda ser explotada por un atacante para reducir la longitud de la clave efectiva y para ganar exito contra AES (Smith, 2007)

Así que, se procederá a realizar ataque por fuerza bruta, utilizando AESphere, en modo local a un espacio delimitado de claves del AES. La longitud mínima de la clave de cifrado AES es de 128 bits, que proporciona 2^{128} claves posibles.

En este caso a modo de prueba, la clave es “ProbarClave12345” y se va a poner un rango de claves para que el sistema pueda encontrarlo. El rango va desde “ProbarClave11111” hasta “ProbarClave13333”.

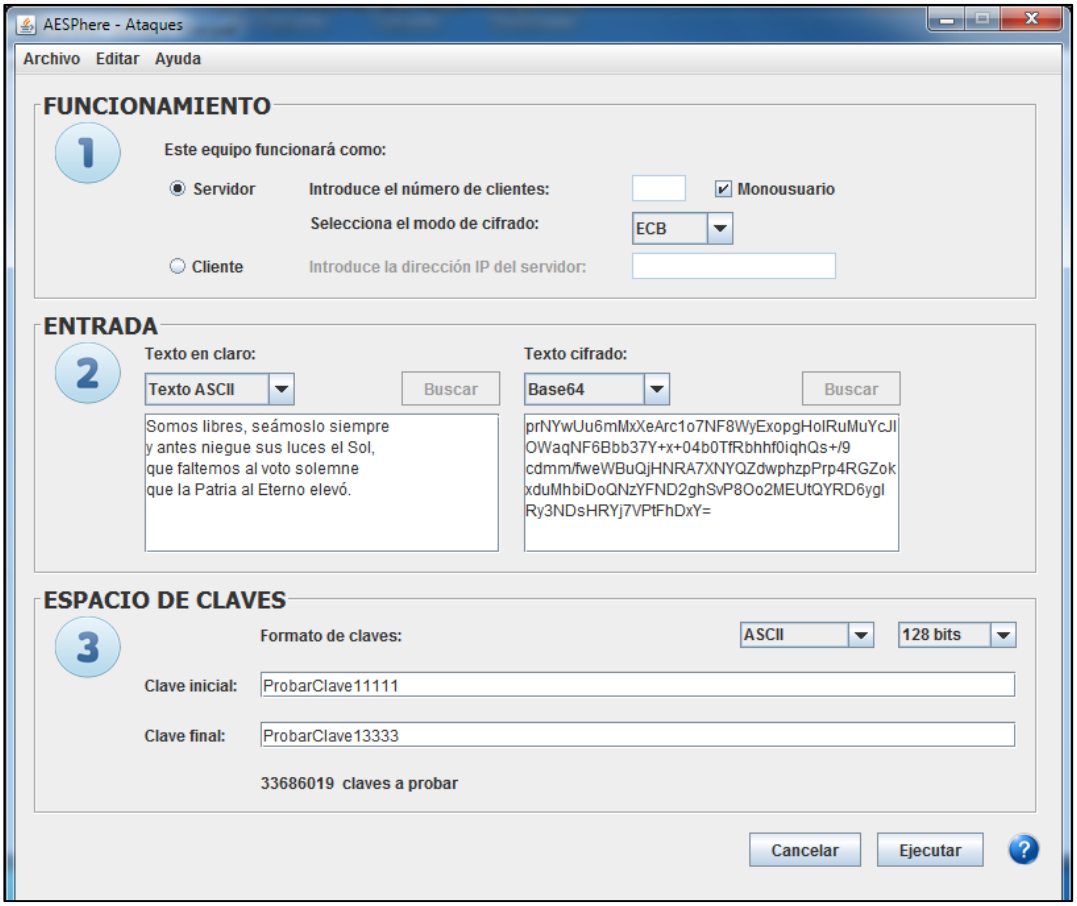


Figura 18. Ataque AES
Fuente: Elaboración propia

Es decir, va a tener un total de 33 686 019 claves para probar

Para atacar el generado de claves RSA, se utiliza el programa genRSA

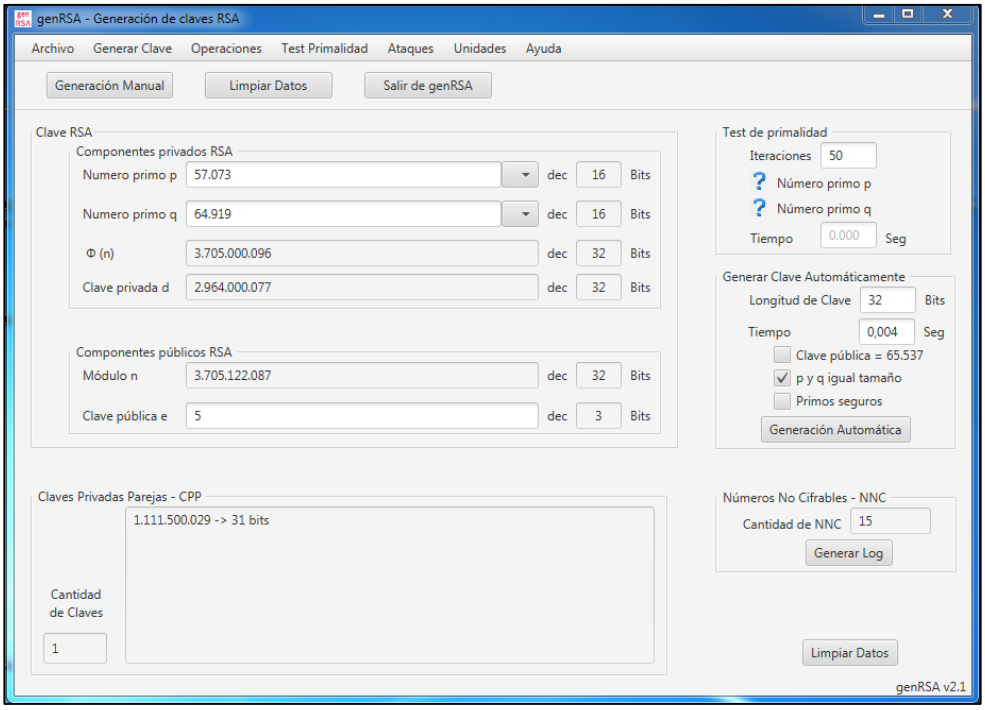


Figura 19. Generar clave RSA
Fuente: Elaboración propia

Y en el caso de los ataques, nos permite atacarlo de tres maneras diferentes: Por factorización, ataque cíclico y ataque por paradoja de cumpleaños.

Ataque por Factorización

El problema de la factorización entera es uno de esos problemas en matemáticas clasificados como abiertos; es decir, siempre puede aparecer un nuevo algoritmo que mejore las prestaciones de los anteriores o bien que presente ciertas ventajas frente a aquellos en casos especiales. El utilizado para este caso, es el algoritmo Rho de Pollard.

Figura 20. Ataque por Factorización
Fuente: Elaboración propia

Ataque Cíclico

Para realizar este ataque, lo único que se necesita tener es el mensaje cifrado. Es decir, no será necesario ni siquiera tener acceso a la clave pública.

Como C = Ne mod n, con N un valor secreto, se realiza cifrados sucesivos de los criptogramas Ci resultantes con la misma clave pública e. Si en uno de estos cifrados obtenemos nuevamente el cifrado C original con el que se ha iniciado el ataque, resulta obvio que el valor del paso anterior será el secreto N buscado.

Esto se debe a que RSA es un grupo mutiplicativo. Para dificultar este tipo de ataques, es interesante usar primos seguros de forma que los subgrupos de trabajo sean lo suficientemente altos.

Por tanto, conocido o capturado el criptograma C, realizaremos los siguientes cifrados: Ci = Cei-1 mod n; para i = 1, 2, ..., con C0 = C

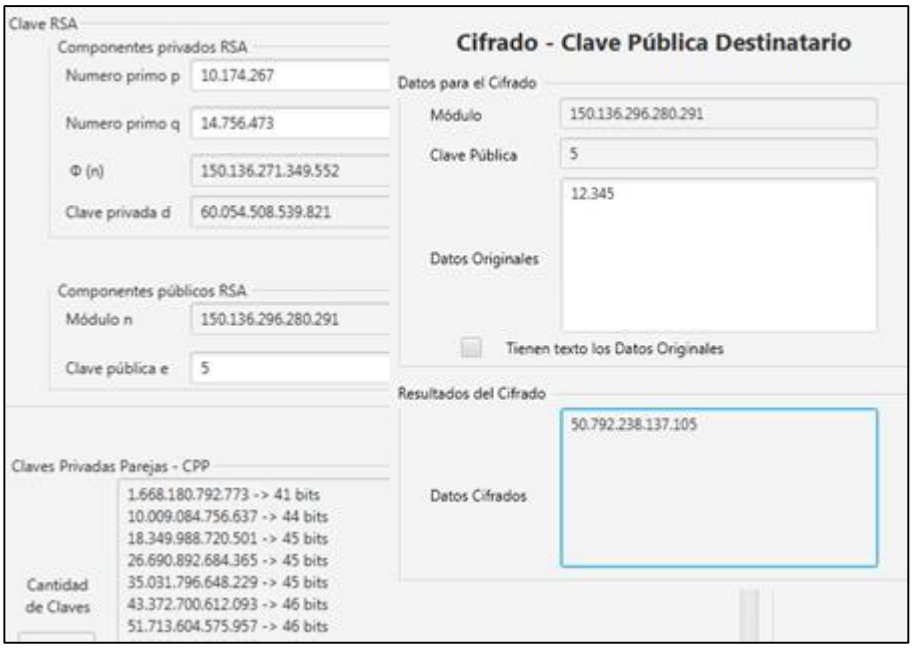


Figura 21. Ataque Cíclico
Fuente: Elaboración propia

Ataque por Paradoja de Cumpleaños

Se elige un valor o número N de ataque aleatorio cualquiera y, conocido el módulo n de la víctima que es un dato público, dividimos ese módulo en dos mitades; la mitad izquierda o baja del módulo y la mitad derecha o alta del módulo, y con dichos valores que supondremos la clave k a la que se cifra el valor de ataque N se realiza la operación de cifra Nk mod n en la zona baja y alta del módulo hasta que se encuentre un resultado igual o colisión, momento en el cual este algoritmo llega a su fin.

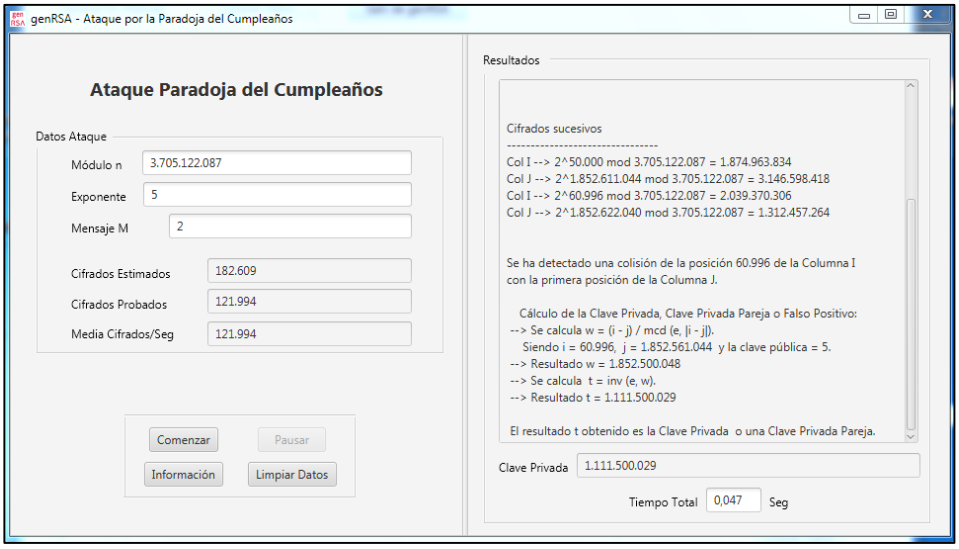


Figura 22. Ataque Paradoja de Cumpleaños
Fuente: Elaboración propia

CAPÍTULO V

ANÁLISIS E INTEPRETACIÓN DE RESULTADOS

5.1 Presentación de resultados obtenidos

5.1.1 Encriptación

El mensaje cifrado con AES nos da como resultado lo que se observa en la imagen.

Para mayor detalle de las rondas, ver Anexo 02

Figura 23. Mensaje cifrado AES
Fuente: Elaboración propia

AES es un sistema de cifrado simétrico, lo cual significa que se utiliza la misma clave para cifrar y para descifrar. AES basa toda su seguridad en la clave, ya que el algoritmo de cifrado/descifrado es conocido. El principal problema de seguridad que tiene es el intercambio de esta clave, ya que para un posible atacante es más fácil intentar interceptar esta clave que averiguar la misma.

Pasemos a explicar el proceso de descifrado de AES. Al igual que a la hora de cifrar, AES opera sobre una matriz de 4x4 bytes llamda state. En cuanto a la clave, ésta puede ser de 128, 192 o 256 bits. Este tamaño de clave marcará el número de rondas que se harán durante el desarrollo del descifrado: 10, 12 o 14.

El proceso a realizar sería el siguiente:

1 - Expansión de la clave. Se usa el esquema de claves de Rijndael.

2 - Etapa inicial:

AddRoundKey

3 - Rondas:

InvShiftRows - Transposición circular de los bytes de las filas.

InvSubBytes - Se aplica una sustitución inversa mediante cajas-S a cada byte del state

AddRoundKey - cada byte del state es combinado con la clave de ronda, obtenida esta mediante una operación XOR.

InvMixColumns - Complejo mezclado de columnas usando una transformación lineal sobre las columnas.

4 - Etapa final, en la que se realizan las siguientes operaciones:

InvShiftRows

InvSubBytes

AddRoundKey

Las fases de la tercera ronda son las que se repiten dependiendo del número de bits de la clave.

Figura 24. Rondas Descifrado AES
Fuente: Elaboración propia

No solamente se puede cifrar mensajes, sino también archivos. Por ejemplo encriptamos un documento pdf de prueba, de 4,80 MB

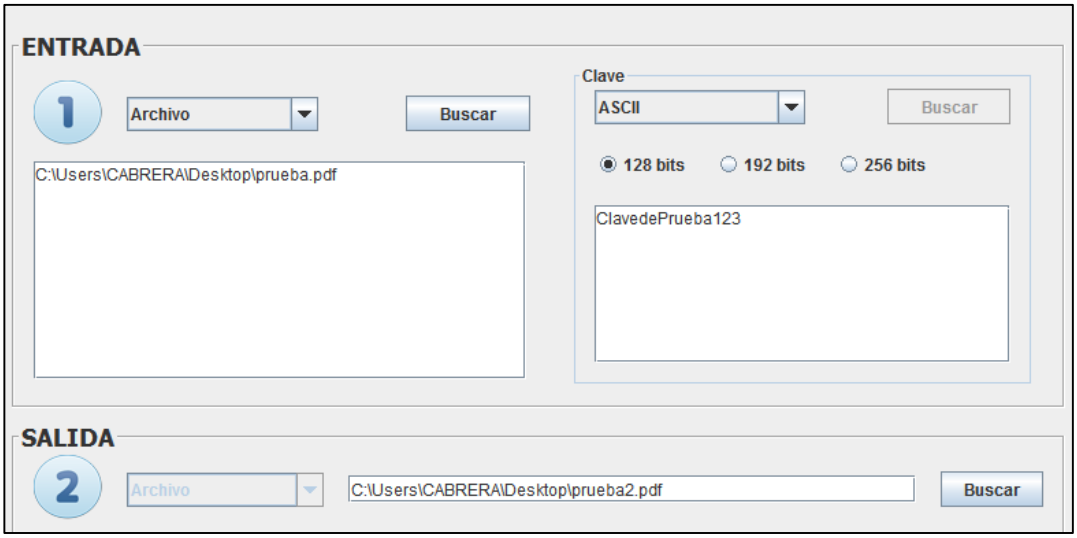


Figura 25. Cifrado AES de archivo
Fuente: Elaboración propia

El archivo se comprueba que efectivamente no se puede abrir, y se procede a descifrarlo con la clave que le hemos dado, para nuevamente poder abrirlo.

De la misma forma, se cifra y descifra con OpenSSL (Detallado en Anexo 03)

```
jhonatan@jhonatan-VirtualBox: ~/Escritorio
jhonatan@jhonatan-VirtualBox:~/Escritorio$ openssl aes-256-cbc -a -salt -in prueba.pdf
 -out cifrapdf.txt
enter aes-256-cbc encryption password:
Verifying - enter aes-256-cbc encryption password:
jhonatan@jhonatan-VirtualBox:~/Escritorio$ openssl aes-256-cbc -d -a -in cifrapdf.txt
-out descifrado.pdf
enter aes-256-cbc decryption password:
```

Figura 26. Cifrado AES en OpenSSL
Fuente: Elaboración propia

Tabla 3

Tabla de cifrado de archivos

ARCHIVO	Tamaño	Tiempo
Mensaje1	21 bytes	menos de 0,005s
Mensaje2	10 bytes	menos de 0,005s
Mensaje3	58 bytes	menos de 0,005s
Mensaje4	126 bytes	menos de 0,005s
Mensaje5	5 bytes	menos de 0,005s
archivo.txt	1,24 KB	menos de 0,005s
archivo2.txt	2,38 KB	menos de 0,005s
archivo3.txt	8,62 KB	menos de 0,005s
archivo4.txt	8,69 KB	menos de 0,005s
archivo5.txt	17,1 KB	menos de 0,005s
Doc1	665 KB	0,08s
Doc2	24,8 MB	3,19s
imagen.jpg	35 KB	0,003s
imagen2.jpg	181 KB	0,02s
imagen3.png	5,35 MB	0,71s
prueba.pdf	4,80 MB	0,61s
prueba2.pdf	6,57 MB	0,8s
prueba3.pdf	10,5 MB	1,35s
videoprueba.mp4	53,9 MB	6,93s
videoprueba2.avi	746 MB	96,65s

Fuente: Elaboración propia

En la Tabla 3 se observa el tiempo que se ha demorado en cifrar los archivos con AES (clave de 256 bits).

El algoritmo AES acepta tres tamaños de clave: 128, 196 y 256 bits. En OpenSSL se puede saber a qué velocidad trabaja, y eso es lo que se ha evaluado. El tiempo de procesamiento es aproximadamente proporcional a la longitud de entrada, y también teniendo en cuenta los recursos de la computadora.

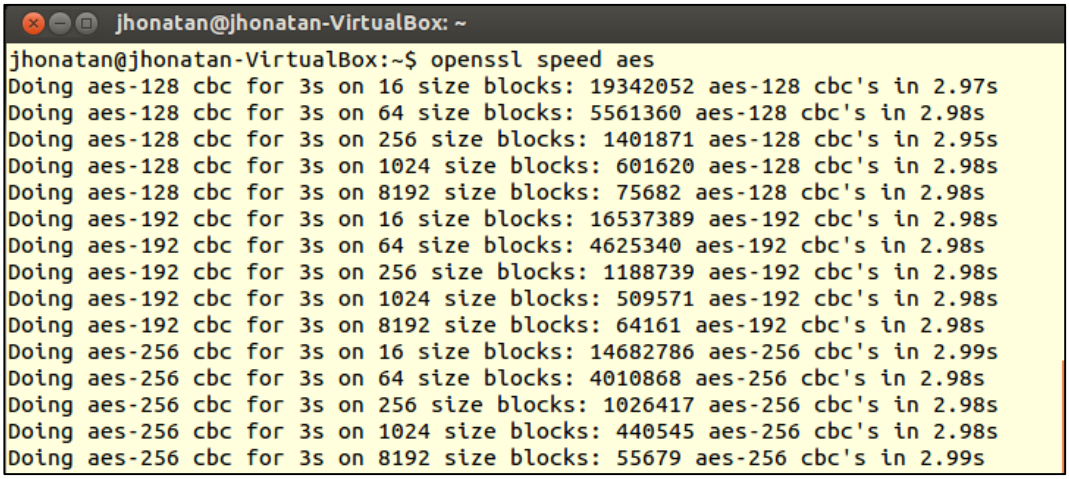


Figura 27. Openssl speed AES
Fuente: Elaboración propia

Para entender, por ejemplo tomamos la parte que dice "`Doing aes-128 cbc for 3s on 16 size blocks: 19 342 052 aes-128 cbc's in 2.97s`", que indica que utilizó AES con clave de 128 bits en mensajes de 16 bytes, y lo hizo de nuevo, y nuevamente, hasta un total de 19 342 052 veces en un marco de tiempo de 3 segundos. Esto implica un ancho de banda de cifrado de aproximadamente 103,15 MB/s (16 por 19 342 052, dividido por 3). Por otro lado, dice lo que sucede cuando OpenSSL con aes-128 encripta (repetidamente) mensajes de 8 192 bytes: puede hacerlo 55 679 veces en 3 segundos, para un ancho de banda total de hash de 152,04 MB/s. Es decir, la primera línea ilustra el costo de cálculo cuando se trata de mensajes pequeños, mientras que la última línea está, cerca del ancho de banda que se puede lograr al procesar una entrada muy larga.

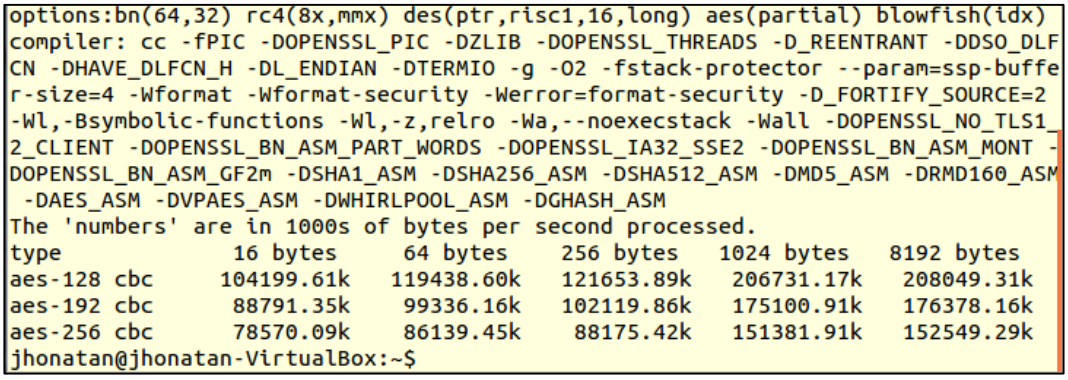


Figura 28. Tasa de cifrado estimado AES en un bucle de 3 segundos
Fuente: Elaboración propia

Como se puede observar en la figura 28, muestra la velocidad estimada realizada por AES, dependiendo del tamaño del archivo a cifrar. En la tabla 4 se ha procedido a colocar los valores obtenidos.

Tabla 4

Tasa de cifrado estimado AES en un bucle de 3s

	16 bytes	64 bytes	256 bytes	1024 bytes	8192 bytes
AES 128	104 MB/s	119 MB/s	121 MB/s	206 MB/s	208 MB/s
AES 192	88 MB/s	99 MB/s	102 MB/s	175 MB/s	176 MB/s
AES 256	78 MB/s	86 MB/s	88 MB/s	151 MB/s	152 MB/s

Fuente: Elaboración propia

Comparado esto con los tipo asimétricos, en este caso, RSA, hay una diferencia sustancial. La diferencia es que mientras que los cifrados simétricos tienen una tasa de cifra del orden de las centenas de megabytes por segundo, los cifrados asimétricos tienen una tasa de cifra de centenas de kilobytes por segundo. Es decir, diferencia de 1 a 1 000.

Por ese motivo RSA, se utiliza para cosas “lights”, como para el intercambio de claves y la firma digital. Al querer encriptar ya un archivo más pesado en RSA, ocurre un error. RSA nos dice: “data too large for key size”

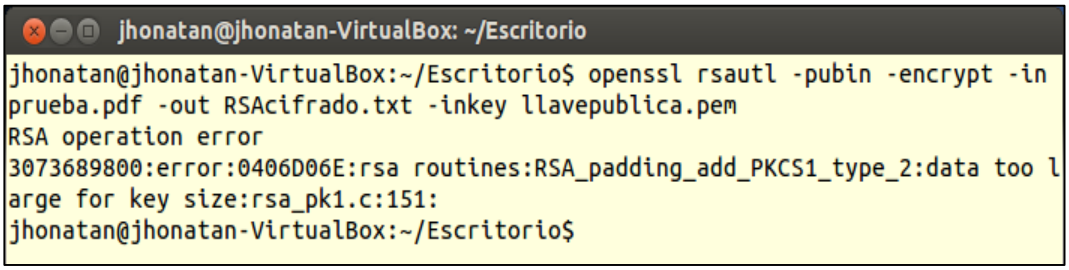


Figura 29. Cifrado RSA en OpenSSL
Fuente: Elaboración propia

Esto se debe a que RSA como tal, no encripta documentos. Es por ello que en criptografía se suele hacer una “mezcla” de cifrado simétrico y asimétrico. Es decir, RSA nos sirve para generar las claves, asegurando que nadie más que el usuario la tenga, y ya posteriormente el mensaje a cifrar se realiza con AES

De manera similar que obtuvimos la velocidad de AES, con OpenSSL también se puede ver de RSA, lo cual nos arrojan los datos que se describen en la siguiente tabla.

Tabla 5

Openssl speed RSA

	sign	verify	sign/s	verify/s
RSA 512	0,000145s	0,000012s	6888,6	86119,8
RSA 1024	0,000800s	0,000037s	1249,2	27321,2
RSA 2048	0,005242s	0,000142s	190,8	7054,0
RSA 4096	0,037378s	0,000524s	26,8	1906,7

Fuente: Elaboración propia

Firmar (sign) y verificar (verify) hacen precisamente lo que dicen. Programan una operación de firma y una operación de verificación con diferentes módulos RSA. Sign/s y Verify/s son las inversiones de Sign and Verify. Es decir, por ejemplo, 1/0,000145s => 6 888,6 firmas por segundo.

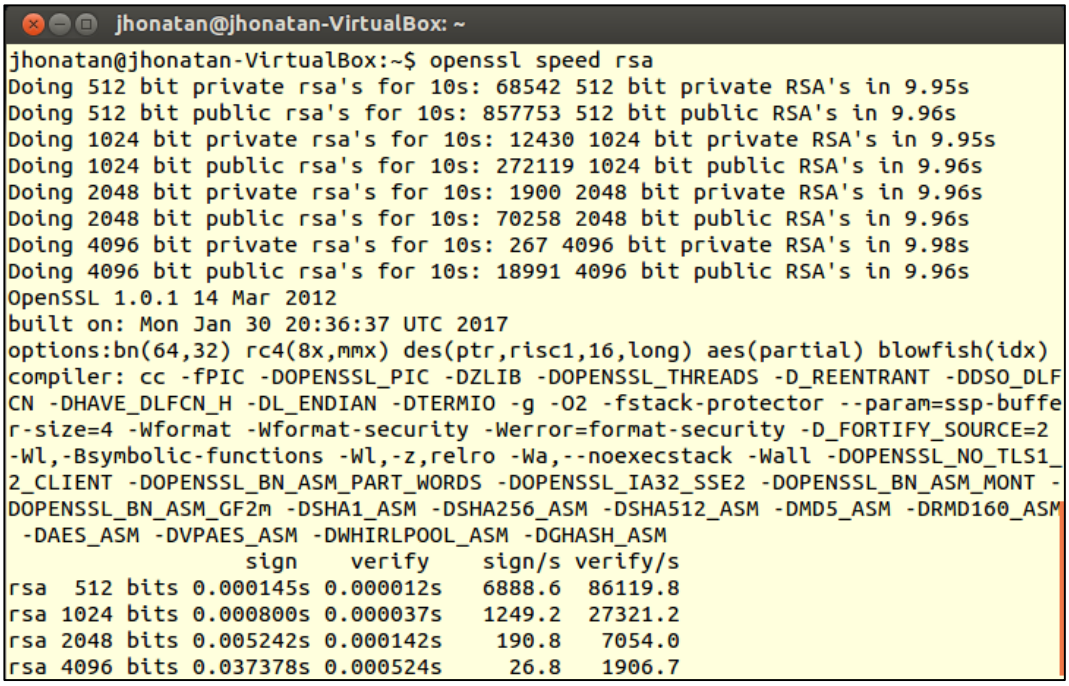


Figura 30. Openssl speed RSA
Fuente: Elaboración propia

5.1.2 Ataques

Los algoritmos criptográficos emplean claves con un elevado número de bits, y usualmente se mide su calidad por la cantidad de esfuerzo que se necesita para romperlos. En los ataques la vulnerabilidad no se mide en cómo está estructurado el algoritmo, sino en la clave. Cuando uno desea romper la seguridad del archivo

cifrado, no se ataca al archivo cifrado, sino a la clave que guarda, dicho de otra forma, no se ataca al algoritmo, sino a la clave. El tipo de ataque más simple es la fuerza bruta, que simplemente trata de ir probando una a una todas las claves.

Para el ataque de fuerza bruta utilizada para romper AES, se logró encontrar la clave "ProbarClave12345" (Aunque lo encontró en hexadecimal, pero si esa clave lo transformamos a ASCII nos daremos cuenta que se trata de nuestra clave)

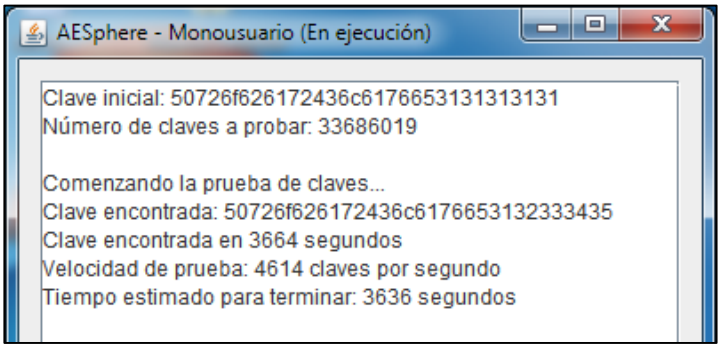


Figura 31. Ataque por fuerza bruta a AES
Fuente: Elaboración propia

El tiempo que demoró en encontrar la clave dentro de 33 686 019 claves posibles fue de 3664 segundos (1 hora con 1 minuto y 4 segundos). Esto, claro está, es a modo de prueba, ya habiéndole dado una 'ayuda' de ponerle un rango a la posible clave.

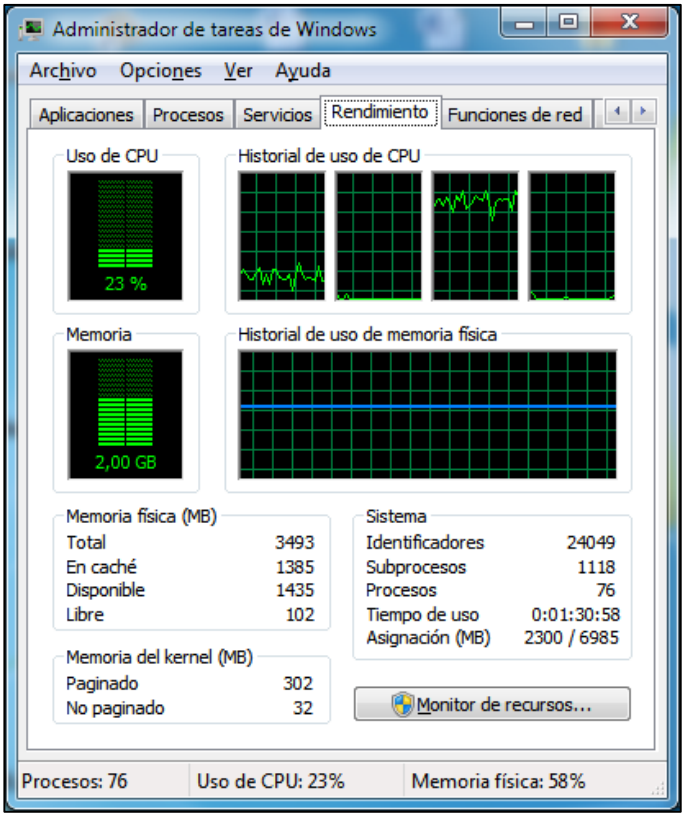


Figura 32. Rendimiento PC ataque fuerza bruta
Fuente: Elaboración propia

Recordemos que un millón es aproximadamente 2^{20}, y que un año tiene más o menos 2^{25} segundos. Si dispusiéramos de un computador capaz de hacer un millón de operaciones por segundo, recorrer completamente un espacio de claves de, por ejemplo, 256 bits a razón de un millón por segundo suponen $2^{256-45} = 2^{211}$ años de cálculo.

Ahora se verá cuáles fueron los resultados de los ataques a RSA, para tomar en cuenta los tiempos en que se tardaron en encontrar la clave, según el número de

bits de N. En la Tabla 6 se pondrá en evidencia la importancia del tamaño de la clave N, y por consecuencia, el tamaño de los números primos p y q.

Tabla 6

Tiempo que tarda en romper RSA por factorización

Tamaño de clave (bits)	Tiempo (segundos)
48	0,172
56	0,188
64	0,273
72	1,605
80	5,266
88	14,542
96	19,761
104	651,344
112	3642,354
120	10201,122
128	18569,685

Fuente: Elaboración propia

Al comienzo, con una prueba una clave de 48 bits, solo toma 0,172 segundos poder romper el algoritmo RSA mediante la factorización; asimismo conforme a que el tamaño de la clave aumente, el tiempo también irá aumentando.

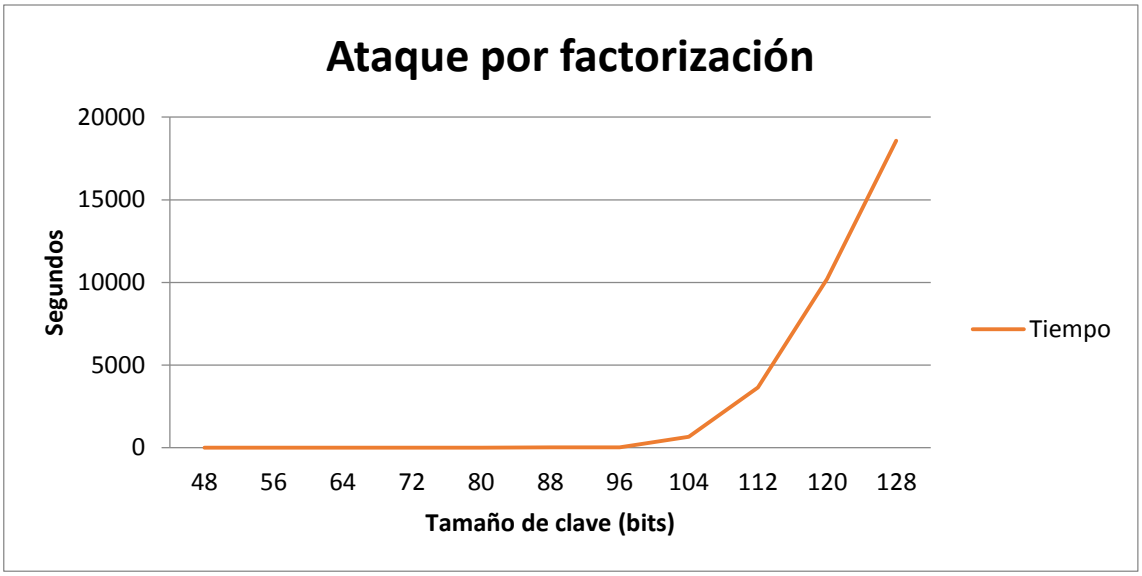


Figura 33. Gráfico del comportamiento del Ataque por Factorización
Fuente: Elaboración propia

Conforme los dígitos del número N vaya aumentando, el computador se le va a hacer más difícil factorizarlo. Entendiendo que N = pxq, es por ello que la seguridad de clave de RSA es más fuerte mientras más grandes sean los números primos p y q.

Para lograr factorizar la clave de 128 bits demoró un total de 18 569,685 segundos, que equivale a un poco más de 5 horas.

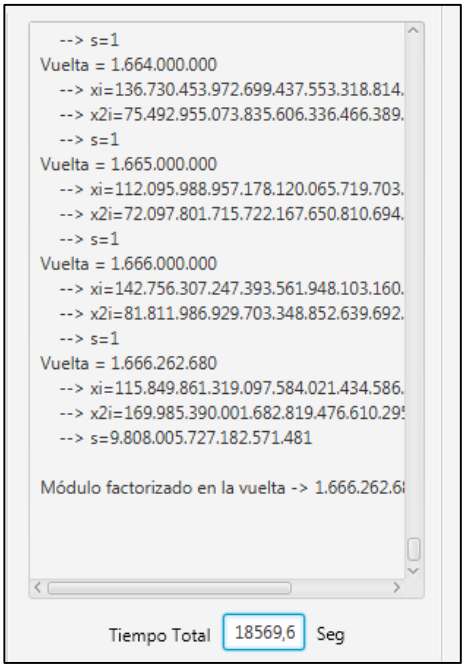


Figura 34. Resultado Ataque por factorización – clave 128 bits
Fuente: Elaboración propia

En el ataque cíclico, la Tabla 7 muestra el tiempo que tarda para poder hallar la clave. Considerando que la clave es de 32 bits, por ejemplo, solamente tarda 0,042 segundos. Sin embargo, si es que la clave es de 64 bits, es decir, solo el doble, el tiempo aumenta considerablemente.

Tabla 7

Tiempo que tarda en romper RSA por ataque cíclico

Tamaño de clave	Tiempo (segundos)
32	0,042
36	0,334
40	6,754
44	106,657
48	235,292
52	1089,99
56	14561,2

Fuente: Elaboración propia

Figura 35. Gráfico del comportamiento del Ataque Cíclico
Fuente: Elaboración propia

De igual manera se puede observar la importancia del tamaño del módulo N. En la gráfica se observa que tiene un comportamiento exponencial a medida que el tamaño del módulo va aumentando, mientras que para una clave de 32 bits

tardó en romper en 0,042 segundos, para una clave de 56 bits tardó un poco más de 4 horas.

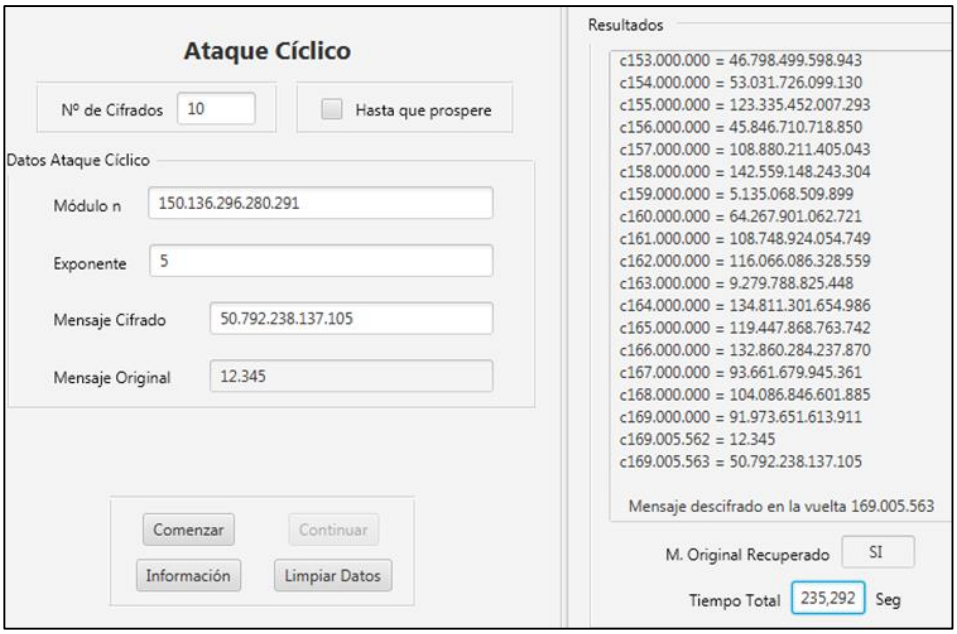


Figura 36. Resultado Ataque Cíclico - clave 48 bits
Fuente: Elaboración propia

En el ataque por paradoja de cumpleaños, se pueden observar los resultados en la Tabla 8 hasta el tamaño de clave de 72 bits.

Tabla 8

Tiempo que tarda en romper RSA por paradoja de cumpleaños

Tamaño de clave (bits)	Tiempo (segundos)
48	8,154
52	30,896
56	105,985
60	435,905
64	1701,87
68	9887,673
72	32310,4

Fuente: Elaboración propia

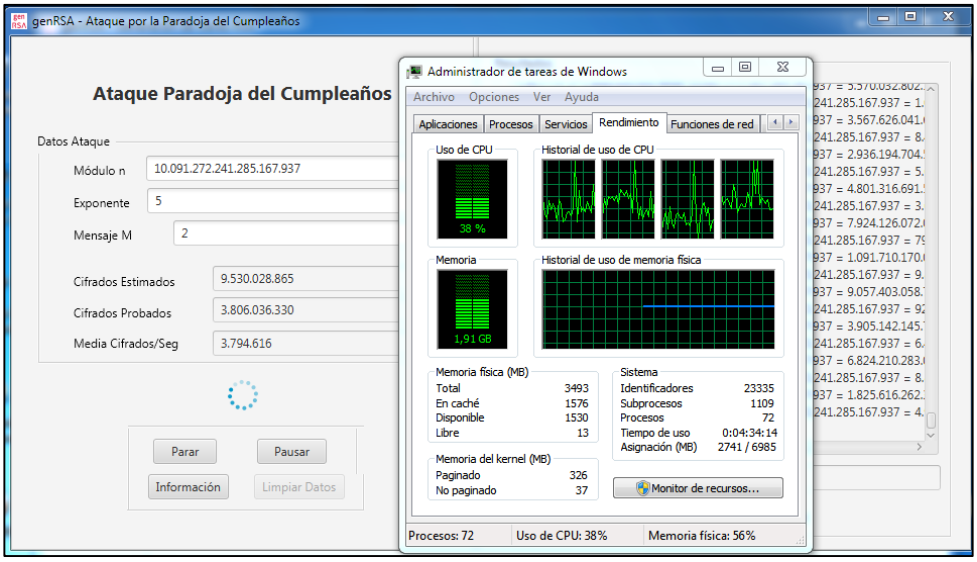


Figura 37. Rendimiento PC Ataque Paradoja de Cumpleaños
Fuente: Elaboración propia

Viendo que para una clave de 72 bits tardó 32 310,4 segundos (8 horas) se decidió dejar hasta ahí el cálculo y no seguir probando con tamaños de clave más grandes, debido a que lógicamente el tiempo iba a aumentar considerablemente. Es importante también ver el trabajo que realizaba la máquina, ya que como se ve en la imagen, el uso de CPU estaba aumentando, a diferencia que cuando no se estaba ejecutando el uso era solo de 1 %

A continuación un gráfico que muestra el aumento del tiempo conforme va aumentando el tamaño de la clave.

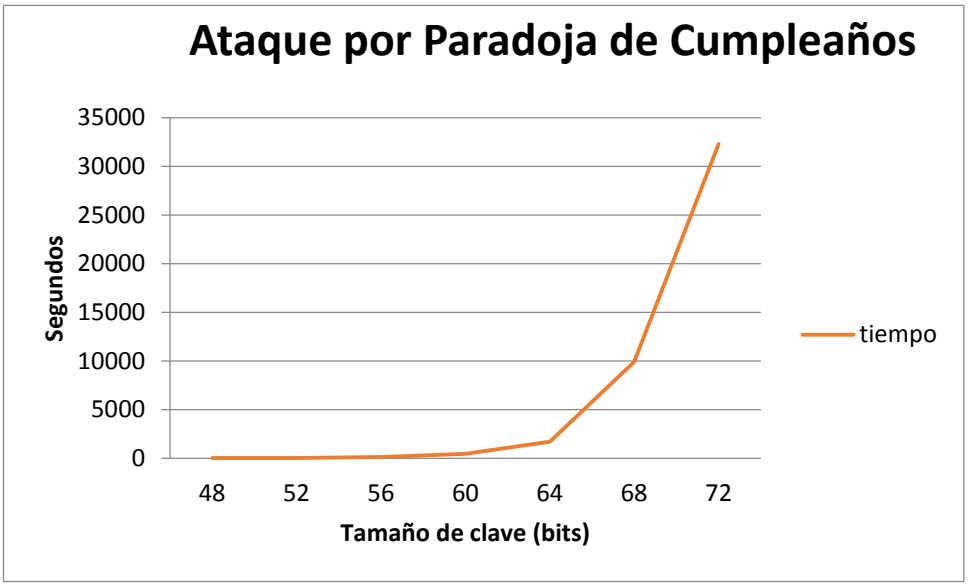


Figura 38. Gráfico del comportamiento del Ataque por Paradoja de Cumpleaños
Fuente: Elaboración propia

De los tres ataques vistos, se puede decir que el ataque por factorización ha permitido hacer pruebas a claves de hasta 128 bits (aproximadamente 5 horas), el ataque cíclico ha permitido a claves de hasta 56 bits (aproximadamente 4 horas) y el ataque por paradoja de cumpleaños a claves de hasta 72bits (aproximadamente 8 horas). Se puede seguir probando en claves más grandes, pero eso claro está tomaría mucho tiempo más.

Habiendo realizado el estudio comparativo, los resultados nos dan a conocer algunas diferencias que existen. RSA implica operaciones exponenciales, mientras que AES utiliza sustitución, rotación, adición y otras operaciones relativamente más simples. No se puede analizar concretamente en forma directa ya que ambos tienen un propósito diferente.

AES como es un algoritmo basado en clave privada, sufre problemas de distribución de claves, es decir, la forma en cómo avisarle al receptor la clave utilizada para cifrar. Sin embargo, este problema se supera en el algoritmo RSA, puesto que la clave para descifrar no la sabe el emisor, y eso no es impedimento para que el emisor cifre el mensaje con su clave pública; pero el cifrado y el descifrado en RSA requieren más tiempo que en AES. Entonces ambos algoritmos tienen sus propias méritos y deméritos.

5.2 Discusiones

Mahajan & Sachdeva (2013) midieron el rendimiento de las técnicas de encriptación existentes, AES, DES y RSA. En base a los archivos de texto utilizados y resultado experimental se concluyó que el algoritmo AES consume menos cifrado y RSA consume más tiempo tiempo de encriptación. Es decir, así como en este estudio se ha visto, que el tiempo de encriptación de AES es menor que RSA.

Hercigonja (2016) concluye que, aunque los algoritmos asimétricos son superiores en seguridad, toman más tiempo para procesar y requieren más memoria. Prácticamente, algoritmos asimétricos como RSA se utilizan para el intercambio de claves y algoritmos simétricos se utilizan para el cifrado/descifrado. Así como en este estudio, que se concluye que una manera

eficiente es combinando ambos algoritmos, el simétrico para los archivos, y el asimétrico para las claves

Muhammad Inaam ul haq, Asfand Yar Galani, & Zafar (2018) después de comparar todos los parámetros que se analizan a través del simulador desarrollado con Windows Azure SDk concluye que AES es la mejor opción porque es rápido y su capacidad de cifrado de datos es alta. RSA es más seguro pero requiere muchos recursos y poder de cómputo. Su capacidad de cifrado de datos es muy baja en comparación con AES, aquí se ve el contexto, ya que como es una plataforma en la nube, son los archivos los que se deben cifrar, por lo que lo óptimo es que se utilice AES. Haciendo contraste con la presente investigación, compartimos conclusiones debido a que en el trabajo realizado por ellos concluyen que AES resulta más óptimo para el cifrado, y RSA para la seguridad.

CONCLUSIONES

Ambos algoritmos resultan eficientes, dependiendo en qué circunstancia se desee aplicar. AES nos será muy útil si se quiere cifrar archivos pesados ya que no se demorará como RSA sí lo hará. RSA resulta computacionalmente más complicado de romper, agregado a ello que AES tiene la debilidad de que al tener que compartir su clave de manera no segura, ésta se pueda interceptar en el camino. Ambos terminan complementándose, ya que en un caso óptimo, el archivo se cifra con AES, y como la clave se debe compartir, ésta se cifra con RSA.

El algoritmo AES tiene una tasa de cifra de megabytes por segundo, mientras que RSA tiene una tasa de cifra de centenas de kilobytes por segundo. Es decir, diferencia de 1 a 1 000. Por lo que encriptar archivos pesados con RSA, resultaría sumamente lento, a diferencia de AES que resulta más rápido.

El algoritmo RSA tiene la dificultad de ser el producto de dos números primos muy grandes, y se ha podido observar que cuanto más grande es la clave, más difícil es poder hallarla computacionalmente usando cualquier método, por ello la importancia de clave mínimo de 1 024 bits, a diferencia de AES que acepta claves hasta de 256 bits. No quiere decir que AES sea insegura, es solo que comparado con RSA, su desventaja es tener que compartir la clave.

RECOMENDACIONES

Al ser tipos de cifrado diferentes no se puede establecer cuál es mejor o peor, ya que ambos funcionan de manera óptima según la necesidad que se tenga. Se puede buscar comparar el algoritmo AES con otros algoritmos de cifrado simétrico, así como el RSA compararlo con otros algoritmos de cifrado asimétrico, y así poder ver la diferencia de ellos con respecto a otros algoritmos de su mismo tipo.

Tener en cuenta que el tiempo para cifrar o descifrar algún archivo, también depende de la computadora con la que se tiene. Así que se recomienda también probar en computadoras más potentes para ver qué tan rápido pueden hacerlo.

De igual manera, al tener una computadora más potente, y en este caso si se habla del tema de vulnerar, si se tienen más computadoras a disposición el tiempo de lograr romper algún algoritmo de encriptación va a ser menor. Esto sin tomar en cuenta la computación cuántica, que eso ya es otro tema.

REFERENCIAS BIBLIOGRÁFICAS

Aguilera, N. (18 de Setiembre de 2013). *Centro Científico Tecnológico Santa Fe.* Recuperado el 10 de junio de 2016, de http://www.santafe-conicet.gov.ar/~aguilera/apuntes/euclides.pdf

Al-Tamimi, A.-k. (2008). *Performance Analysis of Data Encryption Algorithms.*

Arias, F. G. (2012). *El proyecto de investigación. Introducción a la metodología científica. (6ta Edición).* Caracas: Episteme.

Buchmann, J. (2001). *Introduction to Cryptography.* Springer.

Caballero Gil, P. (2002). *Introducción a la Criptografía.* Madrid: Rama.

Chacón Zárate, A. (2009). *Comparativa de seguridad de algoritmos para resúmenes criptográficos.* México: Instituto Politécnico Nacional.

Chuco Güere, M. I. (2013). *Sistema de encriptación RSA para la fiabilidad de transmisión de archivos de texto en la sede campo armiño de Electroperu S.A.* Huancayo: Universidad Nacional del Centro del Perú.

Daemen, J., & Rijmen, V. (2002). *The design of Rijndael.* Berlin: Springer-Verlag.

Fernandez Domingo, J. I. (2006). *LA FIRMA ELECTRONICA.* Madrid: REUS S.A.

Fúster Sabater, A., Guía Martinez (de la), D., Hernández Encinas, L., Montoya Vitini, F., & Muñoz, J. (2000). *Técnicas Criptográficas de Protección de Datos.* Madrid: RA-MA.

Galende Díaz, J. C. (1995). *Criptografía: Historia de la escritura cifrada.* Madrid: Complutense.

Hercigonja, Z. (22 de Diciembre de 2016). Comparative Analysis of Cryptographic Algorithms. *International Journal of DIGITAL TECHNOLOGY & ECONOMY*, pág. 8.

Hernandez Sampieri, R., Fernández Collado, C., & Baptista Lucio, P. (2016). *Metodología de la Investigación (6ta. Edición).* Mexico: McGraw-Hill.

Johnston, P. (08 de Marzo de 2005). *"Mathematics"*. Obtenido de Paj's Home: Cryptography: RSA: Mathematics: http://pajhome.org.uk/crypt/rsa/maths.html

Kahn, D. (1995). *The Codebreakers.* Nueva York: Scribner.

López Egusquiza, C. J. (2013). *Propuesta algorítmica para el ocultamiento de fotografías usando criptografía y esteganografía.* Trujillo: UNT.

Lucena López, M. J. (2009). *Criptografía y Seguridad en Computadores.* España: Dpto. de Informática Universidad de Jaén. Edición virtual.

Mahajan, P., & Sachdeva, A. (2013). *A Study of Encryption Algorithms AES, DES and RSA for security.* USA: Global Journals Inc.

Mamani Ttito, T. (2014). *Modelo de sistema criptográfico de seguridad para las redes de comunicaciones en la región Puno.* Puno: Universidad Nacional del Altiplano.

Menezes, A. J., van Oorschot, P. C., & Vanstone, S. A. (1996). *Handbook of applied cryptography.* EEUU: CRC Press.

Merkle, R. C. (1978). *Secure communication over insecure channels.* Standford: Communications of the ACM.

Morillo, P. (1998). *Las Matemáticas en la Criptología.* Universidad Politécnica de Catalunya.

Muhammad Inaam ul haq, Asfand Yar Galani, & Zafar. (2018). *ANALYTICAL COMPARISON OF RSA AND AES USING WINDOWS AZURE FOR CLOUD COMPUTING ENVIRONMENT.* Pakistan.

Paguay Cuvi, M. H. (2015). *Análisis de algoritmos matemáticos de criptografía pública para mejorar el aprendizaje de la materia de criptografía en la carrera de ingeniería en sistemas de la ESPOCH.* Ecuador: Escuela Superior Politécnica de Chimborazo.

Pastor Franco, J., Sarasa López, M., & Salazar Riaño, J. (1998). *Criptografía digital: fundamentos y aplicaciones.* Zaragoza: Prensas Universitarias.

Ramió Aguirre, J. (2006). *Cifrado de comunicaciones digitales.*

Ramió, J. (1999). *Aplicaciones Criptográficas.* Madrid: Escuela Universitaria de Informática de la Universidad Politécnica, ISBN 83-87238-57-2.

Salomon, D. (2005). *Coding for Data and Computer Communications.* Springer.

Samaniego Zanabria, A. L. (2018). *Evaluación de Algoritmos Criptográficos para mejorar la Seguridad en la Comunicación y Almacenamiento de la Información.* Lima: Universidad Ricardo Palma.

Singh, S. (1999). *The Code Book.* Nueva York: Anchor Books.

Smith, W. D. (08 de Junio de 2007). *1. AES seems weak. 2. Linear time secure cryptography.* Recuperado el 15 de octubre de 2017, de https://eprint.iacr.org/2007/248.pdf

Solana, P. (2009). *Antecedentes y perspectivas de estudio en historia de la Criptografía.* Madrid: Universidad Carlos III.

Stinson, D. (2006). *Cryptography: Teory & Practice.* Chapman & Hall.

Vasquez Fernández, D. (2007). *El algoritmo criptográfico AES para protección de datos.* España: Universidad Pontificia Comillas.

Xifre Solana, P. (2009). *Antecedentes y perspectivas de estudio en historia de la Criptografía.* Madrid: Universidad Carlos III.

Zimmermann, P. R. (2004). *Criptografía para Internet.* Investigación y Ciencia.

ANEXOS

Anexo 01 Matriz de Consistencia

Título: Estudio comparativo de los algoritmos de encriptación Advanced Encryption Standard (AES) y Rivest, Shamir & Adleman (RSA)			
PROBLEMA	**OBJETIVO**	**HIPOTESIS**	**VARIABLES**
General: ¿Cuál es la diferencia de eficiencia del algoritmo de encriptación simétrico AES con respecto al algoritmo de encriptación asimétrico RSA? **Específicos:** • ¿Cuál es la diferencia de la encriptación AES y RSA con respecto al tiempo de ejecución? • ¿Cuál es la diferencia de la encriptación AES y RSA con respecto a la seguridad?	**General:** Realizar un estudio comparativo de los algoritmos AES y RSA para establecer la diferencia que existe entre ambos algoritmos de encriptación. **Específicos:** • Analizar y comparar los algoritmos AES y RSA con respecto al tiempo de ejecución • Analizar y comparar los algoritmos AES y RSA con respecto a la seguridad	**General:** H_0: El algoritmo RSA no es más eficiente que el algoritmo AES H_1: El algoritmo RSA es más eficiente que el algoritmo AES **Específicos:** • H_0:El tiempo de ejecución del algoritmo RSA no es mayor que el algoritmo AES H_1: El tiempo de ejecución del algoritmo RSA es mayor que el algoritmo AES • H_0: El algoritmo RSA no proporciona mayor seguridad que el algoritmo AES H_1: El algoritmo RSA proporciona mayor seguridad que el algoritmo AES	**Variable : Independiente** Algoritmos de encriptación Indicadores : -Tiempo de ejecución **Variable : Dependiente** Seguridad Indicadores: .Privacidad .Integridad

Anexo 02. Cifrado AES para el mensaje: «Texto plano a cifrar.»

Texto en claro:
546578746f20706c616e6f2061206369667261722e0b0b0b0b0b0b0b0b0b0b0b

Texto cifrado:
94e1202131265607b9c9ff7925af027f6a9436cd4d5627bff61cd600bcc4d875

- RONDAS DEL CIFRADO

BLOQUE 1

****

Ronda 1

Comienzo de ronda: 170919020a64153c131b0a420011515a

Después de SubBytes: f001d477674359eb7daf672c6382d1be

Después de ShiftRows: f04367be67afd1777d82d4eb6301592c

Después de MixColumns: e761a448823de83958ee6a1cb0a6a4a5

Después de AddRoundKey: 622e06d162362ff0ca90c8b743e9343d

Ronda 2

Comienzo de ronda: 622e06d162362ff0ca90c8b743e9343d

Después de SubBytes: aa316f3eaa05158c7460e8a91a1e1827

Después de ShiftRows: aa05e827aa60183e741e6f8c1a3115a9

Después de MixColumns: 8fa40d46c97cb8e129753beedbeee143

Después de AddRoundKey: 8c8be9d22a589bbc582fba1859fbf02d

Ronda 3

Comienzo de ronda: 8c8be9d22a589bbc582fba1859fbf02d

Después de SubBytes: 643d1eb5e56a14656a15f4adcb0f8cd8

Después de ShiftRows: 646af4d8e5158cb56a0f1e65cb3d14ad

Después de MixColumns: 5a6f8e99d7f537dcbe33f6657320322e

Después de AddRoundKey: 04c2f51e6a7c6f0672e02f493de6fa6c

Ronda 4

Comienzo de ronda: 04c2f51e6a7c6f0672e02f493de6fa6c

Después de SubBytes: f225e6720210a86f40e1153b278e2d50

Después de ShiftRows: f210155002e12d72408ee66f2725a83b

Después de MixColumns: 8abd38a863de2f2e8019a876b2b50492

Después de AddRoundKey: 68f86f003c12205c13067e286f6c1a8e

Ronda 5

Comienzo de ronda: 68f86f003c12205c13067e286f6c1a8e

Después de SubBytes: 4541a863ebc9b74a7d6ff334a850a219

Después de ShiftRows: 45c9f319eb6fa2637d50a84aa841b734

Después de MixColumns: 20db5ac7bdab7e2de874b8eb0bdcc07d

Después de AddRoundKey: e7ec91ae2550ba36e390aaaedde1cc24

Ronda 6

Comienzo de ronda: e7ec91ae2550ba36e390aaaedde1cc24

Después de SubBytes: 94ce81e43f53f4051160ace4c1f84b36

Después de ShiftRows: 9453ac363f604be411f88105c1cef4e4

Después de MixColumns: 5cebde3471c6feb9b567ff40c0a5cbb1

Después de AddRoundKey: 9c22deab29f43a3de6b12981454e1129

Ronda 7

Comienzo de ronda: 9c22deab29f43a3de6b12981454e1129

Después de SubBytes: de931d62a5bf80278ec8a50c6e2f82a5

Después de ShiftRows: debfa5a5a5c882628e2f1d276e93800c

Después de MixColumns: 7deac432f2d1d47a4cd0f2f5fec4f2b9

Después de AddRoundKey: 1474823ac37d56f62eaaa6b819557c6c

Ronda 8

Comienzo de ronda: 1474823ac37d56f62eaaa6b819557c6c

Después de SubBytes: fa9213802effb14231ac246cd4fc1050

Después de ShiftRows: faff24502eac108031fc1342d492b16c

Después de MixColumns: 8123bd6e23dd39d52ca52d38c34f8b9c

Después de AddRoundKey: e9a4f8f27af6fec517f4be651f8f9614

Ronda 9

Comienzo de ronda: e9a4f8f27af6fec517f4be651f8f9614

Después de SubBytes: 1e494189da42bba6f0bfae4dc07390fa

Después de ShiftRows: 1e42aefadabf9089f07341a6c049bb4d

Después de MixColumns: ae890e216c9dde538973f06eb6c93333

Después de AddRoundKey: 67aa8f3bfc959859222a2539c150fbec

Ronda 10

Comienzo de ronda: 67aa8f3bfc959859222a2539c150fbec

Después de SubBytes: 85ac73e2b02a46cb93e53f1278530fce

Después de ShiftRows: 852a3fceb0e50fe2935373cb78ac4612

Después de AddRoundKey: 94e1202131265607b9c9ff7925af027f

BLOQUE 2

****

Ronda 1

Comienzo de ronda: 251e00044b4f6e5b797e6e696a3a3938

Después de SubBytes: 3f7263f2b3849f39b6f39ff902801207

Después de ShiftRows: 3f849f07b3f312f2b680633902729ff9

Después de MixColumns: 71919754938a69d0b631bb50f4a54502

Después de AddRoundKey: f4de35cd7381ae19244f19fb07ead59a

Ronda 2

Comienzo de ronda: f4de35cd7381ae19244f19fb07ead59a

Después de SubBytes: bf1d96bd8f0ce4d43684d40fc58703b8

Después de ShiftRows: bf0cd4b88f8403bd368796d4c51de40f

Después de MixColumns: 1d78d3692c24d16cbc56e1f85dc71ab3

Después de AddRoundKey: 1e5737fdcf00f231cd0c600edfd20bdd

Ronda 3

Comienzo de ronda: 1e5737fdcf00f231cd0c600edfd20bdd

Después de SubBytes: 725b9a548a6389c7bdfed0ab9eb52bc1

Después de ShiftRows: 7263d0c18afe2b54bdb59ac79e5b89ab

Después de MixColumns: 501ef2bc6944def8f8be7566e8032a26

Después de AddRoundKey: 0eb3893bd4cd8622346dac4aa6c5e264

Ronda 4

Comienzo de ronda: 0eb3893bd4cd8622346dac4aa6c5e264

Después de SubBytes: ab6da7e248bd4493183c91d624a69843

Después de ShiftRows: abbd9143483c98e218a6a793246d44d6

Después de MixColumns: 4321ea4cae6162a3f52e45146de4a0f2

Después de AddRoundKey: a164bde4f1ad6dd16631934ab03dbeee

Ronda 5

Comienzo de ronda: a164bde4f1ad6dd16631934ab03dbeee

Después de SubBytes: 32437a69a1953c3e33c7dcd6e727ae28

Después de ShiftRows: 3295dc28a1c7ae6933277a3ee7433cd6

Después de MixColumns: 34547c4fccb49a434bcda274faf3bdfa

Después de AddRoundKey: f363b726544f5e584029b0312cceb1a3

Ronda 6

Comienzo de ronda: f363b726544f5e584029b0312cceb1a3

Después de SubBytes: 0dfba9f72084586a09a5e7c7718bc80a

Después de ShiftRows: 0d84e70a20a5c8f7098ba96a71fb58c7

Después de MixColumns: 602642608bc50cf8578e75ed6bb368a5

Después de AddRoundKey: a0ef42ffd3f7c87c0458a32cee58b23d

Ronda 7

Comienzo de ronda: a0ef42ffd3f7c87c0458a32cee58b23d

Después de SubBytes: e0df2c166668e810f26a0a71286a3727

Después de ShiftRows: e0680a27666a3716f26a2c1028dfe871

Después de MixColumns: 4e09f51753fd58db7d42f06bb3dfafad

Después de AddRoundKey: 2797b31f6251da571f38a426544e2178

Ronda 8

Comienzo de ronda: 2797b31f6251da571f38a426544e2178

Después de SubBytes: cc886dc0aad1575bc00749f7202ffdbc

Después de ShiftRows: ccd149bcaa07fdc0c02f6d5b208857f7

Después de MixColumns: 1e1250b47b781784dc72d8af6325044a

Después de AddRoundKey: 769515282253d094e7234bf2bfe519c2

Ronda 9

Comienzo de ronda: 769515282253d094e7234bf2bfe519c2

Después de SubBytes: 382a593493ed70229426b38908d9d425

Después de ShiftRows: 38edb3259326d43494d95922082a7089

Después de MixColumns: ca12c75cb78c5a3438f499639745424b

Después de AddRoundKey: 0331464627841c3e93ad4c34e0dc8a94

Ronda 10

Comienzo de ronda: 0331464627841c3e93ad4c34e0dc8a94

Después de SubBytes: 7bc75a5acc5f9cb2dc952918e1867e22

Después de ShiftRows: 7b5f2922cc957e5adc865ab2e1c79c18

Después de AddRoundKey: 6a9436cd4d5627bff61cd600bcc4d875

- RONDAS DEL DESCIFRADO

Texto cifrado:
94e1202131265607b9c9ff7925af027f6a9436cd4d5627bff61cd600bcc4d875

Texto en claro:
546578746f20706c616e6f2061206369667261722e0b0b0b0b0b0b0b0b0b0b0b

BLOQUE 1

****

Ronda 1

Comienzo de ronda: 852a3fceb0e50fe2935373cb78ac4612

Después de InvShiftRows: 85ac73e2b02a46cb93e53f1278530fce

Después de InvSubBytes: 67aa8f3bfc959859222a2539c150fbec

Después de InvAddRoundKey: ae890e216c9dde538973f06eb6c93333

Después de InvMixColumns: 1e42aefadabf9089f07341a6c049bb4d

Ronda 2

Comienzo de ronda: 1e42aefadabf9089f07341a6c049bb4d

Después de InvShiftRows: 1e494189da42bba6f0bfae4dc07390fa

Después de InvSubBytes: e9a4f8f27af6fec517f4be651f8f9614

Después de InvAddRoundKey: 8123bd6e23dd39d52ca52d38c34f8b9c

Después de InvMixColumns: faff24502eac108031fc1342d492b16c

Ronda 3

Comienzo de ronda: faff24502eac108031fc1342d492b16c

Después de InvShiftRows: fa9213802effb14231ac246cd4fc1050

Después de InvSubBytes: 1474823ac37d56f62eaaa6b819557c6c

Después de InvAddRoundKey: 7deac432f2d1d47a4cd0f2f5fec4f2b9

Después de InvMixColumns: debfa5a5a5c882628e2f1d276e93800c

Ronda 4

Comienzo de ronda: debfa5a5a5c882628e2f1d276e93800c

Después de InvShiftRows: de931d62a5bf80278ec8a50c6e2f82a5

Después de InvSubBytes: 9c22deab29f43a3de6b12981454e1129

Después de InvAddRoundKey: 5cebde3471c6feb9b567ff40c0a5cbb1

Después de InvMixColumns: 9453ac363f604be411f88105c1cef4e4

Ronda 5

Comienzo de ronda: 9453ac363f604be411f88105c1cef4e4

Después de InvShiftRows: 94ce81e43f53f4051160ace4c1f84b36

Después de InvSubBytes: e7ec91ae2550ba36e390aaaedde1cc24

Después de InvAddRoundKey: 20db5ac7bdab7e2de874b8eb0bdcc07d

Después de InvMixColumns: 45c9f319eb6fa2637d50a84aa841b734

Ronda 6

Comienzo de ronda: 45c9f319eb6fa2637d50a84aa841b734

Después de InvShiftRows: 4541a863ebc9b74a7d6ff334a850a219

Después de InvSubBytes: 68f86f003c12205c13067e286f6c1a8e

Después de InvAddRoundKey: 8abd38a863de2f2e8019a876b2b50492

Después de InvMixColumns: f210155002e12d72408ee66f2725a83b

Ronda 7

Comienzo de ronda: f210155002e12d72408ee66f2725a83b

Después de InvShiftRows: f225e6720210a86f40e1153b278e2d50

Después de InvSubBytes: 04c2f51e6a7c6f0672e02f493de6fa6c

Después de InvAddRoundKey: 5a6f8e99d7f537dcbe33f6657320322e

Después de InvMixColumns: 646af4d8e5158cb56a0f1e65cb3d14ad

Ronda 8

Comienzo de ronda: 646af4d8e5158cb56a0f1e65cb3d14ad

Después de InvShiftRows: 643d1eb5e56a14656a15f4adcb0f8cd8

Después de InvSubBytes: 8c8be9d22a589bbc582fba1859fbf02d

Después de InvAddRoundKey: 8fa40d46c97cb8e129753beedbeee143

Después de InvMixColumns: aa05e827aa60183e741e6f8c1a3115a9

Ronda 9

Comienzo de ronda: aa05e827aa60183e741e6f8c1a3115a9

Después de InvShiftRows: aa316f3eaa05158c7460e8a91a1e1827

Después de InvSubBytes: 622e06d162362ff0ca90c8b743e9343d

Después de InvAddRoundKey: e761a448823de83958ee6a1cb0a6a4a5

Después de InvMixColumns: f04367be67afd1777d82d4eb6301592c

Ronda 10

Comienzo de ronda: f04367be67afd1777d82d4eb6301592c

Después de InvShiftRows: f001d477674359eb7daf672c6382d1be

Después de InvSubBytes: 170919020a64153c131b0a420011515a

Después de InvAddRoundKey: 546578746f20706c616e6f2061206369

BLOQUE 2

****

Ronda 1

Comienzo de ronda: 7b5f2922cc957e5adc865ab2e1c79c18

Después de InvShiftRows: 7bc75a5acc5f9cb2dc952918e1867e22

Después de InvSubBytes: 0331464627841c3e93ad4c34e0dc8a94

Después de InvAddRoundKey: ca12c75cb78c5a3438f499639745424b

Después de InvMixColumns: 38edb3259326d43494d95922082a7089

Ronda 2

Comienzo de ronda: 38edb3259326d43494d95922082a7089

Después de InvShiftRows: 382a593493ed70229426b38908d9d425

Después de InvSubBytes: 769515282253d094e7234bf2bfe519c2

Después de InvAddRoundKey: 1e1250b47b781784dc72d8af6325044a

Después de InvMixColumns: ccd149bcaa07fdc0c02f6d5b208857f7

Ronda 3

Comienzo de ronda: ccd149bcaa07fdc0c02f6d5b208857f7

Después de InvShiftRows: cc886dc0aad1575bc00749f7202ffdbc

Después de InvSubBytes: 2797b31f6251da571f38a426544e2178

Después de InvAddRoundKey: 4e09f51753fd58db7d42f06bb3dfafad

Después de InvMixColumns: e0680a27666a3716f26a2c1028dfe871

Ronda 4

Comienzo de ronda: e0680a27666a3716f26a2c1028dfe871

Después de InvShiftRows: e0df2c166668e810f26a0a71286a3727

Después de InvSubBytes: a0ef42ffd3f7c87c0458a32cee58b23d

Después de InvAddRoundKey: 602642608bc50cf8578e75ed6bb368a5

Después de InvMixColumns: 0d84e70a20a5c8f7098ba96a71fb58c7

Ronda 5

Comienzo de ronda: 0d84e70a20a5c8f7098ba96a71fb58c7

Después de InvShiftRows: 0dfba9f72084586a09a5e7c7718bc80a

Después de InvSubBytes: f363b726544f5e584029b0312cceb1a3

Después de InvAddRoundKey: 34547c4fccb49a434bcda274faf3bdfa

Después de InvMixColumns: 3295dc28a1c7ae6933277a3ee7433cd6

Ronda 6

Comienzo de ronda: 3295dc28a1c7ae6933277a3ee7433cd6

Después de InvShiftRows: 32437a69a1953c3e33c7dcd6e727ae28

Después de InvSubBytes: a164bde4f1ad6dd16631934ab03dbeee

Después de InvAddRoundKey: 4321ea4cae6162a3f52e45146de4a0f2

Después de InvMixColumns: abbd9143483c98e218a6a793246d44d6

Ronda 7

Comienzo de ronda: abbd9143483c98e218a6a793246d44d6

Después de InvShiftRows: ab6da7e248bd4493183c91d624a69843

Después de InvSubBytes: 0eb3893bd4cd8622346dac4aa6c5e264

Después de InvAddRoundKey: 501ef2bc6944def8f8be7566e8032a26

Después de InvMixColumns: 7263d0c18afe2b54bdb59ac79e5b89ab

Ronda 8

Comienzo de ronda: 7263d0c18afe2b54bdb59ac79e5b89ab

Después de InvShiftRows: 725b9a548a6389c7bdfed0ab9eb52bc1

Después de InvSubBytes: 1e5737fdcf00f231cd0c600edfd20bdd

Después de InvAddRoundKey: 1d78d3692c24d16cbc56e1f85dc71ab3

Después de InvMixColumns: bf0cd4b88f8403bd368796d4c51de40f

Ronda 9

Comienzo de ronda: bf0cd4b88f8403bd368796d4c51de40f

Después de InvShiftRows: bf1d96bd8f0ce4d43684d40fc58703b8

Después de InvSubBytes: f4de35cd7381ae19244f19fb07ead59a

Después de InvAddRoundKey: 71919754938a69d0b631bb50f4a54502

Después de InvMixColumns: 3f849f07b3f312f2b680633902729ff9

Ronda 10

Comienzo de ronda: 3f849f07b3f312f2b680633902729ff9

Después de InvShiftRows: 3f7263f2b3849f39b6f39ff902801207

Después de InvSubBytes: 251e00044b4f6e5b797e6e696a3a3938

Después de InvAddRoundKey: 667261722e0b0b0b0b0b0b0b0b0b0b0b

Anexo 03 Cifrado y descifrado con OpenSSL

Se tiene un archivo de texto plano “prueba.txt”, el cual se procede a encriptar mediante AES y que el documento cifrado se guarde en el archivo “cifrado.txt”. El comando a utilizar es el siguiente

```
openssl aes-256-cbc -a -salt -in prueba.txt -out cifrado.txt
```

Nos pide que coloquemos una contraseña. Ésta vendría a ser la llave del archivo a encriptar.

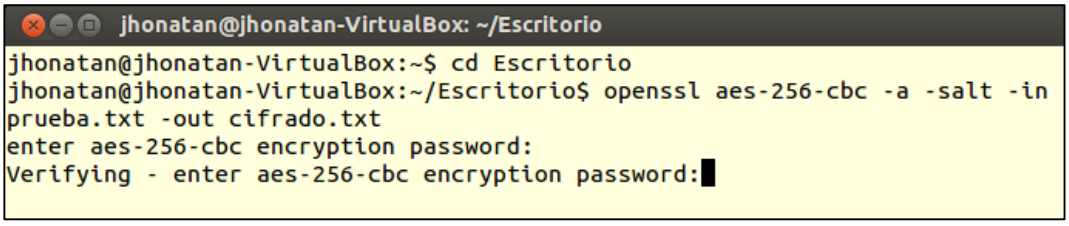


Figura 39. Encriptado AES a "prueba.txt"
Fuente: Elaboración propia

El archivo es encriptado. Para el proceso del desencriptado, el archivo ya descifrado se guardará en “descifrado.txt”. El comando a utilizar es

```
openssl aes-256-cbc -d -a -in cifrado.txt -out descifrado.txt
```

Lógicamente para descifrarlo, se ingresa la clave que se colocó al comienzo

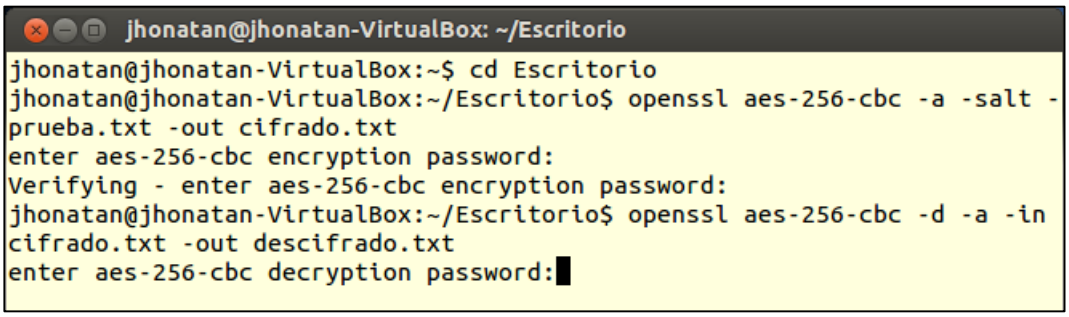


Figura 40. Desencriptado AES a "cifrado.txt"
Fuente: Elaboración propia

Y así tendríamos el archivo original “prueba.txt”, luego el encriptado AES cuyo archivo es “cifrado.txt”, y finalmente el archivo desencriptado “descifrado.txt”, el cual, es igual al original “prueba.txt”

También realizamos la prueba con un archivo tipo PDF (650KB)

```
openssl aes-256-cbc -a -salt -in prueba.pdf -out cifradopdf.txt
```

En este caso, queremos cifrar “prueba.pdf” y que el archivo cifrado se guarde en “cifrapdf.txt”. El archivo encriptado también puede ser del mismo tipo que el original, en este caso PDF, pero en este caso lo guardaremos en .txt

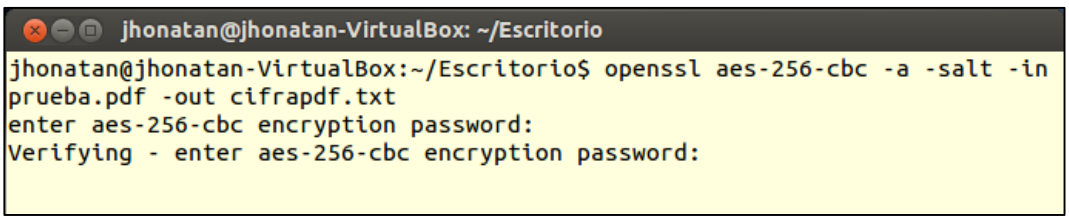


Figura 41. Encriptado AES a "prueba.pdf"
Fuente: Elaboración propia

Ahora se procede a desencriptar

```
openssl aes-256-cbc –d –a –in cifrapdf.txt –out descifrado.pdf
```

El destino del archivo desencriptado es “descifrado.pdf”. Como el archivo original es un PDF, lógicamente el archivo ya desencriptado debe ser el mismo tipo.

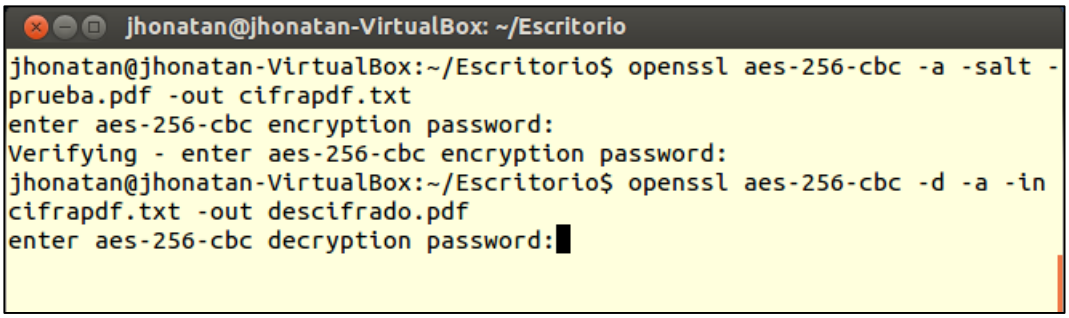


Figura 42. Desencriptado AES a "cifrapdf.txt"
Fuente: Elaboración propia

Ahora se hará el encriptamiento de un video (50MB)

```
openssl aes-256-cbc –a –salt –in video.mp4 –out cifravideo.txt
```

El archivo “video.mp4” se guardará cifrado en “cifravideo.txt”. Colocamos su clave

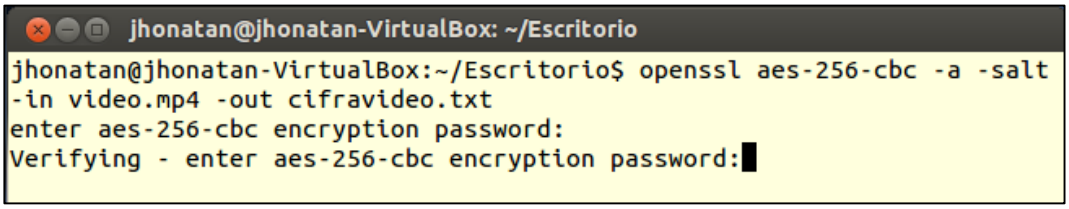


Figura 43. Encriptado AES a "video.mp4"
Fuente: Elaboración propia

Una vez encriptado, ya se puede pasar a desencriptarlo

```
openssl aes-256-cbc -d -a -in cifravideo.txt -out videodescifrado.mp4
```

El destino del archivo desencriptado es “videodescifrado.mp4”. Como el archivo

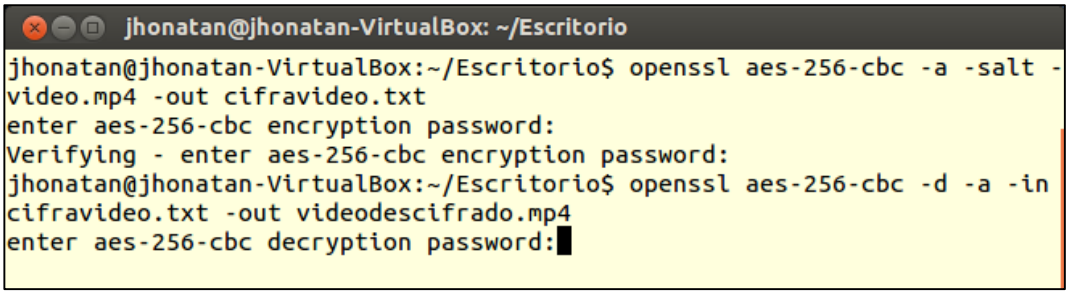


Figura 44. Desencriptado AES a "cifravideo.txt"
Fuente: Elaboración propia

Ahora el encriptamiento se hará aplicando el algoritmo RSA. Se comienza generando la llave privada. En este caso se hará de 1024 bits

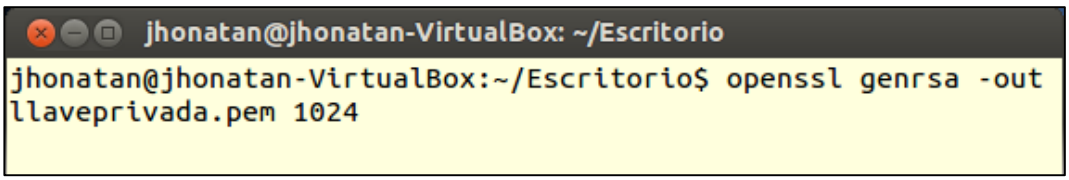


Figura 45. Generación de la llave privada
Fuente: Elaboración propia

El comando para generar la llave privada (archivo tipo pem) es

```
openssl genrsa -out llavepriva.pem 1024
```

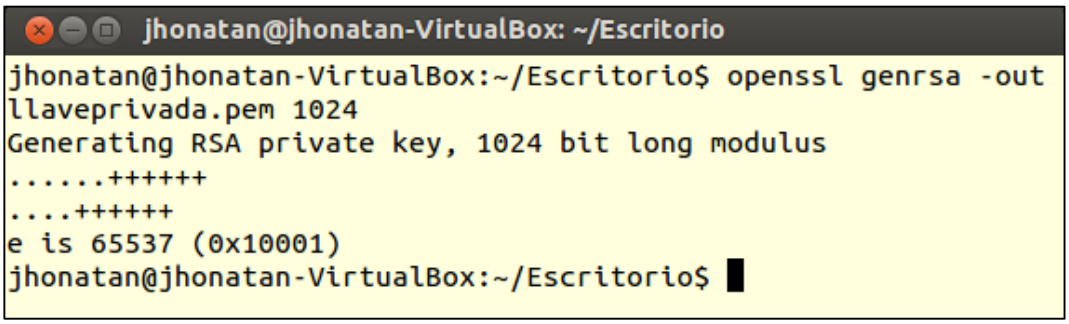


Figura 46. Generando llaveprivada.pem
Fuente: Elaboración propia

A partir de la llave privada se genera la llave pública

```
openssl rsa –in llaveprivada.pem –pubout –out llavepublica.pem
```

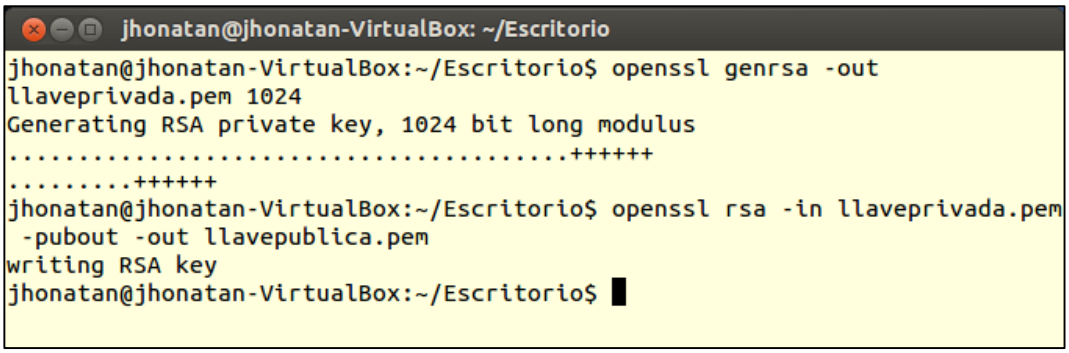


Figura 47. Con llaveprivada.pem se genera llavepublica.pem
Fuente: Elaboración propia

Encriptamos el archivo "prueba.txt" en un archivo "cifradoRSA.txt". Para encriptarlo, utilizamos la llave pública

```
openssl rsautl –pubin –encrypt –in prueba.txt –out cifradoRSA.txt -inkey
llavepublica.pem
```

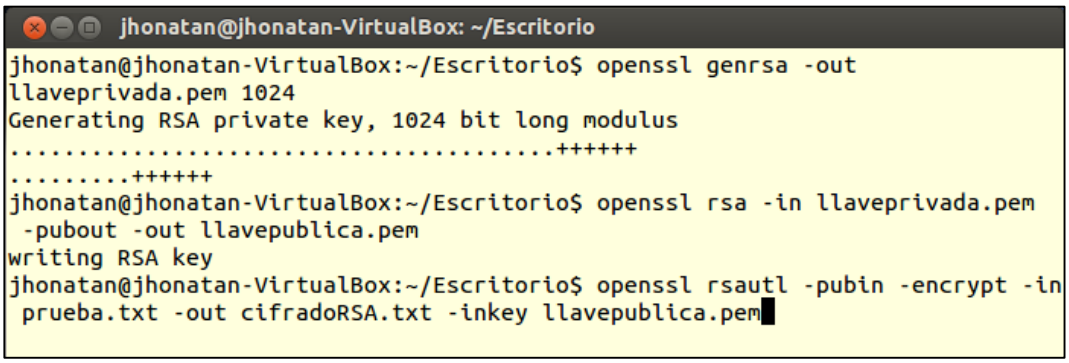


Figura 48. Encriptado RSA a "prueba.txt"
Fuente: Elaboración propia

Para desencriptar utilizamos la llave privada. Lo guardamos en el archivo

```
openssl rsautl –decrypt –in cifradoRSA.txt –out descifradoRSA.txt –
                    inkey llaveprivada.pem
```

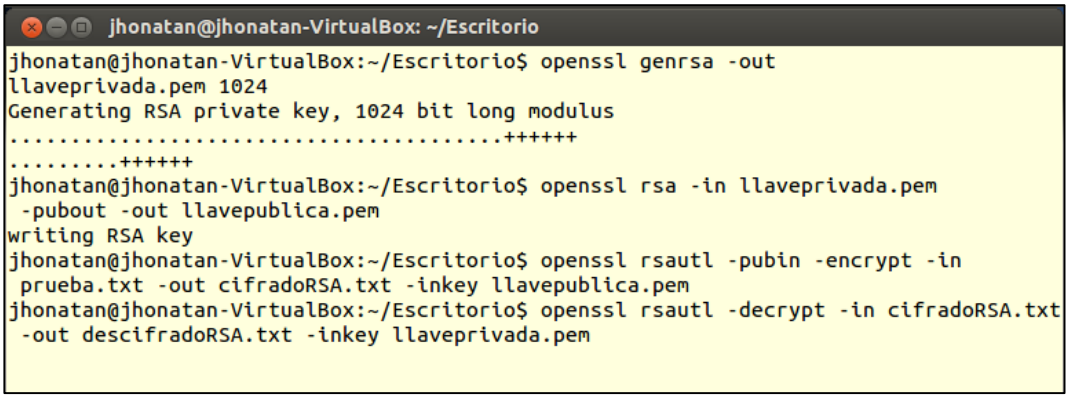


Figura 49. Desencriptado RSA a "cifradoRSA.txt"
Fuente: Elaboración propia