
# DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

## CARRERA DE INGENIERÍA EN ELECTRÓNICA EN REDES Y COMUNICACIÓN DE DATOS

**TESIS PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA EN REDES Y COMUNICACIÓN DE DATOS**

**TEMA: OPTIMIZACIÓN DEL ALGORITMO DE ENCRIPTACIÓN ASIMÉTRICA RSA PARA LA MITIGACIÓN DE ATAQUES DE SEGURIDAD DE REDES**

**AUTOR: FLORES TAIPE, DANIELA KATHERINE**

**DIRECTOR: ING. FUERTES, WALTER PhD.**

**CODIRECTOR: ING. AGUILAR, DARWIN MSc.**

**SANGOLQUÍ**

**2015**

**UNIVERSIDAD DE LAS FUERZAS ARMADAS – ESPE**

**INGENIERÍA EN ELECTRÓNICA EN REDES Y COMUNICACIÓN DE DATOS**

# CERTIFICADO

Ing. Walter Fuertes, PhD.

Ing. Darwin Aguilar, MSc.

## CERTIFICAN

Que el trabajo titulado "Optimización del Algoritmo de Encriptación Asimétrica RSA para la Mitigación de Ataques de Seguridad de Redes", realizado por la Srta. Daniela Katherine Flores Taipe, ha sido guiado y revisado periódicamente y cumple en su totalidad con las normas establecidas por la Universidad de las Fuerzas Armadas – ESPE.

Debido a que se trata de un trabajo de investigación recomiendan su publicación.

Sangolquí, 27 de julio del 2015.

Ing. Walter Fuertes, PhD.

DIRECTOR

Ing. Darwin Aguilar, MSc.

CODIRECTOR

# UNIVERSIDAD DE LAS FUERZAS ARMADAS – ESPE

## INGENIERÍA EN ELECTRÓNICA EN REDES Y COMUNICACIÓN DE DATOS

## AUTORÍA DE RESPONSABILIDAD

Yo, Daniela Katherine Flores Taipe

## DECLARO QUE:

El proyecto de grado denominado "Optimización del Algoritmo de Encriptación Asimétrica RSA para la Mitigación de Ataques de Seguridad de Redes" es de mi autoría y ha sido desarrollado en base a una investigación exhaustiva respetando derechos intelectuales de terceros e incorporando las referencias correspondientes en la bibliografía.

En virtud de esta declaración, me responsabilizo del contenido, veracidad y alcance científico del proyecto de grado en mención.

Sangolquí, 27 de julio del 2015.

Daniela Katherine Flores Taipe

# UNIVERSIDAD DE LAS FUERZAS ARMADAS – ESPE

## INGENIERÍA EN ELECTRÓNICA EN REDES Y COMUNICACIÓN DE DATOS

## AUTORIZACIÓN

Yo, Daniela Katherine Flores Taipe

Autorizo a la Universidad de las Fuerzas Armadas – ESPE la publicación en la biblioteca virtual de la Institución del trabajo titulado "Optimización del Algoritmo de Encriptación Asimétrica RSA para la Mitigación de Ataques de Seguridad de Redes", cuyo contenido, ideas y criterios son de mi exclusiva responsabilidad y autoría.

Sangolquí, 27 de julio del 2014.

Daniela Katherine Flores Taipe

# DEDICATORIA

*A Dios y a la Virgen de Guadalupe, que me han bendecido a cada paso que doy. Por haber guiado mi camino con su divina presencia y haberme permitido culminar esta etapa de mi vida.*

*A mis abuelitos Heriberto y Carmen, por su amor incondicional, mis ángeles desde el cielo, a los que extraño cada día más. A los que me mostraron que la vida es linda pero hay que saberla vivir.*

*A mis mejores amigos, Elvia Taipe y Ramón Flores, por ser mí apoyo, por compartir mis alegrías y mis tristezas, por enseñarme que en la vida hay que luchar para que los sueños se hagan realidad. Por ser los mejores padres del mundo, excelentes maestros de mi vida, porque sin ustedes no sé qué sería de mí.*

*"Y todo lo que hagan, de palabra o de obra, háganlo en el nombre del Señor Jesús, dando gracias a Dios el Padre por medio de él." Colosenses 3:17*

*Daniela Katherine Flores T.*

# AGRADECIMIENTO

En el camino de la vida siempre se encuentran Instituciones y personas que ayudan y apoyan para que otras sobresalgan, y aunque el tiempo pase siempre quedarán en la memoria cada una de ellas que dejaron su huella en uno.

En primer lugar agradezco a Dios por haberme dado fuerzas, salud y valentía, por nunca haberme abandonado cuando ya me sentía decaer.

Agradezco a mi Familia por su apoyo total, y a pesar de tenerlos lejos de mí, siempre tendré de ellos un cariño ilimitado, un amor incondicional y una amistad absoluta.

A mis docentes que fueron aquellas personas que con conocimiento y templanza apoyaron a lo largo de mí caminar educativo.

A mi Director de tesis Dr. Walter Fuertes, Codirector Ing. Darwin Aguilar e Ing. Fausto Meneses miembro del Grupo de Investigación en Sistemas Distribuidos que con esfuerzo, paciencia, tiempo y guía han hecho posible para llegar a feliz término con este proyecto.

A mis compañeros, amigos y a todo los que de alguna manera han contribuido, para que con perseverancia llegue a culminar mi carrera, eternamente gracias de todo corazón.

*Daniela Katherine Flores T.*

# ÍNDICE GENERAL

# ÍNDICE FIGURAS

# ÍNDICE TABLAS

# RESUMEN

Ante el desarrollo de las telecomunicaciones, la información que viaja a través de la red se ha visto afectado por enemigos que intentan quebrantar la seguridad. La criptología es la solución para reducir las amenazas en las redes, mediante algoritmos y protocolos, que permiten cifrar y descifrar sin tomar en cuenta los recursos computacionales. Los algoritmos asimétricos proporcionan una comunicación segura a través del Internet, sin embargo las soluciones matemáticas que estas brindan requieren de un mayor costo computacional. El presente proyecto tiene como objetivo optimizar el algoritmo de encriptación asimétrica RSA para mejorar la seguridad de los mensajes y disminuir el consumo de los recursos. Para llevarlo a cabo se diseñó y desarrolló una solución genérica, en la que el algoritmo permite incrementar la velocidad del cálculo, al disminuir la complejidad matemática. Además se combina tanto el cálculo modular como el probabilístico para mejorar el tiempo de cifrado y descifrado. La información enviada será el mensaje a cifrar y descifrar, junto con los parámetros necesarios para el cálculo del RSA. Estos parámetros consisten en el *p*, *q*, *n* y *una serie de índices aleatorios*, que se obtendrán a través de una llamada a procedimiento remoto. Se realizaron pruebas funcionales tanto en el cliente como en el servidor, con los que se obtuvo varios resultados de las variables evaluadas como el consumo de memoria, consumo del procesador, latencia, reporte estadístico de la red, tiempo cifrado y descifrado. Posteriormente se procedió a la interpretación de estos datos mediante el procesamiento estadístico, determinando que el algoritmo RSA propuesto tiende a una mejoría en cuanto al costo de software y a la seguridad de la información.

**PALABRAS CLAVES:**

- **CRIPTOGRAFIA**
- **ALGORITMO ASIMÉTRICO**
- **RSA**
- **RECURSOS COMPUTACIONALES**
- **SEGURIDAD**

# ABSTRACT

Before the development of telecommunications, the information traveling through the network has been hit by enemies trying to breach security. Cryptology is the solution to reduce threats in the network, using algorithms and protocols that allow you to encrypt and decrypt regardless of computing resources. Asymmetric algorithms provide secure communication over the Internet, however mathematical solutions that provide these require greater computational cost. In the present project it aims to optimize the RSA asymmetric encryption algorithm to improve safety messages and reduce consumption of resources. To carry out was designed and developed a generic solution, where the algorithm allows to increase the calculation speed by decreasing the mathematical complexity. In addition both modular and probabilistic calculation is combined to improve time encryption and decryption. The information sent is the message to encrypt and decrypt, along with the parameters required for the calculation of the RSA. These parameters consist of *p, q, n and a series of random indices*, which will be obtained through a remote procedure call. Functional tests on both the client and server, with several results of the evaluated variables such as memory consumption, processor consumption, latency, network statistics report, time encryption and decryption is performed was obtained. Then we proceeded to the interpretation of these data by the statistical processing, determining that the RSA algorithm proposed having an improvement in the cost of software and information security.

**KEY WORDS:**

- **CRYPTOGRAPHY**
- **ASYMMETRIC ALGORITHM**
- **RSA**
- **RESOURCES CONSUMPTION**
- **SECURITY**

# CAPÍTULO 1

# INTRODUCCIÓN

En el presente capítulo se describe las generalidades, planteamiento del problema, antecedentes, justificación e importancia, alcance, objetivos, metodología y herramientas de software, utilizadas para el desarrollo del software.

## 1.1. Generalidades

La evolución de la tecnología y la comunicación, ha producido cambios notables en la sociedad, exigiendo servicios escalables, rápidos y de bajo costo. Desde finales del siglo XX, un área de las matemáticas conocida como la teoría de números adquirió una importancia notable en los sistemas criptográficos que se usan para la seguridad de la información (Gil, 2000), al mitigar los ataques que frecuentemente se producen en el ciberespacio, disminuye la pérdida de información, la imagen institucional y productividad.

Según los datos encontrados en Identify Theft Resource Center de Estados Unidos (2014), se presentó 783 fugas de información, que afectó a tres sectores importantes como son: Salud, Negocios y Gobierno, con un ataque promedio de 15 violaciones por semana. Los usuarios deberán estar conscientes del riesgo que tienen al exponer su información personal. En la Tabla 1, se observa estadísticas que determinan que las amenazas cibernéticas son detectadas con un tiempo cada vez más reducido, al dar aviso a los intrusos informáticos de la prevención que tienen dichas industrias en las vías de entrada al Internet.

**Tabla 1**

***Resumen de Filtración de datos 2014***

| Categorías | Número de infracciones | Porcentaje de infracciones | Número de archivos | Porcentaje de archivo |
|---|---|---|---|---|
| **Banca/ Crédito/ Financiero** | 43 | 5.5% | 1.198.492 | 1.4% |

**CONTINUA** →

| | | | | |
|---|---|---|---|---|
| **Comercio** | 258 | 33.0% | 68.237.914 | 79.1% |
| **Educación** | 57 | 7.3% | 1.247.812 | 1.5% |
| **Gobierno/ Militar** | 92 | 11.7% | 6.649.319 | 7.8% |
| **Medicina** | 333 | 42.5% | 8.277.991 | 9.7% |
| **Total** | 783 | 100.0% | 85.611,58 | 100.0% |

Fuente: (Identity Theft Resource Center, 2014)

Uno de los mecanismos de mayor contundencia, para evitar ataques informáticos es la encriptación de la información. La encriptación es un proceso de convertir ilegible un mensaje, de esta manera, solo los receptores que tengan y conozcan la clave de encriptación, podrán tener acceso al mensaje, así llega a ser una medida de seguridad que permita almacenar o transmitir información al usar fórmulas matemáticas y para desencriptar usar una clave como parámetro para dichas fórmulas (Sanjuan, 2012).

Los algoritmos de encriptación o criptográficos, tienen como objetivo común que el mensaje sea difícil de desencriptar sin usar la clave. Existen tres diferentes algoritmos de encriptación como son: los asimétricos, simétricos y HASH (Astilla, 2009). En esta investigación se enfocará en los algoritmos asimétricos.

En los algoritmos asimétricos, el tipo de cifrado se basa particularmente en generar un clave privada y pública, con la diferencia de que al momento del envió el mensaje será encriptado por una llave pública, mientras que la clave privada se mantendrá en exclusivo para los diferentes receptores (Sanjuan, 2012). Una vez enviado el paquete encriptado, el destinatario podrá descifrar este mensaje solamente con la llave privada; Se puede decir entonces que la seguridad descansa en la clave.

Sobre la base de las consideraciones anteriores, el actual proyecto tiene como propósito optimizar el algoritmo de criptografía asimétrica RSA (Rivest-Shamir-Adleman, por sus autores), que esencialmente hoy en día es el algoritmo de clave pública más acreditada (López M. J., 2003).

## 1.2. Planteamiento del Problema

En el mundo de la informática, siempre se ha buscado mejorar y desarrollar nuevas maneras de comunicación entre usuarios, sin embargo siempre se ha visto afectado por

enemigos que intentan robar cuanta información puedan. La criptología fortalece la seguridad de la red, esta abarca diferentes algoritmos y protocolos complejos de cifrado y descifrado. No obstante en la actualidad el cifrado de clave pública más empleada, es el método asimétrico RSA llamado así por sus autores (R. Rivest, A. Shamir y L. Adleman). RSA es un sistema criptográfico de llave pública que brinda seguridad, confidencialidad e integridad de la información que navega en la Web.

Ante la gran acogida que tuvo el algoritmo RSA, distintas investigaciones realizadas por la comunidad científica demostraron que: (i) El tiempo para generar las claves tanto del cifrado y descifrado, conllevan un gran cantidad de cálculos matemáticos, que origina su relativa lentitud, lo cual afecta directamente al rendimiento de la red (Gil, 2000). (ii) La seguridad radica en la factorización del producto de los dos números primos, que por garantía requiere una mayor longitud de la clave, lo que implica un incremento en el costo computacional (Yaday, Pankaj, & Yaday, 2012). (iii) Por otro lado al seleccionarse valores pequeños para la clave, el proceso de cifrado es débil y la seguridad disminuye, por lo que facilita el descifrado del mensaje y logra que el algoritmo sea vulnerable (Liu, Li, & Hao, 2011).

Estos problemas han dado lugar a nuevas investigaciones que permitan mejorar tanto el modelo matemático como la algoritmia, para resolver con ello la complejidad computacional y sobre todo la seguridad de la información.

## 1.3. Hipótesis

La optimización del algoritmo de encriptación asimétrica RSA, mediante una nueva algoritmia garantiza la disminución del costo computacional y la incrementación de la seguridad, la velocidad de cálculo y la mitigación de ataques en las redes.

## 1.4. Antecedentes

La criptología es considerada una ciencia, que se ocupa de las escrituras secretas, e incluso de los aspectos relacionados con la nuevas tecnologías y telecomunicaciones (Bosch, 2006). Esta ciencia se divide en varios campos, sin embargo en esta investigación se hace mención especial a la Criptografía.

La criptografía es el término que describe las técnicas de cifrado, destinadas a garantizar la confidencialidad, disponibilidad e integridad de la información. De hecho esta ciencia es una subdivisión de las matemáticas aplicadas y de las ciencias de la computación.

Antes del siglo XX, esta ciencia fue considerada ventajosa para los asuntos políticos y militares. Durante las dos guerras mundiales, se produjo impactos criptográficos que cambiaron el rumbo de la historia. Uno de ellos y el más importante fue la ruptura del cifrado de la máquina alemana Enigma, a través de la máquina de cálculo gigante Colossus, denominada precursora de los ordenadores modernos (Gil, 2000).

Sin embargo, no fue hasta 1976 que las ideas de Claude Shannon, considerado como "El padre de la teoría de la información", fueron usadas para desarrollar el diseño lógico del cifrado, conocido como Estándar de Cifrado de Datos (DES). No obstante ese mismo año W.Diffie y M. Hellman, desarrollaron una nueva forma de cifrar la clave pública a través de cálculos matemáticos (Gil, 2000).

La clave pública se basa en las llamadas funciones unidireccionales con trampa, cuya característica es aplicar funciones fáciles de calcular, pero difícil de invertirlas. Sin embargo, la seguridad de esta es la dificultad de resolver algún problema, un ejemplo de este sistema es: el cifrado de clave pública RSA (Bosch, 2006).

El método de encriptación RSA (Rivest, Shamir, Adleman) es ampliamente utilizado para el cifrado en el mundo. Se basa en el análisis numérico, que consiste en factorizar el producto de dos grandes números primos escogidos al azar. Así, el producto puede ser utilizado como clave de cifrado, y los dos números primos como no son factibles serán usados como claves de descifrado. De esta manera si un atacante factoriza el módulo de "n", entonces la seguridad de RSA está rota y cualquiera puede descifrar fácilmente el mensaje (Patidar & Bhartiya, 2013).

Ante este escenario y en razón de que el sistema criptográfico RSA, juega un papel importante en la protección de la información, la comunidad científica orientó sus esfuerzos en optimizarlo, al desarrollar nuevos esquemas de cifrado de clave pública, que intentan resolver la complejidad computacional. Así por ejemplo, el trabajo

propuesto por Gupta y Sharma (2012), formularon un algoritmo de cifrado híbrido, basado en el algoritmo RSA y el algoritmo de Diffie Hellman, con el objetivo de incrementar la seguridad. Nagar (2012), presenta un nuevo método para intercambiar índices que contienen los valores de las claves públicas y privadas almacenadas en una base de datos. Patidar et al. (2013), proponen el uso de un tercer número primo, con el fin de acelerar el intercambio de datos a través de la red, incluso el módulo n se vuelve difícil de descomponer. Wang (2013), aplica la teoría euclidiana extendida, con el fin de obtener las claves para resolver el problema de la transmisión. Minni et al. (2013) tratan de optimizar el RSA, eliminando la distribución de "n", que al encontrar sus factores comprometerá la seguridad del algoritmo.

Aunque los trabajos anteriores se preocupan del problema del desempeño del RSA, ninguno de ellos hace énfasis en cómo mejorar el nivel de seguridad del mismo. Además tampoco, aún no se ha logrado una solución genérica, debido a que estos procedimientos se enfocan en el consumo elevado de recursos y costos computacionales, sin establecer un proceso de Ingeniería de Software.

### 1.5. Justificación e Importancia

En cuanto al incremento de amenazas, vulnerabilidades en las redes y el desarrollo de las Tecnologías, existen nuevas formas de cifrar los datos, con el fin de mantener la confidencialidad, integridad y disponibilidad de la información.

En particular los métodos de encriptación se enfocan en dos aspectos, el primero en la seguridad de la información y las telecomunicaciones. El segundo en la problemática propia del RSA. En el primer caso, este proyecto se justifica dada a la inseguridad de las redes, por ataques externos e internos que hoy en día son más frecuentes. En relación a la encriptación asimétrica se identifican los siguientes problemas: (i) La complejidad de la solución matemática que utiliza la factorización de números enteros (Johnsonbaugh, 2005); (ii) la vulneración del algoritmo (López P. , 2014); y (iii) existen defectos en la librería criptográfica de java (Zulfikar, 2006).

De acuerdo a estas y a las anteriores consideraciones se requiere optimizar el algoritmo de encriptación RSA, con el fin de aumentar la velocidad de cálculo, además de incrementar la seguridad de los mensajes que son transmitidos por la red, al generar

un modelo de software que utiliza la combinación de técnicas de programación estructurada y orientada a objetos, con un enfoque a la evaluación del algoritmo criptográfico asimétrico y su optimización, con el objetivo de disminuir el impacto de ataques en las redes.

## 1.6. Alcance del Proyecto

El software a desarrollarse utiliza la combinación de técnicas de programación estructurada y orientada a objetos, que emplea iteración y recursividad, apoyado en un modelo matemático, que permite mejorar el método e implementación de la algoritmia para la transmisión segura de mensajes en la red.

Se inicia con un estudio de los sistemas criptográficos simétricos y asimétricos, la misma que describe y analiza los posibles escenarios que se pueden presentar. Con los conceptos básicos y el análisis de los sistemas criptográficos, se establecerá el algoritmo de encriptación asimétrica RSA, para determinar su funcionamiento, algoritmo y método. Dicho método parte de una expresión matemática que optimiza el RSA y que da lugar al diseño e implementación de una solución genérica de cifrado.

Posteriormente para el diseño y desarrollo del algoritmo criptográfico se contempla lo siguiente:

- El software está desarrollado en lenguaje Java, tanto para el sistema operativo Windows como para Linux, haciendo uso del IDE[1] de Netbeans versión 7.4.
- Cuenta con una base de datos en MySQL, los parámetros almacenados permiten calcular el valor RSA y un arreglo de índices generados aleatoriamente que constituye al código.
- Se procede a utilizar una Llamada a un Proceso Remoto (RPC), para acceder a la base de datos creada, con el fin de obtener el juego de caracteres imprimibles del ASCII extendido (alfabeto) y el código.
- Para optimizar el algoritmo RSA se ha validado el uso del juego de caracteres imprimibles del ASCII extendido.

[1] Integrated Development Enviroment. Programa compuesto por herramientas, utilizada para desarrollar código. (Nixon, 2014)

- La aplicación se diseñó y construyó, para funcionar en plataformas Windows y Linux.
- La interfaz es de fácil utilización y amigable con los usuarios.
- La aplicación despliega el acceso válido o rechazado a la base de datos, la cadena de caracteres cifrada como comprobación de su validez y el tiempo que se demora tanto en el cifrado como descifrado.

Finalmente se realiza pruebas de rendimiento a fin de evaluar el sistema, servicio y la red a través de un análisis estadístico, que determinó la validez de la optimización del algoritmo.

## 1.7. Objetivos

### 1.6.1. General

Optimizar el algoritmo de encriptación asimétrica RSA a través de un modelo matemático que da origen a una nueva algoritmia y librería en Java, para garantizar el incremento de la seguridad y la velocidad de cálculo, además disminuir los costos computacionales, para el control y la mitigación de ataques en las redes.

### 1.6.2. Específicos

- Evaluar los sistemas criptográficos asimétricos con el fin de determinar la problemática y nivel de seguridad, para analizar los posibles escenarios que se pueden presentar.
- Definir el algoritmo línea base de la encriptación asimétrica RSA, para determinar su funcionamiento, algoritmo, métodos y herramientas.
- Determinar un modelo matemático que permita optimizar el algoritmo de encriptación RSA.
- Diseñar e implementar el algoritmo experimental que permita optimizar el RSA en base al modelo matemático esperado, para la generación de las claves, la encriptación y la desencriptación RSA.
- Diseñar e implementar una librería de clases y funciones que convierta una solución genérica.
- Realizar el suficiente número de pruebas, evaluar los resultados de manera estadística, validarlos, analizarlos e interpretarlos para difundir los resultados.

## 1.8. Metodología

Para el desarrollo de este software, se adoptó la metodología ágil Proceso Unificado Ágil (Agile Unified Process, AUP) desarrollada por Scott Ambler. Esta metodología consta de cuatro etapas: Concepción, Elaboración, Construcción y Transición. En la Figura 1 se observa las fases del proceso (Ambler, 2006).

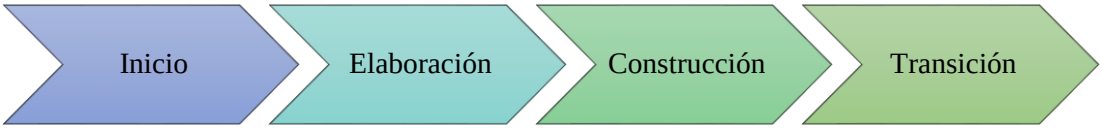


**Figura 1. Fases del Proceso Unificado Ágil (AUP)**

**Fuente:** (Ambler, 2006)

A continuación se describe cada una de sus fases:

- **Inicio:** Se realizó una evaluación de los sistemas criptográficos simétricos y asimétricos, lo que permitió un estudio metódico del modelo de llave pública RSA, que da origen al método línea base.
- **Elaboración:** Está definida por la arquitectura básica e incluye el desarrollo de un diagrama de secuencia, donde se profundiza las acciones del sistema. Por otro lado se elaboró un diagrama de clases, para especificar la relación entre objetos que contienen operaciones específicas, que serán implementadas a través de una Interfaz de Programación de Aplicaciones (API).
- **Construcción:** Se realizó el diseño de optimización, para el mecanismo de encriptación RSA, y se transformó el modelo en código ejecutable, para realizar las diferentes pruebas (individuales, de integración).
- **Transición:** El programa se sometió a pruebas de validación y aceptación, para que así se despliegue en el sistema de producción.

De acuerdo a esta metodología se trabaja a través de disciplinas que permiten desarrollar, validar y entregar el software de acuerdo a las necesidades del usuario. A continuación se detalla las disciplinas usadas:

- **Modelado:** Encontrar una solución genérica que priorice el aumento de la seguridad y la disminución de recursos computacionales;

- **Aplicación:** Transformar el modelo de línea base y optimizado a código entendible y ejecutable;
- **Prueba:** Validar que el sistema funcione de acuerdo al diseño y a los requisitos establecidos, que garantice la calidad del software;
- **Despliegue:** Presentación del software a través del IRC (Internet Relay Chat)[2], que da inicio a una conversación en tiempo real, haciendo uso del algoritmo implementado;
- **Gestión de configuración:** Controlar las versiones de las herramientas usadas, para el desarrollo de este software. Además gestionar los usuarios y contraseñas a usar en la base de datos;
- **Gestión de proyecto:** Seguimiento de actividades mediante los progresos, para la optimización del algoritmo, a través del incremento del nivel de seguridad;
- **Entorno:** Garantizar que el ancho del canal y el procesador no sean restricción, para el proceso adecuado del algoritmo RSA.

## 1.9. Herramientas

A continuación se describe las herramientas elegidas, para el desarrollo del software:

- Windows 8.1: Sistema Operativo introducido por Microsoft, para gestionar tanto hardware como software. Esta versión fue diseñada para adaptarse a la computación moderna, que permite a los usuarios autenticarse con una cuenta de Microsoft, para almacenar y compartir archivos (Northrup, 2013).
- Ubuntu 14.04: Sistema Operativo basado en Linux, desarrollado bajo una licencia libre y de código abierto. Linux permite administrar y mantener ordenadores en buen estado (Ramirez, 2011)
- Power Designer versión 16.1.0.3637: Herramienta producida bajo Microsoft Windows, que es un estándar para los modelos de diseño de software y proporciona un código automatizado de ingeniería inversa y la generación de plantillas personalizadas. (Sybase, 2011).

[2] Proporciona un medio por el cual un usuario puede escribir un mensaje en tiempo real a uno o más usuarios de Internet, y casi instantáneamente, el mensaje aparece en los monitores de todos los demás usuarios que están supervisando la transmisión (Simpson, 2000)

- NetBeans IDE versión 7.4: Entorno de Desarrollo Integrado (IDE) de código abierto, que permite realizar aplicaciones esencialmente con el lenguaje de programación Java (Böck, 2011).
- MySQL versión 5.6.23-log: Base de datos de código abierto con un lenguaje de consulta estructurada (SQL) (Nixon, 2014).

# CAPÍTULO 2

# MARCO TEÓRICO

En el presente capítulo se describe el marco teórico referencial que fundamenta el desarrollo del software para la Optimización del Algoritmo de Encriptación Asimétrica RSA.

## 2.1. Ingeniería de Software

"La Ingeniería del Software es una disciplina de la ingeniería que comprende todos los aspectos de la producción de software desde las etapas iniciales de la especificación del sistema, hasta el mantenimiento de éste después de que se utiliza" (Sommerville, 2005).

Entre varios conceptos que se le acredita a la Ingeniería de Software, se hace énfasis a la elaborada por la IEEE con un significado claro y evidente: "Es la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento del software; es decir, la aplicación de la ingeniería de software" (IEEE, 1993).

La Ingeniería de Software ofrece métodos y técnicas para producir software de una manera organizada y sistemática que garantice su alta calidad. La Figura 2 muestra los estratos que sustenta la producción del software.

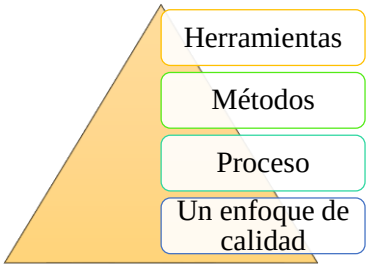


**Figura 2. Estratos de la Ingeniería de Software**

**Fuente:** (Pressman, 2005)

Tal como lo ilustra la Figura 2 la Ingeniería de Software se encuentra estratificada en cuatro partes: (i) Un enfoque de calidad, que visualiza una mejora continua para

lograr el desarrollo efectivo del software; (ii) Proceso, que es la base para mantener juntos los estratos de la tecnología que forma una gestión de proyectos y establece métodos técnicos, para generar los productos del trabajo que asegure la calidad y el manejo apropiado de cambios; (iii) Métodos, que son principios técnicos como la comunicación, requisitos, modelado, construcción, pruebas y soporte que garantizan la construcción del software; y (iv) Herramientas que brindan el soporte automatizado para el proceso y métodos (Pressman, 2005). En el presente trabajo se hace uso de la metodología ágil denominada Proceso Unificado Ágil.

## 2.2. Metodología AUP (Proceso Unificado Ágil)

Es una versión abreviada del conocido Proceso Racional Unificado (Rational Unified Process, RUP), desarrollada por Scott Ambler, anunciada en septiembre de 2005. Su versión actual fue publicada el 13 de mayo de 2006 (Ambler, 2006).

### 2.2.1. Definición

Es un enfoque al desarrollo del Software que utiliza técnicas y conceptos ágiles, para mejorar la productividad del software, que aún se mantiene fiel a las normas dictadas en RUP. Tiene una orientación al desarrollo basado en pruebas (Test Driven Development, TDD), modelado ágil (Agile Model Driven Development, AMDD), gestión del cambio ágil y refactorización de Base de Datos. No obstante se enfoca en la gestión de riesgos, desde las primeras etapas del proyecto y que al final demostrará la validez del producto.

### 2.2.2. Principios

Según Ambler (2006) la metodología AUP, se encuentra apoyada con varios principios:

- El personal sabe lo que está haciendo, es decir, que el grupo no necesita de documentación, sin embargo en ocasiones recurren a una orientación de alto nivel;
- Simplicidad, se describe todo el procedimiento de manera concisa y directa;
- Agilidad, ajuste de los valores y principios del RUP;
- Concentrarse en actividades de alto valor, dicho de otra manera es centrarse en actividades que son importantes en el desarrollo del proyecto;

- La independencia de herramientas, que es el uso de herramientas adecuadas para el trabajo;
- Adaptar el producto, para satisfacer sus necesidades.

### 2.2.3. Ciclo de vida

La Figura 3, muestra el ciclo de vida ágil de la AUP. Como se observa la disciplina denominada Modelo, abarca una parte importante del proceso pero no la domina, más bien se encarga del modelado de negocios, requisitos, análisis y diseño. El resto de las disciplinas, es decir implementación, prueba, despliegue, gestión de configuración y proyecto, y entorno coinciden con disciplinas de RUP.

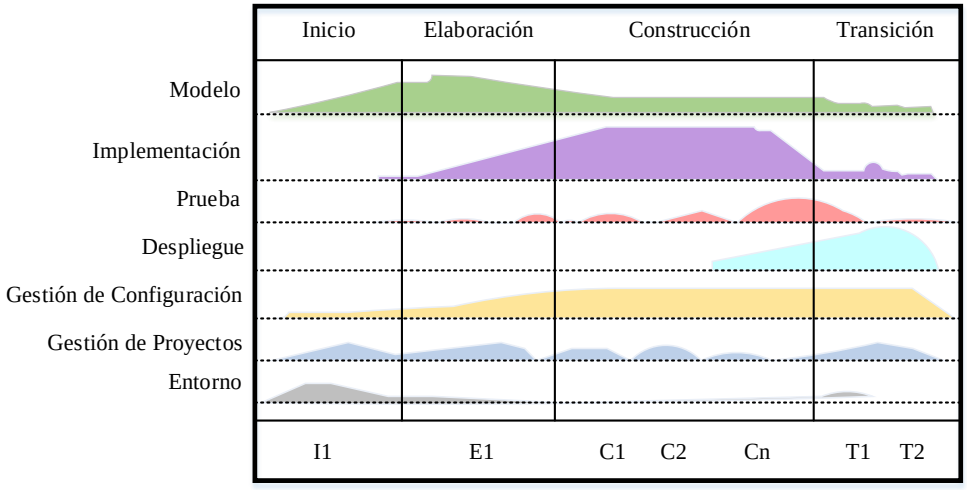


**Figura 3. Ciclo de vida de la Metodología de Proceso Unificado Ágil**

**Fuente:** (Ambler, 2006)

#### 2.2.3.1. Fases

En la siguiente Tabla 2, se describe las cuatro fases de la metodología Ágil UP, estas ocurren de manera sucesiva.

**Tabla 2**

***Fases de la Metodología AUP***

| Etapas | Descripción |
|---|---|
| **Inicio** | Definir el alcance y una arquitectura apta al proyecto, para obtener una aceptación de parte del usuario. |

**CONTINUA** →

| | |
|---|---|
| **Elaboración** | Demostrar la validez de la arquitectura del proyecto. |
| **Construcción** | Desarrollo del software de forma regular e incremental que compense las necesidades de mayor importancia. |
| **Transición** | Validación y presentación del software en el entorno de trabajo. |

Fuente: (Ambler, 2006)

#### 2.2.3.2. Disciplinas

En la Tabla 3, se detalla cada una de las disciplinas. Estas se realizan de manera iterativa, además son actividades que permiten construir, validar y entregar el software con todos los requisitos impuestos desde un principio.

**Tabla 3**

***Disciplinas de la Metodología AUP***

| Disciplinas | Descripción |
|---|---|
| **Modelo** | Entender el problema del proyecto para definir soluciones aptas para resolverlo. |
| **Implementación** | Transformar el modelo a código ejecutable y realizar pruebas básicas. |
| **Prueba** | Evaluar el sistema para detectar fallas y verificar el cumplimiento de los requisitos y así garantizar la calidad del mismo. |
| **Despliegue** | Planificación para la entrega y ejecución del sistema, para que se encuentre disponible a los usuarios. |
| **Gestión de Configuración** | Realizar un seguimiento de las versiones de las herramientas a usar, al igual que controlar y gestionar los cambios respectivos. |
| **Gestión de Proyectos** | Definir las actividades que permitan tener un resultado óptimo en todo sentido. |
| **Entorno** | Garantiza que el proceso sea el adecuado, la orientación y las herramientas sean disponibles para el uso adecuado del sistema. |

Fuente: (Ambler, 2006)

### 2.2.4. Uso de la metodología AUP en el desarrollo de este proyecto

Aunque esta metodología se preocupe especialmente de los elementos que contengan mayor prioridad. Se aborda en etapas tempranas todos los requisitos técnicos que sugiere el usuario, para poder demostrar la validez del producto (Ambler, 2006).

Para garantizar la eficiencia de este proyecto se utiliza el AUP, debido a su proceso simplificado, ya que se incluye explícitamente actividades y herramientas que se usan mediante el desarrollo de este proyecto. Además que contiene una técnica para dirigir las pruebas, que permitan mejorar la productividad del sistema.

## 2.3. Seguridad

La seguridad es una característica, cualidad, síntoma, ambiente, etc. que el hombre siempre ha buscado desde su origen hasta los presentes días. El ser humano ha buscado sentirse seguro mientras desarrolla sus actividades y ahora con el avance de la tecnología y desarrollo de aplicaciones que funcionan sobre el Internet surge la necesidad de generar metodologías, mecanismos y técnicas que permitan generar un nivel de seguridad aceptable para el ser humano.

### 2.3.1. Seguridad Informática

En la actualidad, la informática y las diferentes plataformas tecnológicas disponibles, desempeñan un papel importante en la vida cotidiana del hombre, empresas e instituciones públicas y privadas, además, el solo hecho de que un desastre ocurra ocasionaría la interrupción de los servicios informáticos y con ello el acceso a la información. (Aldegani, 1997)

La seguridad informática abarca múltiples áreas relacionadas a los sistemas de información, de los cuales también existen principios y metodologías para su cuidado. Estas áreas se pueden agrupar en dos grandes bloques o niveles, la protección física, como componentes de hardware, y la protección lógica, que incluye los componentes de software (Ver Figura 4). A continuación se presenta una leve descripción de cada uno de ellos (Aldegani, 1997).

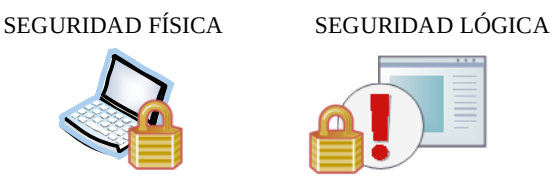


**Figura 4. Seguridad Física y Lógica**

#### 2.3.1.1. Seguridad Física

Se entiende como el conjunto de medidas de prevención, barreras físicas y procedimientos de control que permiten restringir, vigilar y conocer el acceso físico a un elemento del sistema informático, como contramedidas ante amenazas a los recursos, como computadores, elementos de red e información dentro y alrededor del sistema informático. Dentro de las medidas de seguridad se encuentra el acceso de personas no autorizadas a dispositivos de hardware y almacenamiento que integran la infraestructura o plataforma informática (Villalón, 2002).

#### 2.3.1.2. Seguridad Lógica

La seguridad física no constituye el único mecanismo para evitar ataques al sistema informático, hay que mencionar que la mayoría de daños que ocurren o sufren los sistemas, provienen de información almacenada y procesada por la plataforma informática.

El mayor atractivo que contiene un sistema informático es la información que procesa, transmite y almacena, por lo tanto se deben generar protocolos, algoritmos o programas que pueden manipular la información y controlan el acceso a la misma, es decir, aplicar barreras que resguarden el acceso a la información. Los cifrados de mensaje forman parte de la seguridad lógica. Existen objetivos claros que ayudan a mantener un nivel alto de seguridad lógica (Latham, 1983):

- Restringir y controlar el acceso a los programas y archivos;
- Asegurar que miembros no autorizados no puedan modificar programas ni archivos. El término que se adapta a este objetivo es la confidencialidad;
- Sólo el receptor debe recibir la información transmitida;
- La integridad de la información debe mantener hasta que el receptor reciba la información transmitida

Evaluar y controlar permanentemente la seguridad física y lógica es la base fundamental, para poder disminuir siniestros ataques y pérdidas a los equipos del sistema o red informática, y en especial al principal recurso que por ella viaja, la información.

### 2.3.2. Seguridad en Redes

Durante las primeras décadas las redes de computadoras fueron utilizadas para el intercambio de correos electrónicos y compartir recursos, como impresoras o escáneres. Con el crecimiento de las aplicaciones que el hombre ha dado a las redes como transacciones bancarias, declaraciones de impuestos, compras de boletos aéreos, etc. (Aldegani, 1997). La seguridad pasó a ser un eje fundamental que integra la red de computadoras, y los profesionales de la informática deben generar y aportar soluciones para garantizar la seguridad del componente más importante de la red, la información (López M. J., 2003). A continuación se trata la seguridad en redes mediante dos grupos, las redes internas y externas.

#### 2.3.2.1. Redes Internas

Son redes de tamaño pequeño que habitualmente cubren áreas geográficamente limitadas como por ejemplo una casa u oficina. Dentro de estas redes podemos encontrar las LAN (Local Area Network) y WLAN (Wireless Local Area Network).

Al ser pequeñas permite tener un control total sobre el canal de comunicaciones, donde la seguridad física evita la falta de privacidad de información. Los riesgos más comunes y sus posibles soluciones se detallan a continuación.

- **Pérdidas de información:** Estas se dan debido a fallos físicos del sistema, como cortes de energía, daño de discos duros, etc. Lo que se recomienda es crear copias de respaldo periódicamente y almacenarlas en sitios externos a la red;
- **Uso inadecuado de la información:** este tipo de problema proviene de los propios usuarios. Para minimizar este riesgo, se recomienda evitar usuarios con demasiados permisos o privilegios, así como los "super usuarios";
- **Suplantación de identidad:** es común encontrarse con usuarios no identificados o equipos conectados a la red, estos pueden realizar un análisis de tráfico e incluso capturar información de nuestra red. Para ello se debe deshabilitar el uso de conexiones dinámicas y usar identificadores a través de protocolos de autenticación;

- **Espionaje:** al utilizar redes inalámbricas (WLAN) se utiliza ondas de radio como canal de comunicaciones, en este caso se pierde control físico del mismo, por lo que la seguridad física es muy flexible. En este caso la seguridad lógica juega un papel muy importante al utilizar protocolos de cifrado.

(Aldegani, 1997), (López M. J., 2003)

#### 2.3.2.2. Redes Externas

Las redes externas son geográficamente amplias o de mayor tamaño que las internas. Este tipo de redes se apoyan o hacen uso de canales de comunicaciones ajenos a ellas. Al ser este canal ajeno, no se cuenta con un control de la información que viaja sobre el mismo, lo que genera la necesidad de crear o utilizar técnicas de protección efectivas sobre la información, como por ejemplo el uso de la Criptografía.

Existen dispositivos que permiten minimizar los riesgos que existen al utilizar o interconectar redes internas mediante canales de uso compartido o ajeno, uno de los más utilizados es el cortafuegos.

- **Cortafuegos:** dispositivo intermedio, entre la red interna y el canal de comunicación, que filtra el tráfico entrante y saliente de la misma.

Según Aldegani (1997) los potenciales peligros que pueden comprometer la información desde una red externa son:

- **Ataque discriminados:** son los más comunes y menos dañinos. Entre estos se encuentran los virus. El mecanismo de protección es un antivirus que se actualice periódicamente.
- **Ataques a medida:** ataques menos frecuentes que son generados por los hackers. Los hackers siempre buscan grandes corporaciones y muchas veces no lo hacen por la información, sino por demostrar su capacidad para vulnerar

### 2.3.3. Seguridad de la Información

La Real Academia de la Lengua Española define el término seguridad como “cualidad de seguro” y a la palabra seguro como “libre y exento de todo peligro, daño o riesgo”. Combinando estos términos con los sistemas informáticos, es decir, con todo aquel sistema que procese, almacene o transmita información se puede llegar a generar

una definición de seguridad informática como "Un conjunto de métodos y herramientas destinados a proteger la información y por ende los sistemas informáticos ante cualquier amenaza" (Ramió, 2006). La seguridad informática es un proceso para que las personas estén al tanto de los potenciales peligros o riesgos que se pueden presentar, y así poder evitarlos o minimizarlos.

Garantizar la seguridad de un sistema informático, requiere de sistemas, técnicas muy amplias y complejas que abarcan diferentes áreas y factores internos y externos al sistema, que permitan cumplir con los conceptos de integridad, confidencialidad y disponibilidad de la información que dejan considerar a un sistema como seguro.

En este trabajo de investigación se hace referencia a métodos criptográficos como a herramientas que ayudan a potenciar la seguridad en los sistemas informáticos y en las redes de comunicaciones.

#### 2.3.3.1. Principios de seguridad

Existen tres principios o ideas fundamentales (Ver Tabla 4) que la seguridad informática establece para lograr su objetivo como se muestra también en la Figura 5.

**Tabla 4**

***Principios de seguridad***

| Principios | Características |
|---|---|
| **Confidencialidad** | Es la necesidad de que la información se mantenga fuera de las manos de personas no autorizadas. |
| **Integridad** | Es la característica que permite que la información permanezca sin alteraciones. Solo el personal autorizado podrá alterar dicha información. |
| **Disponibilidad** | Llamada operatividad de la información, es la capacidad para estar disponible en cualquier momento a las personas autorizadas para su procesamiento |
| **Autenticación** | Se verifica quién elaboró el documento y cuál es el destinatario |
| **No repudio** | Autenticar que el emisor envió la información y que el receptor lo recibió |

Fuente: (Tanenbaum, 2003)

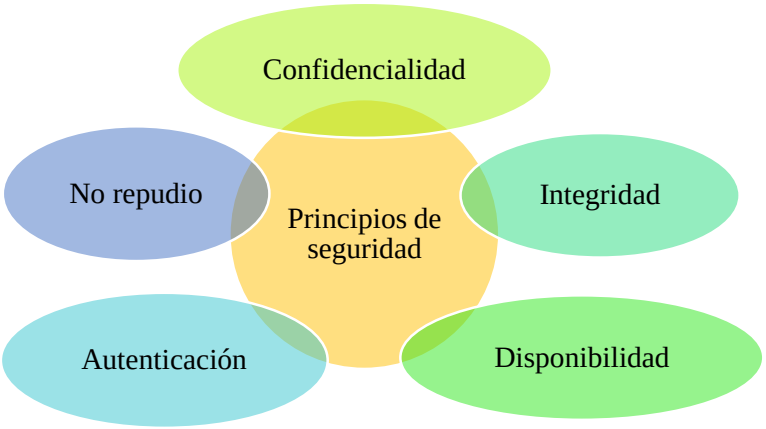


**Figura 5. Principios de seguridad**

Adicionalmente se pueden mencionar dos aspectos que tienen relación con los anteriores (Ver Tabla 5):

**Tabla 5**

***Aspectos de la seguridad***

| Aspectos | Detalles |
|---|---|
| **Control** | Este principio permite establecer y asegurar cómo y cuándo los usuarios autorizados pueden acceder a la información. |
| **Autenticidad** | Confirmar la comunicación con el usuario correcto antes de revelar información delicada o hacer una transferencia de la misma. Del mismo modo permite establecer si el origen y destino de la información es real. |

Fuente: (Aldegani, 1997)

### 2.3.3.2. Elementos de seguridad

Existen elementos de seguridad que tienen una gran relación con los aspectos fundamentales mencionados en el apartado anterior, los mismos que incorporan características particulares al tema de seguridad (Ver Tabla 6).

**Tabla 6**

***Elementos de seguridad***

| Elementos de seguridad | Características |
|---|---|
| **Protección a la réplica** | Este elemento permite evitar que una transacción se pueda ejecutar o reproducir dos veces, al menos que se especifique lo contrario. |
| **No repudio** | Permite que cualquier ente que envíe o reciba un mensaje no pueda manifestar ante organismos que no ha enviado o recibido dicha información. |
| **Consistencia** | El sistema informático y el procesamiento de la información se debe hacer de la forma en que fue planificado o diseñado |
| **Aislamiento** | Establece un control y regulación del acceso al sistema. Impide que personas no autorizadas hagan uso del mismo. |
| **Auditoria** | Permite establecer las acciones y procesos que se llevan a cabo en el sistema, así como saber quién y cuando accedió a determinada información |

Fuente: (Aldegani, 1997)

## 2.4. Criptografía

La seguridad de la información ha sido un tema de gran importancia que ha ido evolucionando con el tiempo. Junto con ella lo ha hecho de la misma forma la Criptografía. Ambos han tenido un interés bastante limitado, pero al ser una disciplina de rápida evolución, éste interés se ha convertido en algo fundamental que ya no solo el profesional en seguridad debe conocer, sino que obliga a usuarios promedios a tener ciertos conocimientos básicos sobre seguridad. La constante evolución y crecimiento del Internet, así como la convergencia de aplicaciones obliga a usuarios de cualquier tipo como expertos, promedio o básicos a utilizar herramientas de seguridad para resguardar su información personal.

La definición de criptografía según el diccionario de la Real Academia es el "arte de escribir con clave secreta o de un modo enigmático" (Española, 2012), esta definición ha quedado un poco obsoleta ya que la criptografía ha pasado de ser un arte a una ciencia, por lo que una definición más técnica sería: "Rama inicial de las Matemáticas y en la actualidad también de la Informática y la Telemática, que hace

uso de métodos y técnicas con el objeto principal de cifrar, y por tanto proteger, un mensaje o archivo por medio de un algoritmo, usando una o más claves" (Ramió, 2006), es decir, es la ciencia que se encarga de ocultar o dificultar la lectura de mensajes o información de carácter confidencial a través del diseño de procedimientos de cifrado.

Hay que recalcar la diferencia entre cifrar y codificar, ya que codificar es presentar una determinada información de forma distinta pero con igual significado y cifrar utiliza métodos lingüísticos o matemáticos para hacer ilegible un mensaje. Además de estas definiciones, se debe mencionar al criptoanálisis. El criptoanálisis es el proceso inverso al cifrado, es decir, acceder de forma ilegítima a un mensaje mediante el quiebre de los procedimientos utilizados para su cifrado.

Para encriptar un mensaje se necesita un texto plano, al cual se lo combina con un algoritmo y una clave para producir un texto cifrado o un criptograma, como se muestra en la Figura 6 los requisitos para cifrar.

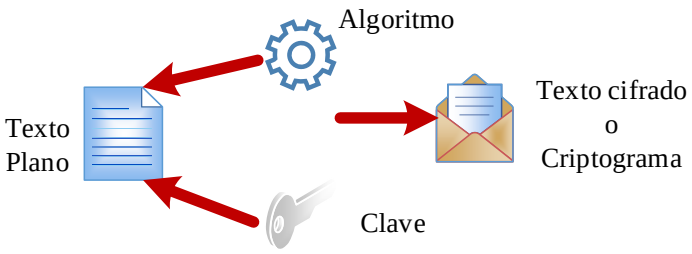


**Figura 6. Requisitos para el cifrado**

El mensaje cifrado debe viajar por un canal de comunicación público o privado hasta llegar a su destino. Durante este transcurso o recorrido del mensaje, por el canal de comunicación, puede ser interceptado y sometido a un proceso de criptoanálisis. Es por eso que un buen sistema de criptografía debe incluir un cifrado sencillo, pero un criptoanálisis o descifrado muy difícil o imposible.

### 2.4.1. Historia

La criptografía existe desde tiempos inmemorables. Los primeros indicios de cifrado datan del antiguo Egipto hace 4500 años donde usaban un "cifrado" por sustitución más no por ocultar cosas o mensajes sino por dar un toque de misterio o dramatismo a la historia que se representaba (Nájera, 2013).

Otro uso de cifrado lo realizaron los Espartanos para proteger sus mensajes, ellos utilizaban una técnica conocida como cifrado por transposición. Esta técnica consistía en enrollar un pergamino sobre una escítala (estaca o vara), que servía para ordenar las letras y mostrar el mensaje. Para poder descifrarlo, el receptor debía contar con una escítala del mismo diámetro (Nájera, 2013).

Durante el Imperio Romano Julio César empleó un sistema de cifrado, que consistía en sustituir la letra a cifrar por otra letra distanciada a tres posiciones más adelante. En el siglo XII Roger Bacon y en el siglo XV León Batista Alberti inventaron y publicaron sendos algoritmos de encriptación, basados en modificaciones del método de Julio César (López M. J., 2003).

Desde la Segunda Guerra Mundial el cifrado toma un gran interés a nivel global, especialmente en las comunicaciones del ejército alemán. En 1923 Arthur Scherbius patenta la máquina ENIGMA. Gracias a esta máquina las comunicaciones alemanas eran indescifrables. Enigma era un instrumento de apariencia simple, parecida a una máquina de escribir. La persona que deseara cifrar un mensaje sólo tenía que teclearlo y las letras correspondientes al mensaje cifrado se irían iluminando en un panel. El destinatario copiaba dichas letras en su propia máquina y el mensaje original aparecía de nuevo. La clave la constituían las posiciones iniciales de tres tambores o rotores que el ingenio poseía en su parte frontal (López M. J., 2003).

### 2.4.2. Evolución

La criptografía evolucionó luego de la Segunda Guerra Mundial. En 1948, Claude Shannon publicó su artículo “A Communications Theory of Secrecy Systems”, donde se modernizaron las técnicas de codificación para transformarlas en procesos matemáticos avanzados. Shannon introduce un nuevo concepto “distancia de unicidad” que marca la longitud de un texto cifrado que se necesita para poder descifrarlo (Shannon, 1949).

La evolución de la criptografía se da gracias al avance tecnológico y al crecimiento en el nivel de procesamiento de datos de las computadoras, las cuales se convierten en pieza y herramienta clave en el proceso de cifrado. En 1976, IBM desarrollo el algoritmo de cifrado DES (Data Encryption Standard) que forma parte de la criptografía simétrica. En los mismos años 70’s Whitfield Diffie y Martin Hellman a través de su artículo “New Directions in Cryptography” establecen las bases de la

criptografía asimétrica, hoy en día utilizada en transacciones utilizadas en Internet en páginas con protocolo HTTPS, firmas digitales o para cifrar mensaje utilizando PGP (Pretty Good Privacy, combinación de criptología simétrica y asimétrica) (Gil, 2000).

### 2.4.3. Justificación

La criptografía clásica fue creada en sus principios para mantener la confidencialidad de los mensajes. Según Sanjuan (2012) hoy en día, la criptografía es utilizada para mantener las comunicaciones seguras y cumplir con tres propósitos que justifican su uso:

- **Mantener la confidencialidad del mensaje**: solo puede ser visto por aquellos usuarios que tienen la autorización de ver la información.
- **Garantizar la autenticidad del destinatario y remitente**: certificar la identidad del emisor y receptor, y que el acceso a sus componentes solo puedan hacerlo los usuarios autorizados;
- **Garantizar la integridad en el mensaje**: la información enviada por el cliente debe ser la misma que llega al servidor, además que los componentes del mensaje solo puedan ser creados o modificados por los usuarios autorizados.

### 2.4.4. Tipos de Criptografía

La criptología o sistemas criptográficos se pueden clasificar en clásica, tratamiento del mensaje y por el tipo de clave (llave pública o privada que permite cifrar un documento y descifrar el correspondiente criptograma). (i) Según el cifrado clásico se divide en transposición y sustitución, fueron propuestas por Shannon; (ii) Por el tratamiento del mensaje se dividen en cifrado en bloque, donde el cifrado se lo realiza en bloques de 64 o 128 bits, y el cifrado en flujo, que se lo ejecuta bit a bit (Sanjuan, 2012); y (iii) De acuerdo al tipo de clave se puede clasificar en simétricos y asimétricos.

#### 2.4.4.1. Cifrado Clásico

Existen dos tipos de cifrado clásico (Ver Tabla 7) que se aplican básicamente a los caracteres del mensaje, además están orientadas a sistemas de clave privada.

**Tabla 7**

***Tipos de cifrado clásico***

| Cifrado | Características |
|---|---|
| **Sustitución** | Las letras del alfabeto corresponden a un símbolo igual o diferente. El emisor al enviar el mensaje sustituye por estos elementos, por otra parte el receptor tiene conocimiento de este cambio, de esta manera sustituye por el símbolo original para obtener el mensaje. |
| **Transposición** | Cambia el orden de los símbolos del mensaje original, es decir, el emisor envía los elementos pero colocados de forma distinta y el receptor los organiza a su posición original. |

Fuente: (Sanjuan, 2012), (Stallings, 2011).

### 2.4.4.2. Tratamiento del mensaje

Existen dos tipos de tratamiento de la información (Ver Tabla 8).

**Tabla 8**

***Métodos de Cifrado Simétrico***

| Cifrado | Características | Ventaja | Desventaja |
|---|---|---|---|
| **Bloque** | Agrupa el mensaje original en bloques de información, estos pueden ser caracteres, bits, etc. de tamaño fijo. | Inmunidad debido a que es imposible introducir bloque extraño sin detectarlo. | Baja velocidad de cifrado, ya que lee primero el bloque entero y es propenso a errores de cifra. |
| **Flujo** | El cifrado del mensaje, se lo realiza bit a bit o byte a byte, mediante un flujo de clave aleatoria y de mayor tamaño que el mensaje. | La velocidad de cifrado es rápido, ya que no toma en cuenta otros elementos y resistente a errores. | Es vulnerable, ya que pueden modificarse o alterarse los elementos por separado. |

Fuente: (Sanjuan, 2012), (Simson & Spafford, 2002)

### 2.4.4.3. Tipos de clave

#### Simétricos

Los métodos simétricos son aquellos en los que el emisor y receptor comparten una única clave. La clave se utiliza para los procesos de cifrado y descifrado. No obstante siempre tendrá un mismo tamaño (Ver Figura 7).

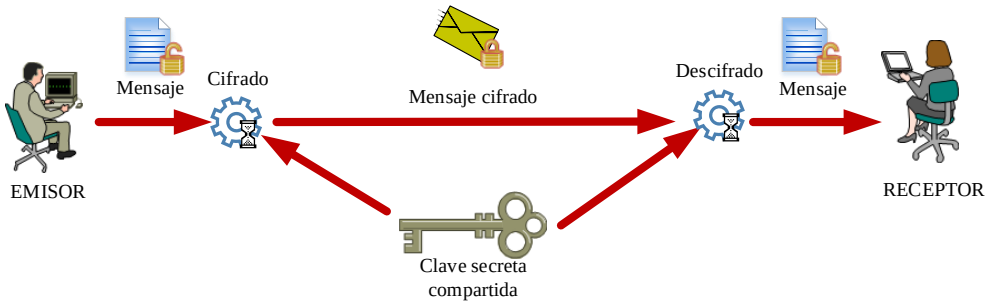


**Figura 7. Método simétrico**

El uso de una única clave requiere de seguridad, ya que al perderla los demás usuarios pueden descifrar los mensajes. Además, requiere de un canal seguro para proteger la clave (Stallings, 2011).

Las desventajas que presentan los sistemas criptográficos simétricos, a parte de la necesidad de la distribución de las claves, es el peligro de mantener, proteger y gestionar gran cantidad de claves diferentes. Las ventajas que presenta este tipo de criptografía es ser rápida y utilizar menos recursos computacionales. Como todo tipo de cifrado, colabora con la seguridad y prevención de riesgos de la información. Los algoritmos simétricos más conocidos son: DES, RC5 e IDEA. Estos algoritmos se describen en el CAPÍTULO 3.

#### Asimétricos

En este tipo de criptografía utiliza dos claves, una pública y la otra privada. En la Figura 8 se observa que estas claves son complementarias y funcionan solo en un sentido, es decir, una se utiliza en el lado de emisión para el cifrado y la otra para el proceso de descifrado en el lado de recepción (Stallings, 2011).

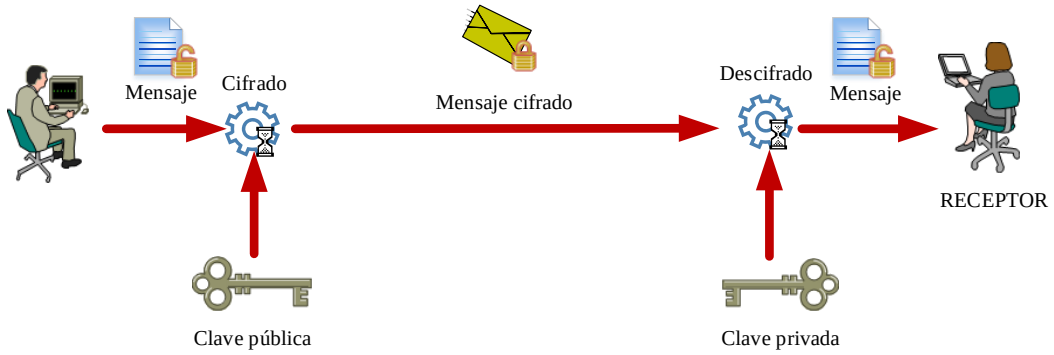


**Figura 8. Método Asimétrico**

La seguridad del sistema reside en la complejidad computacional de descubrir la clave privada a partir de la clave pública. Las longitudes de las claves empleadas en este tipo de cifrado son mayores a las utilizadas en el cifrado simétrico (Stallings, 2011). Gracias a esto su utilización ha sido de gran interés en redes de comunicaciones inseguras como el Internet. Los sistemas asimétricos más conocidos son: Diffie-Hellman, ElGamal, Curvas Elípticas y RSA. Estos algoritmos se describen en el CAPÍTULO 3.

## 2.5. Teoría de Números

La teoría de números es una rama de las matemáticas puras que se ocupa de los números naturales o enteros positivos. La historia dice que desde los años 5700 A.C. los sumerios contaban con calendarios que dio origen al desarrollo de la Aritmética. Tiempo después les siguieron los babilonios que ampliaron sus habilidades a calcular, pese a esto otras antiguas civilizaciones empezaron a especular sobre las propiedades del misticismo numérico. Los primeros estudios científicos que dan origen a la teoría de números se les atribuye a los griegos, por los años 600 A.C. No obstante alrededor de 300 A.C. surgió el estudio de los Elementos de Euclides convirtiendo a las matemáticas de la Numerología en una ciencia deductiva (Apostol, 1984). Aunque a finales del siglo XX la teoría de números revolucionó la seguridad de las comunicaciones, por adquirir una mayor importancia en los sistemas criptográficos (Johnsonbaugh, 2005).

### 2.5.1. Divisores

En esta sección se encuentra definiciones básicas como: (i) Números primos, y (ii) Máximo Común Divisor. Se hacen mención para una mejor comprensión del Algoritmo Euclidiano.

#### 2.5.1.1. Números primos

Un número es primo cuando es un entero mayor que 1, distinto de 0, y cuyos únicos divisores positivos son uno y él mismo (Johnsonbaugh, 2005).

Él número 1 se toma sólo por conveniencia, ya que no perjudica en nada y se tiene cierta ventaja en la declaración de teoremas (Mora, 2014)

*Ejemplo 1:* Los números primos son 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31,…….

**Principios:**

- Todo entero positivo n > 1 tiene un divisor primo;
- El número 2 es el único número primo que es par;
- Hay un número infinito de primos, es decir, por grande que sea un número primo siempre habrá otro número primo mayor (Arteaga & Santillán, 2013).

*Ejemplo 2:* El entero 17 es primo porque sus únicos divisores son 1 y 17.

Para determinar si un número es un entero positivo no primo (compuesto), se menciona a continuación dos maneras de comprobación:

(i) Probar si alguno de los número enteros *2, 3,……., n-1* divide a *n.* Si un entero divide a *n*, se denomina compuesto, caso contrario es primo.

(ii) Un entero positivo *n* mayor que 1 es compuesto si *n* tiene un divisor *d que satisface* $2 \leq d \leq \sqrt{n}$ (Johnsonbaugh, 2005).

#### 2.5.1.2. Máximo común divisor

El máximo común divisor de dos números enteros a y b (diferentes de cero) es el entero positivo más grande que divide a ambos a y b (Johnsonbaugh, 2005).

*Ejemplo 3: Calcular mcd (15,50).*

Los divisores positivos de 15 son: 1, 3, 5, 15.

Y los divisores positivos de 50 son: 1, 2, 5, 10, 25, 50.

Los divisores comunes de 15 y 50 son: 1, 5.

Por tanto el máximo común divisor de 15 y 50, mcd (15, 50)=5.

Aunque el método expuesto anteriormente es el más utilizado existe otro método donde a y b son números enteros, a > 1, b > 1, con factorizaciones primas (Johnsonbaugh, 2005).

*Ejemplo 4: Calcular mcd (105,30).*

La factorización prima de 30: 2*3*5

La factorización prima de 105: 3*5*7

El 3 y 5 son divisores comunes en ambos números, puesto que el producto de 3*5 es 15 y también es divisor para 30 y 105, se define como 15 el máximo común divisor (Johnsonbaugh, 2005).

### 2.5.2. Algoritmos enteros

En esta segunda sección se menciona la base para calcular $a^n mod\ z$, que es parte fundamental para el estudio del CAPÍTULO 3 en la descripción del algoritmo asimétrico RSA.

Continuando con él contexto se asume que a y n son números grandes, por tanto $a^n$ será mayor. Lo óptimo en este caso será calcular el residuo después de cada multiplicación, con el fin de mantener números pequeños (Johnsonbaugh, 2005). A continuación se describe esta técnica.

#### 2.5.2.1. Aritmética Modular

Sistema aritmético, denominado relaciones de congruencia para los números enteros, a continuación se presenta las tres ecuaciones de las diferentes propiedades (Stallings, 2011):

$$[\,(a\ mod\ z) + (b\ mod\ z)]mod\ z = (a + b)mod\ z$$

Ecuación (1)

$$[\,(a\ mod\ z) - (b\ mod\ z)]mod\ z = (a - b)mod\ z$$

Ecuación (2)

$$[\,(a\ mod\ z) * (b\ mod\ z)]mod\ z = (a * b)mod\ z$$

Ecuación (3)

En este proyecto se demuestra cómo surge la tercera ecuación, ya que es parte indispensable del CAPÍTULO 3 para el algoritmo RSA.

**Teorema**

Según Johnsonbaugh (2005) "*a, b y z* son enteros positivos, donde: $w = ab\ mod\ z, x = a\ mod\ z$ y $y = b\ mod\ z$. Como w es el residuo cuando $a * b$ se divide por el teorema del cociente-residuo, existe $q_1$como se observa en la ecuación (4):

$$ab = q_1 z + w$$

Ecuación (4)

Luego resulta la ecuación (5):

$$w = ab - q_1 z$$

Ecuación (5)

De manera similar, existen $q_2$ y $q_3$ que se expresan en las ecuaciones (6).

$$a = q_2 z + x, \qquad b = q_3\, z + y$$

Ecuaciones (6)

Ahora $w = ab - q_1 z$

$$= (q_2 z + x)(q_3 z + x) - q_1 z$$

$$= (q_2 q_3 z + q_2 y + q_3 x - q_1)z + xy$$

$$= qz + xy,$$

Ecuación (7)

Donde $q = q_2 q_3 z + q_2 y + q_3 x - q_1$. Dando como resultado la ecuación (8).

$$xy = -qz + w;$$

Ecuación (8)

Es decir, w es el residuo cuando $xy$ se divide entre z. Así, $w = xy \ mod \ z$, lo que se entiende por la ecuación (9).

$$[\ (a \ mod \ z) * (b \ mod \ z)] mod \ z = (a * b) mod \ z$$

Ecuación (9)"

*Ejemplo 5: Calcular* $28^{23} mod\ 15$

De acuerdo a Johnsonbaugh (2005) para calcular $a^{23}$se eleva al cuadrado una y otra vez, como se muestra a continuación (Ver Tabla 9):

**Tabla 9**

***Cálculo de $a^{23}$elevando al cuadrado repetidas veces***

| x | Valor de n | n mod 2 | Resultado | Cociente cuando n actual se divide entre 2 |
|---|---|---|---|---|
| **a** | 23 | 1 | a | 11 |
| **$a^2$** | 11 | 1 | $a * a^2 = a^3$ | 5 |
| **$a^4$** | 5 | 1 | $a^3 * a^4 = a^7$ | 2 |
| **$a^8$** | 2 | 0 | Sin cambio | 1 |
| **$a^{16}$** | 1 | 1 | $a^7 * a^{16} = a^{23}$ | 0 |

En un comienzo x se hace igual a $a$ y $n$ al valor del exponente, en este caso 23. Después se calcula $n \ mod \ 2$, con un valor igual a 1, por tanto $a^1$ se incluye en el producto. Luego se calcula cuando 23 se divide entre 2, el cociente es 11, volviéndose el nuevo valor de $n$. El proceso se repite hasta cuando la nueva n sea 0.

Por tanto se calculó sucesivamente:

$a$, $a * a^2 = a^3$, $a^3 * a^4 = a^7$, $a^7 * a^{16} = a^{23}$

Luego para calcular $a^{23} \ mod \ z$, se calcula de manera sucesiva:

$a \ mod \ z$, $a^3 \ mod \ z$, $a^7 \ mod \ z$, $a^{23} \ mod \ z$,

Para cada multiplicación se utiliza el Teorema antes explicado, es decir, se calcula $a^2$ con la ecuación (10):

$$[\,(a \ mod \ z) * (a \ mod \ z)]mod \ z = a^2 mod \ z$$

Ecuación (10)

Se calcula $a^4$ con la ecuación (11):

$$[\,(a^2 \ mod \ z) * (a^2 \ mod \ z)]mod \ z = a^2 * a^2 mod \ z = a^4 \ mod \ z$$

Ecuación (11)

Igualmente se calcula para $a^8$ y $a^{16}$. A continuación se indica cómo se realiza el cálculo de $a^3$, usando la ecuación (12).

$$a^3 \ mod \ z = a * a^2 mod \ z = \ [\,(a \ mod \ z) * (a^2 \ mod \ z)]mod \ z$$

Ecuación (12)

Y así sucesivamente para $a^7$ y $a^{23}$. Con las fórmulas antes mostradas, se presentan los cálculos realizados, para determinar $28^{23} mod \ 15$ , en las siguientes ecuaciones (13):

$$28^2 \text{mod } 15 = (28 \text{ mod } 15)(28 \text{ mod } 15) \text{mod } 15 = 28^2 \text{mod } 15 = 4$$

$$28^4 \text{mod } 15 = (28^2 \text{ mod } 15)(28^2 \text{ mod } 15) \text{mod } 15 = 4^2 \text{mod } 15 = 1$$

$$28^8 \text{mod } 15 = (28^4 \text{ mod } 15)(28^4 \text{ mod } 15) \text{mod } 15 = 1^2 \text{mod } 15 = 1$$

$$28^{16} \text{mod } 15 = (28^8 \text{ mod } 15)(28^8 \text{ mod } 15) \text{mod } 15 = 1^2 \text{mod } 15 = 1$$

$$28^3 \text{mod } 15 = (28 \text{ mod } 15)(28^2 \text{ mod } 15) \text{mod } 15 = 28 * 4 \text{ mod } 15 = 7$$

$$28^7 \text{mod } 15 = (28^3 \text{ mod } 15)(28^4 \text{ mod } 15) \text{mod } 15 = 7 * 1 \text{ mod } 15 = 7$$

$$28^{23} \bmod 15 = (28^7 \bmod 15)(28^{16} \bmod 15) \bmod 15 = 7 * 1 \bmod 15 = 7$$

Ecuaciones (13)

### 2.5.3. Algoritmo Euclidiano

El algoritmo Euclidiano es un algoritmo antiguo, conocido y eficiente en el cálculo del máximo común divisor de dos enteros. Se aplica en diversas áreas, como es en las ciencias de la computación (Johnsonbaugh, 2005).

#### 2.5.3.1. Teorema

Si *a* es un entero no negativo, *b* es un entero positivo y *r=a mod b*, por tanto la ecuación (14) representa al algoritmo euclidiano para encontrar el máximo común divisor.

$$mcd(a, b)=mcd(b, r)$$

Ecuación (14)

*Ejemplo 6: En las siguientes ecuaciones en (15) se encuentra el máximo común divisor a través del algoritmo euclidiano.*

Como 50 mod 15=5,

mcd (50, 15) = mcd (15, 5)

Como 15 mod 5=0

mcd (15, 5) = mcd(5,0)

mcd (5, 0)=5. Por tanto mcd(50, 15)=mcd(15,5)=mcd(5,0)=5

Ecuación (15)

#### 2.5.3.2. Algoritmo de Euclides extendido

El algoritmo de Euclides extendido es importante para los cálculos que se realiza posteriormente en el sistema criptográfico asimétrico RSA (Stallings, 2011). Permite calcular el máximo común divisor y dos números enteros *s* y *t* adicionales, que

satisfacen la siguiente ecuación (16), donde $a$ y $b$ son enteros no negativos diferentes de cero (Johnsonbaugh, 2005).

$$mcd(a,b) = sa + tb$$

Ecuación (16)

Para una mejor demostración de este algoritmo se explica a continuación un ejemplo.

*Ejemplo 7: Calcular el mcd (660,29).*

Según Johnsonbaugh (2005) comienza con $a = 660$ y $b = 29$. El algoritmo euclidiano calcula primero la ecuación (17).

$$r = 660\ mod\ 29 = 22$$

Ecuación (17)

Después establece $a = 29$ y $b = 22$. Luego el algoritmo calcula la ecuación (18)

$$r = 29\ mod\ 22 = 7$$

Ecuación (18)

Después establece $a = 22$ y $b = 7$. El algoritmo calcula la ecuación (19)

$$r = 22\ mod\ 7 = 1$$

Ecuación (19)

Luego establece $a = 7$ y $b = 1$. Ahora el algoritmo calcula la ecuación (20)

$$r = 7\ mod\ 1 = 0$$

Ecuación (20)

Como $r = 0$, el algoritmo termina por encontrar el máximo común divisor de 660 y 29 como 1. Para encontrar $s$ y $t$, se opera hacia atrás, comenzando con la ecuación (19), que se reescribe como la ecuación (21), donde se divide el 22 para 7, con su cociente igual a 3.

$$1 = 22 - 7 * 3$$

Ecuación (21)

La ecuación (18) se reescribe en la siguiente ecuación (22). Luego se sustituye esta ecuación para 7 en la ecuación (21), formando la ecuación (23).

$$7 = 29 - 22 * 1$$

Ecuación (22)

$$1 = 22 - (29 - 22 * 1) * 3 = 4 * 22 - 3 * 29$$

Ecuación (23)

La ecuación (17) se reescribe en la siguiente ecuación (24). Luego se sustituye esta ecuación para 22 en la ecuación (23), formando la ecuación (25).

$$22 = 660 - 29 * 22$$

Ecuación (24)

$$1 = 4 * (660 - 29 * 22) - 3 * 29 = 4 * 660 - 91 * 29$$

Ecuación (25)

Así, si $s = 4$ y $t = -91$, se obtiene la ecuación (26):

$$mcd(660,29) = 1 = s * 660 + t * 29$$

Ecuación (26)

#### 2.5.3.3. Cálculo de inverso del módulo de un entero

Para el cálculo inverso del módulo se tiene que los dos números enteros $n > 0$ y $\varphi > 1$ , den como resultado $mcd(n, \varphi) = 1$. Dando apertura para calcular un entero s, $0 < s < \varphi$ tal que $ns \ mod \ \varphi = 1$. Definiendo a $s$ como el inverso de $n \ mod \ \varphi$ , este cálculo es indispensable en el algoritmo RSA (Johnsonbaugh, 2005).

Para una mejor comprensión de este cálculo, se explica a continuación con un ejemplo.

*Ejemplo 8:*

Sea n = 29 y φ = 660. En el ejemplo 9 se demostró que el $mcd(660,29) = 1$ y se pudo obtener $s$ y $t$; Sin embargo en este cálculo es necesario encontrar números $s'$ y $t'$. Lo cual demuestra que $s'n + t'\varphi = 1$, donde $s' = -91$ y $t' = 4$.

Entonces $s'n = -t'\varphi + 1$, por tanto $\varphi > 1$ y el residuo es 1; Dando como resultado la ecuación (27).

$$29(-91) mod\ 660 = ns' mod\ \varphi = 1$$

Ecuación (27)

Puede ser que $s'$, sea el valor deseado, pero tal vez no satisfaga con $0 < s' < \varphi$, sin embargo se puede convertir $s'$ en el mejor valor a través de la ecuación (28).

$$s = s' mod\ \varphi = -91\ mod\ 660 = 569$$

Ecuación (28)

Por lo tanto el inverso de 29 módulo 660 es 569, es denominado un número único.

## 2.6. Herramientas de Desarrollo

El presente proyecto usa diferentes herramientas que son gratuitas y no tienen ningún tipo de restricción, como son: Netbeans IDE y MySQL. El usuario puede instalar dichos programas en plataformas, como Windows/Linux sin problema alguno. Por otro lado, esta sección presenta una breve introducción de las técnicas de programación como: los sockets y la llamada a procedimiento remoto; las cuales permiten garantizar la comunicación cliente/servidor.

### 2.6.1. Netbeans IDE

Es un Entorno de Desarrollo Integrado (IDE) de código abierto, multiplataforma, que permite realizar aplicaciones esencialmente con el lenguaje de programación Java e incluso permite ejecutar interacciones a base de datos, web, UML y telefonía móvil.

Se originó en 1996 como un trabajo estudiantil, con el objetivo de escribir un entorno de desarrollo integrado (IDE) para Java (Penarrieta, 2011). Esta plataforma Netbeans está basada en Java API, con un kit de herramientas de ventana abstracta

(Abstrarct Window Toolkit, AWT), incluido Swing (biblioteca gráfica), todo esto integra lo que es el Java Standard Edition (JSE) (Böck, 2011).

Netbeans, permite realizar varias tareas propias de la programación: como editar, compilar, ejecutar y depurar todo tipo de código. Como se muestra en la Figura 9 el entorno de desarrollo de Netbeans, es amigable al usuario y accesible al público.

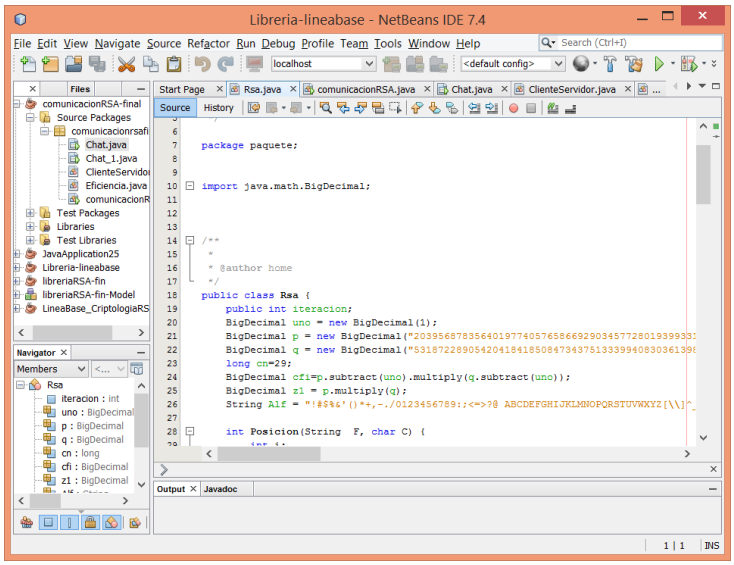


**Figura 9. Entorno de Desarrollo Netbeans**

El software propuesto, se desarrolló con esta herramienta al estar libre de restricciones, además facilita la conexión a la base de datos creada en MySQL, y la comunicación Cliente/Servidor en tiempo real.

### 2.6.2. MySQL

Es una base de datos de código abierto, con un Lenguaje de consulta estructurada (SQL). Se ejecuta bajo la Licencia Pública General de GNU (llamada GPL), es ampliamente usado por ser amigable con el usuario. Permite la creación de cualquier tipo de aplicación con diversos lenguajes de programación, es de fácil instalación y configuración (Nixon, 2014). A continuación se detalla varias características importantes que fundamenta el uso de esta herramienta en el presente proyecto (Casillas, Gibert, & Pérez).

- Se encuentra desarrollado en C/C++.
- Está modificado para equipos multiprocesadores, debido a su implementación multihilo.
- Se utiliza como cliente/servidor o introducido en aplicaciones.
- Contiene una administración apoyada en usuarios, privilegios y contraseñas, que permite mantener un alto nivel de seguridad de los datos.
- Consta de API's, con una disponibilidad de lenguajes como: C, C++, Eiffel, Java, Perl, PHP, Python, Ruby y TCL.

Se ha utilizado esta herramienta, porque permite almacenar información con un alto nivel de seguridad, además de trabajar con un lenguaje amigable.

### 2.6.3. Sockets

Un socket es una abstracción, a través de la cual una aplicación puede enviar y recibir datos, son utilizadas en este proyecto, para las comunicaciones TCP en Java y transmitir información importante. Esta aplicación puede conectarse y mantener una comunicación con aplicaciones que estén conectadas a la misma red. Permite programar comunicaciones sin importar los niveles inferiores (Calvert, 2011).

Los sockets se identifican de acuerdo al destino de los paquetes, mediante la dirección IP (nodo destino) y el puerto correspondiente (valor entero). Existen dos maneras de utilizar un socket: como cliente (Socket) y servidor (ServerSocket). A continuación en la Figura 10 se muestra un esquema para establecer las comunicaciones TCP (Calvert, 2011).

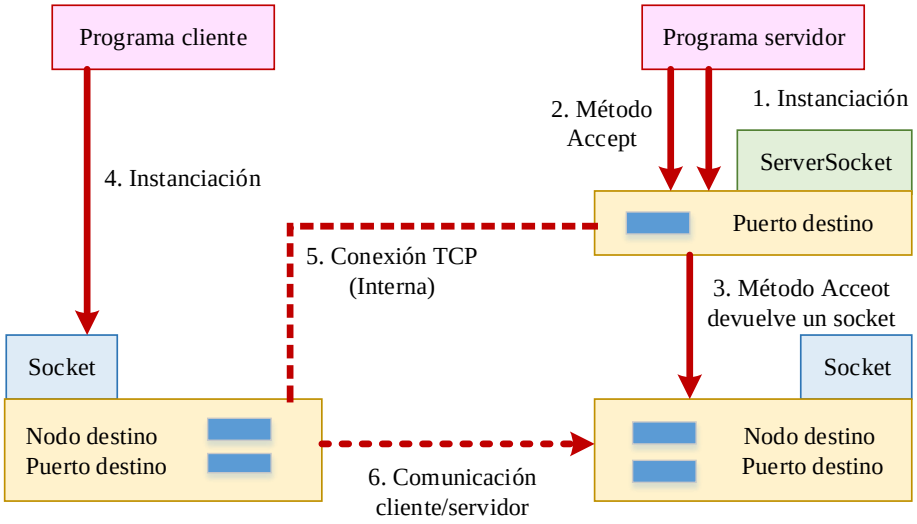


**Figura 10. Establecer comunicaciones TCP**

**Fuente:** (Alulema, 2014)

Para establecer la comunicación TCP por medio de los sockets se involucra 6 operaciones principales: (1) se crea una instancia de la clase ServerSocket con el puerto del servicio; (2) el programa servidor, invoca el método Accept sobre el ServerSocket, el programa se bloquea hasta que haya una conexión de un cliente; (3) el método Accept entrega un Socket, para realizar la comunicación cliente/servidor; (4) en el cliente se crea una instancia de clase Socket con la dirección IP y el puerto servicio; (5) se establece una comunicación interna con el Socket del cliente y el ServerSocket del servidor; y (6) realizado los pasos anteriores se empieza a comunicar datos entre cliente-servidor.

### 2.6.4. Llamada a Procedimiento Remoto

Por sus siglas en inglés (Remote Procedure Call, RPC), es un paradigma útil que proporciona comunicación a través de la red entre programas con un lenguaje de alto nivel (Andrew, Birrell, & Bruce, 1984). Este protocolo evita que el programador trabaje con la interfaz de los Sockets. Además en la comunicación, el servidor brinda procesos al cliente como procedimientos locales, incluso tiene de vuelta resultados de esta operación (Ver Figura 11).

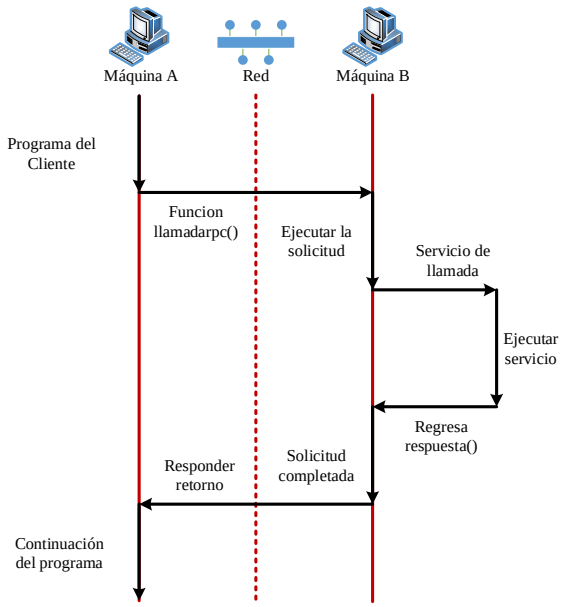


**Figura 11. Llamada a Procedimiento Remoto**

**Fuente:** (Murali, Anusha, Shirisha, & Sravya, 2011)

El funcionamiento toma su curso cuando la Máquina A (Cliente) realiza la llamada a una función. La Máquina B (Servidor) recibe el mensaje y ejecuta la solicitud, este realiza el servicio de la llamada, ejecuta y regresa la respuesta empaquetada en un mensaje. Finalmente se completa la solicitud y se transmite el mensaje al cliente.

# CAPÍTULO 3

# EVALUACIÓN DE LOS SISTEMAS CRIPTOGRÁFICOS

Estudio breve de los sistemas criptográficos simétricos y asimétricos, donde se describen y analizan los posibles escenarios que se pueden presentar. Finalmente se realiza un enfoque más detallado del sistema criptográfico RSA.

## 3.1. Criptografía de clave privada

La criptografía simétrica, también conocida como encriptación de clave única, era el tipo de cifrado en uso, antes del desarrollo de encriptación de clave pública en la década de 1970 (Stallings, 2011). Este método es utilizado tanto para cifrar como descifrar un mensaje con una sola clave (Simson & Spafford, 2002).

Según Stallings (2011), los elementos que usa la encriptación simétrica (Ver Figura 12), son los siguientes:

- **Mensaje Original:** Mensaje legible que se introducen al algoritmo;
- **Algoritmo de encriptación:** El algoritmo de encriptación realiza los cambios pertinentes con el mensaje original;
- **Clave privada:** La clave secreta es introducida en el algoritmo de encriptación, los cambios realizados por el algoritmo dependen de la clave introducida;
- **Mensaje encriptado:** Mensaje cambiado y producido como salida, que depende del mensaje y la clave privada;
- **Algoritmo de desencriptación:** Este algoritmo funciona inversamente, toma el mensaje encriptado y la clave secreta, para obtener el documento original.

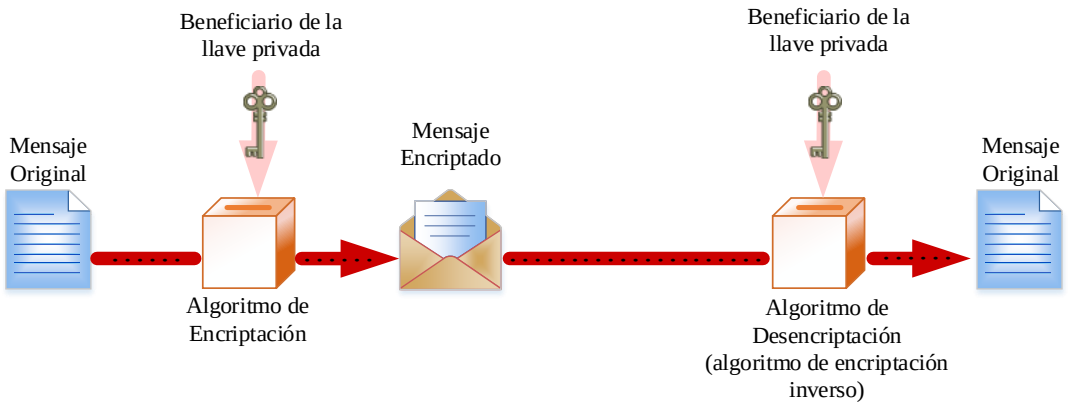


**Figura 12. Criptografía de clave privada usada para cifrar mensajería**

**Fuente:** (Simson & Spafford, 2002)

Los algoritmos de clave simétrica brindan varias ventajas, como es la velocidad, la facilidad de implementar y un número largo de posibles claves (Simson & Spafford, 2002). Sin embargo, tiene varias limitaciones que pueden perjudicar la seguridad de la información: (i) El intercambio de la clave puede ser difícil, debido al canal de comunicación; (ii) La clave se ve comprometida cuando se ha realizado una copia de ella; y (iii) Cuando un par de usuarios necesitan comunicarse, será indispensable usar una clave única, para esto se emplea la siguiente ecuación: $(N^2 - N)/2$, que permite predecir el número de claves necesarios de acuerdo al número N de los usuarios, por ejemplo: para 8 usuarios será necesario 28 claves (Simson & Spafford, 2002). A continuación se describen tres algoritmos simétricos que revolucionaron la seguridad de las redes.

### 3.1.1. Algoritmos de clave simétrica

#### 3.1.1.1. Algoritmo DES

Es un algoritmo de cifrado en bloque, que se basa en el Standard de Cifrado de Datos (Data Encryption Standard, DES) adoptado en 1977 por el NIST (National Institute of Standards and Technology). Sin embargo en 1981, el ANSI (American National Standards Institute) adoptó a DES como DEA (Data Encryption Algorithm), (Simson & Spafford, 2002). Pese a esto la DES y la DEA son diferentes, la primera es implementada en un circuito integrado con una descripción en forma de FIPS (Federal Information Processing Standards), mientras que la segunda no exige implementación con circuito integrado y puede ser programado libremente (Fúster, Guía, Hernández, Montoya, & Muñoz, 2001).

El algoritmo DES tiene dos características importantes: (i) La longitud de bloque es de 64 bits; y (ii) La longitud de la clave 56 bits, lo que representa $2^{56} = 7{,}2 * 10^{16}$ claves diferentes (Fúster, Guía, Hernández, Montoya, & Muñoz, 2001).

La Figura 13 muestra el esquema del algoritmo DES, como se puede observar tiene dos entradas para realizar el cifrado. Estas dos entradas se refieren: (i) Texto que debe ser de 64 bits, y (ii) La clave con una longitud de 56 bits, los otros 8 bits son usados como bits de paridad.

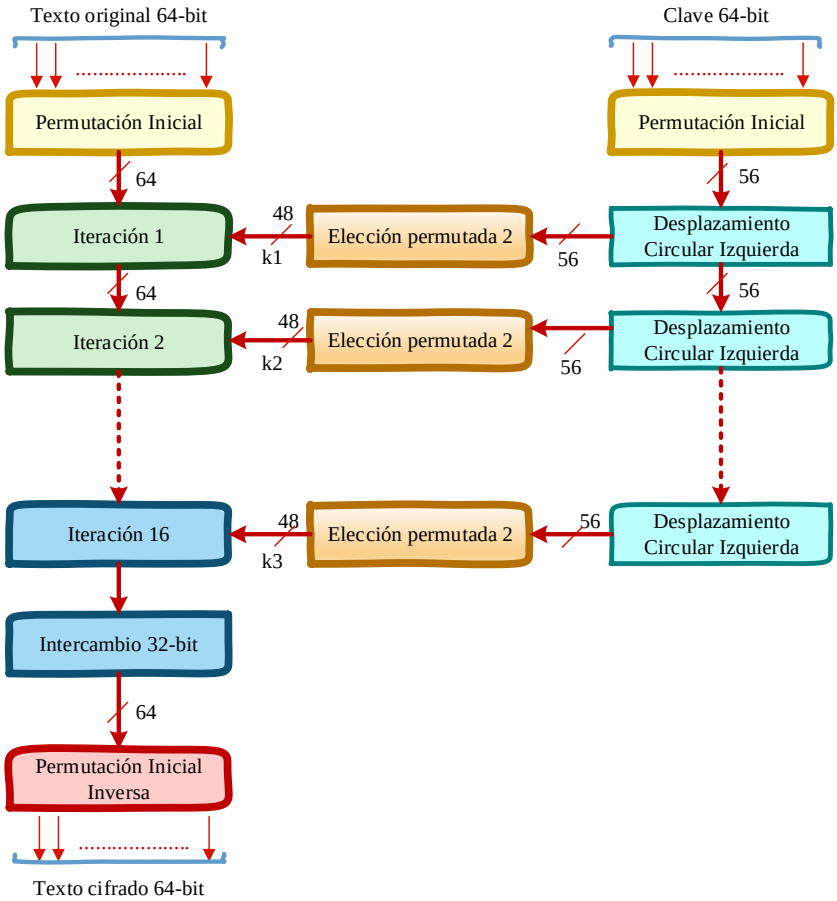


**Figura 13. Esquema del Algoritmo DES**

**Fuente:** (Arteaga & Santillán, 2013)

Para cifrar la información con este algoritmo, se inicia con la Permutación Inicial en la parte del texto, esta función solo cambia el orden de los bits, las siguientes 16 etapas realizan el mismo trabajo pero con una parte diferente de la clave. Luego intercambian 32 bits, tanto izquierda como derecha. Finalmente pasa por la permutación $[IP^{-1}]$ que se refiere a la función de la permutación inicial inversa, dando

como resultado el texto cifrado. Por otro lado la Clave de 56-bits, ingresa a una función de permutación. Luego, para cada iteración se produce una subclave $k_i$ a través del desplazamiento circular izquierda y una permutación, esta permutación será la misma en cada iteración generando un subclave diferente (Stallings, 2011). Para descifrar el DES, solo se repite las iteraciones y esto conducirá a los datos originales.

Este algoritmo es considerado inseguro gracias al tamaño de su clave, que contiene 56 bits. Hoy en día esa cantidad es insuficiente para garantizar la seguridad de la información, debido a la potencia de los ordenadores. Finalmente hay que mencionar que el entorno de aplicación de este algoritmo es un sistema embebido.

#### 3.1.1.2. Algoritmo IDEA

El Algoritmo Internacional de Cifrado de Datos (International Data Encryption Algorithm, IDEA) fue propuesto por James L. Massey y Xuejia Lai, en Zurich 1990 (Simson & Spafford, 2002). Este algoritmo cifra bloques de 64 bits con una clave de 128 bits, se basa principalmente en mezclar operaciones aritméticas de grupos algebraicos diferentes. Además consiste en ocho transformaciones idénticas y seguida por una de salida de texto cifrado de 64 bits (Fúster, Guía, Hernández, Montoya, & Muñoz, 2001) (Ver Figura 14).

En la Figura 15, para cifrar los datos se usa operaciones aritméticas sobre pares de sub-bloques de 16 bits, cada vuelta trabaja con una sub-clave diferente. Por otro lado, para descifrar se usa el mismo proceso de encriptación. A pesar de usar una clave de 128 bits, se hace uso de 52 sub-bloques de clave de 16 bits cada uno (Fúster, Guía, Hernández, Montoya, & Muñoz, 2001).

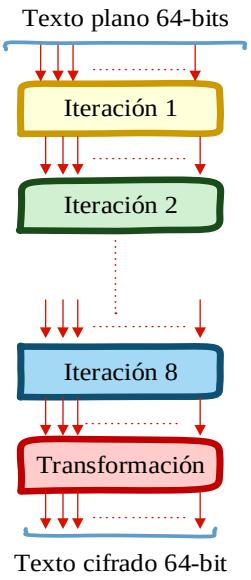


**Figura 14. Esquema del Algoritmo IDEA**

**Fuente:** (Arteaga & Santillán, 2013)

Este algoritmo es considerado un sistema criptográfico seguro, inmune al criptoanálisis diferencial. Pese a esto el ataque de fuerza bruta sería un hecho imposible de hacerlo, ya que sería probar $10^{38}$ claves.

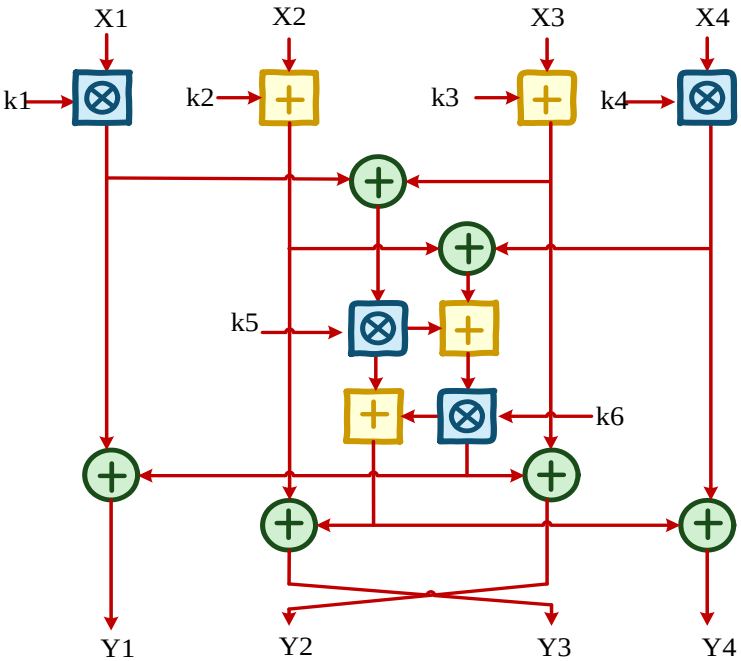


**Figura 15. Diseño de la primera iteración del IDEA**

**Fuente:** (Fúster, Guía, Hernández, Montoya, & Muñoz, 2001)

### 3.1.1.3. Algoritmo RC5

Es un algoritmo de cifrado en bloque, publicado por Ronald Rivest en 1994 (Kalishi & Yiqum, 1998). Trabaja con tres componentes básico: (i) Palabras de tamaño variable (16, 32 o 64 bits); (ii) Número de vueltas variable (0, 1, 2….255); y (iii) Clave privada con longitud variable (octetos desde 1 hasta 255).

El algoritmo RC5 utiliza tres operaciones: (i) Suma del módulo $2^w$; (ii) OR exclusivo bit a bit; y (iii) Rotación de un número a la izquierda. A continuación el algoritmo de cifrado se describe en pseudocódigo para dos palabras de bloque de entrada y salida (A, B), como se indica en las ecuaciones (29). Para descifrar se realiza las operaciones inversas en orden inverso.

$$A = A + S[0];$$

$$B = B + S[1];$$

$$for\ i = 1 \quad to \quad r \quad do;$$

$$A = \left((\bar{A}B + A\bar{B}) \ll B\right) + S[2 * i];$$

$$A = \left((\bar{B}A + B\bar{A}) \ll A\right) + S[2 * i + 1];$$

Ecuaciones (29)

Este algoritmo proporciona un nivel alto de seguridad dependiendo de los parámetros elegidos. Necesita de las rotaciones dependientes de los datos, para que el criptoanálisis diferencial y lineal no tenga ninguna respuesta.

## 3.2. Criptografía de clave pública

Los algoritmos asimétricos fueron introducidos en 1975 por Whitfield Diffie y Martin Hellman (Simson & Spafford, 2002). Estos investigadores de la Universidad de Stanford, proponen la técnica de encriptar un mensaje mediante una clave pública, mientras que para descifrar es necesario utilizar la clave privada, con la única particularidad de formarse en pares dichas claves. En la Figura 16 se muestra el uso de la criptología de clave pública usada para encriptar mensajes.

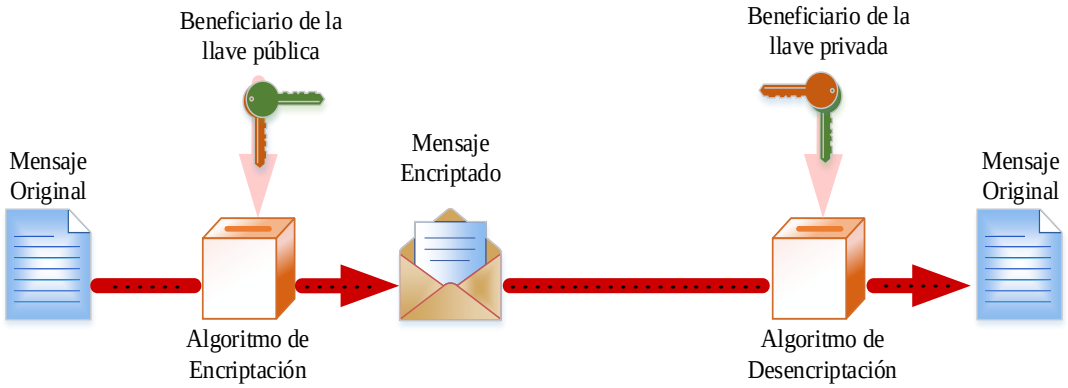


**Figura 16. Criptografía de clave pública usada para cifrar mensajería**

**Fuente:** (Simson & Spafford, 2002)

Tal como muestra la Figura 16, los elementos que tiene un esquema de encriptación asimétrica son los siguientes:

- **Mensaje Original:** Mensaje legible que se introducen al algoritmo;
- **Algoritmo de encriptación:** El algoritmo de encriptación realiza los cambios pertinentes con el mensaje que se introdujo.;
- **Clave pública y privada:** Par de llaves seleccionadas por el usuario para la encriptación y desencriptación, los cambios realizados por el algoritmo dependen de la clave introducida;
- **Mensaje encriptado:** Mensaje cambiado producido como salida, que depende del mensaje y la clave ingresada;
- **Algoritmo de desencriptación:** Este algoritmo recepta al mensaje cifrado y al par de la clave, devolviendo al texto original.

El sistema de clave pública, se caracteriza por el uso de dos claves en el algoritmo criptográfico, ya sea el emisor con la clave pública y el receptor la clave privada o viceversa. A continuación en la Tabla 10 se detalla los principales usos de la clave pública.

**Tabla 10**

***Sistema de clave pública: usos***

| | |
|---|---|
| **Encriptación/Desencriptación** | El emisor cifra el mensaje con la clave pública del receptor |
| **Firma Digital** | Se aplica la firma digital siempre y cuando el emisor aplica a su mensaje la clave privada |
| **Intercambio de claves** | Tanto el emisor como el receptor cooperan para intercambiar una sesión de claves |

Fuente: (Stallings, 2011)

La clave pública se basa especialmente en una familia de funciones unidireccionales con trampa, es decir, para cada clave es fácil calcular una trampa pero difícil de invertirla (Fúster, Guía, Hernández, Montoya, & Muñoz, 2001). No obstante, trata de plantear al atacante problemas matemáticos imposibles de solucionar, que de acuerdo al desarrollo de nuevos algoritmos asimétricos requiere de la identificación de ecuaciones matemáticas con propiedades particulares. Existen dos funciones candidatas a ser denominadas con trampa como: (i) Producto de números enteros que de acuerdo a su inversa es la factorización de este número; y (ii) Exponenciación discreta, cuya inversa es el algoritmo inverso.

Generalmente los algoritmos asimétricos cuentan con algunas desventajas: (i) Un costo computacional elevado. De acuerdo al cifrado y descifrado de llave pública, este requiere de 1000 veces más energía del computador que para un algoritmo de encriptación simétrica (Simson & Spafford, 2002); y (ii) La longitud de la clave deber ser más grande en comparación a la simétrica, sin embargo su lentitud es considerable por los cálculos que se realiza tanto en encriptar como desencriptar. En el siguiente contexto se describe tres sistemas asimétricos que impactaron a la seguridad de la información.

### 3.2.1. Sistemas Criptográficos de clave pública

#### 3.2.1.1. Intercambio de claves Diffie-Hellman

El primer algoritmo de clave pública fue propuesto en 1976 por Whitfield Diffie y Martin Hellman, este se conoce generalmente como Diffie-Hellman. El algoritmo se volvió famoso y ampliamente utilizado, no obstante un número grande de productos comerciales emplean esta técnica de intercambio de claves. Su eficiencia radica en la dificultad para el cálculo de logaritmos discretos.

Este algoritmo permite un intercambio y elaboración de claves privadas entre usuarios activos a través de un canal de comunicación inseguro. Dicho de otra manera se intercambian valores matemáticos para la respectiva clave, que permite el cálculo de una tercera clave. A continuación en la Figura 17 se describe brevemente el algoritmo de Diffie-Hellman:

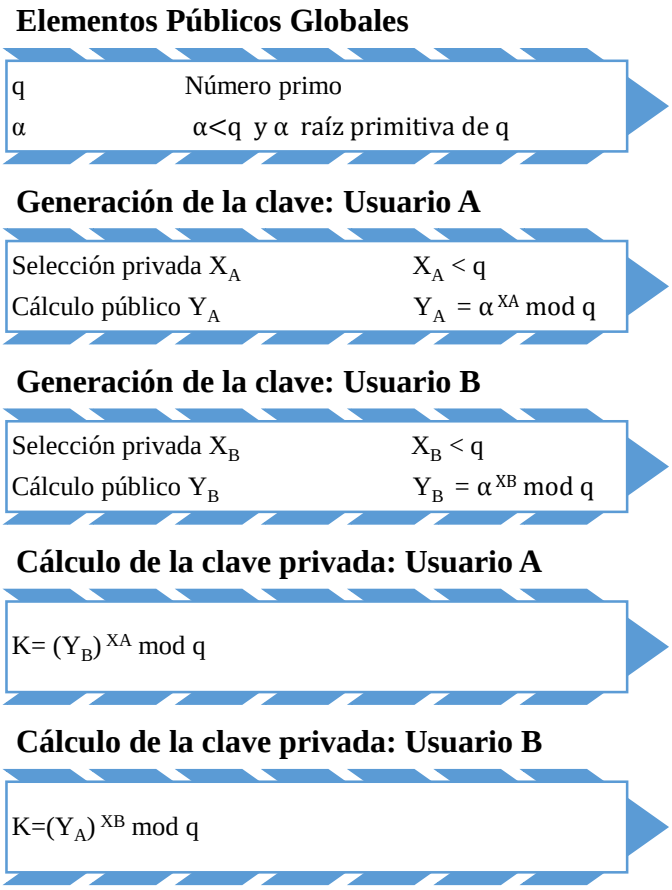


**Figura 17. Algoritmo de Diffie-Hellman**

**Fuente:** (Stallings, 2011)

De acuerdo con Stallings (2011), para este esquema se conoce los dos números, un numero primo q y un número entero α, los que permite generar las claves, tanto para el usuario A como el B. De esta manera cada lado mantiene el valor de X en secreto, mientras que el valor de Y es público para el otro lado. El resultado es que las dos partes intercambien valores secretos, haciendo que el adversario se vea obligado a determinar la clave mediante un logaritmo secreto. La seguridad de este algoritmo radica en el cálculo de logaritmos discretos.

### 3.2.1.2. Sistema Criptográfico ElGamal

En un sistema criptográfico asimétrico propuesto por Taher Elgamal en 1984, que se basa en algoritmos discretos e incluso está relacionado con la técnica Diffie-Hellman. No obstante es usado como una serie de normas que se incluye en el Standard de Firma Digital (Digital Signature Standard, DSS). En la Figura 18 se demuestra con un ejemplo el procedimiento que se realiza para cifrar y descifrar haciendo uso del sistema ElGamal.

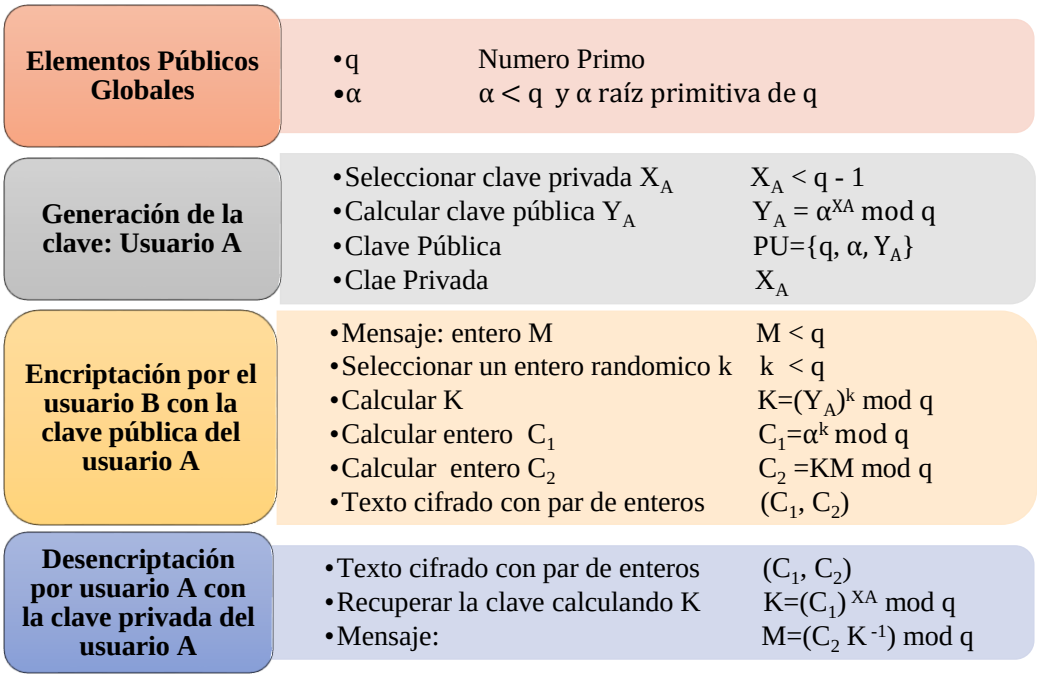


**Figura 18. Sistema Criptográfico ElGamal**

**Fuente:** (Stallings, 2011)

Según Stallings (2011) y sobre la base del esquema anterior se elige números primos q y α para generar las claves privada y pública del usuario A. El usuario B tiene

acceso a la clave pública A que permite encriptar el mensaje. Sin embargo para descifrar los datos se recupera la clave mediante el cálculo de K (definida durante el proceso de encriptación por las reglas de aritmética modular). Finalmente para recuperar el texto original a través del cálculo de M.

Se ha hecho mención a la manera de generar la clave secreta y privada. Pese a esto, cabe recalcar que la seguridad de este sistema radica en el cálculo de logaritmos discretos, por esta razón se supone que se hace uso en un solo sentido.

#### 3.2.1.3. Sistema criptográfico basado en Curvas Elípticas

Es un sistema criptográfico denominado Elliptic Curve Criptosystem (ECC), propuesto en 1985 por Neil Koblitz y Victor Miller. La seguridad se basa en la dificultad del logaritmo discreto. Usa puntos de la Curva Elíptica en vez de los números enteros, como símbolos del alfabeto del mensaje a cifrar (Belingueres, 2000). La curva elíptica está definida por la ecuación (30):

$$y^2 + axy + by = x^3 + cx^2 + dx + e$$

Ecuación (30)

Donde x e y son las indeterminadas, y a, b, c, d, e son números reales (elementos constantes de un campo). Una curva elíptica se define por una ecuación de dos variables con coeficientes. No obstante, para la criptografía las variables y los coeficientes se restringen a elementos de un campo finito.

Este algoritmo es el mejor candidato a reemplazar a otros sistemas, sin embargo su complejidad para implementar y comprender su lógica detiene a varias compañías de usarse como solución. Pese a esto la ECC es la respuesta a la seguridad cuando el espacio de almacenamiento y memoria es limitado, ya que una corta clave puede ofrecer un alto grado de seguridad, sin dejar de ser rápido en su cálculo de manera eficiente en hardware. Finalmente, el entorno de aplicación de este algoritmo es en las Redes Wireless, específicamente usado en la informática móvil.

## 3.3. Sistema Criptográfico RSA

Es un sistema criptográfico de clave pública desarrollado en 1977 por Ronald Rivest, Adi Shamir y Leonard Adleman, en el Instituto Tecnológico de Massachusetts

y fue publicado en 1978. El Rivest-Shamir-Adleman (RSA), nombrado así por sus creadores, fue patentado en 1983 por Public Key Partners (PKP), el cual permitía el uso no comercial de este algoritmo a fines personales, académicos o intelectuales. Sin embargo en el 2000 esta patente expira y deja de estar restringido su uso comercial (Fúster, Guía, Hernández, Montoya, & Muñoz, 2001).

El RSA se sustenta en la problemática de factorizar números grandes, debido a que es el producto de dos números primos, indispensables para el cálculo de las claves. En este sistema cada usuario hace público la clave para el cifrado y oculta la clave del descifrado, en este caso el emisor busca la clave pública para enviar un mensaje y el receptor busca la clave privada para descifrarlo (Ver Figura 19). El RSA está basada en los parámetros primos *p* y *q* y en los valores de las claves públicas y privadas. La seguridad de la clave debe contar con al menos 1.024 bits, y con primos p y q de al menos 500 bits que difieran unos cuantos dígitos (Ramió, 2006).

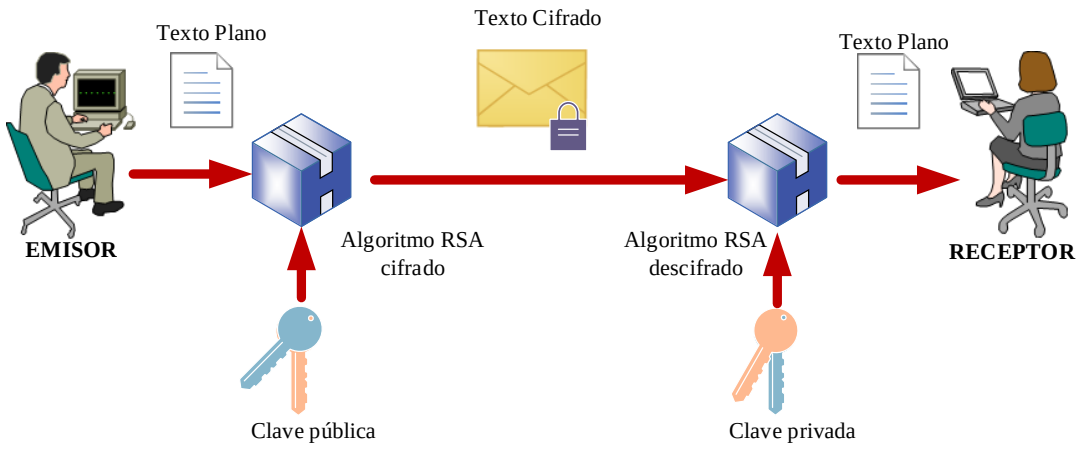


**Figura 19. Modelo Simplificado del Sistema Criptográfico RSA**

Este algoritmo es ampliamente usado en los protocolos de comercio electrónico (Wang & Zhang, 2011), que proporciona privacidad y garantiza la autenticidad de los datos digitales. Incluso es usado por servidores Web y navegadores que asegura el correo electrónico, sesiones de acceso remoto y los sistemas de pago de tarjetas de crédito electrónica (Boneh, 1999). Tomando como línea base la investigación propuesta por Johnsonbaugh (2005), a continuación se explica en tres secciones el algoritmo RSA: (i) La generación de claves; (ii) El proceso de cifrado; y (iii) El proceso de descifrado.

### 3.3.1. Generación de clave pública y privada

En esta primera sección se enumera los pasos necesarios para generar la clave pública y privada, para ello se define las respectivas condiciones que los parámetros deben cumplir. El algoritmo es el siguiente:

i. Elegir dos números primos $p$ y $q$, $p$ no debe ser igual a $q$ y cada uno debe contener 100 o más dígitos;

ii. Calcular $z = p * q$, donde $z$ representa a la longitud de la clave que generalmente se expresa en bits, incluso es el producto que se hace público para procesar ambas claves;

iii. Calcular $\varphi = (p-1) * (q-1)$, donde φ es la función de Euler;

iv. Elegir un número entero $n$, tal que el máximo común divisor de $n$ y $\varphi$ sea 1. Estos números deben ser coprimos[3] y satisfacer la condición $1 < n < \varphi$;

v. Calcular un número único $s$, que cumpla con la condición $0 < s < \varphi$, que satisface $ns \; mod \; \varphi = 1$. Este número se guarda en privado, ya que es usado para descifrar los mensajes.

La clave pública consiste en el par $(n, z)$ y la clave privada en el par $(s, z)$, donde z es el dato primordial para garantizar seguridad, es decir, por la incapacidad de descubrir los números p y q así este número sea público. Los números $p$ y $q$ son números primos elegidos en privado, deben diferir en pocos dígitos, aunque deben ser lo suficientemente grandes para resguardar la seguridad y no ser demasiados cercanos.

Continuando con las condiciones de los parámetros, n suele ser primo, para cumplir con lo estipulado $mcd(n, \varphi) = 1$, estos datos deben satisfacer que $n > 0$ y $\varphi > 1$. Además para corroborar esta información se utiliza el Algoritmo de Euclides Extendido (Ver Capitulo 2).

El parámetro $s$, es calculado por el receptor, es un número único que satisface $ns \; mod \; \varphi = 1$. Para calcular s de una manera eficiente se realiza el cálculo inverso

[3] Según Kisbye (2005), los números enteros son coprimos entre sí si no tienen ningún divisor común, excepto el 1 y el -1.

del módulo de un entero que se describe en el Capítulo 2. Este número se guarda en secreto para desencriptar la información.

### 3.3.2. Encriptación de mensajes

El usuario A (emisor) envía un mensaje (texto plano) al usuario B (receptor). El emisor cifra los mensajes, para esto es necesario que los caracteres sean codificados a números. Este número entero *a*, donde $0 \leq a \leq z - 1$.

Se utiliza la clave pública que consiste en el par $(n, z)$ y el número entero a. Se emplea la aritmética modular (Ver Capítulo 2) a la ecuación (31) para obtener *C*, como texto cifrado que es enviado al receptor.

$$C = a^n mod\ z$$

Ecuación (31)

### 3.3.3. Desencriptación de mensajes

Un vez que el texto *C*, es recibido por el usuario B. El receptor descifra el mensaje con la ayuda de la clave privada $(s, z)$ y el entero *C*. Se vuelve aplicar la aritmética modular a la ecuación (32), que como resultado nos da D, demostrando que *D = a*.

$$D = C^s mod\ z$$

Ecuación (32)

### 3.3.4. Algoritmo Matemático

Para la implementación de la línea base este proyecto se basó en el trabajo propuesto por R. Johnsonbaugh (2005), donde se considera una codificación del alfabeto, que transforma el juego de caracteres imprimibles del ASCII extendido en los números del (0 al 221), cada uno de ellos tendrá tres dígitos.

Se toma dos números primos al azar, cada uno con 300 dígitos que corresponderán a *p* y *q*. No obstante para demostrar el algoritmo desarrollado se utiliza los valores 13 y 11 respectivamente. Otro dato Importante es el *n*=29, que como se observa es un número primo.

Se supone que el usuario A envía un mensaje a B, para una mejor explicación este mensaje solo tiene un carácter que es “A”. Para esta letra se busca la posición en un String denominado Alf (alfabeto ASCII), misma que se representa con el número “032”. Si hubiera más caracteres en el mensaje los números se irían concatenando, para formar un solo número entero. A continuación se realiza los cálculos respectivos para cifrar y descifrar.

- **Generación de la clave pública y privada**

i. Los números $p$ y $q$ tienen un valor correspondiente a 13 y 11 respectivamente;

ii. Se calcula z, que es el producto entre $p$ y $q$, que es igual a 143

iii. Se calcula $\varphi = (13 - 1)(11 - 1) = 120$

iv. Para comprobar que el $mcd(n, \varphi) = 1$, se utiliza el Algoritmo de Euclides Extendido donde $n = 29$ y $\varphi = 120$. En las siguientes ecuaciones (33-36), se usa el algoritmo euclidiano:

$$r = 29 \; mod \; 120 = 29$$

Ecuación (33)

Después establece $a = 120$ y $b = 29$. Luego el algoritmo calcula la ecuación (34)

$$r = 120 \; mod \; 29 = 4$$

Ecuación (34)

Después establece $a = 29$ y $b = 4$. El algoritmo calcula la ecuación (35)

$$r = 29 \; mod \; 4 = 1$$

Ecuación (35)

Luego establece $a = 4$ y $b = 1$. Ahora el algoritmo calcula la ecuación (36)

$$r = 4 \; mod \; 1 = 0$$

Ecuación (36)

Como $r = 0$, el algoritmo termina por encontrar el máximo común divisor de 29 y 120 como 1. Para encontrar $s$ y $t$, se opera hacia atrás, comenzando con la ecuación

(35), que se reescribe como la ecuación (37), donde se divide el 29 para 4 con un cociente igual a 7.

$$1 = 29 - 4 * 7$$

Ecuación (37)

La ecuación (34) se reescribe en la siguiente ecuación (38). Luego se sustituye esta ecuación para 4 en la ecuación (37) formando la ecuación (39).

$$4 = 120 - 29 * 4$$

Ecuación (38)

$$1 = 29 - (120 - 29 * 4) * 7 = 29 * 29 - 7 * 120$$

Ecuación (39)

La ecuación (33) se reescribe en la siguiente ecuación (40). Luego se sustituye esta ecuación para 29 en la ecuación (39) formando la ecuación (41).

$$29 = 29 - 120 * 0$$

Ecuación (40)

$$1 = 29 * (29 - 120 * 0) - 7 * 120 = 29 * 29 - 7 * 120$$

Ecuación (41)

Así, si $s = 29$ y $t = -7$, se obtiene la ecuación (42):

$$mcd(29, 120) = 1 = s * 29 + t * 120$$

Ecuación (42)

iv. Calcular el número *s*, que cumpla con la condición $0 < s < \varphi$, que satisface $ns\ mod\ \varphi = 1$. Sea $s' = 29$ y $t' = -7$, dando como resultado la ecuación (43).

$$29(29) mod\ 120 = ns' mod\ \varphi = 1$$

Ecuación (43)

Puede ser que $s'$ sea el valor deseado, pero tal vez no satisfaga con $0 < s' < \varphi$, Sin embargo se puede convertir $s'$ en el mejor valor a través de la ecuación (44).

$$s = s' mod\ \varphi = 29\ mod\ 120 = 29$$

Ecuación (44)

Como resultado el inverso de 29 módulo 120 es 29, este valor es denominado un número único.

- **Proceso de Cifrado**

La clave pública es $(29, 143)$ y el número entero a=32. Se utiliza la aritmética modular para obtener *C* (texto cifrado) a través de la ecuación (45).

$$C = 32^{29} mod\ 143$$

Ecuación (45)

Para calcular $a^{29}$se muestra a continuación la siguiente Tabla 11:

**Tabla 11**

***Cálculo de $a^{29}$elevando al cuadrado repetidas veces***

| x | Valor de n | n mod 2 | Resultado | Cociente cuando n actual se divide entre 2 |
|---|---|---|---|---|
| **a** | 29 | 1 | a | 14 |
| **$a^2$** | 14 | 0 | Sin cambio | 7 |
| **$a^4$** | 7 | 1 | $a * a^4 = a^5$ | 3 |
| **$a^8$** | 3 | 1 | $a^5 * a^8 = a^{13}$ | 1 |
| **$a^{16}$** | 1 | 1 | $a^{13} * a^{16} = a^{29}$ | 0 |

A continuación se realiza los cálculos respectivos para poder obtener $32^{29} mod\ 143$ en las siguientes ecuaciones (46):

$$32^2 \text{mod } 143 = (32 \text{ mod } 143)(32 \text{ mod } 143) \text{mod } 143 = 32^2 \text{mod } 143 = 23$$

$$32^4 \text{mod } 143 = (32^2 \text{ mod } 143)(32^2 \text{ mod } 143) \text{mod } 143 = 23^2 \text{mod } 143 = 100$$

$$32^8 \text{mod } 143 = (32^4 \text{ mod } 143)(32^4 \text{ mod } 143) \text{mod } 143 = 100^2 \text{mod } 143 = 133$$

$$32^{16} \text{mod } 143 = (32^8 \text{ mod } 143)(32^8 \text{ mod } 143) \text{mod } 143 = 133^2 \text{mod } 143 = 100$$

$$32^5 \bmod 143 = (32 \bmod 143)(32^4 \bmod 143) \bmod 143 = 54$$

$$32^{13} \bmod 143 = (32^5 \bmod 143)(32^8 \bmod 143) \bmod 143 = 32$$

$$32^{29} \bmod 143 = (32^{13} \bmod 143)(32^{16} \bmod 143) \bmod 143 = 54$$

Ecuación(46)

Como resultado de texto cifrado se obtuvo un valor igual a 54, este número será enviado al receptor.

- **Proceso de Descifrado**

La clave privada es $(29, 143)$ y el número entero a=54. Se utiliza la aritmética modular para obtener el texto descifrado a través $54^{29} mod\ 143$ en las siguientes ecuaciones (47):

$$54^2 \bmod 143 = (54 \bmod 143)(54 \bmod 143) \bmod 143 = 54^2 \bmod 143 = 56$$

$$54^4 \bmod 143 = (54^2 \bmod 143)(54^2 \bmod 143) \bmod 143 = 56^2 \bmod 143 = 133$$

$$54^8 \bmod 143 = (54^4 \bmod 143)(54^4 \bmod 143) \bmod 143 = 133^2 \bmod 143 = 100$$

$$54^{16} \bmod 143 = (54^8 \bmod 143)(54^8 \bmod 143) \bmod 143 = 100^2 \bmod 143 = 133$$

$$54^5 \bmod 143 = (54 \bmod 143)(54^4 \bmod 143) \bmod 143 = 32$$

$$54^{13} \bmod 143 = (54^5 \bmod 143)(54^8 \bmod 143) \bmod 143 = 54$$

$$54^{29} \bmod 143 = (54^{13} \bmod 143)(54^{16} \bmod 143) \bmod 143 = 32$$

Ecuación(47)

Como resultado del texto descifrado se tiene un valor de 32, codificando a caracteres sería la letra “A”.

# CAPÍTULO 4

# DISEÑO Y DESARROLLO DEL PROYECTO

Este capítulo presenta la especificación de requisitos de software y la aplicación de la metodología AUP, para el diseño del software optimizado. No obstante se indica el desarrollo del software, las herramientas usadas y pruebas implementadas, para garantizar la calidad del producto.

## 4.1. Diseño

La finalidad de este proyecto "Optimización del algoritmo de encriptación asimétrica RSA, para la mitigación de ataques de seguridad de redes", es mejorar tres indicadores de desempeño, tales como: tiempo de cifrado y descifrado, consumo de recursos del sistema y rendimiento de la red, con el fin de incrementar la seguridad y disminuir los recursos respecto a la versión base de la metodología RSA. Para lograr este cometido el proyecto se encaminó a través de dos fases: (1) Optimización del modelo matemático con su respectiva algoritmia; y (2) Generación de la librería .jar, que asegure el algoritmo y las claves usadas, para el ingreso a la base de datos.

Uno de los aspectos a considerar en este diseño es el mecanismo de encriptación asimétrica RSA, que para su optimización está acorde a los objetivos de la seguridad de la norma ISO 27000, con el reto de incrementar (i) la confidencialidad: esto se logra operacionalmente a través de la clave privada y pública; (ii) la integridad: a través de la creación de una librería .jar; y (iii) la disponibilidad: accesibilidad del usuario de enviar a destino los mensajes que requieren un grado mayor de seguridad. Estos tres desafíos se han considerado como punto clave para la optimización del algoritmo RSA, con lo cual se consigue el incremento de seguridad. A pesar de esto se buscó la manera de usar números primos de una manera eficiente alcanzando la disminución de recursos.

### 4.1.1. Especificación del Alcance y Requisitos del Proyecto

#### 4.1.1.1. Alcance

Acorde al análisis realizado en el capítulo 3, se acoge la línea base propuesta por R. Johnsonbaugh, para la optimización del algoritmo RSA, que permite determinar

una expresión matemática mejorada. Para proponer una solución genérica de dicho algoritmo se constó de tres opciones:

- El servidor debe mantener una conexión a la base de datos, para tomar los números primos p y q. Estos números son necesarios para el cálculo RSA. Además fue indispensable seleccionar el juego de caracteres imprimibles del ASCII extendido (alfabeto), y el arreglo de índices generado aleatoriamente que constituye al código.
- Una vez almacenados los números primos p y q, se realizó los cálculos de Z y φ, que fueron indispensables para la generación de las claves tanto la privada como la pública.
- Se generó aleatoriamente un número primo n. Este fue indispensable para el cálculo de la clave privada y pública.

Este algoritmo forma parte de una librería que se encuentra en la interfaz. Esta interfaz permite simular una conversación en tiempo real entre el cliente y el servidor. Los mensajes de este diálogo viajaron por la red de forma cifrada.

#### 4.1.1.2. Requisitos

Las funciones que debe realizar el proyecto son las siguientes:

**Requisitos funcionales**

A continuación en la Tabla 12 se observan los requisitos funcionales del proyecto "Optimización del algoritmo de encriptación asimétrica RSA, para la mitigación de ataques de seguridad de redes".

**Tabla 12**

***Requisitos funcionales del proyecto***

| Requerimiento | Actores | Descripción |
|---|---|---|
| **Envió de mensaje** | Cliente | La aplicación permite ingresar datos que se encuentren disponibles en el juego de caracteres imprimibles ASCII extendido. |

**CONTINUA** ⟶

| | | |
|---|---|---|
| **Recepción de mensaje** | Servidor | La aplicación permite recibir los datos que el cliente haya enviado. |
| **Utilización del algoritmo** | Cliente<br>Servidor | Tanto el cliente como el servidor harán uso del algoritmo para cifrar y descifrar respectivamente |
| **Visualización del mensaje** | Cliente<br>Servidor | Tanto el cliente como el servidor pueden observar el resultado del cifrado. |
| **Selección de datos** | Servidor | El servidor se encuentra almacenando la base de datos que consta de dos tablas. |
| **Llamada a Procedimiento Remoto** | Cliente | En este caso el cliente deberá obtener los datos almacenados en el servidor, sin que afecte en el funcionamiento del sistema. |

**Requisitos de Rendimiento**

- Para una mayor rapidez en el cálculo de las claves públicas y privadas se necesita de un procesador como mínimo CORE i3 o versiones superiores.
- Para la programación de la Interfaz gráfica, se puede utilizar el programa Netbeans a partir de la versión 6.9.1 hasta las 8.0.2. Ninguna de las versiones presenta problemas con la conexión a MySQL.
- Para la base de datos, se usa cualquier versión de MySQL es necesario cambiar la codificación predefinida a UTF-8. Además su instalación es permitida en las plataformas Linux y Windows. Incluso se debe configurar que el ingreso a la base de datos se puedan conectar desde varias máquinas y no solo desde el localhost.
- Es necesario que la transferencia de datos culmine, para volver a enviar más datos y así el socket del servidor, recepte más información sin perder la comunicación.
- El sistema soporta la conexión de dos usuarios, es decir, conexión punto a punto.

**Requisitos no funcionales**

A continuación en la Tabla 13 se observan los requisitos no funcionales del proyecto "Optimización del algoritmo de encriptación asimétrica RSA para la mitigación de ataques de seguridad de redes"

**Tabla 13**

***Requisitos no funcionales del proyecto***

| Requerimiento | Descripción |
|---|---|
| **Base de Datos** | Los datos son almacenados en la información en la base de datos MYSQL |
| **Plataforma Linux/Windows** | El mecanismo de encriptación podrá ser implementado tanto en Linux como Windows, a partir de la versión 9.1 y Windows 7 en adelante. |
| **Funcionamiento síncrono** | La interfaz debe funcionar en tiempo real, para que la mensajería sea instantánea. |
| **Comunicación TCP** | Se establecerá la conexión cliente/servidor, se transfiere los datos y finalmente se dará fin a la misma. |

### 4.1.2. Diseño conceptual

En este apartado se utiliza la primera etapa de la metodología AUP, dónde se modela la semántica del dominio del sistema como son Diagramas de Secuencia, Diagrama de Clases y Arquitectura del Sistema.

#### 4.1.2.1. Diagrama de secuencia

Este diagrama de secuencias, para optimizar el algoritmo RSA, se lo realiza con la finalidad de modelar la interacción entre objetos del sistema. Tal como muestra la Figura 20 la optimización del algoritmo RSA inicia desde:

(i) La ejecución de la aplicación, tanto el cliente como el servidor ①y ②;

(ii) Se realiza el envío y recepción del mensaje ③;

(iii) El cliente escribe un mensaje en la aplicación, y así accionar la sincronización Cliente-Servidor ④. Luego se presionará el Botón Enviar que se encuentra en la interfaz del usuario, dando origen al Flujo de Datos ⑤;

(iv) La acción anterior permite recopilar los parámetros indispensables para el cálculo RSA, almacenados en una base de datos;

(v) El mensaje será cifrado y enviado al servidor ⑥, no obstante para cifrar llamará a la base de datos ⑦ y ⑧;

(vi) El mensaje cifrado será presentado en la pantalla de la aplicación del cliente ⑨. Luego este mensaje será enviado al Servidor ⑩, este llamar a la base de datos, para obtener los parámetros que corresponde el cálculo del RSA⑪-⑮;

(vii) Se presentará el mensaje en la aplicación del servidor⑯;

(viii) Para finalizar el programa tanto el cliente como el servidor accionará el botón Cerrar ⑰, con lo cual finaliza el programa de recepción y envío ⑱y ⑲.

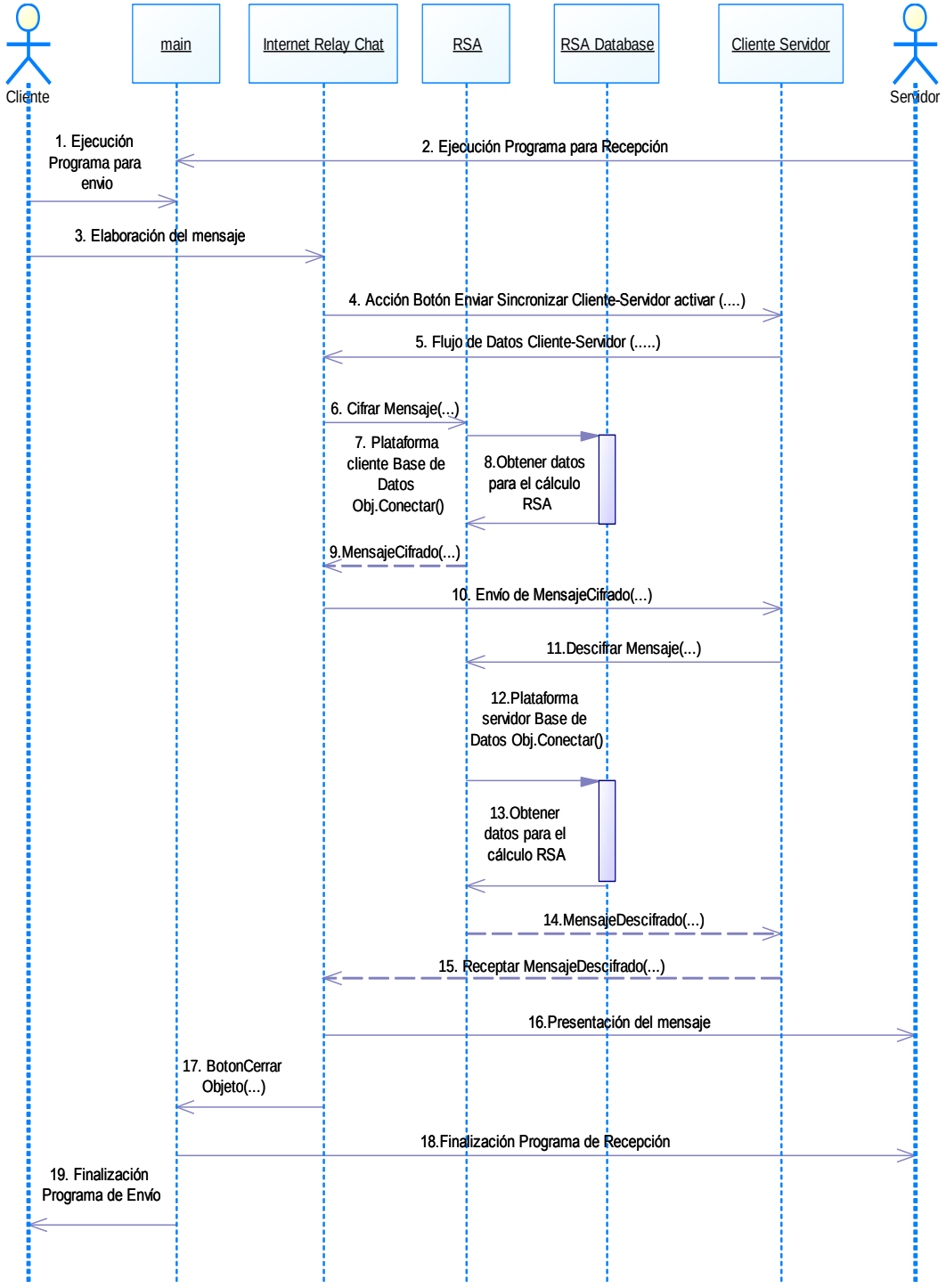


**Figura 20. Diagramas de secuencia del proyecto**

### 4.1.2.2. Diagrama de clases

Se modeló este diagrama, para describir las clases, operaciones y las relaciones entre los objetos que involucran a este sistema. (Ver Figura 21).

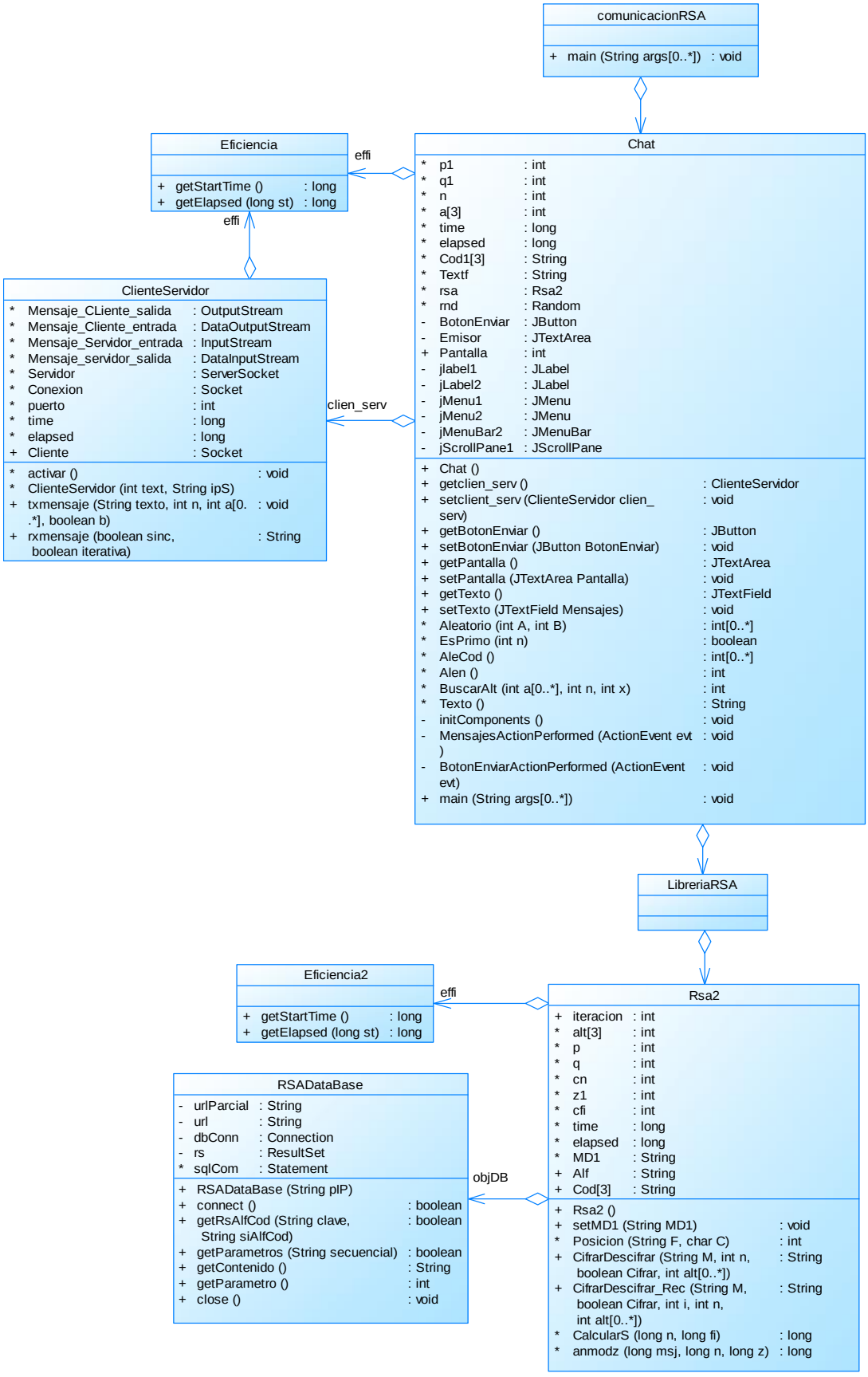


**Figura 21. Diagrama de clases del proyecto**

#### 4.1.2.3. Arquitectura del Sistema

De acuerdo a consideraciones de diseño, se implementa la arquitectura Cliente-Servidor de tres capas, donde la interfaz del usuario utiliza la librería .jar para poder crear una comunicación cliente/servidor, y hacer uso del algoritmo optimizado RSA, finalmente se recupera los datos almacenados para el cálculo RSA tal como lo muestra la Figura 22.

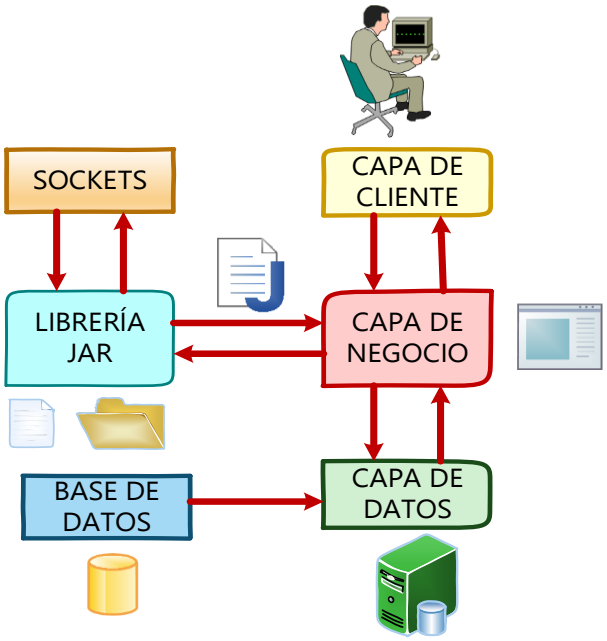


**Figura 22. Arquitectura del sistema**

La arquitectura Cliente-Servidor de tres capas permite construir componentes físicos a partir de lo lógico. La aplicación puede estar distribuida en múltiples ordenadores, a pesar de esto los problemas para la conexión a la base de datos se disminuye, ya que no es gestionado por el cliente, solamente por el servidor que realiza un mantenimiento y control de los datos almacenados. (Meza, 2010)

Considerando lo antes mencionado, este sistema permite brindar una solución genérica para la optimización del RSA, en donde, tanto el cliente como el servidor inician el programa. Se establece la comunicación entre ambos y mediante una interfaz de programación de aplicaciones (API), se intercambia los caracteres (Ver Figuras 23 y 24). Para esta solución se debe cumplir con las diferentes funciones:

(i) **Comunicación cliente/servidor mediante el protocolo TCP**: Se realiza mediante el uso de sockets programados en Java. Esto permite establecer la sincronización entre el cliente y el servidor para mantener una comunicación en tiempo real como se muestra en la Figura 23.

(ii) **Acceso a la base de datos**, compuesta por: (i) El alfabeto que contiene los caracteres imprimibles del ASCII extendido; (ii) El código, que es un arreglo de índices con cadenas de caracteres generadas aleatoriamente; y (iii) Los parámetros p y q que permiten calcular el valor del RSA. La base de datos se crea para un usuario específico con la capacidad de poder conectarse desde cualquier máquina. El computador que contiene la base de datos se denomina servidor. Por otra parte, el cliente como no contiene la base de datos, realiza la llamada de procedimiento remoto (RPC), para solicitar al servidor los parámetros adquiridos de la base de datos, enviando a estos de vuelta como resultado de dicha función (Ver Figura 23).

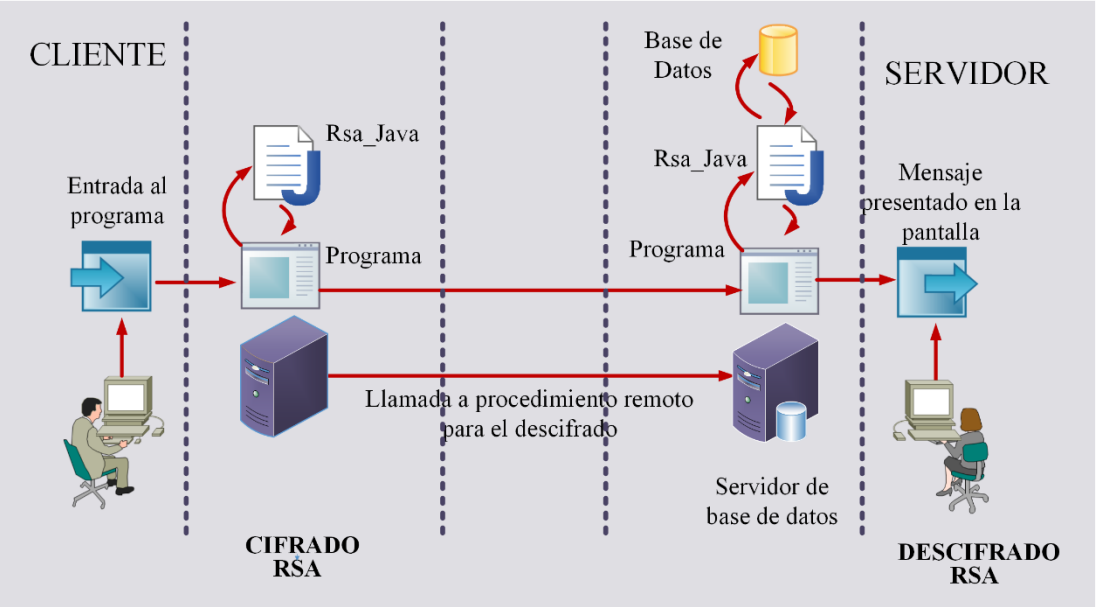


**Figura 23. Esquema global de la Arquitectura del sistema**

(iii) **Modelo matemático del algoritmo RSA:** Para optimizar el algoritmo RSA en la etapa de cifrado, se aplicó diferentes formas para la selección de los parámetros que se hacen uso en la fórmula $a^n \ mod \ z$, donde $a$ representa la posición en el alfabeto; $n$ es el parámetro que se hace público desde el emisor hacia el receptor; y $z$ es la longitud del alfabeto. Esta fórmula permite calcular el valor RSA, mismo que representa la posición en el código, cuyo carácter va a formar parte del mensaje

cifrado. Este proceso se repite hasta cuando se haya barrido todo el mensaje. En el proceso de descifrado, en cambio, se realizó el procedimiento anterior de manera inversa, para obtener el mensaje enviado por el cliente, un esquema global de este procedimiento se observa en la Figura 24.

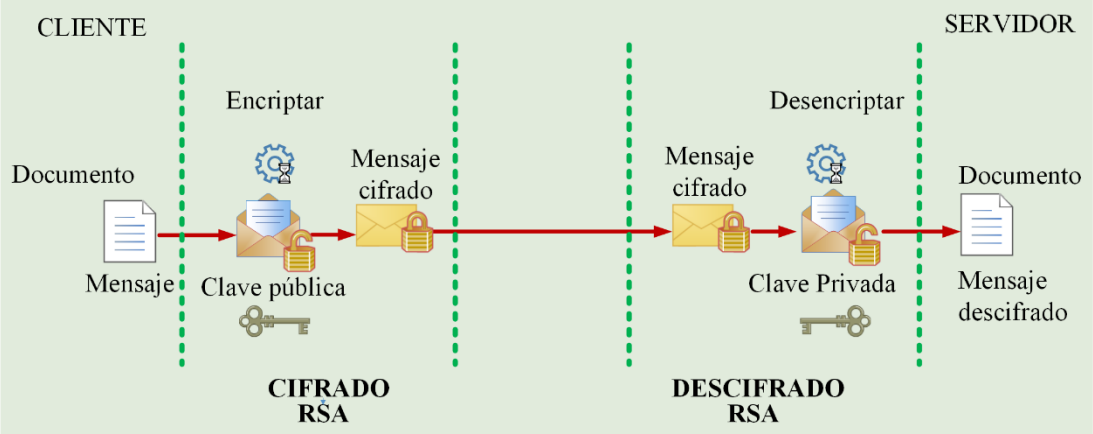


**Figura 24. Esquema global de la Arquitectura del sistema**

## 4.2. Desarrollo del sistema

De acuerdo a las consideraciones planteadas en la sección de diseño se ha establecido cuatro etapas para su respectivo desarrollo, donde se indica las herramientas de software que se utilizó y la demostración de validez de la arquitectura.

### 4.2.1. Comunicación cliente/servidor

La primera parte consiste en la relación entre la capa de cliente y la capa de negocio, básicamente el usuario-servidor ingresa a la aplicación, para establecer la comunicación con el usuario-cliente.

#### 4.2.1.1. Netbeans

Para utilizar Netbeans en la programación del sistema se realizó el siguiente proceso:

**Instalación:**

- Ingresar a https://netbeans.org/.
- Descargar la versión completa de Netbeans, es necesario tener instalado el JDK versión 7 para la instalación y funcionamiento del Java SE y Java EE. Para este

proyecto se utilizó la versión 7.4 All (Gratis) en Windows y en Linux la versión 8.0.1 All (Gratis). (Ver Figura 25).

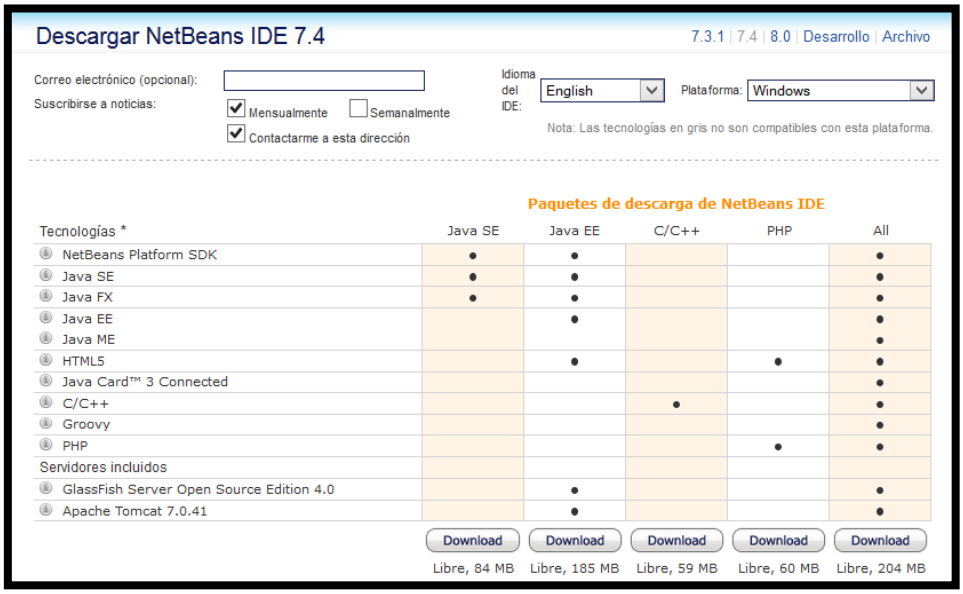


**Figura 25. Licencias de Netbeans**

- Al terminar la instalación, este programa no da problemas de activación por ser una versión gratis, más bien esta es la ventaja de trabajar con esta herramienta que no pide requisito alguno para su uso. Por tanto se puede dar inicio haciendo clic en el ícono que se encuentra en el escritorio.

### 4.2.1.2. Implementación del Proyecto

Al dar inicio a Netbeans se presenta la ventana principal con las noticias y tutoriales que este programa proporciona. Se dirige a la pestaña Archivo→Nuevo Proyecto. En la cual se elige la categoría, el proyecto a realizar y se define el nombre, en este caso, se escoge Java y Java Application respectivamente, como muestra en la Figura 26.

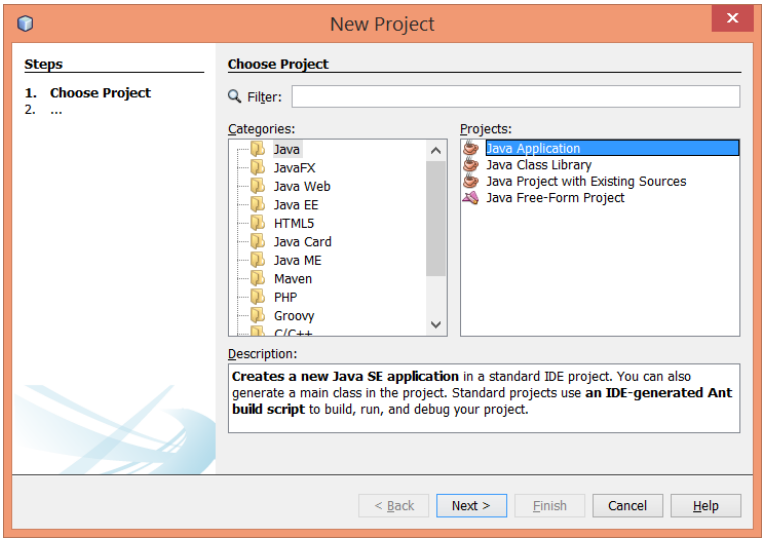


**Figura 26. Ventana Creación Proyecto ChatRSA**

El proyecto inicial muestra por defecto el Class Main, en el que se define el comando que presenta el Panel y que simula a un Chat. Para la creación del Panel se da un clic derecho en el paquete Java y se selecciona un archivo JFrame Form. Este último se muestra en blanco, para crear un interfaz de usuario amigable al usuario (Ver Figura 27).

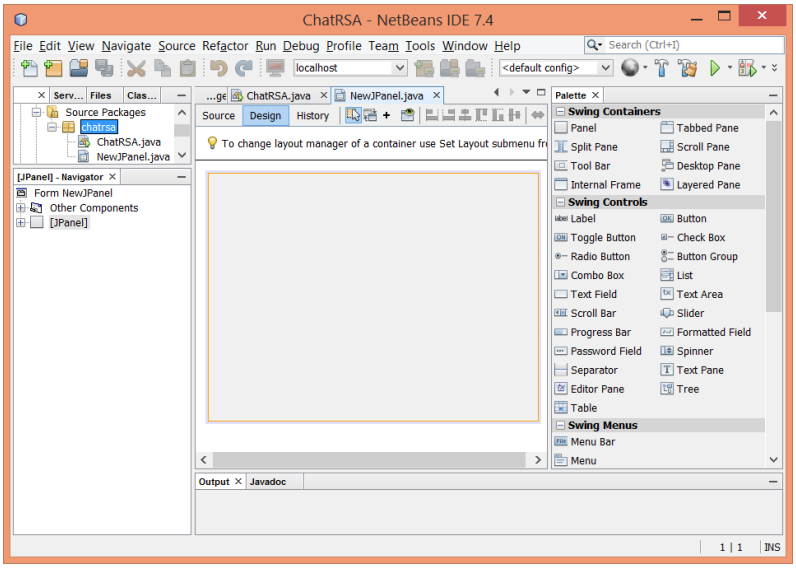


**Figura 27. Ventana de Creación Interfaz del Proyecto**

#### 4.2.1.3. Interfaz de usuario

La interfaz de usuario en Netbeans, está compuesta por dos áreas: (i) escribir el mensaje (emisor); y (ii) recibir el mensaje (receptor). Se le incorpora a esta interfaz un botón para enviar los datos. Se debe aclarar que la intención de hacer esta interfaz es sólo simular una conversación, para corroborar si el algoritmo RSA funciona correctamente (Ver Figura 28). A continuación se explica cada elemento de la interfaz.

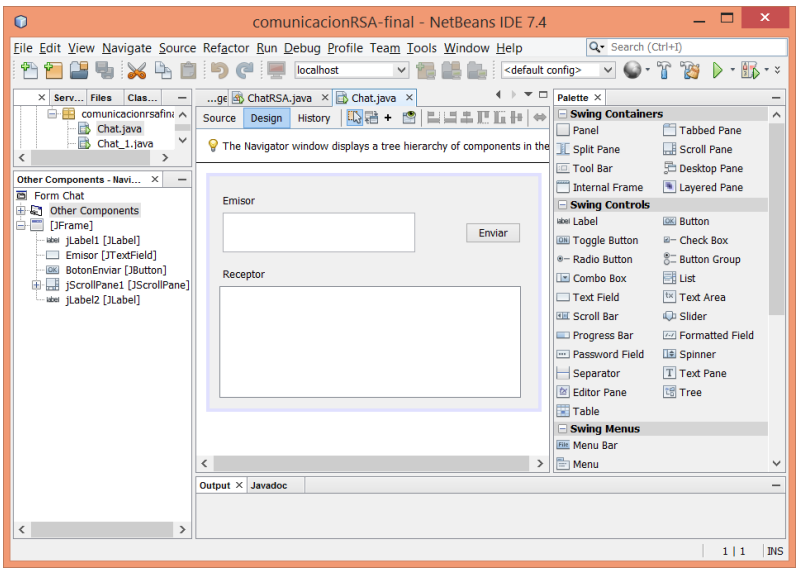


**Figura 28. Interfaz de Usuario Netbeans**

(i) **Escribir el mensaje:** Es donde el emisor escribe el mensaje, este Panel permite enviar una cadena de caracteres, cuya longitud está limitada al ancho del canal y del procesador.

(ii) **Recibir el mensaje:** Muestra el mensaje, que el emisor envió al receptor. Este mensaje se visualiza ya descifrada (texto original), es decir, cuando la conexión y el socket hayan cerrado la comunicación.

(iii) **Botón Enviar:** Los eventos que realiza el Botón Enviar, son indispensables para la transmisión de datos. Se comienza con el inicio del flujo, luego se activa la acción para tomar el valor aleatorio *n*, que es un número primo entre 1001 al 9999. Este rango puede cambiar de acuerdo a las condiciones que desee el usuario. Posteriormente se adquiere un arreglo aleatorio de los índices, cada índice contiene una cadena de caracteres del código ASCII extendido. Cada uno de los parámetros anteriores es enviado a la función CifrarDescifrar que se encuentra en la librería RSA,

para su respectivo cifrado. Finalmente el mensaje encriptado es enviado a través del flujo de datos, hasta que el Socket del servidor permita el ingreso de esa información.

#### 4.2.1.4. Sockets

Para realizar la transmisión de datos se encuentran diseñados los sockets. Estos sockets cumplen con dicha función a través de los Streams que se encuentran definidos en Java, específicamente en el paquete java.io.

Como se mencionó anteriormente, la interfaz de usuario tienen tres áreas que son implementadas por un hilo conformado por tres excepciones como: (i) activar; (ii) enviar; y (iii) recibir. Para realizar cada una de estas tareas, es necesario que al servidor se le establezca un puerto del servicio y una dirección IP del ordenador, luego se da inicio a la ejecución del hilo.

Posteriormente al accionar el Botón Enviar el cliente ingresa a una de las excepciones del hilo llamado "activar", en él se crea una instancia de tipo Socket donde se establece la dirección IP destino y el puerto del servicio. Esta clase Socket emplea dos métodos para la transmisión de datos: *InputStream→getInputStream()* y *OutputStream→getOutputStream()*. En este caso se usa el Stream de salida del socket del cliente y el Stream de entrada del socket del servidor, que se refiere a *OutputStream→getOutputStream(),* con el que se establece un flujo de datos. En último lugar se define que el flujo de salida pertenece a la clase *DataOutputStream,* que permite escribir todo tipo de dato.

Después de haber cifrado el mensaje se acciona la excepción "enviar", la cual recibe como datos el texto cifrado, el parámetro *n* y un arreglo aleatorio de los índices. Estas son escritas en el *Stream* de salida del socket cliente y se cierra el socket con el método *close().*

Seguidamente, el área para presentar el mensaje activa a la excepción del hilo denominado "recibir", se crea una instancia de clase ServerSocket en la que se establece el puerto del servicio, hasta que el cliente establezca la conexión. El servidor abre la conexión con el método *accept(),* que regresa un socket de tipo Socket para establecer la comunicación desde el clienta al servidor. Luego se utiliza el método *InputStream→getInputStream(),* para permitir la entrada del flujo al servidor. Este

flujo de entrada pertenece a la clase *DataInputStream*, que permite leer los datos enviados por el cliente. Finalmente se desencripta el mensaje cifrado, para luego cerrar el Socket y el ServerSocket.

### 4.2.2. Acceso a la base de datos

Esta segunda parte consiste en la relación entre la capa de cliente, la capa de negocio y la capa de datos. Tanto el servidor como el cliente necesitan de los parámetros almacenados en la base de datos, estos son indispensables para el algoritmo RSA.

#### 4.2.2.1. MySQL

Para utilizar MySQL en la creación de la base de datos se realizó el siguiente proceso:

**Instalación:**

- Ingresar a https://www.mysql.com/.
- Descargar la versión MySQL versión 5.6 o la más actual, mediante el proceso de instalación se debe cumplir varios prerrequisitos que engloban el paquete completo de esta herramienta. Para este proyecto se utiliza la versión 5.6, que es un software libre y amigable con el usuario. Tanto para la plataforma Windows y en Linux se puede utilizar la versión existente en la página oficial de MySQL (Ver Figura 29).


**Figura 29. Programa MySQL**

- Es necesario realizar las configuraciones respectivas como ingresar la contraseña del Root MySQL o seleccionar las diferentes aplicaciones para el Servidor. Luego permite crear un usuario con su respectiva contraseña, este usuario cuenta con el uso de credenciales que son proporcionadas en la configuración del servidor (Ver Figura 30).

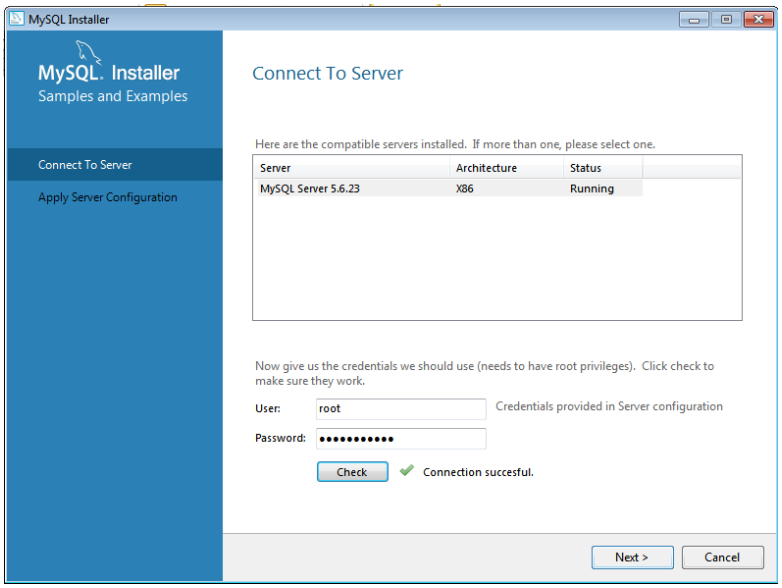


**Figura 30.Contraseña del usuario MySQL**

- Una vez instalado, se presenta una consola de MySQL denominada "Command Line Client", en el cual se debe ingresar con la contraseña ya establecida para poder crear el usuario y la tabla.

#### 4.2.2.2. Creación de la base de datos

Para la creación y configuración de la base de datos es necesario cambiar ciertas codificaciones que vienen de fábrica, estos permiten interactuar de una manera eficaz con el algoritmo programado en Netbeans. Para realizar estos cambios se debe tomar en cuenta la plataforma en la que se trabaja, es el caso de Linux que por defecto no viene con la codificación, para reconocer caracteres especiales o parecidos al idioma, por otro lado en Windows no es indispensable realizar ninguna modificación porque ya viene pre establecido.

**Cambiar codificación UTF-8 en MySQL: Linux**

En esta primera etapa, cambiar es preciso la codificación latin1 (codificación que tiene menos de 256 caracteres) a utf-8 (codificación de caracteres Unicode), esta última es utilizada por Netbeans por esta razón es fundamental realizar los cambios pertinentes. Primero se ingresa a MySQL para comprobar que valores se tiene determinado en las variables (Ver Figura 31).

```
mysql> show variables like 'character_set%';
+--------------------------+----------------------------+
| Variable_name            | Value                      |
+--------------------------+----------------------------+
| character_set_client     | latin1                     |
| character_set_connection | latin1                     |
| character_set_database   | latin1                     |
| character_set_filesystem | binary                     |
| character_set_results    | latin1                     |
| character_set_server     | latin1                     |
| character_set_system     | latin1                     |
| character_sets_dir       | /usr/share/mysql/charsets/ |
+--------------------------+----------------------------+
8 rows in set (0.00 sec)

mysql> show variables like 'collation%';
+----------------------+--------------------+
| Variable_name        | Value              |
+----------------------+--------------------+
| collation_connection | latin1_general_ci  |
| collation_database   | latin1 swedish ci  |
| collation_server     | latin1_general_ci  |
+----------------------+--------------------+
3 rows in set (0.00 sec)
```

**Figura 31.Codificación Latin1-MySQL**

Como se muestra en las figuras anteriores, la gran mayoría de variables están configuradas en latín1, en este caso se modifica el archivo de configuración de MySQL, en la que se cambia las diferentes cabeceras (Ver Figura 32).

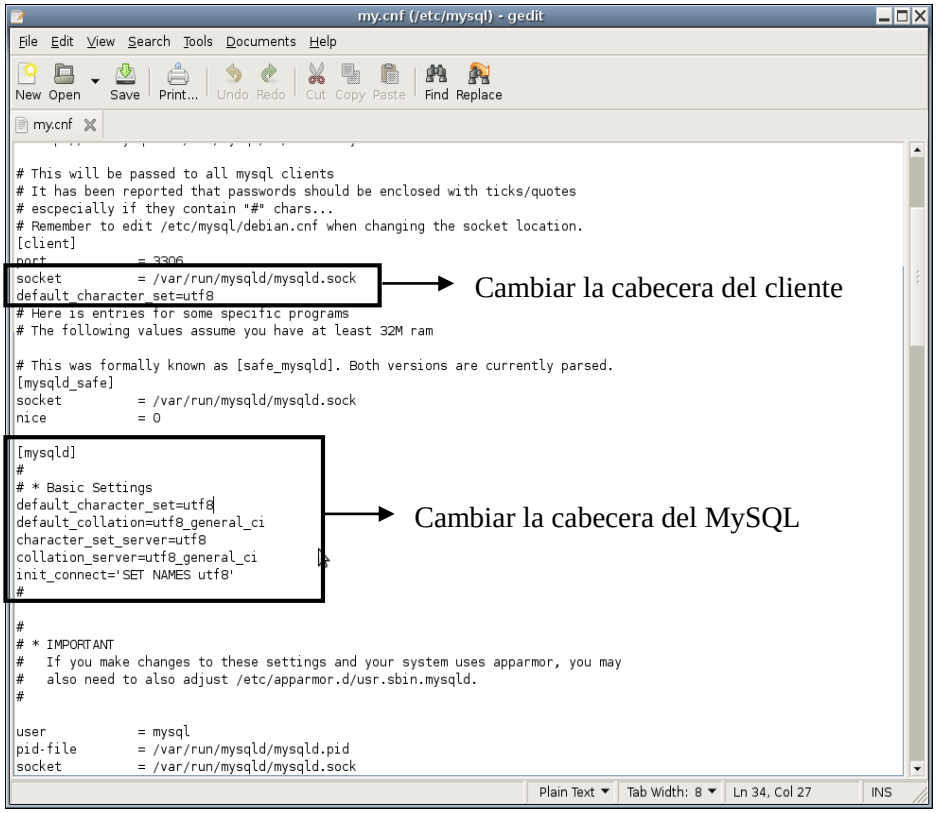


**Figura 32. Codificación de lati1 a utf-8**

Se confirman los cambios realizados al ejecutar los comandos anteriores como se muestran en las Figuras 33.

```
mysql> show variables like 'character_set%';
+--------------------------+----------------------------+
| Variable_name            | Value                      |
+--------------------------+----------------------------+
| character_set_client     | utf8                       |
| character_set_connection | utf8                       |
| character_set_database   | latin1                     |
| character_set_filesystem | binary                     |
| character_set_results    | utf8                       |
| character_set_server     | utf8                       |
| character_set_system     | utf8                       |
| character_sets_dir       | /usr/share/mysql/charsets/ |
+--------------------------+----------------------------+
8 rows in set (0.00 sec)

mysql> show variables like 'collation%';
+----------------------+-------------------+
| Variable_name        | Value             |
+----------------------+-------------------+
| collation_connection | utf8_general_ci   |
| collation_database   | latin1_swedish_ci |
| collation_server     | utf8_general_ci   |
+----------------------+-------------------+
3 rows in set (0.00 sec)
```

**Figura 33. Codificación utf-8 MySQL**

**Creación de la base de datos y tablas**

En esta segunda etapa, se realiza la base de datos, para esto se utiliza la sentencia CREATE DATABASE y para seleccionarla se emplea el comando USE, que es una opción propia de MySQL. Luego se crea la tabla, tanto del alfabeto como la de los parámetros con la sentencia CREATE TABLE, que permite establecer las columnas que se le indique. Finalmente se inserta nuevas filas en la tabla con la sentencia INSERT. En la siguientes Tablas 14 y 15 se muestra los datos que fueron ingresados para ser utilizados en Netbeans.

**Tabla 14**

***Parámetros-Base de Datos***

| Secuencial | Descripción | Parámetro | Fecha |
|---|---|---|---|
| **1** | p | 13 | 2015-03-30 |
| **2** | q | 17 | 2015-03-30 |

**Tabla 15**

***Alfabeto y Código-Base de Datos***

| Clave | Contenido | Si_Alf_Cod | Fecha |
|---|---|---|---|
| **1** | 221 caracteres del alfabeto ASCII extendido | s | 2015-03-30 |
| **2** | 221 caracteres del alfabeto ASCII extendido (aleatorio) | n | 2015-03-30 |
| **3** | 221 caracteres del alfabeto ASCII extendido (aleatorio) | n | 2015-03-30 |
| **4** | 221 caracteres del alfabeto ASCII extendido (aleatorio) | n | 2015-03-30 |

**Creación de los usuarios**

Una vez elaborada la base de datos y las respectivas Tablas 14 y 15, se crea un usuario que tenga permisos personalizados. En este caso se utiliza el comando:

*CREATE USER 'usuario'@'%' IDENTIFIED BY 'contraseña'*, donde % permite la conexión desde varias máquinas. Luego será necesario proporcionar el acceso a la base de datos desde el usuario con la siguiente sentencia: GRANT ALL PRIVILEGES ON base.* TO 'usuario'@ '%' IDENTIFIED BY 'contraseña'.

El usuario creado tiene permiso de manejar la base de datos que contienen las 2 tablas anteriormente indicadas, estas tablas son fundamental para la optimización del algoritmo RSA, incluso de la seguridad que permite proporcionar MySQL con los valores de p y q. En la siguiente sección se muestra la conexión entre la base de datos y Java.

#### 4.2.2.3. Conexión MySQL con Java

En el presente proyecto, la conexión a MySQL está directamente relacionada con el RSA, por tanto pertenecerá a la librería .jar al igual que el algoritmo. Por lo que al momento de activar la interfaz del usuario se conecta a la base de datos, es necesario definir la dirección IP del ordenador que la contiene.

La conexión de la base de datos a través de una aplicación Java se denomina JDBC (Java Database Connectivity). El JDBC consta de clases e interfaces escritas en un lenguaje de programación Java, que permiten establecer un vínculo a una fuente de datos, e incluso admite realizar las consultas de MySQL. En el presente proyecto se procede a trabajar con una arquitectura tal como se observa en la Figura 34, que muestra la ubicación del gestor de controladores con respecto a los controladores JDBC y la aplicación Java.

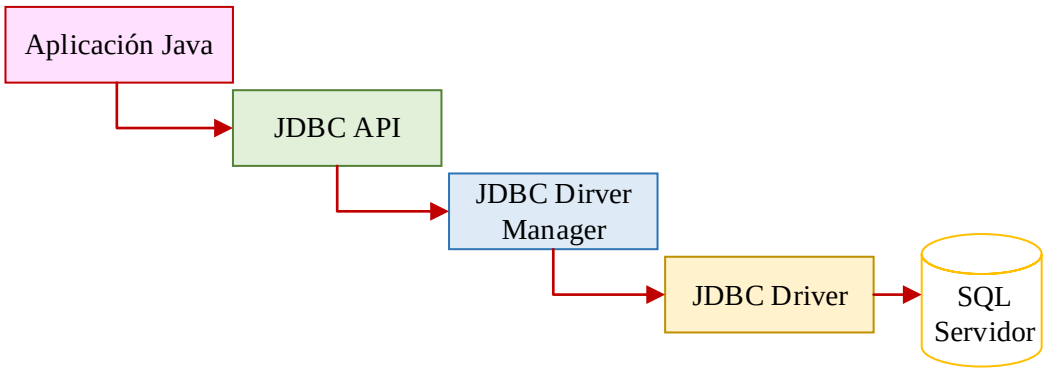


**Figura 34. Arquitectura JDBC**

Para acceder desde Java a la base de datos es necesario trabajar con un driver o un controlador JDBC. Este driver se lo puede obtener a través de la página Web: https://www.mysql.com/products/connector/, que debe ser incluido en el proyecto Netbeans en la parte de librerías, como una nueva dependencia para el conector. El JDBC contiene un paquete llamado *java.sql*, que permite ingresar y procesar datos almacenados en una fuente de datos a través de Java. A continuación se muestra en la Tabla 16 el nombre del controlador JDB y la URL de la base de datos a utilizar.

**Tabla 16**

***Nombre del Controlador JDBC***

| | **Nombre del controlador JDBC** | **Formato URL** |
|---|---|---|
| **MySQL** | com.mysql.jdbc.Driver | jdbc:mysql://”hostname/databaseName” |

Para usar el paquete *java.sql* se debe cargar el driver mediante el método *Class.forName(Nombre del controlador JDBC),* que registra un controlador configurable y portátil. Posteriormente para abrir conexión se usa el objeto *Connection,* el cual recibe como parámetros el formato URL, el usuario y la contraseña de la base de datos, a través del método *getConnection()*de la interfaz *Driver Manager*. Este último permite localizar el driver que puede conectar a la fuente de datos con la URL definida. Luego se utiliza un objeto *Statement* para enviar sentencias SQL a la fuente de datos, este objeto se crea con el método *createStatement()* del objeto *Connection.*

Para obtener los datos almacenados en la base de datos se utiliza el comando *SELECT(),* que emplea el método *executeQuery(),* cuyo resultado es de tipo *ResultSet,* que sirve para retornar las filas de consulta. Con esto se comprueba que los datos de comparación concuerdan con la información que se encuentra almacenada en la fuente de datos. Finalmente se debe obtener el contenido de cada tabla para esto es necesario utilizar el método *getString(),* que retorna el valor de la columna a petición. Es fundamental que el *ResultSet()* finalice con el método *close(),* para cerrar todas las conexiones a MySQL.

### 4.2.3. Modelo matemático del algoritmo RSA

En esta tercera parte se refiere en sí a la optimización del algoritmo RSA (capa de negocio), utilizado para cifrar y descifrar los mensajes que se envían entre el cliente y servidor. Se debe tener en cuenta que esta forma parte de una librería .jar acompañado de la interface para la conexión de la base de datos. En las siguientes secciones se proporciona una explicación: (i) algoritmo RSA iterativo, y (ii) algoritmo RSA recursivo.

#### 4.2.3.1. Algoritmo RSA iterativo

Como se pudo observar en el capítulo tres se menciona el trabajo propuesto por R. Johnsonbaugh (2005) como línea base de este proyecto. No obstante para optimizarlo se empleó diferentes métodos que permitieron desarrollar una solución genérica, que disminuyó el costo computacional e incremento la seguridad de los mensajes. A continuación se muestra el pseudocódigo del algoritmo optimizado iterativo.

**CLIENTE**

**Algoritmo para enviar el mensaje**

- Primero se selecciona el alfabeto, el arreglo con el respectivo Código y los parámetros *p* y *q*, que se encuentran almacenados en la base de datos.
- Como se mencionó en la primera sección el cliente escribe un mensaje *(msje)*. este es capturado para ser enviado a *Cifrar*.
- Se genera un número aleatorio primo entre *1001* y *9999*, este representa a *n*.
- Se debe generar aleatoriamente un arreglo aleatorio de índices del código (*a*).

**Algoritmo para cifrar el mensaje**

- Para cada carácter del mensaje *(msje)*, se debe calcular la posición de este en el alfabeto, con la siguiente expresión matemática $ps = Posicion(Alfa, msje_i)$, donde Alfa es el alfabeto seleccionado de la base de datos, y $msje_i$ es cada uno de los caracteres.
- Luego se emplea la expresión $a = Posicion\left(Alfa, Cod_{a_{i \bmod 3},\ ps}\right)$, donde *Alfa* es el alfabeto seleccionado desde la base de datos y el $Cod_{a_{i \bmod 3},\ ps}$ es el carácter de acuerdo a la posición *ps* y al arreglo respectivo del código.

- Aplicar la fórmula $x = a^n mod\ z$, para obtener *(x)*.
- Luego se obtiene el carácter que se encuentre en la posición x que forma parte del mensaje cifrado.
- Se vuelve a repetir los cuatro pasos anteriores para cada carácter del mensaje *(msje)*.
- El resultado final se obtiene un mensaje cifrado *(msjec)* del mensaje original *(msje)*.
- Se envía al servidor msjec, n y el arreglo con el respectivo Código.

**SERVIDOR**

**Algoritmo para recibir el mensaje**

- Se recepta el número *n*, el arreglo de índices del código (a) y el mensaje cifrado (msjec).
- Es necesario calcular la clave (s), a través de la expresión $s = CalcularS(n, \varphi)$, donde φ= (p-1)*(q-1).

**Algoritmo para descifrar el mensaje**

- Para cada carácter del mensaje cifrado *(msjec)*, se debe calcular la posición de este en el alfabeto, con la siguiente expresión matemática $a = Posicion(Alfa, msjec_i)$, donde Alfa es el alfabeto seleccionado de la base de datos, y $msjec_i$ es cada uno de los caracteres.
- Aplicar la fórmula $x = a^s mod\ z$, para obtener *(x)*.
- Luego se emplea la expresión $ps = Posicion(Cod_{alt_{i\ mod\ 3}}, Alf_x)$ donde $Cod_{alt_{i\ mod\ 3}}$, representa a un arreglo del código y $Alf_x$ es el carácter de acuerdo a valor x.
- Luego se obtiene del alfabeto el carácter que corresponde a la posición ps, el cual forma parte ya del mensaje original o descifrado.
- Se vuelve a repetir los anteriores cuatro pasos para cada carácter del mensaje *(msjec)*.
- El resultado final es un mensaje original *(msjc)* del mensaje cifrado *(msjec)*.

#### 4.2.3.2. Algoritmo RSA recursivo

A continuación se presenta el pseudocódigo del algoritmo optimizados recursivo tanto de cifrado como descifrado:

**CLIENTE**

**Algoritmo para enviar el mensaje**

- Primero se selecciona el alfabeto, el arreglo con el respectivo Código y los parámetros *p* y *q*, que se encuentran almacenados en la base de datos.
- Como se mencionó en la primera sección el cliente escribe un mensaje *(msje)*. este es capturado para ser enviado a *Cifrar*.
- Se genera un número aleatorio primo entre *1001* y *9999*, este representa a *n*.
- Se debe generar aleatoriamente un arreglo aleatorio de índices del código (*a*).

**Algoritmo para cifrar el mensaje**

- Para cada carácter del mensaje *(msje)*, se debe calcular la posición de este en el alfabeto (ps), con la siguiente expresión matemática $ps = Posicion(Alfa, msje_i)$, donde Alfa es el alfabeto seleccionado de la base de datos, y $msje_i$ es cada uno de los caracteres.
- Luego se emplea la expresión $a = Posicion\left(Alfa, Cod_{a_{i \bmod 3},\ ps}\right)$, donde *Alfa* es el alfabeto seleccionado desde la base de datos y el $Cod_{a_{i \bmod 3},\ ps}$ es el carácter de acuerdo a la posición *ps* y al arreglo respectivo del código.
- Aplicar la fórmula $x = a^n mod\ z$, para obtener *(x)*.
- Luego se obtiene el carácter que se encuentre en la posición x que forma parte del mensaje cifrado.
- Se envía de nuevo a *cifrar*, es decir se llama a sí mismo, hasta que se obtenga todo el mensaje cifrado.
- El resultado final es un mensaje cifrado *(msjec)* del mensaje original *(msje)*.
- Se envía al servidor msjec, n y el arreglo con el respectivo Código.

**SERVIDOR**

**Algoritmo para recibir el mensaje**

- Se recepta el número *n*, el arreglo de índices del código (a) y el mensaje cifrado (msjec).
- Es necesario calcular la clave (s), a través de la expresión $s = CalcularS(n, \varphi)$, donde φ= (p-1)*(q-1).

**Algoritmo para descifrar el mensaje**

- Para cada carácter del mensaje cifrado *(msjec)*, se debe calcular la posición de este en el alfabeto, con la siguiente expresión matemática $a = Posicion(Alfa, msjec_i)$, donde Alfa es el alfabeto seleccionado de la base de datos, y $msjec_i$ es cada uno de los caracteres.
- Aplicar la fórmula $x = a^s mod\ z$, para obtener *(x)*.
- Luego se emplea la expresión $ps = Posicion(Cod_{alt_{i\ mod\ 3}}, Alf_x)$ donde $Cod_{alt_{i\ mod\ 3}}$, representa a un arreglo del código y $Alf_x$ es el carácter de acuerdo a valor x.
- Luego se obtiene del alfabeto el carácter que corresponde a la posición ps, el cual forma parte ya del mensaje original o descifrado.
- Se envía de nuevo a *descifrar*, es decir se llama a sí mismo, hasta que se obtiene todo el mensaje descifrado.
- El resultado final es un mensaje original *(msjc)* del mensaje cifrado *(msjec)*.

## 4.3. Pruebas del software

Para realizar pruebas al software se ha tomado en cuenta el proceso expuesto por Pressman (2005), en el que afirma que las pruebas evalúan la calidad y de manera pragmática el descubrimiento de errores. El proceso a seguir permite probar este proyecto en los diferentes aspectos: unidad, integración, validación y alto nivel. Como se observa en la Figura 35.

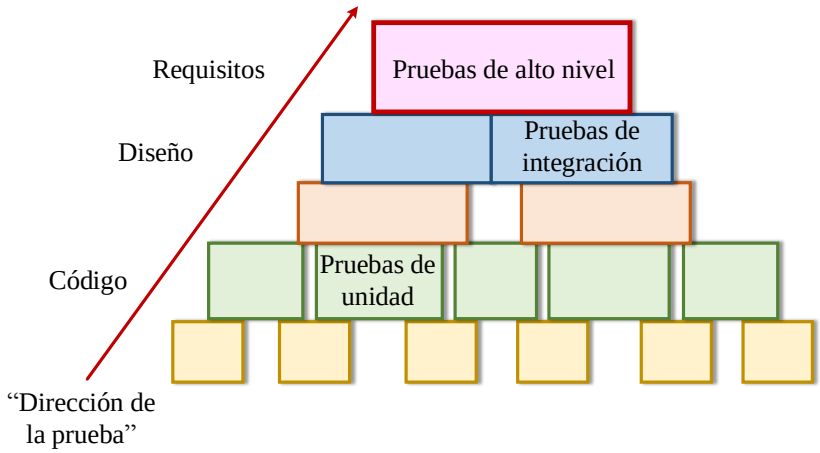


**Figura 35. Pasos de la prueba de software**

**Fuente:** (Pressman, 2005)

### 4.3.1. Prueba de Unidad

Se ha realizado esta prueba para verificar la lógica del procedimiento interno de un módulo, por lo que en la Tabla 17 se indica los errores encontrados con su respectiva solución:

**Tabla 17**

***Prueba de Unidad***

| Módulo | Error | Solución |
|---|---|---|
| **EsPrimo (int n)** | El número entero *n* como condición en el bucle *for*, el cual no permitía saber si era o no un número primo. | Utilizar un nuevo número entero *divisor* como condición del bucle *for*. |
| **CifrarDescifrar_Rec (…)** | El String *MD1* se retorna dentro del bucle *if* ,para devolver el mensaje cifrado y/o descifrado | Retornar el String *MD1* fuera de la sentencia. |

### 4.3.2. Prueba de Integración

Se realizó esta prueba con el objetivo de descubrir errores asociados con la interfaz, por lo que en la Tabla 18 se toma la estrategia denominada integración descendente, que comienza desde el módulo principal hacia los submódulos.

**Tabla 18**

***Prueba de Integración***

| Módulo | Error | Solución |
|---|---|---|
| **Emisor** | No se encontró | |
| **JTextArea** | No se encontró | |
| **BotónEnviar** | El parámetro *n* ingresa en el submódulo *EsPrimo(n)*, donde no permite saber si es o no *n* un número primo. | Se declara una nueva variable tipo entero denominada *divisor*. |

### 4.3.3. Prueba de Validación

Esta prueba se realizó con el objetivo de saber si se satisface las expectativas ante cualquier usuario, es decir saber si se cumple los requisitos, para lograr una solución genérica de la optimización del algoritmo RSA. (Ver Tabla 19).

**Tabla 19**

***Prueba de Validación***

| | Error | Solución |
|---|---|---|
| **Portabilidad** | No se encontró | |
| **Compatibilidad** | No se encontró | |
| **Facilidad de mantenimiento** | El parámetro *n* y *el arreglo de índices (código)*, son declarados en el método Chat una sola vez. | Para cambiar el número primo *n* y el arreglo de índices son declarados dentro del BotónEnviar. |

### 4.3.4. Prueba de Alto Nivel

Una vez validado el software se realiza esta prueba para verificar que el software, hardware, usuario y base de datos interactúen de manera apropiada, por lo que se tomó en cuenta a dos personas, las cuales encontraron los errores que se muestra en la Tabla 20.

**Tabla 20**

***Prueba de Alto Nivel***

| | **Error** | **Solución** |
|---|---|---|
| **Base de datos** | En Linux la codificación de la base de datos es Latin1, esto no permite la comparación con el alfabeto. | Cambiar la codificación de Latín a UTF-8 |
| **Hardware** | Los ordenadores Core 2 Duo limitan realizar pruebas de hasta 200 caracteres. | Utilizar computadoras Core i5 o i7 permite evaluar hasta con cadenas de 5120 caracteres. |
| **Software** | No se deben conectar al mismo tiempo el cliente y el servidor, ya que los dos desearán comenzar la sincronización. | El servidor debe ingresar primero a la aplicación, para iniciar la sincronización hacia el cliente. |

# CAPÍTULO 5

# ANÁLISIS Y EVALUACIÓN DE RESULTADOS

Este capítulo muestra los resultados obtenidos, luego del procesamiento estadístico que valida la optimización del algoritmo de encriptación RSA. Se ha organizado en la metodología de evaluación, evaluación del algoritmo RSA línea base vs optimizado, iterativo vs recursivo y discusión.

## 5.1. Metodología de evaluación

Para evaluar la calidad del algoritmo optimizado de encriptación asimétrico RSA, se ha divido en dos partes: (i) Se realizaron pruebas tanto del algoritmo base como del optimizado RSA y luego se comparó sus resultados; (ii) Se realizó más pruebas para comparar el gasto computacional entre el algoritmo optimizado RSA iterativo vs el recursivo. Estas pruebas consistieron en variar el número de caracteres en una cadena. Las variables evaluadas fueron separadas en tres diferentes rendimientos: servicio, sistema y red (Ver Tabla 21 y 22).

**Tabla 21**

***Matriz de definición conceptual de las variables evaluadas***

| **Rendimiento de servicios** | |
|---|---|
| **Variable** | **Definición conceptual** |
| **RSA** | Es un sistema criptográfico de clave pública, tiene dos claves tanto para cifrar como descifrar. |
| **Nivel de seguridad** | Garantiza que la información y los recursos se usen para únicos propósitos por medio de un conjunto de medidas técnicas y legales que aseguren: la Integridad, confidencialidad, disponibilidad, autenticación y no repudio. |
| **Tiempo de cifrado y descifrado** | Tiempo que tarda en realizar los cálculos para el cifrado y descifrado |
| **Número de iteraciones** | Número de caracteres que se encuentra en una cadena. |

**CONTINUA** ⟶

| Consumo de recursos computacionales | |
|---|---|
| **Variable** | **Definición conceptual** |
| **Consumo de memoria** | Permite realizar la compilación de información, su función principal es almacenar temporalmente la información del disco para ser ejecutada por el CPU. |
| **Consumo del procesador** | Es una medida de la cantidad de trabajo computacional que un sistema informático realiza |
| **Rendimiento de red** | |
| **Variable** | **Definición conceptual** |
| **Latencia** | Es la suma de tres elementos indispensables: transmisión, propagación y tiempo en cola, es decir es la suma de los retardos temporales que tienen los paquetes. |
| **Reporte estadístico de la red** | Número total de paquetes recibidos y transmitidos por segundo |

## Tabla 22

### *Matriz de definición operacional de las variables evaluadas*

| Rendimiento del servicio | | | |
|---|---|---|---|
| **Variable** | **Definición operacional** | **Indicadores** | **Herramienta** |
| **RSA** | p & q elegidos al azar que al multiplicarse pueda contener los dígitos del mensaje | Números primos | Netbeans |
| | n escogido al azar | Números primos entre 1001-9999 | Netbeans |
| **Nivel de seguridad** | Confiabilidad | Clave pública y privada | Netbeans |
| | Integridad | Rsa creado como una librería .jar | |
| | Disponibilidad | Accesibilidad del usuario de enviar a destino un mensaje | |

**CONTINUA** ⟶

| | | | | |
|---|---|---|---|---|
| **Tiempo de cifrado y descifrado** | Cifrado: cliente | Tiempo | Milisegundos(ms) | Netbeans |
| | Descifrado: servidor | Tiempo | Milisegundos(ms) | |
| **Número de iteraciones** | Número de iteraciones | | Números enteros | Netbeans |

**Consumo de recursos computacionales**

| **Variable** | **Definición operacional** | | **Indicador** | **Herramienta** |
|---|---|---|---|---|
| **Consumo de memoria** | kbmemfree | Memoria libre | Kilobytes | sar |
| | kbmemused | Memoria usada | Kilobytes | |
| | %memused | Porcentaje de memoria usada | Porcentaje % | |
| | kbbuffers | Memoria usada como amortiguadores por el núcleo | Kilobytes | |
| | Kbcached | Memoria utilizada para los datos de caché por el núcleo | Kilobytes | |
| **Consumo del procesador** | user% | Utilización del CPU que se produjo durante la ejecución a nivel de usuario | | sar |
| | nice% | Utilización del CPU que se produjo durante la ejecución a nivel de usuario con prioridad | | |
| | system% | Utilización del CPU que se produjo durante la ejecución a nivel del sistema (kernel) | | |
| | iowait% | Porcentaje de tiempo que la CPU estaba libre mientras que el sistema tenía un disco con una solicitud de E/S | | |
| | steal% | Procesador de tiempo dedicado a la CPU mientras se atiende a otro procesador virtual | | |
| | idle% | Porcentaje de tiempo en la que el CPU se encuentra inactivo. | | |

**CONTINUA** →

| Rendimiento de red | | | | |
|---|---|---|---|---|
| **Variable** | **Definición operacional** | | **Indicador** | **Herramienta** |
| **Latencia** | Last | Latencia del último paquete enviado | Milisegundos(ms) | mtr |
| | avg | Es la latencia promedio de los paquetes | Milisegundos(ms) | |
| | best | El tiempo de ida y vuelta más corto para un paquete | Milisegundos(ms) | |
| | wrst | El tiempo de ida y vuelta más largo para un paquete | Milisegundos(ms) | |
| | StDev | La desviación estándar de las latencias a cada host | Milisegundos(ms) | |
| **Reporte estadístico de la red** | rxpck/s | Número total de paquetes recibidos por segundo | | sar |
| | txpck/s | Número total de paquetes transmitidos por segundo | | |
| | rxkB/s | Número total de kilobytes recibidos por segundo | | |
| | txkB/s | Número total de kilobytes transmitidos por segundo | | |
| | rxmcst/s | Número de paquetes de multidifusión recibidos por segundo | | |

Para obtener las diferentes medidas tanto del Rendimiento de red, como el de CPU se utilizaron las herramientas que se mencionan en la Tabla 18, que son My TracerRoute (MTR) (B.V, 1998) y System Activity Report (SAR) (Googlux, s.f.). Estas dos son herramientas de software libre, que permiten realizar pruebas en tiempo real ya sea en el cliente o en el servidor, realizando funciones como es cifrar y descifrar un mensaje. Por otro lado en el rendimiento del servicio lo que es el tiempo encriptado y desencriptado se realizó un programa en código Java.

En las dos secciones siguientes se comparan los resultados obtenidos tanto de la línea base como del optimizado, y el optimizado iterativo, recursivo. Se utilizó varias

técnicas de investigación, en el campo de la estadística como: (i) Histograma de densidad; y (ii) Análisis de la varianza (ANOVA).

## 5.2. Evaluación del algoritmo RSA: línea base vs optimizado

Para la evaluación del algoritmo RSA se plantea 13 escenarios, que consisten en variar el número de caracteres en una cadena, que van desde: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 50, 100 y 200 cifras. Luego se compara los valores de las variables evaluadas tanto del servidor como los del cliente y se analiza el comportamiento del algoritmo RSA línea base vs optimizado. Posterior a esto se utiliza la estadística, para el procesamiento de datos.

### 5.2.1. Evaluación del servicio

#### 5.2.1.1. Tiempo de cifrado y descifrado

**Histograma de densidad y análisis de varianza: Cliente**

Los resultados de la línea base y la propuesta optimizada (Figuras 36 y 37), ilustran un histograma de densidad con su respectivo diagrama de caja, estas estadísticas descriptivas, proporcionan los datos de las medianas respectivamente en función del tiempo de cifrado del mensaje.

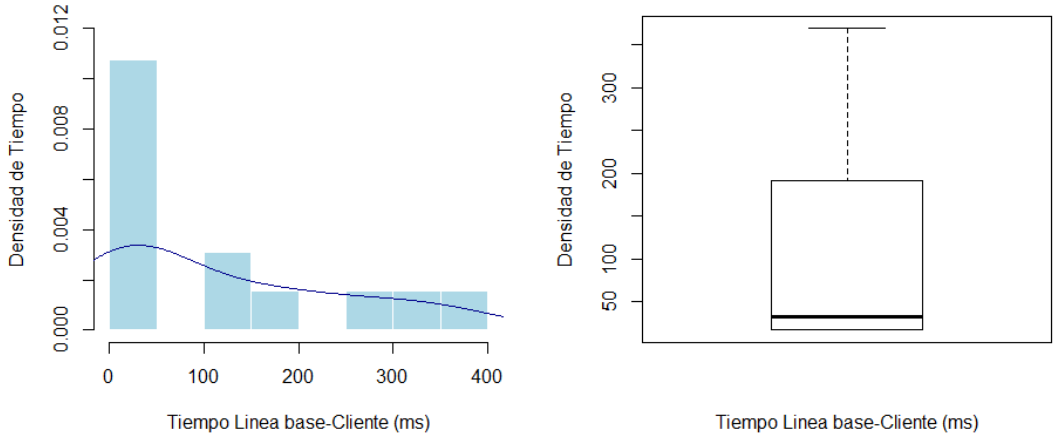


**Figura 36. Histograma de densidad y diagrama de caja del Tiempo Línea Base-Cliente**

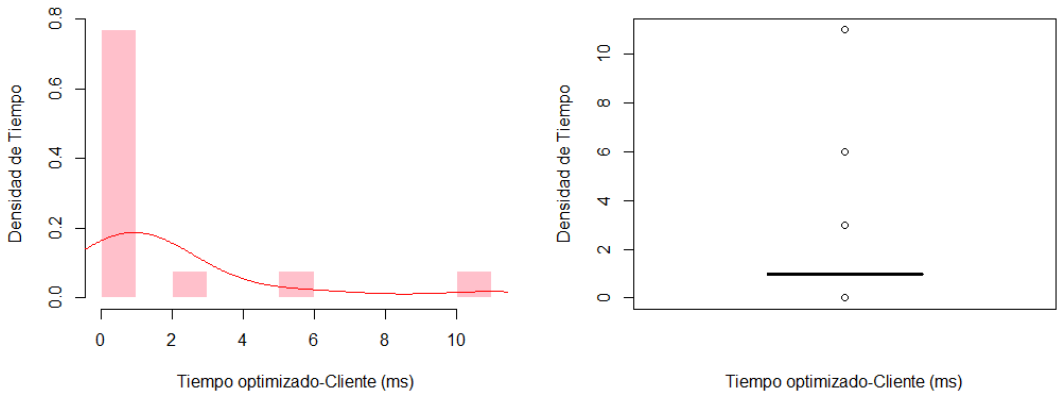


**Figura 37. Histograma de densidad y diagrama de caja del Tiempo Optimizado-Cliente**

Como se muestra en las Figuras 36 y 37 los histogramas de frecuencia son equivalentes a los diagramas de caja. Esto permite analizar parámetros estadísticos, con la finalidad de resolver o reajustar los valores extremos y atípicos, y así mejorar el nivel de significación menor al 5%. A través de los diagramas de caja se obtiene información de los valores mínimos, extremos y atípicos. Además del primer cuartil, segundo cuartil (mediana), media y tercer cuartil. La Tabla 23 describe un resumen de los datos, generado mediante el software estadístico R.

**Tabla 23**

***Datos de los diagramas de caja-Tiempo-Cliente***

| **Tiempo Línea Base-Cliente** | | | **Tiempo Optimizado-Cliente** | | |
|---|---|---|---|---|---|
| **Mínimo** | 17.00 | 4.60% | **Mínimo** | 0.00 | 0% |
| **Primer cuartil** | 18.00 | 4.87% | **Primer cuartil** | 1.00 | 9.09% |
| **Mediana** | 33.00 | 8.94% | **Mediana** | 1.00 | 9.09% |
| **Media** | 119.2 | 32.30% | **Media** | 2.154 | 19.58% |
| **Tercer cuartil** | 192.0 | 52.03% | **Tercer cuartil** | 1.00 | 9.09% |
| **Máximo** | 369.0 | 100% | **Máximo** | 11.00 | 100% |

Para el análisis de los diagramas de cajas, se considera trabajar con un Rango Inter Cuartil (RIC), que es la diferencia entre el tercer cuartil (Q3) y el primer cuartil (Q1), donde se omiten los valores extremos y atípicos, con la finalidad de volver a reajustar la información de la investigación, y así los datos estén concentrados en función de la

mediana. Como en el caso de la Figura 36, la diferencia entre el segundo cuartil y el primer cuartil es de 4.07%. Este resultado es menor al 25% de las observaciones y desde la mediana al tercer cuartil se tiene 43.09% es mayor al 25%. Esto quiere decir que la distribución de los tiempos es asimétrica por la derecha. Existiendo valores atípicos y por arriba del tercer cuartil en donde se encuentran gran cantidad de datos.

Por otro lado en la Figura 37 se tiene que la diferencia entre el segundo cuartil y el primer cuartil es de 0%, al igual que la diferencia del tercer y segundo cuartil. Esto quiere decir que la distribución de los tiempos es simétrica normalizada, con escases de valores atípicos por arriba del tercer cuartil.

Así como se realizó el análisis de diagrama de caja, es necesario analizar los histogramas de densidad mediante el cálculo de la Medida Cuartílica, a través de la ecuación (48):

$$As = \frac{Q_1 + Q_3 - 2Q_2}{Q_3 - Q_1}$$

Ecuación (48)

Para la Figura 36 el histograma de densidad muestra una curva de la distribución de frecuencia de forma asimétrica, con una medida cuartílica igual a 0.827 > 0, lo que significa que tienen una distribución asimétrica positiva, es decir los valores están concentrados en niveles superiores a la media aritmética. Por el contrario el histograma de la Figura 37, tiene una medida cuartílica igual a 0. Esto significa que la distribución de frecuencia es simétrica normalizada.

Para la evaluación de los diagramas es necesario comparar las medianas (porcentaje) donde el 8.94% corresponde al Tiempo Línea Base-Cliente y el 9.09% al Tiempo Optimizado-Cliente. Se observa que el Tiempo Optimizado es mayor al Tiempo línea Base (9.09% > 8.94%), con una diferencia del 0.15%, es decir que el 99.85% de las observaciones indica que el tiempo se optimiza de manera favorable.

La segunda opción garantiza y comprueba la optimización del algoritmo mediante el modelo de Regresión Lineal (Ver Figura 38).

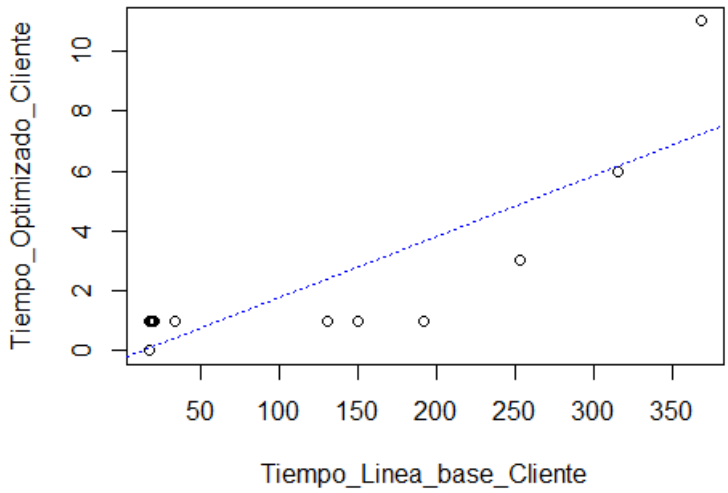


**Figura 38. Regresión Lineal del Tiempo-Cliente**

Mediante el método de Regresión Lineal se obtiene los siguientes resultados: (i) la pendiente $\beta_1 = 0.02037$ con un comportamiento creciente, y (ii) una ordenada de -0.27424. (Obsérvese ec. 49)

Tiempo-Optimizado-Cliente = - 0.27424 + 0.02037 * Tiempo-Línea base-Cliente

Ecuación (49)

Posteriormente se verifica esta información con el análisis de la varianza (ANOVA), con el objetivo de confirmar el adecuado ajuste de las cifras que son generadas por el algoritmo asimétrico optimizado RSA. (Ver Tabla 24).

**Tabla 24**

***Análisis de Varianza: Tiempo-Cliente***

| Fuente | Grados de Libertad | Suma de cuadrado | Cuadrado medio | F |
|---|---|---|---|---|
| **Regresión** | 1 | 80.410 | 80.410 | 26.576 |
| **Error Residual** | 11 | 33.282 | 3.026 | |

A partir de la Tabla de ANOVA, se realiza las diferentes pruebas de hipótesis, que consisten en:

- Hipótesis Nula $H_0$ $\beta_1 = 0$;

- Hipótesis Alternativa $H_1\ \beta_1 \neq 0$;
- Estadístico de prueba es F=26.576 (visto en el cuadro de ANOVA);
- Región de Rechazo: De acuerdo a la tabla de distribución de Fisher con un nivel de significación del 0.05, se tiene que $F_{0.05}(1,11) = 4.84$. Donde $F > F_{0.05}(1,11)$, es decir 26.576 > 4.84.
- Toma de decisión: Se rechaza la hipótesis nula, por la magnitud del estadístico de prueba. Se deduce que el nivel de significación del contraste es bajo. Con esto se comprueba que el tiempo del cliente si se ha optimizado en función de la línea base del cliente, lo que determina la existencia de un ajuste de la información.

**Histograma de densidad: Servidor**

A continuación se presenta los resultados de la línea base y la propuesta optimizada (Figuras 39 y 40). Estas ilustran un histograma de densidad, al igual que el diagrama de caja que proporciona el dato de la mediana, en función del tiempo de descifrado del mensaje.

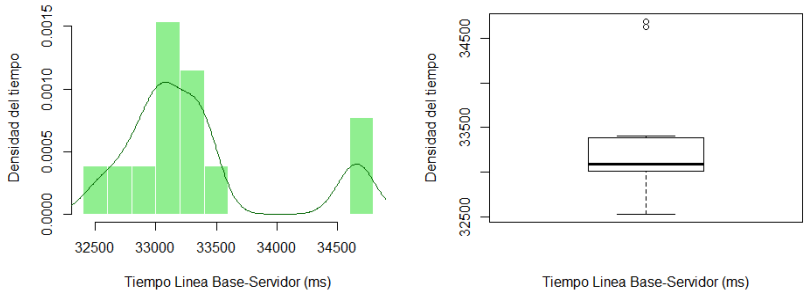


**Figura 39. Histograma de densidad y diagrama de caja del Tiempo Línea Base-Servidor**

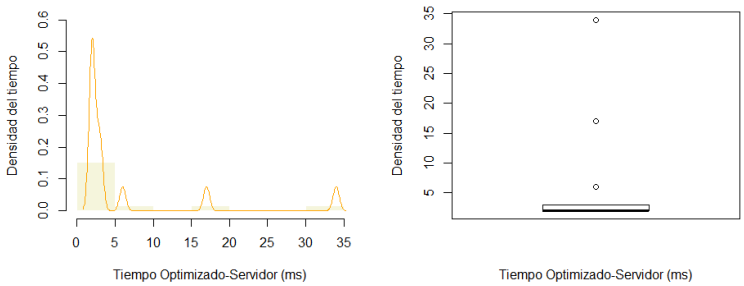


**Figura 40. Histograma de densidad y diagrama de caja del Tiempo Optimizado-Servidor**

Los histogramas de frecuencia son equivalentes a los diagramas de caja tal como se muestra en las Figuras 39 y 40. La Tabla 25 muestra un resumen de los datos de los diagramas de caja, generado mediante el software estadístico R.

**Tabla 25**

***Datos de los diagramas de caja-Tiempo-Servidor***

| **Tiempo Línea Base-Servidor** | | | **Tiempo Optimizado-Servidor** | | |
|---|---|---|---|---|---|
| **Mínimo** | 32530 | 93.76% | **Mínimo** | 2.00 | 5.88% |
| **Primer cuartil** | 33010 | 95.14% | **Primer cuartil** | 2.00 | 5.88% |
| **Mediana** | 33090 | 95.38% | **Mediana** | 2.00 | 5.88% |
| **Media** | 33310 | 96.01% | **Media** | 6.154 | 18.1% |
| **Tercer cuartil** | 33380 | 52.03% | **Tercer cuartil** | 3.00 | 8.82% |
| **Máximo** | 34693 | 96.21% | **Máximo** | 34.00 | 100% |

La Figura 39 ilustra la diferencia entre el segundo cuartil y el primer cuartil del 0.24%, este resultado es menor al 25% de las observaciones y desde el tercer cuartil a la mediana se tiene -43.35% menor al 25%, esto quiere decir que la distribución de los tiempos es asimétrica por la izquierda, tiene escases de valores atípicos y por debajo del segundo cuartil se encuentran gran cantidad de datos.

Por otro lado en la Figura 40 se tiene que la diferencia entre el segundo cuartil y el primer cuartil es de 0%, y la diferencia del tercer y segundo cuartil es 2.94%. Esto quiere decir que las observaciones tienden hacia la derecha, con tres valores atípicos por arriba del tercer cuartil.

A pesar del análisis anterior es necesario examinar los histogramas de densidad mediante el cálculo de la Medida Cuartílica. En la Figura 39 el histograma de densidad muestra una curva de la distribución de frecuencia de forma asimétrica, con una medida cuartílica igual a $0.56 > 0$. Al igual que el histograma de la Figura 40, tiene una medida cuartílica igual a $1 > 0$, esto significa que la distribución de frecuencia es asimétrica positiva, es decir los valores de ambas están concentrados en niveles superiores a la media aritmética.

Para la evaluación de los diagramas es necesario comparar las medianas (porcentaje) donde el 95.38% corresponde al Tiempo Línea Base-Servidor y el 5.88% al Tiempo Optimizado-Servidor. Se observa que el Tiempo Optimizado es menor al Tiempo línea Base (5.88% < 95.38%), con una diferencia del 89.5%. Eso quiere decir que el tiempo descifrado es menor en un 89.5% con respecto a la línea base.

De acuerdo a los tiempos máximos tanto del Tiempo optimizado-Cliente (11 ms) y línea base (369 ms), se probó que al usar el algoritmo propuesto el tiempo es 33 veces más rápido que el de línea base. Al igual que en el tiempo optimizado-servidor (34 ms) y el línea base (34693 ms), se demostró que el tiempo usado es 1020 veces más rápido que el algoritmo básico. Obteniendo así una respuesta positiva con una mejora del 89.5 % y 99% de optimización.

### 5.2.2. Evaluación del CPU

#### 5.2.2.1. Consumo de memoria

Para la evaluación de esta variable, se toma el parámetro kbmemused (i.e. cantidad de memoria usada), con una unidad de almacenamiento igual al Kilobyte abreviado como KB y es equivalente a $10^3$ bytes de memoria del ordenador.

**Histograma de densidad y análisis de varianza: Cliente**

Los resultados de la línea base y la propuesta optimizada (Figuras 41 y 42), ilustran un histograma de densidad con su respectivo diagrama de caja, estas estadísticas descriptivas proporcionan los datos de las medianas en función de la memoria al cifrar el mensaje.

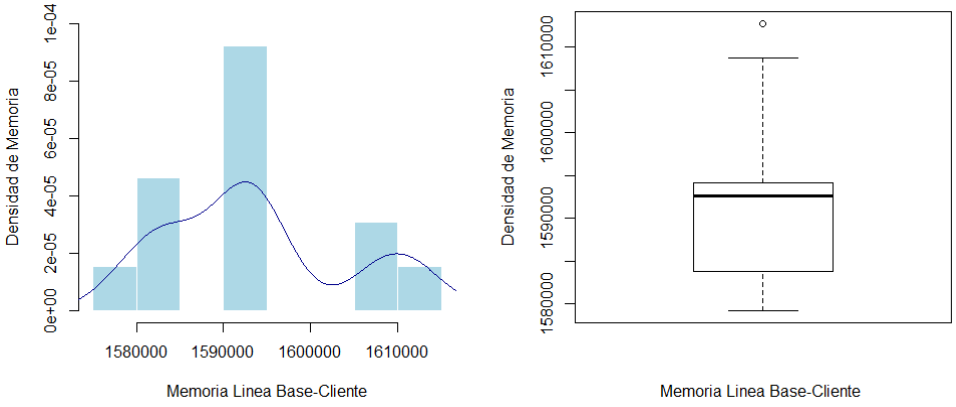


**Figura 41.Histograma de densidad y diagrama de caja Consumo Memoria Línea Base-Cliente**

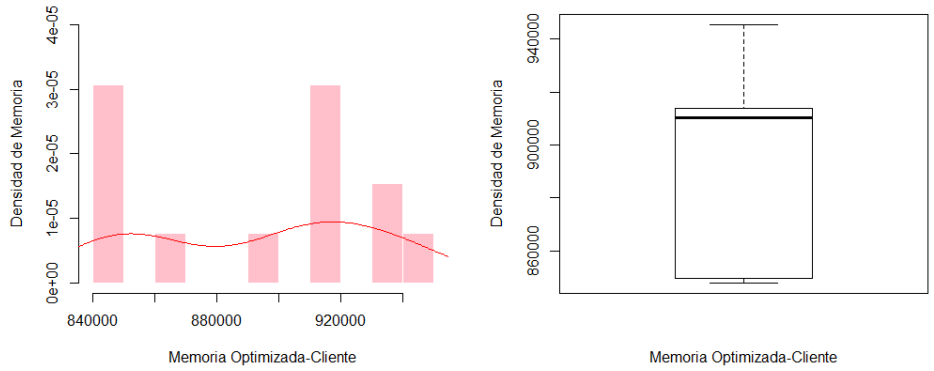


**Figura 42.Histograma de densidad y diagrama de caja Consumo Memoria Optimizado-Cliente**

Los histogramas de frecuencia, son equivalentes a los diagramas de caja tal como se muestra en las Figuras 41 y 42. La Tabla 26 describe un resumen de los datos, generado por el software estadístico R.

**Tabla 26**

***Datos de los diagramas de caja-Memoria-Cliente***

| **Memoria Línea Base-Cliente** | | | **Memoria Optimizada-Cliente** | | |
|---|---|---|---|---|---|
| **Mínimo** | 1579000 | 97.89% | **Mínimo** | 847900 | 89.68% |
| **Primer cuartil** | 1584000 | 98.20% | **Primer cuartil** | 849600 | 89.86% |
| **Mediana** | 1593000 | 98.76% | **Mediana** | 910200 | 96.27% |
| **Media** | 1594000 | 98.82% | **Media** | 893800 | 94.54% |
| | | | **CONTINUA** | | |

| | | | | | |
|---|---|---|---|---|---|
| **Tercer cuartil** | 1594000 | 98.82% | **Tercer cuartil** | 914000 | 96.67% |
| **Máximo** | 1613000 | 100% | **Máximo** | 945400 | 100% |

La Figura 41 ilustra la diferencia entre el segundo cuartil y el primer cuartil del 0.56%, este resultado es menor al 25% de las observaciones y desde el tercer cuartil a la mediana se tiene 0.06% menor al 25%, esto quiere decir que la distribución de la memoria es asimétrica a la izquierda.

Por otro lado en la Figura 42 se tiene que la diferencia entre el segundo cuartil y el primer cuartil es de 6.41%, y la diferencia del tercer y segundo cuartil es 0.4%. Esto quiere decir que las observaciones tienden hacia la izquierda con varios valores atípicos por arriba del tercer cuartil.

A pesar del análisis anterior es necesario examinar los histogramas de densidad mediante el cálculo de la Medida Cuartílica. En la Figura 41 el histograma de densidad muestra una curva de la distribución de frecuencia de forma asimétrica, con una medida cuartílica igual a $0.8 < 0$. Al igual que el histograma de la Figura 42, tiene una medida cuartílica igual a $0.88 < 0$, esto significa que la distribución de frecuencia es asimétrica negativa, es decir los valores de ambas están concentrados en niveles inferiores a la media aritmética.

Para la evaluación de estos diagramas es necesario comparar las medianas (porcentaje), donde el 98.76% corresponde a la Memoria-Línea Base-Cliente y el 96.27% a la Memoria Optimizado-Cliente. Se observa que la Memoria Optimizada es menor a la Memoria línea Base ($96.27\% < 98.76\%$), con una diferencia del 2.49%, es decir que el consumo de la memoria con el algoritmo RSA optimizado es menor en un 97.51%, con respecto a la línea base.

La segunda opción permite garantizar y comprobar la optimización del algoritmo, mediante el modelo de Regresión Lineal (Ver Figura 43).

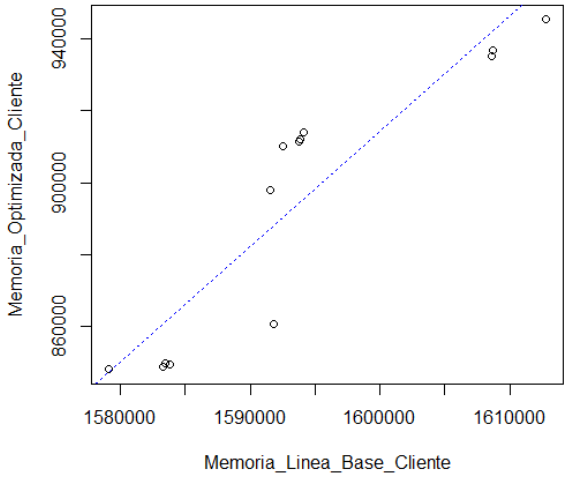


**Figura 43. Regresión Lineal de la Memoria-Cliente**

Mediante el método de Regresión Lineal se obtiene los siguientes resultados: (i) la pendiente $\beta_1 = 3.22$ con un comportamiento creciente, y (ii) una ordenada de -4238000. (Obsérvese ec. 50)

Memoria-Optimizada-Cliente = - 4238000 + 3.22 * Memoria-Línea base-Cliente

Ecuación (50)

Posteriormente, se verifica esta información con el análisis de la varianza (ANOVA), con el objetivo de confirmar el adecuado ajuste de las cifras, que son generadas por el algoritmo asimétrico optimizado RSA. (Ver Tabla 27).

**Tabla 27**

***Análisis de Varianza: Memoria-Cliente***

| Fuente | Grados de Libertad | Suma de cuadrado | Cuadrado medio | F |
|---|---|---|---|---|
| **Regresión** | 1 | $1.3921x10^{10}$ | $1.3921x10^{10}$ | 53.838 |
| **Error Residual** | 11 | $2.8442x10^{9}$ | $2.5857x10^{8}$ | |

A partir de la Tabla de ANOVA, se realiza las diferentes pruebas de hipótesis, que consisten en:

- Hipótesis Nula $H_0$ $\beta_1 = 0$;
- Hipótesis Alternativa $H_1$ $\beta_1 \neq 0$;

- Estadístico de prueba es F=53.838 (visto en el cuadro de ANOVA);
- Región de Rechazo: De acuerdo a la tabla de distribución de Fisher con un nivel de significación del 0.05, se tiene que $F_{0.05}(1,11) = 4.84$. Donde $F > F_{0.05}(1,11)$, es decir 53.838 > 4.84.
- Toma de decisión: Se rechaza la hipótesis nula, por la magnitud del estadístico de prueba. Se deduce que el nivel de significación del contraste es bajo. Con esto se comprueba que el consumo de la memoria en el cliente si se ha optimizado con respecto a la línea base del cliente, lo que determina la existencia de un ajuste de los datos.

**Histograma de densidad: Servidor**

A continuación, se presenta los resultados de la línea base y la propuesta optimizada (Figuras 44 y 45). Estas ilustran un histograma de densidad, al igual que el diagrama de caja que proporciona el dato de la mediana, respectivamente en función de la memoria del servidor.

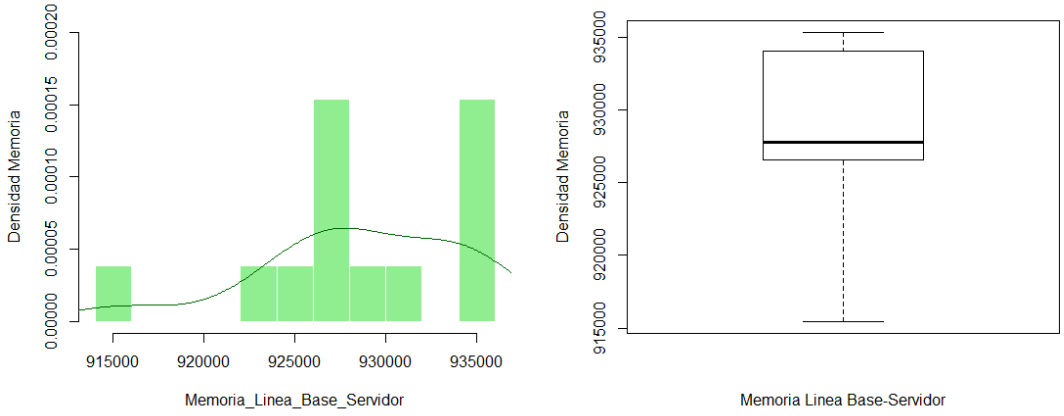


**Figura 44. Histograma de densidad y diagrama de caja Consumo Memoria Línea Base-Servidor**

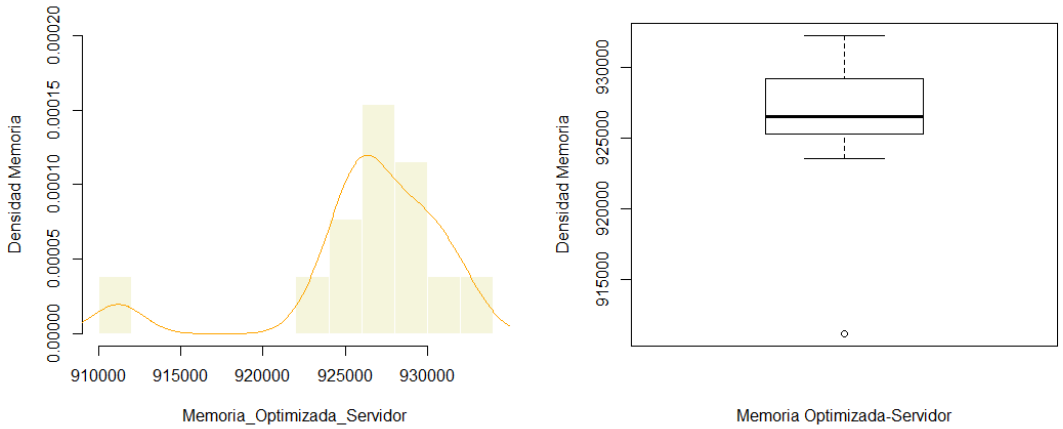


**Figura 45.Histograma de densidad y diagrama de caja Consumo Memoria Optimizada-Servidor**

Los histogramas de frecuencia son equivalentes a los diagramas de caja tal como se muestra en las Figuras 44 y 45. La Tabla 28 muestra un resumen de los datos de los diagramas de caja, generado mediante el software estadístico R.

**Tabla 28**

***Datos de los diagramas de caja-Memoria-Servidor***

| **Memoria Línea Base-Servidor** | | | **Memoria Optimizado-Servidor** | | |
|---|---|---|---|---|---|
| **Mínimo** | 915500 | 97.88% | **Mínimo** | 911200 | 97.74% |
| **Primer cuartil** | 926500 | 99.059% | **Primer cuartil** | 925300 | 99.25% |
| **Mediana** | 927800 | 99.19% | **Mediana** | 926500 | 99.38% |
| **Media** | 928600 | 99.28% | **Media** | 926400 | 99.37% |
| **Tercer cuartil** | 934000 | 99.86% | **Tercer cuartil** | 929200 | 99.67% |
| **Máximo** | 935300 | 100% | **Máximo** | 932200 | 100% |

La Figura 44, ilustra la diferencia entre el segundo cuartil y el primer cuartil del 0.131%, este resultado es menor al 25% de las observaciones y desde el tercer cuartil a la mediana se tiene 0.67% menor al 25%, esto quiere decir que la distribución de los datos es asimétrica por la derecha, tiene varios valores atípicos por debajo del primer cuartil.

Por otro lado en la Figura 45 se tiene que la diferencia entre el segundo cuartil y el primer cuartil es de 0.13%, y la diferencia del tercer y segundo cuartil es 0.29%.

Esto quiere decir que las observaciones tienden hacia la derecha, con varios valores atípicos por arriba del tercer cuartil.

Así como se realiza el análisis de diagrama de caja, es necesario examinar los histogramas de densidad mediante el cálculo de la Medida Cuartílica. La Figura 44, tiene una medida cuartílica igual a 0.65 > 0. Al igual que el histograma de la Figura 45, posee una medida cuartílica igual a 0.38 > 0, esto significa que la distribución de frecuencia es asimétrica positiva, es decir los valores de ambas están concentrados en niveles superiores a la media aritmética.

Para realizar la evaluación de los diagramas es necesario comparar las medianas (porcentaje) donde el 99.19% corresponde a la Memoria Línea Base-Servidor y el 99.38% a la Memoria Optimizada-Servidor. Se observa que la Memoria Optimizada es mayor a la Memoria línea Base (99.38% > 99.19%), con una diferencia del 0.19%, es decir que el 99.81% de las observaciones indica que la memoria se optimizó.

De acuerdo a los valores máximos tanto de la Memoria Optimizada-Cliente (945400) y línea base (1613000), se demuestra que se disminuye 1.70 veces el consumo de la memoria. De la misma forma en la Memoria Optimizada-Servidor (932200) y línea base (935300), se indica que la disminución de la memoria es mínima. Pese a esto los dos casos presentan que el consumo de la memoria ha sido optimizado.

#### 5.2.2.2. Consumo de procesador

**Análisis de varianza: Cliente**

El análisis de varianza garantiza la optimización del algoritmo RSA mediante el modelo de Regresión Lineal como se muestra a continuación en la Figura 46:

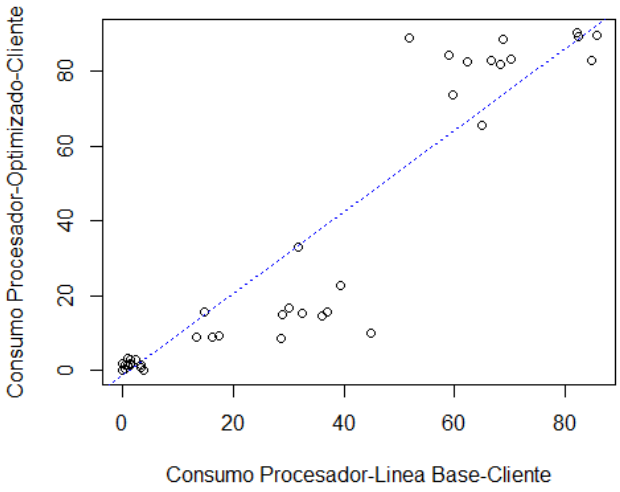


**Figura 46. Regresión lineal del Consumo del Procesador-Cliente**

A través del método de Regresión Lineal se obtiene los siguientes resultados: (i) la pendiente $\beta_1 = 1.096$ con un comportamiento creciente, y (ii) una ordenada de -1.590. (Obsérvese ec. 51)

Procesador-Optimizado-Cliente = - 1.590 + 1.096 * Procesador-Línea base-Cliente

Ecuación (51)

Posteriormente, se verifica esta información con el análisis de la varianza (ANOVA), con el objetivo de confirmar el adecuado ajuste de las cifras, que son generadas por el algoritmo asimétrico optimizado RSA. (Ver Tabla 29).

**Tabla 29**

***Análisis de Varianza: Procesador-Cliente***

| Fuente | Grados de Libertad | Suma de cuadrado | Cuadrado medio | F |
|---|---|---|---|---|
| **Regresión** | 1 | 65841 | 65841 | 718.39 |
| **Error Residual** | 76 | 6965 | 92 | |

A partir de la Tabla de ANOVA, se realiza las diferentes pruebas de hipótesis, que consisten en:

- Hipótesis Nula $H_0$ $\beta_1 = 0$;

- Hipótesis Alternativa $H_1$ $\beta_1 \neq 0$;
- Estadístico de prueba es F=718.39 (visto en el cuadro de ANOVA);
- Región de Rechazo: De acuerdo a la tabla de distribución de Fisher con un nivel de significación del 0.05, se tiene que $F_{0.05}(1,11) = 4.84$. Donde $F > F_{0.05}(1,11)$, es decir 718.39 > 4.84.
- Toma de decisión: Se rechaza la hipótesis nula, por la magnitud del estadístico de prueba. Se deduce que el nivel de significación del contraste es bajo. Con esto se comprueba que el consumo del procesador se ha optimizado y que la ecuación conseguida es óptima para el modelado de los datos.

**Análisis de varianza: Servidor**

El análisis de varianza garantiza y comprueba la optimización del algoritmo, se usa el modelo de Regresión Lineal como se muestra a continuación en la Figura 47:

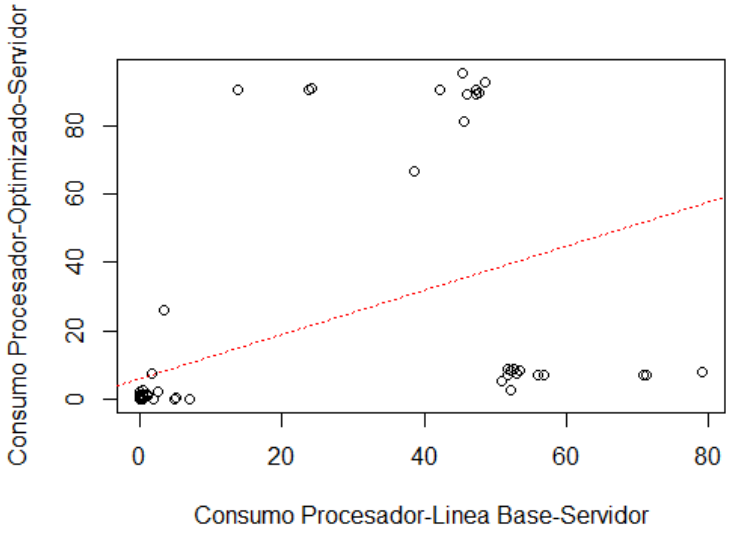


**Figura 47. Regresión lineal del Consumo del Procesador-Servidor**

Mediante el método de Regresión Lineal se obtiene los siguientes resultados: (i) la pendiente $\beta_2 = 0.6506$ con un comportamiento creciente, y (ii) una ordenada de 5.8168. (Obsérvese ec. 52)

Procesador-Optimizado-Serv = 5.8168 + 0.6506 * Procesador-Línea base-Serv

Ecuación (52)

Posteriormente, se verifica esta información con el análisis de la varianza (ANOVA), con el objetivo de confirmar el adecuado ajuste de las cifras, que son generadas por el algoritmo asimétrico optimizado RSA. (Ver Tabla 30).

**Tabla 30**

***Análisis de Varianza: Procesador-Servidor***

| Fuente | Grados de Libertad | Suma de cuadrado | Cuadrado medio | F |
|---|---|---|---|---|
| **Regresión** | 1 | 19096 | 19096 | 23.23 |
| **Error Residual** | 76 | 62474 | 822 | |

A partir de la Tabla de ANOVA, se realiza las diferentes pruebas de hipótesis, que consisten en:

- Hipótesis Nula $H_0\ \beta_1 = 0$;
- Hipótesis Alternativa $H_1\ \beta_1 \neq 0$;
- Estadístico de prueba es F=23.23 (visto en el cuadro de ANOVA);
- Región de Rechazo: De acuerdo a la tabla de distribución de Fisher con un nivel de significación del 0.05, se tiene que $F_{0.05}(1,11) = 4.84$. Donde $F > F_{0.05}(1,11)$, es decir 23.23 > 4.84.
- Toma de decisión: Se rechaza la hipótesis nula, por la magnitud del estadístico de prueba. Se deduce que la dispersión de Y con respecto a X es baja. Con esto se demuestra que el consumo del procesador (servidor) se ha mejorado.

### 5.2.3. Evaluación de la red

#### 5.2.3.1. Latencia

**Histograma de densidad: Cliente**

Los resultados de la línea base y del proyecto optimizado (Figuras 48 y 49), se muestran graficados en un histograma de densidad con su respectivo diagrama de caja, estas estadísticas descriptivas, proporcionan los datos de las medianas.

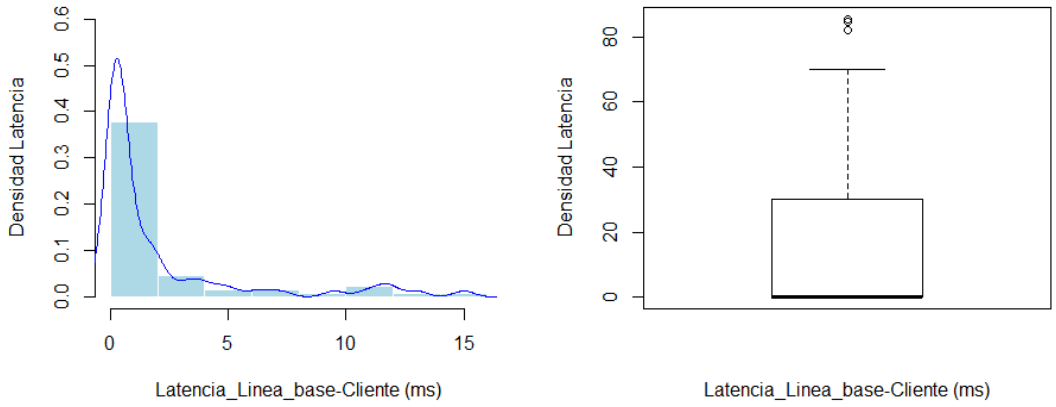


**Figura 48. Histograma de densidad y diagrama de caja de la Latencia Línea Base-Cliente**

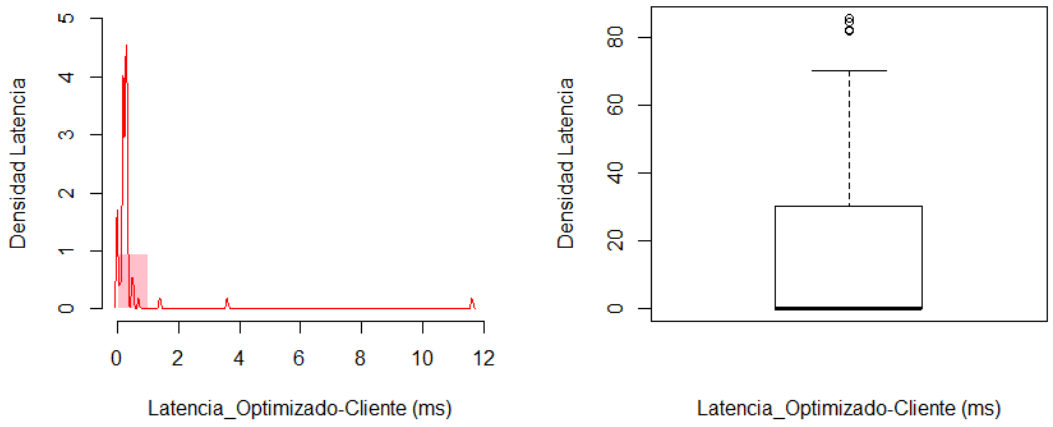


**Figura 49. Histograma de densidad y diagrama de caja de la Latencia Optimizada-Cliente**

Los histogramas de frecuencia son equivalentes a los diagramas de caja, tal como se muestra en las Figuras 48 y 49. La Tabla 31 muestra un resumen de los datos de los diagramas de cajas.

**Tabla 31**

***Datos de los diagramas de caja-Latencia-Cliente***

| **Latencia Línea Base-Cliente** | | | **Latencia Optimizado-Cliente** | | |
|---|---|---|---|---|---|
| **Mínimo** | 0.00 | 0% | **Mínimo** | 0.00 | 0% |
| **Primer cuartil** | 0.30 | 2.00% | **Primer cuartil** | 0.20 | 1.724% |
| **Mediana** | 0.30 | 2.00% | **Mediana** | 0.20 | 1.724% |
| | | | **CONTINUA** | → | |

| | | | | | |
|---|---|---|---|---|---|
| **Media** | 2.115 | 14.1% | **Media** | 0.4754 | 4.098% |
| **Tercer cuartil** | 1.90 | 12.66% | **Tercer cuartil** | 0.30 | 2.58% |
| **Máximo** | 15.00 | 100% | **Máximo** | 11.60 | 100% |

La Figura 48, ilustra la diferencia entre el segundo cuartil y el primer cuartil del 0%, este resultado es menor al 25% de las observaciones y desde el tercer cuartil a la mediana se tiene 10.66% menor al 25%, esto quiere decir que la distribución de los datos es asimétrica por la derecha, tiene varios valores atípicos por arriba del tercer cuartil.

Por otro lado en la Figura 49 se tiene que la diferencia entre el segundo cuartil y el primer cuartil es de 0%, y la diferencia del tercer y segundo cuartil es 0.856%. Esto quiere decir que las observaciones tienden hacia la derecha, con varios valores atípicos por arriba del tercer cuartil.

A pesar del análisis anterior es necesario examinar los histogramas de densidad mediante el cálculo de la Medida Cuartílica. En la Figura 48 el histograma de densidad muestra una curva de la distribución de frecuencia de forma asimétrica, con una medida cuartílica igual a $1 > 0$. Al igual que el histograma de la Figura 49, tiene una medida cuartílica igual a $1 > 0$, esto significa que la distribución de frecuencia es asimétrica positiva, es decir los valores de ambas están concentrados en niveles superiores a la media aritmética.

Para la evaluación de estos diagramas es necesario comparar las medianas (porcentaje) donde el 2.00% corresponde a la Latencia Línea Base-Cliente y el 1.724% a la Latencia Optimizada-Cliente. Por tanto se observa que la Latencia Optimizada es menor a la Latencia línea Base (1.74%<2.0%), con una diferencia del 0.26%, es decir que la latencia en el cliente es menor en un 0.26% con respecto a la línea base.

**Histograma de densidad: Servidor**

A continuación, se presenta los resultados de la línea base y de la propuesta optimizada (Figuras 50 y 51). Estas ilustran un histograma de densidad con su respectivo diagrama de caja, estas estadísticas descriptivas proporcionan los datos de las medianas.

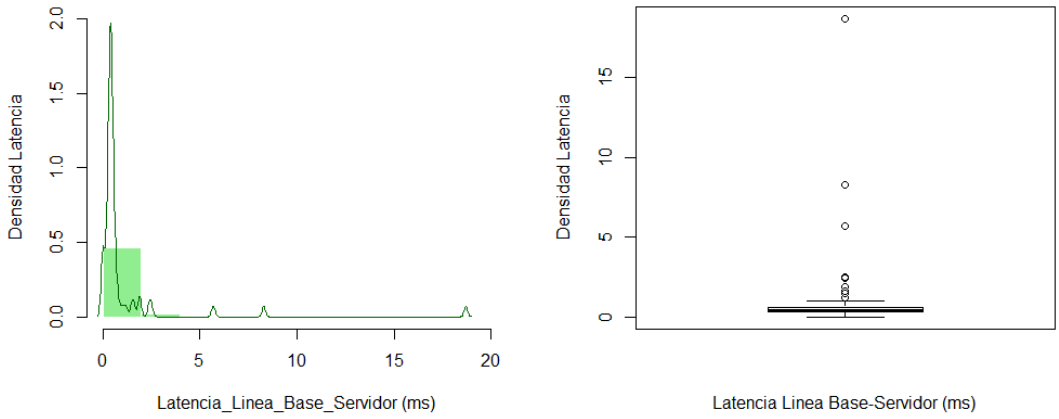


**Figura 50. Histograma de densidad y diagrama de caja de la Latencia Línea Base-Servidor**

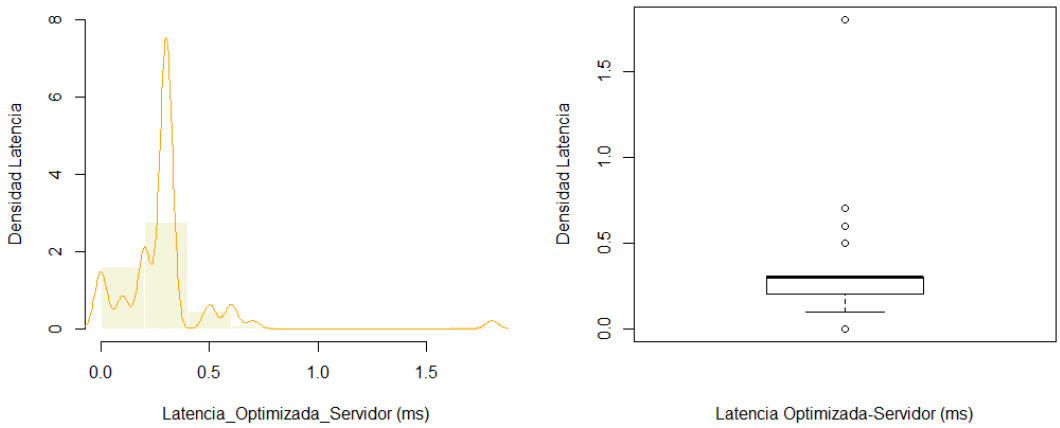


**Figura 51.Histograma de densidad y diagrama de caja de la Latencia Optimizado-Servidor**

Los histogramas de frecuencia, son equivalentes a los diagramas de caja, como se muestra en las Figuras 50 y 51. La Tabla 32 muestra un resumen de los datos de los diagramas de cajas, generado mediante el software estadístico R.

**Tabla 32**

***Datos de los diagramas de caja-Latencia-Servidor***

| **Latencia Línea Base-Servidor** | | | **Latencia Optimizado-Servidor** | | |
|---|---|---|---|---|---|
| **Mínimo** | 0.0 | 0.0% | **Mínimo** | 0.00 | 0.0% |
| **Primer cuartil** | 0.30 | 1.60% | **Primer cuartil** | 0.20 | 11.1% |
| **Mediana** | 0.40 | 2.13% | **Mediana** | 0.30 | 16.67% |
| | | | **CONTINUA** → | | |

| | | | | | |
|---|---|---|---|---|---|
| **Media** | 1.029 | 5.50% | **Media** | 0.2923 | 16.23% |
| **Tercer cuartil** | 0.60 | 3.20% | **Tercer cuartil** | 0.300 | 16.67% |
| **Máximo** | 18.70 | 100% | **Máximo** | 1.80 | 100% |

La Figura 50 ilustra una diferencia entre el segundo cuartil y el primer cuartil del 0.53%, este resultado es menor al 25% de las observaciones y desde el tercer cuartil a la mediana se tiene 1.07% menor al 25%, esto quiere decir que la distribución de los datos es asimétrica por la derecha y tiene varios de valores atípicos por arriba del tercer cuartil.

Por otro lado en la Figura 51 se tiene que la diferencia entre el segundo cuartil y el primer cuartil es de 5.57%, y la diferencia del tercer y segundo cuartil es 0%. Esto quiere decir que las observaciones tienden hacia la izquierda, con valores atípicos por debajo del primer cuartil.

Pese al análisis anterior es necesario examinar los histogramas de densidad, mediante el cálculo de la Medida Cuartílica. En la Figura 50 el histograma de densidad, posee una medida cuartílica igual a $0.33 > 0$, lo que significa que tienen una distribución asimétrica positiva, es decir los valores se hallan en niveles superiores a la media aritmética. Por el contrario el histograma de la Figura 51, tiene una medida cuartílica igual a $1 < 0$, esto significa que la distribución de frecuencia es asimétrica negativa, es decir los valores de ambas están concentrados en niveles inferiores a la media aritmética.

Para la evaluación de estos diagramas es necesario comparar las medianas (porcentaje) donde el 2.13% que corresponde a la Latencia Línea Base-Servidor y el 16.67% a la Latencia Optimizado-Servidor. Se observa que la Latencia Optimizado es mayor a la Latencia línea Base ($16.67\% > 2.13\%$), con una diferencia del 14.54%. Eso quiere decir que el la latencia se optimizó en un 85.46% con respecto al de línea base.

### 5.3. Evaluación del algoritmo RSA optimizado: iterativo vs recursivo

Como se menciona al inicio de este capítulo se plantea 13 escenarios, que consisten en variar el número de caracteres en una cadena, que van desde: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 50, 100 y 200 cifras. A continuación se compara los valores de las

variables del servidor y del cliente y se analiza el comportamiento del algoritmo RSA optimizado de manera iterativa y recursiva. Finalmente se utiliza la estadística, para el procesamiento de los datos.

### 5.3.1. Evaluación del servicio

#### 5.3.1.1. Tiempo de cifrado y descifrado

**Histograma de densidad y análisis de varianza: Cliente**

Los resultados de la propuesta optimizada de manera recursiva e iterativa (Figuras 52 y 53), ilustran un histograma de densidad, al igual que un diagrama de caja que proporciona los datos de las medianas respectivamente en función del tiempo de cifrado del mensaje.

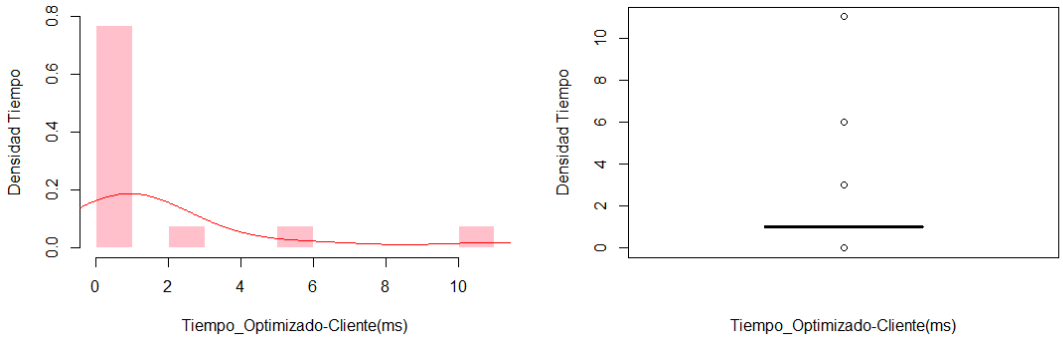


**Figura 52. Histograma de densidad y diagrama de caja del Tiempo Optimizado-Cliente**

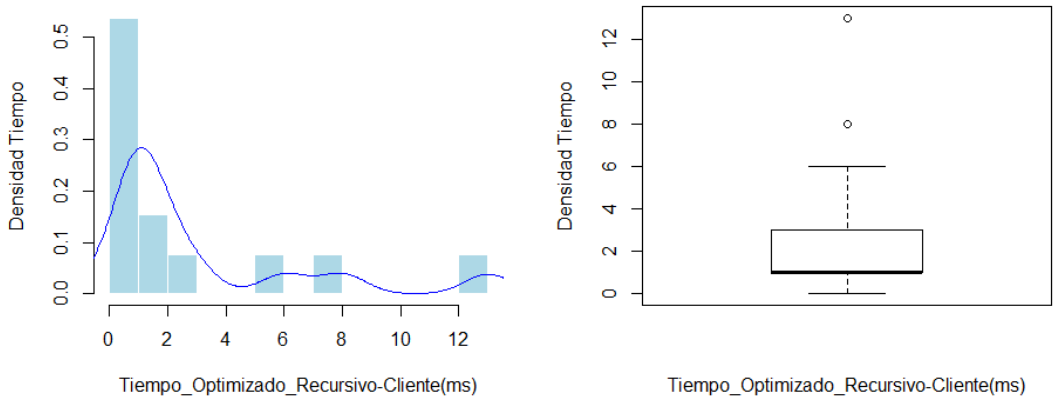


**Figura 53. Histograma de densidad y diagrama de caja del Tiempo Recursivo-Cliente**

Como se muestra en las Figuras 52 y 53, los histogramas de frecuencia son equivalentes a los diagramas de caja. La Tabla 33 muestra un resumen de los datos del diagrama de cajas, generado mediante el software estadístico R.

**Tabla 33**

***Datos de los diagramas de caja-Tiempo-Cliente***

| **Tiempo Optimizado-Cliente** | | | **Tiempo Recursivo-Cliente** | | |
|---|---|---|---|---|---|
| **Mínimo** | 0.00 | 0.0% | **Mínimo** | 0.00 | 0% |
| **Primer cuartil** | 1.00 | 9.09% | **Primer cuartil** | 1.00 | 7.69% |
| **Mediana** | 1.00 | 9.09% | **Mediana** | 1.00 | 7.69% |
| **Media** | 2.154 | 19.58% | **Media** | 3.077 | 23.66% |
| **Tercer cuartil** | 1.00 | 9.09% | **Tercer cuartil** | 3.00 | 23.07% |
| **Máximo** | 11.0 | 100% | **Máximo** | 13.00 | 100% |

La Figura 52 ilustra la diferencia entre el segundo cuartil y el primer cuartil es de 0%, al igual que la diferencia del tercer y segundo cuartil. Esto quiere decir que la distribución de los tiempos es simétrica normalizada, con escases de valores atípicos por arriba del tercer cuartil. Sin embargo en la Figura 53 se obtiene una diferencia entre el segundo cuartil y el primer cuartil del 0% y desde la mediana al tercer cuartil se tiene 15.38% menor al 25%, esto quiere decir que la distribución de los tiempos es asimétrica por la izquierda, tiene dos valores atípicos y por arriba del tercer cuartil se encuentran una cantidad regular de datos.

Además del análisis anterior, es necesario examinar los histogramas de densidad como en la Figura 52 se muestra una curva de la distribución de frecuencia de forma simétrica normalizada, con una medida cuartílica igual a 0. Por el contario el histograma de la Figura 53, tiene una medida cuartílica igual a $1 > 0$, esto significa que la distribución de frecuencia es asimétrica positiva, es decir que los valores están concentrados en niveles superiores a la media aritmética.

Para evaluar los diagramas es indispensable comparar las medianas (porcentaje) donde el 7.69% que corresponde al Tiempo Recurso-Cliente y el 9.09% al Tiempo Optimizado-Cliente. Esto permite definir que el Tiempo Optimizado es mayor al

Tiempo línea Base (9.09% > 7.69%), con una diferencia del 1.4%, es decir que el 98.6% de las observaciones indica que el Tiempo optimizado es mucho mejor que el Tiempo recursivo.

Para garantizar que el algoritmo RSA iterativo use menos recursos que el recursivo, se hace un estudio a través del modelo de regresión lineal, como se muestra a continuación en la Figura 54:

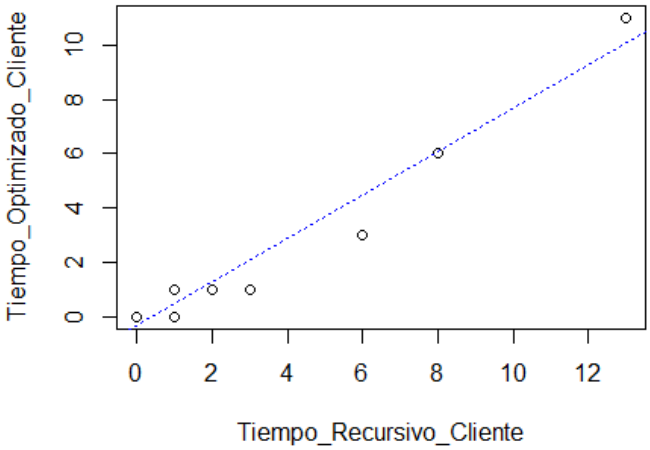


**Figura 54. Regresión lineal del Tiempo-Cliente**

A través del método de Regresión Lineal se obtiene los siguientes resultados: (i) la pendiente $\beta_1 = 0.7983$ con un comportamiento creciente, y (ii) una ordenada de -0.3024. (Obsérvese ec. 53)

Tiempo-Optimizado-Cliente = - 0.3024 + 0.7983 * Tiempo-Recursivo-Cliente

Ecuación (53)

Luego se verifica esta información con el análisis de la varianza (ANOVA), para confirmar el adecuado ajuste de las cifras, que son generadas por el algoritmo asimétrico optimizado RSA-iterativo. (Ver Tabla 34).

**Tabla 34**

***Análisis de Varianza: Tiempo-Cliente***

| Fuente | Grados de Libertad | Suma de cuadrado | Cuadrado medio | F |
|---|---|---|---|---|
| **Regresión** | 1 | 107.644 | 107.64 | 195.76 |
| **Error Residual** | 11 | 6.049 | 0.55 | |

A partir de la Tabla 34 de ANOVA, se realiza las diferentes pruebas de hipótesis, que consisten en:

- Hipótesis Nula $H_0\ \beta_1 = 0$;
- Hipótesis Alternativa $H_1\ \beta_1 \neq 0$;
- Estadístico de prueba es F=195.76 (visto en el cuadro de ANOVA);
- Región de Rechazo: De acuerdo a la tabla de distribución de Fisher con un nivel de significación del 0.05, se tiene que $F_{0.05}(1,11) = 4.84$. Donde $F > F_{0.05}(1,11)$, es decir 195.76 > 4.84.
- Toma de decisión: Se rechaza la hipótesis nula y por la magnitud del estadístico de prueba. Se deduce que el nivel de significación del contraste es sumamente bajo. Con esto se comprueba que el tiempo iterativo es mejor que el tiempo recursivo, lo que determina el óptimo ajuste de los datos.

**Histograma de densidad y Análisis de varianza: Servidor**

A continuación, se presenta los resultados de la propuesta optimizada de manera recursiva e iterativa (Figuras 55 y 56). Estas ilustran un histograma de densidad, al igual que el diagrama de caja que proporciona los datos de las medianas en función del tiempo de descifrado del mensaje.

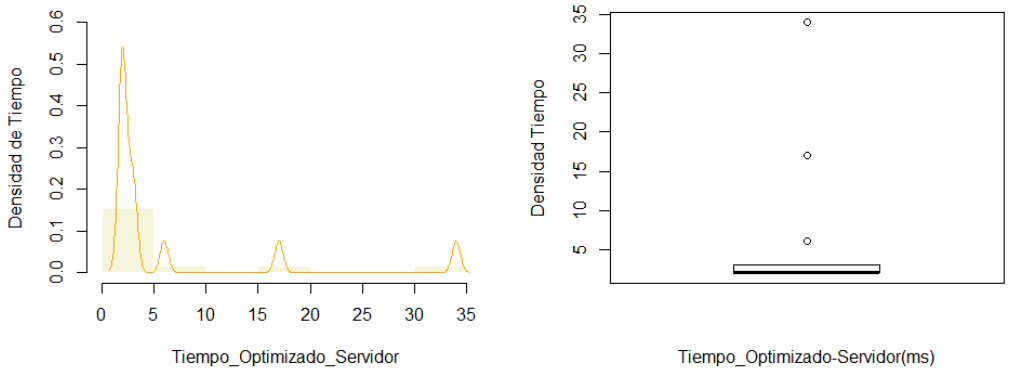


**Figura 55. Histograma de densidad y diagrama de caja del Tiempo Optimizado-Servidor**

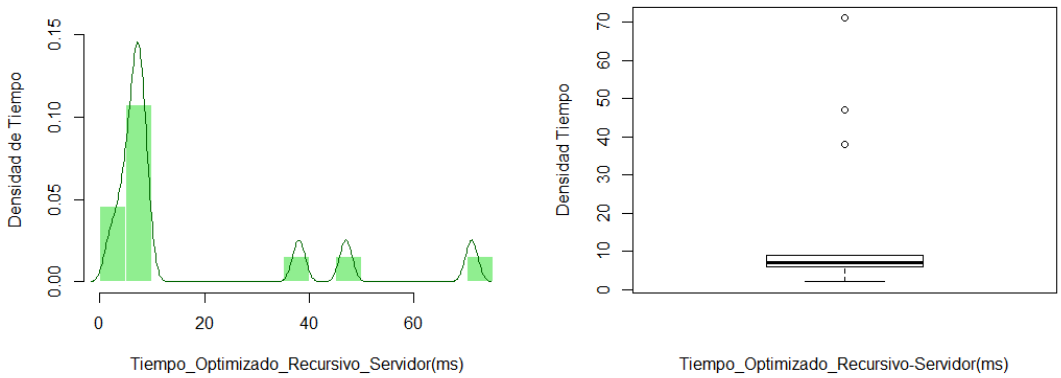


**Figura 56. Histograma de densidad y diagrama de caja del Tiempo Recursivo-Servidor**

Los histogramas de frecuencia son equivalentes a los diagramas de caja tal como se muestra en las Figuras 55 y 56. La Tabla 35 muestra un resumen de los datos de los diagramas de caja.

**Tabla 35**

***Datos de los diagramas de caja-Tiempo-Servidor***

| **Tiempo Optimizado-Servidor** | | | **Tiempo Recursivo-Servidor** | | |
|---|---|---|---|---|---|
| **Mínimo** | 2.00 | 5.88% | **Mínimo** | 2.00 | 2.81% |
| **Primer cuartil** | 2.00 | 5.88% | **Primer cuartil** | 6.00 | 8.45% |
| **Mediana** | 2.00 | 5.88% | **Mediana** | 7.00 | 9.86% |
| **Media** | 6.154 | 18.1% | **Media** | 16.85 | 23.73% |
| **Tercer cuartil** | 3.00 | 8.82% | **Tercer cuartil** | 9.00 | 12.67% |
| **Máximo** | 34.00 | 100% | **Máximo** | 71.00 | 100% |

La Figura 55 ilustra una diferencia entre el segundo cuartil y el primer cuartil del 0%, y la diferencia del tercer y segundo cuartil es 2.94%. Esto quiere decir que las observaciones tienden hacia la derecha, con tres valores atípicos por arriba del tercer cuartil. Por otro lado en la Figura 56, brinda una diferencia entre el segundo cuartil y el primer cuartil del 1.41%, este resultado es menor al 25% de las observaciones, y desde el tercer cuartil a la mediana se tiene 2.81% menor al 25%, esto quiere decir que la distribución de los tiempos es asimétrica por la derecha, tiene tres valores atípicos y por arriba de la mediana se encuentran gran cantidad de datos.

En la Figura 55 el histograma de densidad muestra una curva de la distribución de frecuencia de forma asimétrica, con una medida cuartílica igual a $1 > 0$. Al igual que el histograma de la Figura 56, tiene una medida cuartílica igual a $0.33 > 0$, esto significa que la distribución de frecuencia es asimétrica positiva, es decir los valores de ambas están concentrados en niveles superiores a la media aritmética.

Para la evaluación de estos diagramas es necesario comparar las medianas (porcentaje) donde el 9.86% corresponde al Tiempo Optimizado Recursivo y el 5.88% al Tiempo Optimizado-Servidor. Por tanto se observa que el Tiempo Optimizado iterativo es menor recursivo ($5.88\% < 9.86\%$), con una diferencia del 3.98%. Eso quiere decir que el tiempo optimizado iterativo es menor en un 96.02% con respecto al recursivo.

Para garantizar que el algoritmo RSA iterativo usa menos recursos que el recursivo, se hace un estudio a través del modelo de regresión lineal como se muestra a continuación en la Figura 57:

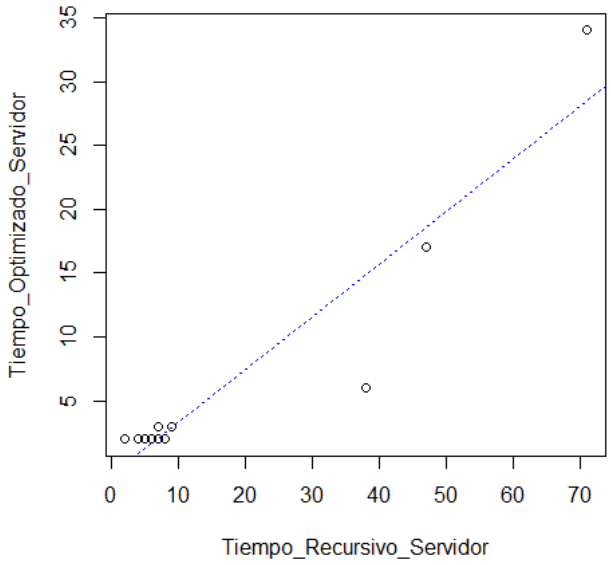


**Figura 57. Regresión lineal del Tiempo-Servidor**

Mediante el método de regresión lineal se obtiene los siguientes resultados: (i) la pendiente $\beta_2 = 0.4121$ con un comportamiento creciente, y (ii) una ordenada de -0.7878. (Observar ec. 54)

Tiempo-Optimizado-Servidor = - 0.7878 + 0.4121 * Tiempo-Recursivo-Servidor

Ecuación (54)

Posteriormente, se verifica esta información con el análisis de la varianza (ANOVA), con el objetivo de confirmar el adecuado ajuste de las cifras, que son generadas por el algoritmo asimétrico optimizado RSA-iterativo. (Ver Tabla 36).

**Tabla 36**

***Análisis de Varianza: Tiempo-Servidor***

| Fuente | Grados de Libertad | Suma de cuadrado | Cuadrado medio | F |
|---|---|---|---|---|
| **Regresión** | 1 | 923.97 | 923.97 | 84.891 |
| **Error Residual** | 11 | 119.73 | 10.88 | |

A partir de la Tabla de ANOVA, se realiza las diferentes pruebas de hipótesis, que consisten en:

- Hipótesis Nula $H_0\ \beta_1 = 0$;
- Hipótesis Alternativa $H_1\ \beta_1 \neq 0$;
- Estadístico de prueba es F=84.891 (visto en el cuadro de ANOVA);
- Región de Rechazo: De acuerdo a la tabla de distribución de Fisher con un nivel de significación del 0.05, se tiene que $F_{0.05}(1,11) = 4.84$. Donde $F > F_{0.05}(1,11)$, es decir 84.891 > 4.84.
- Toma de decisión: Se rechaza la hipótesis nula, por la magnitud del estadístico de prueba. Se deduce que el nivel de significación del contraste es bajo. Con esto se comprueba que el tiempo iterativo del servidor es mejor al tiempo recursivo, lo que determina la existencia de un excelente ajuste de la información.

De acuerdo a los tiempos máximos tanto del Tiempo optimizado-Cliente (11 ms) y recursivo (13 ms), se prueba que al usar el algoritmo optimizado iterativo el tiempo es 1.8 veces más rápido que el recursivo. Al igual que en el Tiempo Optimizado-Servidor iterativo (34 ms) y el recursivo (71 ms), se demuestra que el tiempo usado es 2.08 veces más rápido que el algoritmo recursivo. Obteniendo así una respuesta positiva del algoritmo RSA iterativo respecto al recursivo con un 98.6 % y 96.02% de optimización.

### 5.3.2. Evaluación del CPU

#### 5.3.2.1. Consumo de memoria

Para la evaluación de esta variable se toma en cuenta el parámetro kbmemused (i.e. cantidad de memoria usada), con una unidad de almacenamiento igual al Kilobyte abreviado como KB y es equivalente a $10^3$ bytes de memoria del ordenador.

**Histograma de densidad: Cliente**

Los resultados de la propuesta optimizada de manera recursiva e iterativa (Figuras 58 y 59), ilustran un histograma de densidad con su respectivo diagrama de caja, estas estadísticas descriptivas proporcionan los datos de las medianas en función de la memoria al cifrar el mensaje.

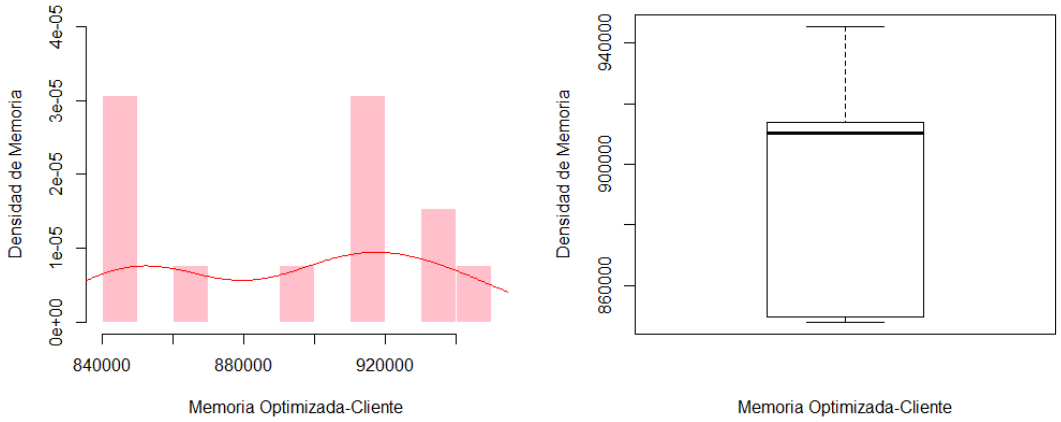


**Figura 58.Histograma de densidad y diagrama de caja Consumo Memoria Optimizada-Cliente**

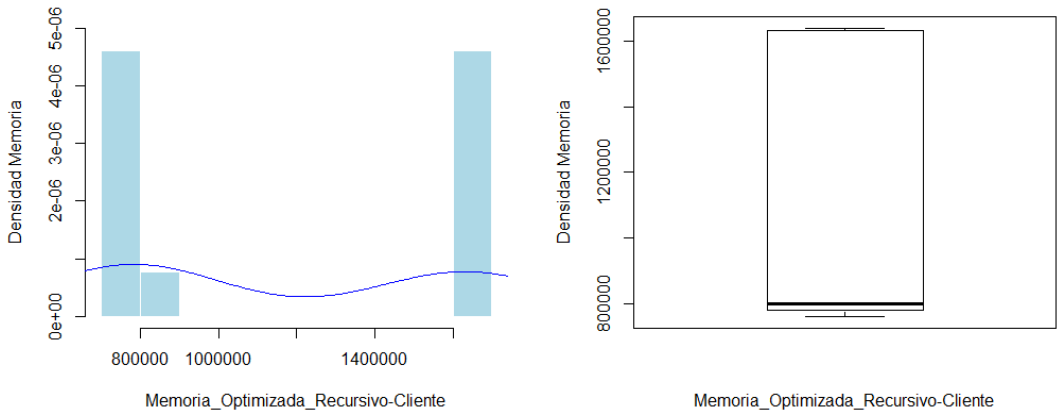


**Figura 59.Histograma de densidad y diagrama de caja Consumo Memoria Recursivo-Cliente**

Los histogramas de frecuencia son equivalentes a los diagramas de caja tal como se muestra en las Figuras 58 y 59. La Tabla 37 muestra un resumen de los datos de los diagramas de caja.

**Tabla 37**

***Datos de los diagramas de caja-Memoria-Cliente***

| **Memoria Optimizada-Cliente** | | | **Memoria Recursiva-Cliente** | | |
|---|---|---|---|---|---|
| **Mínimo** | 847900 | 89.68% | **Mínimo** | 761900 | 46.51% |
| **Primer cuartil** | 849600 | 89.86% | **Primer cuartil** | 779000 | 47.55% |
| **Mediana** | 910200 | 96.27% | **Mediana** | 801100 | 48.90% |
| | | | **CONTINUA** | → | |

| | | | | | |
|---|---|---|---|---|---|
| **Media** | 893800 | 94.54% | **Media** | 1176000 | 71.79% |
| **Tercer cuartil** | 914000 | 96.67% | **Tercer cuartil** | 1633000 | 99.69% |
| **Máximo** | 945400 | 100% | **Máximo** | 1638000 | 100% |

La Figura 58, ilustra una diferencia entre el segundo cuartil y el primer cuartil del 6.41%, este resultado es menor al 25% de las observaciones y desde el tercer cuartil a la mediana se tiene 0.4% menor al 25%. Esto quiere decir que la distribución de la memoria es asimétrica a la izquierda.

Por otro lado en la Figura 59 se tiene que la diferencia entre el segundo cuartil y el primer cuartil es de 1.35%, y la diferencia del tercer y segundo cuartil es 50.79%. Esto quiere decir que las observaciones tienden hacia la derecha con varios valores atípicos por abajo del primer cuartil.

A pesar del análisis anterior es necesario examinar los histogramas de densidad mediante el cálculo de la Medida Cuartílica. En la Figura 58 el histograma de densidad muestra una curva de la distribución de frecuencia de forma asimétrica, con una medida cuartílica igual a $0.88 < 0$, lo que significa que tienen una distribución asimétrica negativa, es decir los valores están concentrados en niveles inferiores a la media aritmética. Por el contario el histograma de la Figura 59, tiene una medida cuartílica igual a $0.97 > 0$, esto significa que la distribución de frecuencia es asimétrica positiva, es decir los valores están concentrados en niveles superiores a la media aritmética.

Para la evaluación de estos diagramas es necesario comparar las medianas (porcentaje) donde el 48.90% corresponde a la Memoria-Recursiva-Cliente y el 96.27% a la Memoria Optimizado-Cliente. Se observa que la Memoria Optimizada es mayor a la Memoria Recursiva (96.27% > 48.90%), con una diferencia del 47.37%, es decir que el 52.63% de las observaciones indica que el consumo de la memoria con el algoritmo RSA optimizado-iterativo es favorable con respecto a la línea base.

**Histograma de densidad: Servidor**

A continuación, se presenta los resultados de la propuesta optimizada de manera recursiva e iterativa (Figuras 60 y 61). Estas ilustran un histograma de densidad, al

igual que el diagrama de caja que proporciona el valor de la mediana, respectivamente en función de la memoria del servidor.

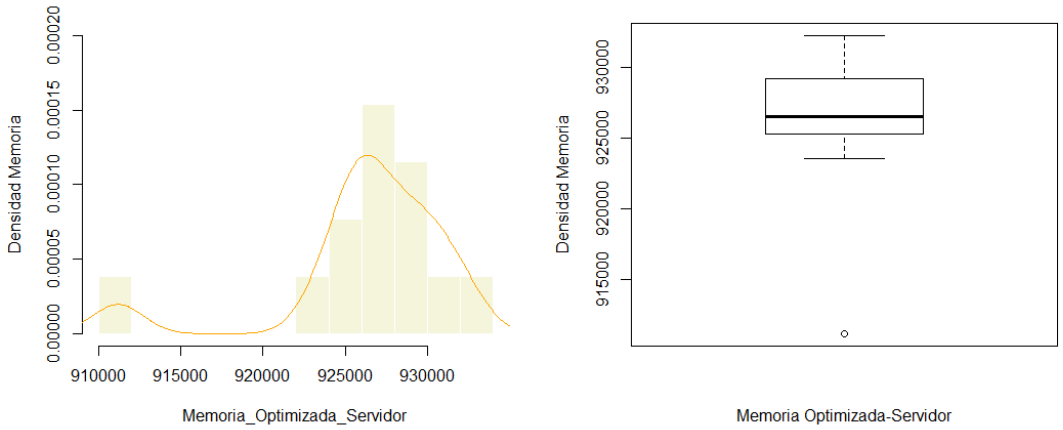


**Figura 60. Histograma de densidad y diagrama de caja Consumo Memoria Optimizada-Servidor**

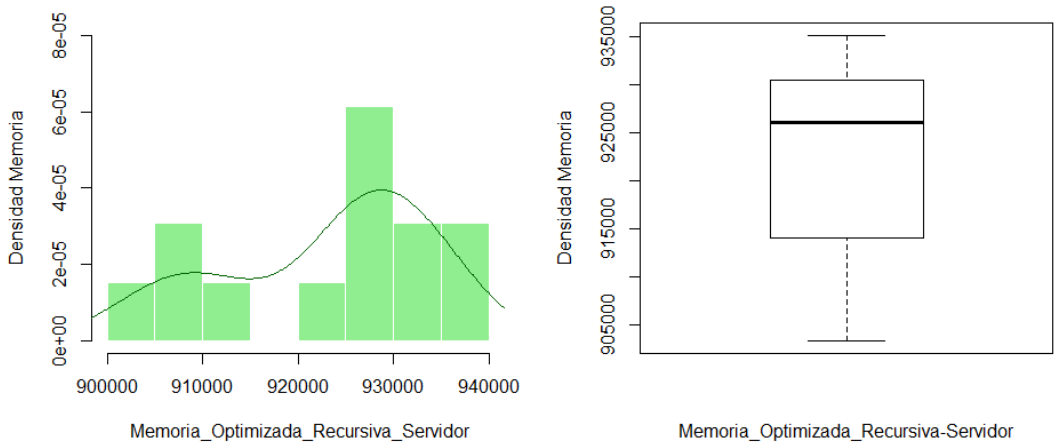


**Figura 61.Histograma de densidad y diagrama de caja Consumo Memoria Recursiva-Servidor**

Los histogramas de frecuencia son equivalentes a los diagramas de caja tal como se muestra en las Figuras 60 y 61. La Tabla 38 muestra un resumen de los datos del diagrama de caja, generado mediante el software estadístico R.

**Tabla 38**

***Datos de los diagramas de caja-Memoria-Servidor***

| **Memoria Optimizada-Servidor** | | | **Memoria Recursiva-Servidor** | | |
|---|---|---|---|---|---|
| **Mínimo** | 911200 | 97.74% | **Mínimo** | 903400 | 96.59% |
| **Primer cuartil** | 925300 | 99.25% | **Primer cuartil** | 914100 | 97.74% |
| **Mediana** | 926500 | 99.38% | **Mediana** | 926100 | 99.02% |
| **Media** | 926400 | 99.37% | **Media** | 922900 | 98.68% |
| **Tercer cuartil** | 929200 | 99.67% | **Tercer cuartil** | 930600 | 99.50% |
| **Máximo** | 932200 | 100% | **Máximo** | 935200 | 100% |

La Figura 60 ilustra una diferencia entre el segundo cuartil y el primer cuartil del 0.13%, este resultado es menor al 25% de las observaciones y desde el tercer cuartil a la mediana se tiene 0.29% menor al 25%, esto quiere decir que la distribución de los tiempos es asimétrica por la derecha y tiene más valores atípicos por arriba del tercer cuartil.

Por otro lado en la Figura 61 se tiene que la diferencia entre el segundo cuartil y el primer cuartil es de 1.28%, y la diferencia del tercer y segundo cuartil es 0.48%. Esto quiere decir que las observaciones tienden hacia la izquierda, con más valores atípicos por debajo del segundo cuartil.

Así como se realizó el análisis de diagrama de caja, es necesario analizar los histogramas de densidad mediante el cálculo de la Medida Cuartílica. La Figura 60, tiene una medida cuartílica igual a $0.38 > 0$, lo que significa que tienen una distribución asimétrica positiva, es decir los valores están concentrados en niveles superiores a la media aritmética. Por el contrario en el histograma de la Figura 61, posee una medida cuartílica igual a $0.45 < 0$, esto significa que la distribución de frecuencia es asimétrica negativa, es decir los valores están concentrados en niveles inferiores a la media aritmética.

Para realizar la evaluación de los diagramas es necesario comparar las medianas (porcentaje) donde el 99.02% corresponde a la Memoria Recursiva-Servidor y el 99.38% a la Memoria Optimizada-Servidor. Se observa que la Memoria Optimizada

es mayor a la Memoria Recursiva (99.38% > 99.02%), con una diferencia del 0.36%, es decir que el 99.64% de las observaciones indica que la memoria sigue siendo óptima con el algoritmo RSA-iterativo.

De acuerdo a los valores máximos tanto de la Memoria Optimizada iterativa-Cliente (945400) y recursiva (1638000), se demuestra que disminuye 1.73 veces el consumo de la memoria. De la misma forma en la Memoria Optimizada iterativa-Servidor (932200) y recursiva (935200), se indica que la disminución de la memoria es mínima. Pese a esto, los dos casos presentan que el consumo de la memoria con el algoritmo RSA iterativo sigue siendo óptima.

#### 5.3.2.2. Consumo de procesador

**Análisis de varianza: Cliente**

Se garantiza la optimización del algoritmo RSA mediante el modelo de Regresión Lineal como se muestra a continuación en la Figura 62:

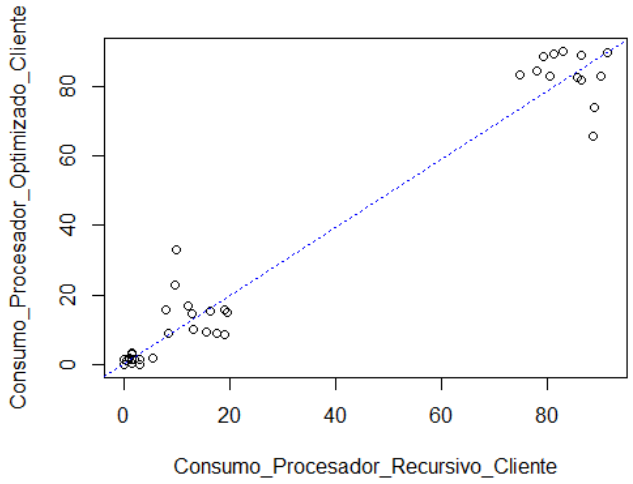


**Figura 62. Regresión lineal del Consumo del Procesador-Cliente**

A través del método de Regresión Lineal se obtuvo los siguientes resultados: (i) la pendiente $\beta_1 = 0.9784$ con un comportamiento creciente, y (ii) una ordenada de 0.3600. (Obsérvese ec. 55)

Procesador-Optimizado-Cliente = 0.3600 + 0.9784 * Procesador-Recursivo-Cliente

Ecuación (55)

Posteriormente, se verifica esta información con el análisis de la varianza (ANOVA), con el objetivo de confirmar el adecuado ajuste de las cifras, que son generadas por el algoritmo asimétrico optimizado RSA. (Ver Tabla 39).

**Tabla 39**

***Análisis de Varianza: Procesador-Cliente***

| Fuente | Grados de Libertad | Suma de cuadrado | Cuadrado medio | F |
|---|---|---|---|---|
| **Regresión** | 1 | 70581 | 70581 | 2409.4 |
| **Error Residual** | 76 | 2226 | 29 | |

A partir de la Tabla de ANOVA, se realiza las diferentes pruebas de hipótesis, que consisten en:

- Hipótesis Nula $H_0\ \beta_1 = 0$;
- Hipótesis Alternativa $H_1\ \beta_1 \neq 0$;
- Estadístico de prueba es F=2409.4 (visto en el cuadro de ANOVA);
- Región de Rechazo: De acuerdo a la tabla de distribución de Fisher con un nivel de significación del 0.05, se tiene que $F_{0.05}(1{,}11) = 4.84$. Donde $F > F_{0.05}(1{,}11)$, es decir 2409.4 > 4.84.
- Toma de decisión: Se rechaza la hipótesis nula, por la magnitud del estadístico de prueba. Se deduce que el nivel de significación del contraste es sumamente bajo. Al emplear el algoritmo iterativo RSA, el consumo del procesador sigue siendo óptimo y que la ecuación determina el ajuste para el modelado de los datos.

**Análisis de varianza: Servidor**

Se garantiza la optimización del algoritmo mediante el modelo de Regresión Lineal como se muestra a continuación en la Figura 63:

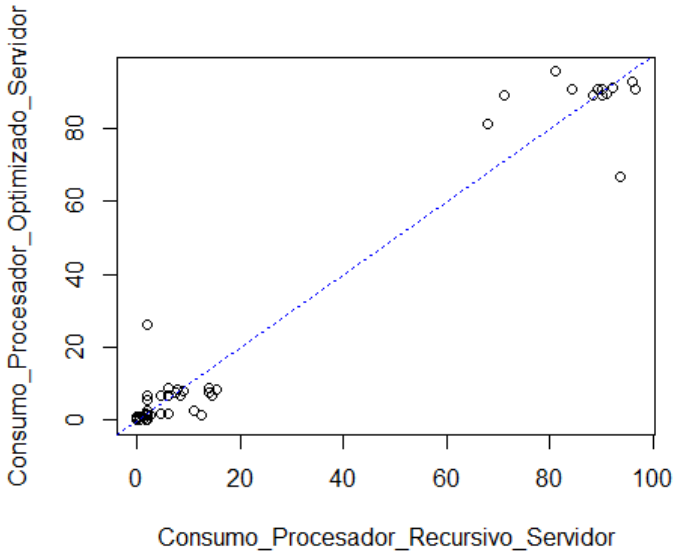


**Figura 63. Regresión lineal del Consumo del Procesador-Servidor**

Mediante el método de Regresión Lineal se obtiene los siguientes resultados: (i) la pendiente $\beta_2 = 0.99835$ con un comportamiento creciente, y (ii) una ordenada de 0.02142. (Obsérvese ec. 56)

Procesador-Optimizado-Serv = 0.02142 + 0.99835 * Procesador-Recursivo-Serv

Ecuación (56)

Posteriormente, se verifica esta información con el análisis de la varianza (ANOVA), con el objetivo de confirmar el adecuado ajuste de las cifras, que son generadas por el algoritmo asimétrico optimizado RSA. (Ver Tabla 40).

**Tabla 40**

***Análisis de Varianza: Procesador-Servidor***

| Fuente | Grados de Libertad | Suma de cuadrado | Cuadrado medio | F |
|---|---|---|---|---|
| **Regresión** | 1 | 79018 | 79018 | 2353.4 |
| **Error Residual** | 76 | 2552 | 34 | |

A partir de la Tabla de ANOVA, se realiza las diferentes pruebas de hipótesis, que consisten en:

- Hipótesis Nula $H_0\ \beta_1 = 0$;
- Hipótesis Alternativa $H_1\ \beta_1 \neq 0$;
- Estadístico de prueba es F=2353.4 (visto en el cuadro de ANOVA);
- Región de Rechazo: De acuerdo a la tabla de distribución de Fisher con un nivel de significación del 0.05, se tiene que $F_{0.05}(1,11) = 4.84$. Donde $F > F_{0.05}(1,11)$, es decir 2353.4 > 4.84.
- Toma de decisión: Se rechaza la hipótesis nula, por la magnitud del estadístico de prueba. Se deduce que la dispersión de Y con respecto a X es baja. Esto indica que al usar el algoritmo RSA iterativo, el consumo del procesador en el servidor es óptimo ante el recursivo.

### 5.3.3. Evaluación de la red

#### 5.3.3.1. Latencia

**Histograma de densidad: Cliente**

Los resultados de la de la propuesta optimizada de manera recursiva e iterativa (Figuras 64 y 65). Estas ilustran un histograma de densidad, al igual que un diagrama de caja, estas estadísticas descriptivas proporcionan los datos de las medianas.

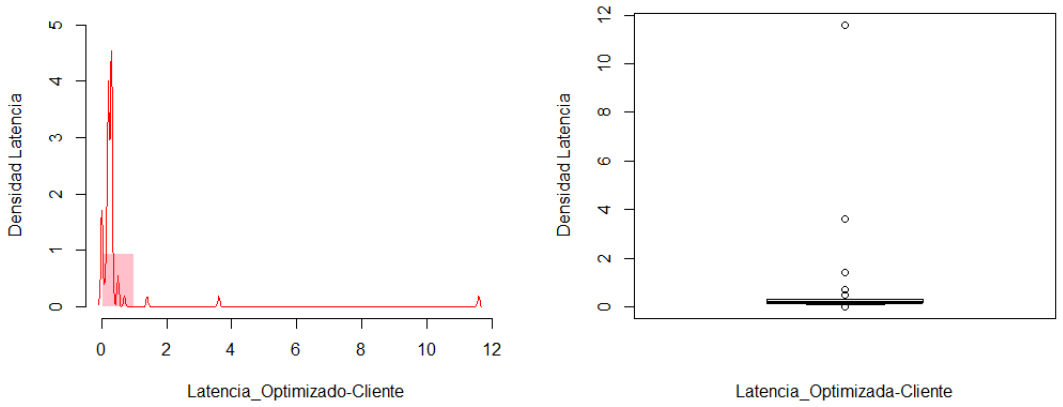


**Figura 64. Histograma de densidad y diagrama de caja de la Latencia Optimizada-Cliente**

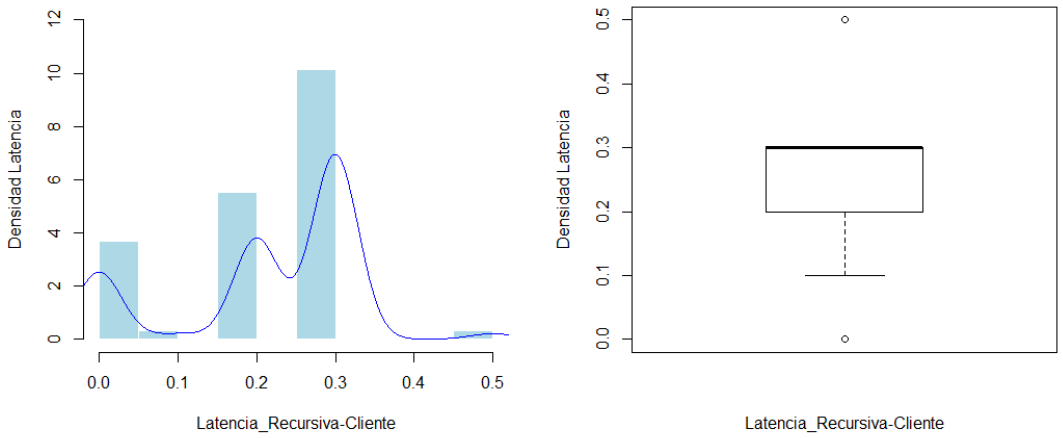


**Figura 65. Histograma de densidad y diagrama de caja de la Latencia Recursiva-Cliente**

Los histogramas de frecuencia son equivalentes a los diagramas de caja tal como se muestra en las Figuras 64 y 65. La Tabla 41 muestra un resumen de los datos generados de los diagramas de clases, generado mediante el software estadístico R.

**Tabla 41**

***Datos de los diagramas de caja-Latencia-Cliente***

| **Latencia Optimizado-Cliente** | | | **Latencia Recursivo-Cliente** | | |
|---|---|---|---|---|---|
| **Mínimo** | 0.00 | 0% | **Mínimo** | 0.00 | 0% |
| **Primer cuartil** | 0.20 | 1.724% | **Primer cuartil** | 0.20 | 40% |
| **Mediana** | 0.20 | 1.724% | **Mediana** | 0.30 | 60% |
| **Media** | 0.4754 | 4.098% | **Media** | 0.2169 | 43.38% |
| **Tercer cuartil** | 0.30 | 2.58% | **Tercer cuartil** | 0.30 | 60% |
| **Máximo** | 11.60 | 100% | **Máximo** | 0.50 | 100% |

La Figura 64 ilustra una diferencia entre el segundo cuartil y el primer cuartil del 0%, este resultado es menor al 25% de las observaciones y desde el tercer cuartil a la mediana se tiene 0.856% menor al 25%. Esto quiere decir que la distribución de los datos es asimétrica por la derecha y tiene varios valores atípicos por arriba del tercer cuartil.

Por otro lado en la Figura 65 se tiene que la diferencia entre el segundo cuartil y el primer cuartil es de 0.1%, y la diferencia del tercer y segundo cuartil es 0%. Esto

quiere decir que las observaciones tienden hacia la izquierda, con varios valores atípicos por abajo del segundo cuartil.

A pesar del análisis anterior es necesario examinar los histogramas de densidad mediante el cálculo de la Medida Cuartílica. En la Figura 64 el histograma de densidad posee una medida cuartílica igual a 1 > 0, lo que significa que tiene una distribución asimétrica positiva, es decir los valores están concentrados en niveles superiores a la media aritmética. Por el contrario el histograma de la Figura 65, tiene una medida cuartílica igual a 1 < 0, esto significa que la distribución de frecuencia es asimétrica positiva, es decir los valores están concentrados en niveles inferiores a la media aritmética.

Para la evaluación de estos diagramas es necesario comparar las medianas (porcentaje) donde el 60% corresponde a la Latencia Recursiva-Cliente y el 1.724% a la Latencia Optimizada-Cliente. Por tanto se observa que la Latencia Optimizada es menor a la Latencia Recursiva (1.74%<60%), con una diferencia del 58.26%, es decir que la latencia con el algoritmo iterativo es menor en un 58.26% con respecto al algoritmo recursivo.

**Histograma de densidad: Servidor**

A continuación se presenta los resultados de la propuesta optimizada de manera recursiva e iterativa (Figuras 66 y 67). Estas ilustran un histograma de densidad con su respectivo diagrama de caja, estas estadísticas descriptivas proporcionan los datos de las medianas.

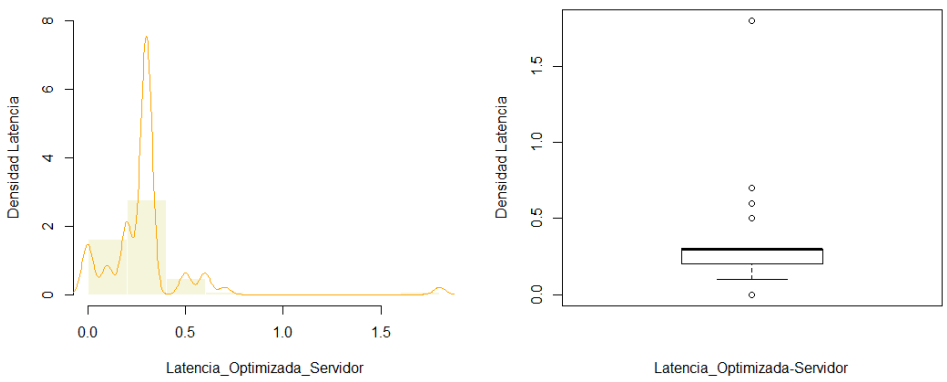


**Figura 66. Histograma de densidad y diagrama de caja de la Latencia Optimizada-Servidor**

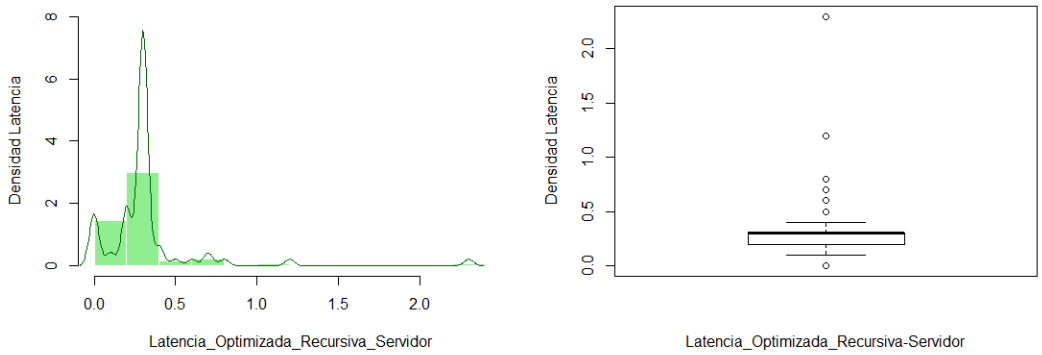


**Figura 67. Histograma de densidad y diagrama de caja de la Latencia Recursiva-Servidor**

Los histogramas de frecuencia son equivalentes a los diagramas de caja, tal como se muestra en las Figuras 66 y 67. La Tabla 42 muestra un resumen de los datos de los diagramas de caja.

**Tabla 42**

***Datos de los diagramas de caja-Latencia-Servidor***

| **Latencia Optimizado-Servidor** | | | **Latencia Recursiva-Servidor** | | |
|---|---|---|---|---|---|
| **Mínimo** | 0.00 | 0.0% | **Mínimo** | 0.00 | 0.0% |
| **Primer cuartil** | 0.20 | 11.1% | **Primer cuartil** | 0.20 | 8.69% |
| **Mediana** | 0.30 | 16.67% | **Mediana** | 0.30 | 13.047% |
| **Media** | 0.2923 | 16.23% | **Media** | 0.32 | 13.91% |
| **Tercer cuartil** | 0.300 | 16.67% | **Tercer cuartil** | 0.30 | 13.047% |
| **Máximo** | 1.80 | 100% | **Máximo** | 2.30 | 100% |

La Figura 66, ilustra una diferencia entre el segundo cuartil y el primer cuartil del 5.57%, este resultado es menor al 25% de las observaciones y desde el tercer cuartil a la mediana se tiene 0% menor al 25%. Esto quiere decir que la distribución de los datos es asimétrica por la izquierda, tiene varios de valores atípicos por arriba del tercer cuartil.

Por otro lado en la Figura 67 se tiene que la diferencia entre el segundo cuartil y el primer cuartil es de 4.357%, y la diferencia del tercer y segundo cuartil es 0%. Esto

quiere decir que las observaciones tienden hacia la izquierda, con valores atípicos por arriba del tercer cuartil.

Pese al análisis anterior es necesario examinar los histogramas de densidad mediante el cálculo de la Medida Cuartílica. En la Figura 66 el histograma de densidad posee una medida cuartílica igual a 1 < 0. Al igual que el histograma de la Figura 67, tiene una medida cuartílica igual a 1 < 0, esto significa que la distribución de frecuencia es asimétrica negativa, es decir los valores de ambas están concentrados en niveles inferiores a la media aritmética.

Para la evaluación de estos diagramas es necesario comparar las medianas (porcentaje) donde el 13.047% que corresponde a la Latencia Recursiva-Servidor y el 16.67% a la Latencia Optimizado-Servidor. Se observa que la Latencia Optimizado es mayor a la Latencia Recursiva (16.67% > 13.047%), con una diferencia del 3.623%. Eso quiere decir que el la latencia con el algoritmo RSA iterativo sigue siendo optima en un 96.377% con respecto al recursivo.

### 5.3.3.2. Reporte de la red

**Histograma de densidad: Cliente**

Los resultados de la propuesta optimizada de manera recursiva e iterativa (Figuras 68 y 69). Estas ilustran un histograma de densidad con su respectivo diagrama de caja, estas estadísticas descriptivas proporcionan los datos de las medianas.

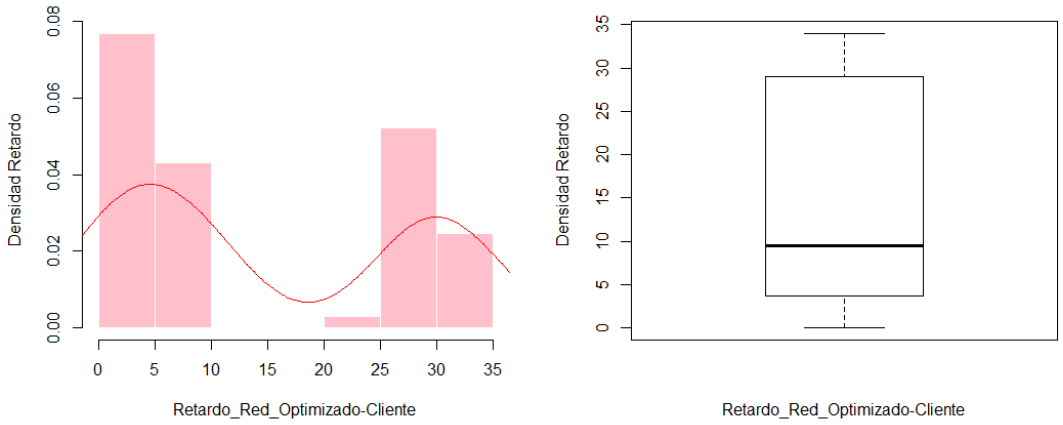


**Figura 68. Histograma de densidad y diagrama de caja Retardo Red Optimizado-Cliente**

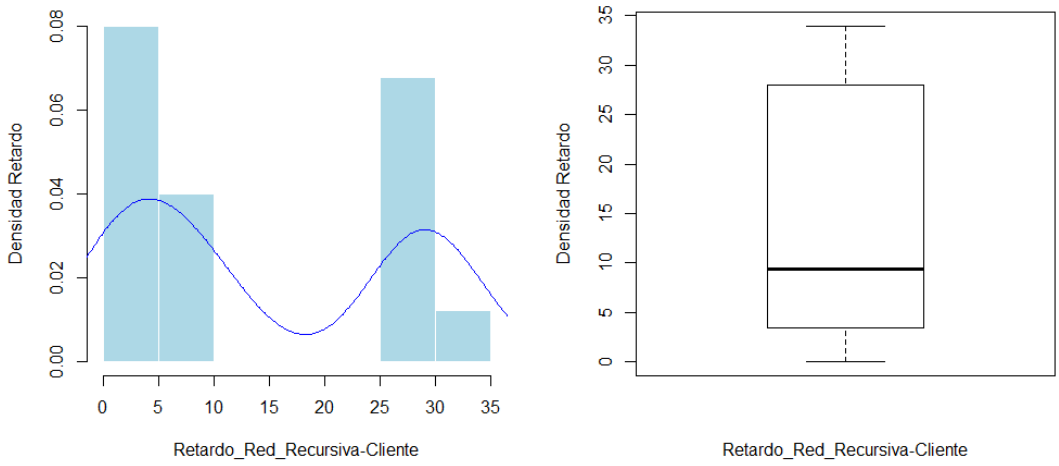


**Figura 69. Histograma de densidad y diagrama de caja Retardo Red Recursivo-Cliente**

Los histogramas de frecuencia son equivalentes a los diagramas de caja tal como se muestra en las Figuras 68 y 69. La Tabla 43 muestra un resumen de los datos de los diagramas de caja, generado mediante el software estadístico R.

**Tabla 43**

***Datos de los diagramas de caja-Retardo-Cliente***

| **Retardo Optimizado-Cliente** | | | **Retardo Recursivo-Cliente** | | |
|---|---|---|---|---|---|
| **Mínimo** | 0.00 | 0% | **Mínimo** | 0.00 | 0% |
| **Primer cuartil** | 3.73 | 10.97% | **Primer cuartil** | 3.47 | 10.20% |
| **Mediana** | 9.42 | 27.70% | **Mediana** | 9.42 | 27.70% |
| **Media** | 14.92 | 43.88% | **Media** | 14.51 | 42.67% |
| **Tercer cuartil** | 29.00 | 85.29% | **Tercer cuartil** | 28.00 | 82.35% |
| **Máximo** | 34.00 | 100% | **Máximo** | 34.00 | 100% |

La Figura 68, ilustra una diferencia entre el segundo cuartil y el primer cuartil del 16.73%, este resultado es menor al 25% de las observaciones y desde el tercer cuartil a la mediana se tiene 57.59% mayor al 25%. Esto quiere decir que la distribución de los datos es asimétrica por la derecha, tiene mayor cantidad de valores atípicos por arriba del tercer cuartil.

Por otro lado en la Figura 69 se tiene que la diferencia entre el segundo cuartil y el primer cuartil es de 17.5%, y la diferencia del tercer y segundo cuartil es 54.65%. Esto quiere decir que la distribución de los datos es asimétrica hacia la derecha, con una gran cantidad de valores atípicos por arriba del tercer cuartil.

A pesar del análisis anterior es necesario examinar los histogramas de densidad mediante el cálculo de la Medida Cuartílica. En la Figura 68 el histograma de densidad muestra una curva de la distribución de frecuencia de forma asimétrica, con una medida cuartílica igual a 0.54 > 0. Al igual que el histograma de la Figura 69, tiene una medida cuartílica igual a 0.67 > 0, esto significa que la distribución de frecuencia es asimétrica positiva, es decir los valores de ambas están concentrados en niveles superiores a la media aritmética.

Para la evaluación de estos diagramas es necesario comparar las medianas (porcentaje) donde el 27.70% corresponde a la Retardo Recursivo-Cliente y la misma cantidad a la Retardo Optimizada-Cliente. Por tanto se demuestra que las gráficas brindan valores prácticamente iguales, es decir que el retardo no sufre ningún cambio con el algoritmo recursivo.

**Histograma de densidad: Servidor**

Finalmente se presenta los resultados de la propuesta optimizada de manera recursiva e iterativa (Figuras 70 y 71), a través de un histograma de densidad con su respectivo diagrama de caja que proporcionan los datos de las medianas.

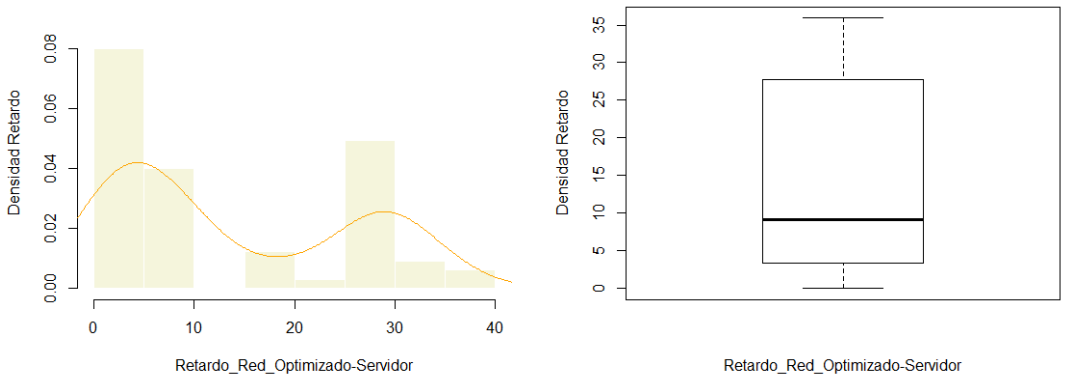


**Figura 70. Histograma de densidad y diagrama de caja Retardo Red Optimizado-Servidor**

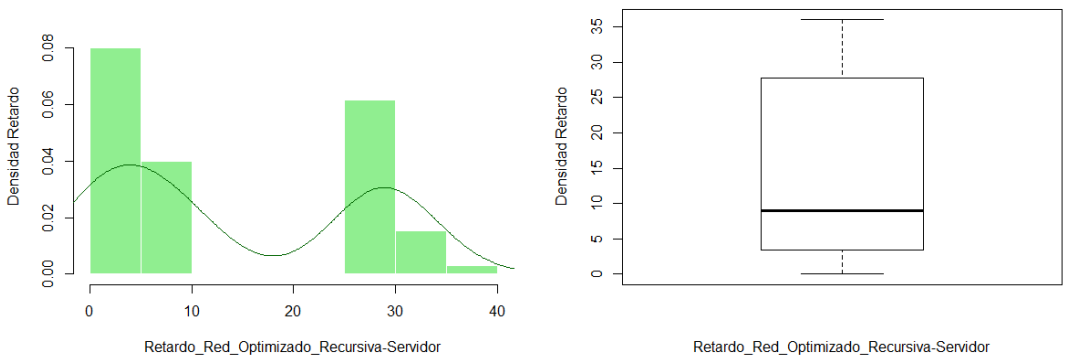


**Figura 71. Histograma de densidad y diagrama de caja Retardo Red Recursivo-Servidor**

Los histogramas de frecuencia son equivalentes a los diagramas de caja, tal como se muestra en las Figuras 70 y 71. La Tabla 44 muestra un resumen de los datos de los diagramas de caja, generado mediante el software estadístico R.

**Tabla 44**

***Datos de los diagramas de caja-Retardo-Servidor***

| **Retardo Optimizado-Servidor** | | | **Retardo Recursivo-Servidor** | | |
|---|---|---|---|---|---|
| **Mínimo** | 0.0 | 0.0% | **Mínimo** | 0.00 | 0.0% |
| **Primer cuartil** | 3.44 | 9.57% | **Primer cuartil** | 3.44 | 9.55% |
| **Mediana** | 9.13 | 25.42% | **Mediana** | 9.05 | 25.13% |
| **Media** | 13.94 | 38.82% | **Media** | 14.31 | 39.75% |
| **Tercer cuartil** | 27.72 | 77.19% | **Tercer cuartil** | 27.72 | 77.00% |
| **Máximo** | 35.91 | 100% | **Máximo** | 36.00 | 100% |

La Figura 70, ilustra una diferencia entre el segundo cuartil y el primer cuartil del 15.85%, este resultado es menor al 25% de las observaciones y desde el tercer cuartil a la mediana se tiene 51.77% mayor al 25%, esto quiere decir que la distribución de los tiempos es asimétrica por la derecha y tiene una gran cantidad de valores atípicos por arriba del segundo cuartil.

Por otro lado en la Figura 71 se tiene que la diferencia entre el segundo cuartil y el primer cuartil es de 15.58%, y la diferencia del tercer y segundo cuartil es 51.87%.

Esto quiere decir que las observaciones tienden hacia la derecha, con valores atípicos por arriba del tercer cuartil.

Pese al análisis anterior es necesario examinar los histogramas de densidad mediante el cálculo de la Medida Cuartílica. En la Figura 70 el histograma de densidad posee una medida cuartílica igual a 0.53 > 0. Al igual que el histograma de la Figura 71, tiene una medida cuartílica igual a 0.53 > 0. Esto significa que la distribución de frecuencia es asimétrica negativa, es decir los valores de ambas están concentrados en niveles inferiores a la media aritmética.

Para la evaluación de estos diagramas es necesario comparar las medianas (porcentaje) donde el 25.13% corresponde al Retardo Recursivo-Servidor y el 25.42% al Retardo Optimizado-Servidor. Se observa que el Retardo Optimizado es mayor al Retardo Recursivo (25.42% > 25.13%), con una diferencia del 0.29%. Eso quiere decir que el retardo con el algoritmo iterativo es óptimo en un 99.71% con respecto al recursivo.

## 5.4. Discusión

En el presente proyecto se demuestra que el algoritmo de encriptación RSA optimizado es capaz de cifrar y descifrar la información disminuyendo costos computacionales. Además cabe destacar que este algoritmo responde favorablemente a las tres técnicas que prueban el nivel de seguridad como son: la confidencialidad, disponibilidad e integridad.

- La confidencialidad, se logró incrementar la seguridad de los parámetros que son primordiales para la generación de claves, mediante la base de datos que es gestionada por un usuario y una clave;
- Disponibilidad: Accesibilidad que tiene el usuario al enviar al destino la información que requiere mayor seguridad;
- Integridad: Se generó una librería .jar, a la que solo tiene acceso el autor de este nuevo algoritmo RSA;

Para lograr los resultados obtenidos fue necesario implementar diferentes modelos matemáticos, algoritmias y herramientas. El proyecto base tiene características elementales como es el uso de números primos grandes $p$ y $q$ cada uno con 300 cifras,

encontrados en Internet. Sin embargo estos números limitan al mensaje ya que su longitud depende de ellos. Por otro lado el proyecto propuesto se optimizó en cuanto a la longitud del mensaje ya que esta no tiene ninguna limitación de software, sino más bien a la velocidad de transmisión y a la del procesador. De esta manera garantiza que la restricción no se deba al algoritmo sino a la infraestructura.

# CAPÍTULO 6

# CONCLUSIONES Y RECOMENDACIONES

## 6.1. Conclusiones

- Este proyecto se enfocó en optimizar el algoritmo de encriptación RSA logrando una solución genérica, que disminuye el consumo de los recursos y costes de software, además de incrementar el nivel de seguridad de los mensajes transmitidos por la red.
- Se evaluaron los diferentes sistemas criptográficos determinando la velocidad de cálculo, nivel de seguridad y los escenarios en los que se presentan. Esto permitió llevar a cabo el análisis y optimización de la expresión matemática RSA combinando cálculo modular y probabilístico, con lo cual se mejoró el tiempo de cifrado y descifrado. Al mejorar este algoritmo se garantizó la confidencialidad, disponibilidad e integridad de la información.
- Se desarrolló e implementó un algoritmo experimental que permite optimizar el RSA en base a la expresión matemática mejorada que compensa la necesidad de asegurar la transmisión de mensajes sobre la red. La optimización se realizó a través de un proceso de Ingeniería de Software mediante la metodología ágil AUP. AUP permitió desarrollar este proyecto de manera organizada y sistemática. Para esto se utilizó las herramientas Netbeans y MySQL, que son herramientas de libre distribución. Netbeans permitió realizar una API con Java, amigable para el usuario. En esta API se utilizó el algoritmo asimétrico optimizado RSA. Por otro lado para gestionar de mejor forma los parámetros para el cálculo del RSA se empleó una base de datos en MySQL con la ayuda de las técnicas de programación: sockets y RPC, con esto brinda una mayor confidencialidad e integridad de la información. No obstante se puede recalcar que el cifrado y descifrado de la información tiene ciertas limitaciones pero no por las herramientas utilizadas sino más bien por la limitación en la velocidad de la transmisión y la del procesador.
- Se realizaron varias pruebas funcionales, para verificar y validar el comportamiento del algoritmo. Luego se procedió a la tabulación, procesamiento estadístico, inferencia, análisis e interpretación de datos tanto en el cliente como el servidor durante 13 escenarios diferentes, descritos en el

Capítulo 5. De estos análisis se tomó los datos máximos para las siguientes interpretaciones: (i) Tiempo del cliente es 33 veces más rápido que el de línea base, mientras que el tiempo del servidor es 1020 veces más rápido que el algoritmo básico, con esto alcanza una mejora del 89.5% y 99% de optimización; (ii) memoria del cliente disminuye su consumo a 1.7 veces, por otra parte la memoria del servidor la disminución es mínima con respecto a la memoria base; (iii) la variable procesador utiliza el análisis de varianza, el cual demuestra que el estadístico de prueba tanto en el cliente como en el servidor es mayor al nivel de significación, con esto se confirma que el consumo del procesador ha sido optimizado; (iv) latencia del cliente es 1.20 veces menor a la línea base, de igual forma en la latencia del servidor disminuye a 10.38 veces. Por tanto se llega a la conclusión que las variables evaluadas se optimizaron desde un 85% hasta el 99% con respecto al algoritmo de la línea base, como se demuestra en las Figuras 36-51.

- Adicionalmente se realizó otra prueba, en la que el objetivo era el uso de recursos ante el método RSA iterativo vs el recursivo, en las cuales se hizo la evaluación de 13 variables distintas (Ver Tablas 47-50 del Anexo C, D, E, F); obteniéndose como respuesta en todas las variables evaluadas que el algoritmo recursivo consumía mayor número de recursos computacionales. Los resultados de cada variable evaluada de acuerdo a los datos máximos de cada uno son: (i) el tiempo del cliente con el uso del algoritmo optimizado iterativo es 1.8 veces más rápido frente al recursivo, por otro lado en el servidor el tiempo iterativo es 2.08 veces más rápido que el algoritmo recursivo; (ii) Memoria del cliente con el algoritmo iterativo disminuye 1.73 veces frente al recursivo, por el contrario la memoria del servidor su disminución es mínima pero de todas formas en ambos casos se optimiza el consumo de la memoria; (iii) como se mencionó en el caso anterior para el procesador se usó el análisis de varianza, tanto en el cliente como en el servidor el estadístico de prueba es mayor al nivel de significación, comprobando que el procesador es óptimo en los dos casos; (iv) mediante el estudio de las medianas se obtuvo que en la latencia del cliente con el algoritmo iterativo es menor en un 58.26% ante al recursivo, y en el cliente la latencia iterativa disminuye en un 96.37% ante la

recursiva; y (v) el Reporte de la red tanto en el cliente como el servidor no sufren un cambio significativo. Por tanto la memoria, procesador, latencia, reporte de la red, tiempo cifrado y descifrado se optimizan del 2% al 40% más frente al algoritmo RSA iterativo.

- Se incrementa el nivel de seguridad ya que se realiza periódicamente un mezclado del código ASCII mediante un programa en código Java, asegurando así que un intruso tenga el patrón del arreglo de índices (código).

## 6.2. Recomendaciones

Es recomendable cambiar la codificación de MySQL, que vienen por defecto en Linux, es decir de UTF-8 a Latin, ya que al no realizar esta acción no encuentra la posición exacta del carácter ingresado y por tanto afecta en el cálculo de cifrar y descifrar, dando como resultado en el servidor un mensaje distinto al original.

Respecto al algoritmo propuesto en este estudio es necesario que los números aleatorios que son parte fundamental en el cálculo del RSA, como son *p, q,* y *n,* sean primos caso contrario en la generación de claves será errónea y por tanto al descifrar el mensaje no será el mismo que envió el cliente.

Es indispensable indicar que para la transmisión de datos se debe sincronizar tanto el cliente como el servidor, y así lograr el flujo de datos sin que exista la pérdida de información.

Debido a los cálculos probabilísticos que se utilizan para obtener los parámetros n y el arreglo de índices que corresponde al código, se recomienda que *n* emplee un número con un rango de 4 a 6 cifras, para que el intruso tenga mayor dificultad al momento de realizar cálculos modulares inversos. Por otro lado en el arreglo de índices se puede ingresar de 1 a 3 filas más, para asegurar que el mensaje tenga mayor complejidad de entendimiento.

## 6.3. Trabajo Futuro

Para reforzar este proyecto se planea generar una librería en lenguaje C++, creando un archivo de enlace dinámico que permita interactuar en Windows y/o un archivo .so en Linux. A través de esto se respondería favorablemente a uno de los retos

de seguridad que es la integridad, ya que solamente el autor podría manipular y modificar la librería de enlace dinámico.

Además se puede incorporar este algoritmo en equipos con mejores características tanto de hardware como de software, con los que se podría encriptar no solo a caracteres sino más bien ya documentos o archivos que requieran de una alta seguridad.

Finalmente para ampliar el manejo de este algoritmo se proyecta subirlo a la nube, así se ofrecería más recursos al usuario, es decir no solamente el uso de PCS sino también de teléfonos móviles para la transmisión segura de datos, ya que la seguridad que provee la nube es la mejor solución para este servicio de encriptación.

# BIBLIOGRAFÍA


Aldegani, G. (1997). *Seguridad Informática.* Buenos Aires: MP.

Alulema, D. (2014). *Modelos de Comunicaciones.* Quito: Tecnologías de Software para Electrónica.

Ambler, S. (13 de Mayo de 2006). *The Agile Unified Process (AUP)*. Obtenido de Ambysoft: http://www.ambysoft.com/unifiedprocess/agileUP.html

Andrew, D., Birrell, & Bruce, J. (1984). Implemening Remote Procedure Calls. *Transactions on Computer System*, 39-59.

Apostol, T. (1984). *Introducción a la teoría analítica de números.* Sevilla: Reverté,S.A.

Arteaga, J., & Santillán, I. (31 de Enero de 2013). *Encriptación de información, protocolo segurid y sistemas computacionales.* Obtenido de Repositorio Digital Universidad Técnica el Norte: http://repositorio.utn.edu.ec/bitstream/123456789/1088/3/04%20ISC%20042-Capitulo%20II.pdf

Astilla, M. B. (2009). *Diseño de un Esquema de Comunicación segura utilizando como criptosistema simétrico a triple DES y asimétrico a RSA.* (Tesis inédita de maestría), Instituto Politécnico Nacional "Centro de Innovación y Desarrollo Tecnológico en Cómputo", México.

B.V, B. (Octubre de 1998). *MTR*. Obtenido de The Linux Experts: https://www.bitwizard.nl/mtr/

Belingueres, G. (2000). Introducción A los criptosistemas de Curva Elíptica. *Red Mundial 5*, 1-30.

Böck, H. (2011). *The Definitive Guide to Netbeans™ Platform 7.* Apress.

Boneh, D. (1999). Twenty Years of attacks on the RSA cryptosystem. *Notices of the AMS*, 203-213.

Bosch, P. (2006). Las matemáticas e la criptología. *Encuentros multidisciplinares, 8*(23), 56-66.

Calvert, K. D. (2011). *TCP/IP sockets in Java: practical guide for programmers.* USA,San Diego: Academic Press.

Casillas, L., Gibert, M., & Pérez, O. (s.f.). *Bases de datos en MySQL.* Obtenido de Bases de datos en MySQL: http://ocw.uoc.edu/computer-science-technology-and-multimedia/bases-de-datos/bases-de-datos/P06_M2109_02151.pdf

Española, R. A. (2012). *Diccionario Real Academia de Lengua Española.* Madrid: Edición 22º.

Fúster, A., Guía, D., Hernández, L., Montoya, F., & Muñoz, J. (2001). *Técnicas criptográficas de protección de datos.* Enero: Alfaomega.

Gil, P. C. (2000). Algunos hitos de la criptografía del siglo XX. *Números*, 405-408.

Googlux. (s.f.). *SAR(system activity reporter).* Obtenido de Googlux: http://www.googlux.com/sar.html

Gupta, S., & Sharma, J. (2012). A hybrid encryption algorithm based on RSA and Diffie-Hellman. *Computational Intelligence and Computing Research*, 1-4.

Identity Theft Resource Center. (31 de Diciembre de 2014). *ITRC Corporate Sponsors.* Obtenido de Data Bresch Reports: http://www.idtheftcenter.org/images/breach/DataBreachReports_2014.pdf

IEEE. (1993). Ingeniería de Software Estandar IEEE 830. *IEEE Recommended Practice for Software Requirements Specificactions.*

Ivy, B., & Kumar, P. (2012). A modified RSA cryptosystem based on "n" prime numbers. *International Journal of Engineering and computer Science ISSN*, 2319-7242.

Johnsonbaugh, R. (2005). *Matemáticas Discretas.* Mexico: Person Education.

Kalishi, B., & Yiqum, L. (1998). *On the Security of the RC5 Encryption Algorithm.* RSA Laboratories Technical Report TR-602.

Kisbye, N. (2005). Sistemas de claves públicas y privadas. *Revista de Educación Matemática,* 20-35.

Latham, D. (15 de Agosto de 1983). *Department of defense trusted computer system evaluation criteria.* Obtenido de Department of Defense Standard: http://csrc.nist.gov/publications/history/dod85.pdf

Liu, Q., Li, T., & Hao, L. (Marzo de 2011). The research of the bath RSA decryption performance. *Journal of Computational Information Systems, 7*(3), 948-955.

López, M. J. (2003). *Criptografía y Seguridad en Computadores* (Tercera ed.). España, Jaén. Obtenido de http://www.uned.es/413042/material/Criptografia.pdf

López, P. (24 de Febrero de 2014). *RSA permite que una nueva genración de servicio de seguridad administrados ayude a las empresas a mejorar la detección de amenazas avanzadas y la respuesta a dichas amenazas.* Obtenido de EMC2: http://mexico.emc.com/about/news/press/20140224-01.htm

Meza, M. (3 de septiembre de 2010). *Arquitectura cliente servidor de tres capas*. Obtenido de Base de datos: https://miriammeza.wordpress.com/2010/09/03/arquitectura-cliente-servidor-de-tres-capas/

Minni, R., Sultania, K., Mishra, S., & Vincent, D. (Julio de 2013). An algorithm to enhance security in RSA. *Computing,Communications and Networking Technologies (ICCCNT)*, 1-4.

Mora, W. (2014). *Introducción a la Teoría de Números.* Cartago: Tecdigital.

Murali, G., Anusha, K., Shirisha, A., & Sravya, S. (2011). Remote Procedure calls implementing using distributed algorithm. 1742-1746.

Nagar, S., & Alshamma, S. (21-24 de Marzo de 2012). High speed implementation of RSA algorithm with modified keys exchange. *Sciences of Electronics, Technologies of Information and Telecommunications (SETIT),* 639-642.

Nájera, L. (2013). El poder de los cifrados. *Revista Tech. Mexico,* Conferencia.

Nixon, R. (2014). *Learning PHP,MySQL,JavaScript,CSS and HTML 5.* O'Really Media,Inc.

Northrup, T. (2013). *Windows 8.1 inside out.* Microsoft press.

Oracle. (2012). *NetBeans*. Obtenido de Oracle Corporation: http://plugins.netbeans.org/plugin/52568/netbeanspdfviewer

Oracle. (2015). *MySQL*. Obtenido de MySQL: The world's most popular open source database: https://www.mysql.com/

Patidar, R., & Bhartiya, R. (Diciembre de 2013). Modified RSA cryptosystem based on offline storage and prime number. *IEEE International Conference on Computational Intelligence and Conputing Research*.

Penarrieta, R. (2011). *Java y Netbeans.* Obtenido de Programación Java-Netbeans v.7.0: http://javaagricola.wikispaces.com/file/view/0_Java+y+NetBeans.pdf

Pressman, R. (2005). *Ingeniería del Software: Un enfoque práctico*. España: McGrawHill.

Ramió, J. (2006). *Libro Electrónico de Seguridad Informática y Criptografía.* Madrid: JRA.

Ramirez, E. (1 de Mayo de 2011). *Manual Básico de Ubuntu.* Obtenido de Introducción al software libre: uso básico de sistema operativo y programas de ofimática: http://solcr.org/sites/default/files/manual%20ubuntu.pdf

Rivest, R. (1995). The RC% encryption algorithm. *Fast Software Encryption. Springer Berlin Heidelberg*, 86-96.

Sanjuan, L. (2012). Criptografía I. *Seguridad en desarrollo del Software*, (pág. 34). Universidad del Norte.

Shannon, C. (1949). Communication Theory of secrecy systems. *Bellsystem technical journal*, 656-715.

Simpson, C. (2000). Internet Relay Chat. *Teacher Librarian*, 18.

Simson, G., & Spafford, G. (2002). *Web Security,Privacy and Commerce.* United States of America: O´Reilly and Associates.

Sommerville. (2005). *Ingeniería del Software.* Madrid: Pearson Addison Wesley.

Stallings, W. (2011). *Cryptography and Network Security Principles and Practice.* United States of America: Pearson Education, Inc.

Sybase. (Noviembre de 2011). *Power Designer 16.1.* Obtenido de Sybase an SAP Company: http://infocenter.sybase.com/help/topic/com.sybase.infocenter.dc38094.1610/doc/pdf/quick_reference.pdf

Tanenbaum, A. (2003). *Redes de Computadoras.* México: Pearson Prentice Hall.

Villalón, A. (Julio de 2002). *Seguridad en Unix y Redes.* Obtenido de Seguridad en Unix y Redes: https://www.rediris.es/cert/doc/unixsec/unixsec.pdf

Wang, H. (2013). Key generation research of RSA public cryptosystem and Mat lab implement. *Sensor Network Security Tchnology and Privacy Communication System (SNS and PCS),* 125-129.

Wang, L., & Zhang, Y. (2011). A new personal information protection approach based on RSA cryptography. *IT in Medicine and Education (ITME), 2011 International Symposium*, 591-593.

Yaday, P., Pankaj, S., & Yaday, K. (2012). Implementation of RSA algorithm using Elliptic curve algorithm for security and performance enhancement. *International Journal of Scientific Technology Research 1.4*, 102-105.

Zulfikar, R. (06 de Octubre de 2006). *A common RSA implementation mistake explained.* Obtenido de Symantec: http://www.symantec.com/connect/blogs/common-rsa-implementation-mistake-explained

# ANEXOS

# ANEXOS A

## Cliente: Algoritmo RSA línea base

### Tabla 45

***Variables evaluadas con el algoritmo RSA línea base: Cliente***

| NÚMERO ITERACIONES | NÚMERO DE LETRAS | TIEMPO DE CIFRADO | CONSUMO DE MEMORIA | | CONSUMO DEL PROCESADOR | | LATENCIA | | REPORTE ESTADÍSTICO DE LA RED | |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 17 | kbmemfree | 456204 | %user | 14,78 | Last | 0,3 | rxpck/s | 4,00 |
| | | | kbmemused | 1579152 | %nice | 0,00 | avg | 0,3 | txpck/s | 2,00 |
| | | | %memused | 77,59 | %system | 0,49 | best | 0,3 | rxkB/s | 0,35 |
| | | | kbbuffers | 746832 | %iowait | 0,00 | wrst | 0,3 | txkB/s | 0,12 |
| | | | kbcached | 326204 | %steal | 0,00 | StDev | 0,0 | rxmcst/s | 0,00 |
| | | | | | %idle | 84,73 | | | | |
| 1 | 2 | 17 | kbmemfree | 452064 | %user | 17,56 | Last | 0,3 | rxpck/s | 9,00 |
| | | | kbmemused | 1583292 | %nice | 0,00 | avg | 0,9 | txpck/s | 9,00 |
| | | | %memused | 77,79 | %system | 0,49 | best | 0,3 | rxkB/s | 0,60 |
| | | | kbbuffers | 747740 | %iowait | 0,00 | wrst | 6,2 | txkB/s | 0,74 |
| | | | kbcached | 327176 | %steal | 0,00 | StDev | 1,9 | rxmcst/s | 0,00 |
| | | | | | %idle | 81,95 | | | | |
| 1 | 3 | 17 | kbmemfree | 451568 | %user | 16,34 | Last | 0,3 | rxpck/s | 9,09 |
| | | | kbmemused | 1583788 | %nice | 0,00 | avg | 0,6 | txpck/s | 10,10 |
| | | | %memused | 77,81 | %system | 1,49 | best | 0,4 | rxkB/s | 0,59 |
| | | | kbbuffers | 747232 | %iowait | 0,00 | wrst | 1,8 | txkB/s | 0,86 |
| | | | kbcached | 327480 | %steal | 0,00 | StDev | 5,1 | rxmcst/s | 2,02 |
| | | | | | %idle | 82,18 | | | | |

**CONTINUA** →

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 4 | 18 | kbmemfree | 451860 | %user | 13,43 | Last | 0,3 | rxpck/s | 9,00 |
| | | | kbmemused | 1583496 | %nice | 0,00 | avg | 1,4 | txpck/s | 10,00 |
| | | | %memused | 77,8 | %system | 1,00 | best | 0,2 | rxkB/s | 0,60 |
| | | | kbbuffers | 746828 | %iowait | 0,00 | wrst | 11,7 | txkB/s | 0,96 |
| | | | kbcached | 327988 | %steal | 0,00 | StDev | 3,6 | rxmcst/s | 0,00 |
| | | | | | %idle | 85,57 | | | | |
| 1 | 5 | 19 | kbmemfree | 443496 | %user | 32,50 | Last | 0,3 | rxpck/s | 13,86 |
| | | | kbmemused | 1591860 | %nice | 0,00 | avg | 1,2 | txpck/s | 11,88 |
| | | | %memused | 78,21 | %system | 1,00 | best | 0,2 | rxkB/s | 0,92 |
| | | | kbbuffers | 747840 | %iowait | 0,00 | wrst | 9,5 | txkB/s | 1,18 |
| | | | kbcached | 330640 | %steal | 0,00 | StDev | 2,9 | rxmcst/s | 1,98 |
| | | | | | %idle | 66,50 | | | | |
| 1 | 6 | 20 | kbmemfree | 443808 | %user | 39,30 | Last | 0,3 | rxpck/s | 7,00 |
| | | | kbmemused | 1591548 | %nice | 0,00 | avg | 0,9 | txpck/s | 0,00 |
| | | | %memused | 78,2 | %system | 1,00 | best | 0,2 | rxkB/s | 6,81 |
| | | | kbbuffers | 747224 | %iowait | 0,00 | wrst | 7,2 | txkB/s | 0,00 |
| | | | kbcached | 331012 | %steal | 0,00 | StDev | 2,2 | rxmcst/s | 11,00 |
| | | | | | %idle | 59,70 | | | | |
| 1 | 7 | 33 | kbmemfree | 442808 | %user | 44,78 | Last | 0,3 | rxpck/s | 8,00 |
| | | | kbmemused | 1592548 | %nice | 0,00 | avg | 1,3 | txpck/s | 10,00 |
| | | | %memused | 78,24 | %system | 3,48 | best | 0,2 | rxkB/s | 0,52 |
| | | | kbbuffers | 746496 | %iowait | 0,00 | wrst | 10,9 | txkB/s | 1,17 |
| | | | kbcached | 331440 | %steal | 0,00 | StDev | 3,4 | rxmcst/s | 0,00 |
| | | | | | %idle | 51,74 | | | | |
| 8 | 8 | 130 | kbmemfree | 442167 | %user | 36,14 | Last | 0,3 | rxpck/s | 9,10 |
| | | | kbmemused | 1593746 | %nice | 0,00 | avg | 1,5 | txpck/s | 9,87 |
| | | | %memused | 78,27 | %system | 1,49 | best | 0,2 | rxkB/s | 0,61 |
| | | | kbbuffers | 747124 | %iowait | 0,00 | wrst | 11,8 | txkB/s | 1,21 |
| | | | kbcached | 331958 | %steal | 0,00 | StDev | 2,1 | rxmcst/s | 0,99 |
| | | | | | %idle | 62,38 | | | | |

**CONTINUA** ⟶

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 9 | 150 | kbmemfree | 441196 | %user | 37,07 | Last | 0,3 | rxpck/s | 10,00 |
| | | | kbmemused | 1594160 | %nice | 0,00 | avg | 1,8 | txpck/s | 10,00 |
| | | | %memused | 78,32 | %system | 3,90 | best | 0,2 | rxkB/s | 0,66 |
| | | | kbbuffers | 747712 | %iowait | 0,00 | wrst | 15,0 | txkB/s | 1,25 |
| | | | kbcached | 332040 | %steal | 0,00 | StDev | 4,6 | rxmcst/s | 2,00 |
| | | | | | %idle | 59,02 | | | | |
| 1 | 10 | 192 | kbmemfree | 441484 | %user | 28,71 | Last | 0,3 | rxpck/s | 10,00 |
| | | | kbmemused | 1593872 | %nice | 0,00 | avg | 0,3 | txpck/s | 8,00 |
| | | | %memused | 78,31 | %system | 2,48 | best | 0,3 | rxkB/s | 0,88 |
| | | | kbbuffers | 747496 | %iowait | 0,00 | wrst | 0,3 | txkB/s | 1,11 |
| | | | kbcached | 332524 | %steal | 0,00 | StDev | 0,0 | rxmcst/s | 0,00 |
| | | | | | %idle | 68,81 | | | | |
| 1 | 50 | 253 | kbmemfree | 426792 | %user | 31,68 | Last | 0,3 | rxpck/s | 9,00 |
| | | | kbmemused | 1608564 | %nice | 0,00 | avg | 0,3 | txpck/s | 10,00 |
| | | | %memused | 79,03 | %system | 3,47 | best | 0,2 | rxkB/s | 0,60 |
| | | | kbbuffers | 746616 | %iowait | 0,00 | wrst | 0,3 | txkB/s | 1,25 |
| | | | kbcached | 332488 | %steal | 0,00 | StDev | 0,0 | rxmcst/s | 0,00 |
| | | | | | %idle | 64,85 | | | | |
| 1 | 100 | 315 | kbmemfree | 426648 | %user | 30,20 | Last | 0,3 | rxpck/s | 13,00 |
| | | | kbmemused | 1608708 | %nice | 0,00 | avg | 1,5 | txpck/s | 12,00 |
| | | | %memused | 79,04 | %system | 1,49 | best | 0,2 | rxkB/s | 0,97 |
| | | | kbbuffers | 747250 | %iowait | 0,00 | wrst | 13,0 | txkB/s | 1,38 |
| | | | kbcached | 332084 | %steal | 0,00 | StDev | 4,0 | rxmcst/s | 2,00 |
| | | | | | %idle | 68,32 | | | | |
| 1 | 200 | 369 | kbmemfree | 1198364 | %user | 28,86 | Last | 0,2 | rxpck/s | 7,00 |
| | | | kbmemused | 836992 | %nice | 0,00 | avg | 0,3 | txpck/s | 8,00 |
| | | | %memused | 41,12 | %system | 1,00 | best | 0,2 | rxkB/s | 0,45 |
| | | | kbbuffers | 23760 | %iowait | 0,00 | wrst | 0,4 | txkB/s | 1,11 |
| | | | kbcached | 325004 | %steal | 0,00 | StDev | 0,1 | rxmcst/s | 2,00 |
| | | | | | %idle | 70,15 | | | | |

**ANEXOS B**

**Servidor: Algoritmo RSA línea base**

**Tabla 46**

***Variables evaluadas con el algoritmo RSA línea base: Servidor***

| NÚMERO ITERACIONES | NÚMERO DE LETRAS | TIEMPO DE DESCIFRADO | CONSUMO DE MEMORIA | | CONSUMO DEL PROCESADOR | | LATENCIA | | REPORTE ESTADÍSTICO DE LA RED | |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 33047 | kbmemfree | 9624 | %user | 55,94 | Last | 0,3 | rxpck/s | 9,00 |
| | | | kbmemused | 934808 | %nice | 0,00 | avg | 0,3 | txpck/s | 3,00 |
| | | | %memused | 98,98 | %system | 1,98 | best | 0,3 | rxkB/s | 2,28 |
| | | | kbbuffers | 5436 | %iowait | 3,47 | wrst | 0,4 | txkB/s | 0,23 |
| | | | kbcached | 74304 | %steal | 0,00 | StDev | 0,0 | rxmcst/s | 6,00 |
| | | | | | %idle | 38,61 | | | | |
| 1 | 2 | 32859 | kbmemfree | 9124 | %user | 51,72 | Last | 0,3 | rxpck/s | 2,97 |
| | | | kbmemused | 935308 | %nice | 0,00 | avg | 0,7 | txpck/s | 0,00 |
| | | | %memused | 99,03 | %system | 0,00 | best | 0,3 | rxkB/s | 0,15 |
| | | | kbbuffers | 6252 | %iowait | 0,99 | wrst | 1,9 | txkB/s | 0,00 |
| | | | kbcached | 76020 | %steal | 0,00 | StDev | 2,5 | rxmcst/s | 1,98 |
| | | | | | %idle | 47,29 | | | | |
| 1 | 3 | 33348 | kbmemfree | 14720 | %user | 71,21 | Last | 0,4 | rxpck/s | 3,00 |
| | | | kbmemused | 929712 | %nice | 0,00 | avg | 0,4 | txpck/s | 2,00 |
| | | | %memused | 98,44 | %system | 5,05 | best | 0,4 | rxkB/s | 0,18 |
| | | | kbbuffers | 7504 | %iowait | 0,00 | wrst | 0,5 | txkB/s | 0,00 |
| | | | kbcached | 72984 | %steal | 0,00 | StDev | 0,0 | rxmcst/s | 0,00 |

**CONTINUA** ⟶

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 4 | 33277 | | | %idle | 23,74 | | | |
| | | | kbmemfree | 16988 | %user | 70,79 | Last | 0,6 | rxpck/s | 3,96 |
| | | | kbmemused | 927444 | %nice | 0,00 | avg | 0,3 | txpck/s | 1,98 |
| | | | %memused | 98,2 | %system | 4,95 | best | 0,3 | rxkB/s | 0,36 |
| | | | kbbuffers | 7924 | %iowait | 0,00 | wrst | 0,6 | txkB/s | 0,18 |
| | | | kbcached | 73692 | %steal | 0,00 | StDev | 0,1 | rxmcst/s | 2,97 |
| 1 | 5 | 33408 | | | %idle | 24,26 | | | | |
| | | | kbmemfree | 10060 | %user | 51,74 | Last | 0,4 | rxpck/s | 2,97 |
| | | | kbmemused | 934372 | %nice | 0,00 | avg | 0,4 | txpck/s | 1,98 |
| | | | %memused | 98,93 | %system | 0,50 | best | 0,4 | rxkB/s | 0,22 |
| | | | kbbuffers | 10812 | %iowait | 0,00 | wrst | 0,5 | txkB/s | 0,18 |
| | | | kbcached | 72452 | %steal | 0,00 | StDev | 0,0 | rxmcst/s | 0,00 |
| 1 | 6 | 34691 | | | %idle | 47,76 | | | | |
| | | | kbmemfree | 10420 | %user | 51,00 | Last | 0,4 | rxpck/s | 14,00 |
| | | | kbmemused | 934012 | %nice | 0,00 | avg | 0,5 | txpck/s | 3,00 |
| | | | %memused | 98,9 | %system | 0,50 | best | 0,4 | rxkB/s | 4,32 |
| | | | kbbuffers | 2576 | %iowait | 0,00 | wrst | 0,8 | txkB/s | 0,27 |
| | | | kbcached | 68204 | %steal | 0,00 | StDev | 0,2 | rxmcst/s | 13,00 |
| 1 | 7 | 32756 | | | %idle | 48,50 | | | | |
| | | | kbmemfree | 19588 | %user | 52,97 | Last | 1,5 | rxpck/s | 2,97 |
| | | | kbmemused | 924844 | %nice | 0,00 | avg | 2,4 | txpck/s | 0,99 |
| | | | %memused | 97,93 | %system | 0,50 | best | 0,3 | rxkB/s | 0,13 |
| | | | kbbuffers | 8160 | %iowait | 0,50 | wrst | 18,7 | txkB/s | 0,09 |
| | | | kbcached | 68176 | %steal | 0,00 | StDev | 5,7 | rxmcst/s | 2,97 |
| 8 | 8 | 34627 | | | %idle | 46,04 | | | | |
| | | | kbmemfree | 15873 | %user | 52,57 | Last | 1,2 | rxpck/s | 2,99 |
| | | | kbmemused | 926547 | %nice | 0,00 | avg | 0,7 | txpck/s | 0,99 |
| | | | %memused | 98,12 | %system | 0,00 | best | 0,2 | rxkB/s | 0,17 |
| | | | kbbuffers | 9427 | %iowait | 1,75 | wrst | 1,6 | txkB/s | 0,08 |
| | | | kbcached | 68845 | %steal | 0,00 | StDev | 8,3 | rxmcst/s | 3,05 |
| | | | | | %idle | 45,69 | | | | |

**CONTINUA** ⟶

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 9 | 33089 | kbmemfree | 16660 | %user | 52,22 | Last | 0,5 | rxpck/s | 3,00 |
| | | | kbmemused | 927772 | %nice | 0,00 | avg | 0,6 | txpck/s | 1,00 |
| | | | %memused | 98,24 | %system | 0,00 | best | 0,4 | rxkB/s | 0,61 |
| | | | kbbuffers | 10080 | %iowait | 2,46 | wrst | 1,0 | txkB/s | 0,10 |
| | | | kbcached | 69872 | %steal | 0,00 | StDev | 0,2 | rxmcst/s | 3,00 |
| | | | | | %idle | 45,32 | | | | |
| 1 | 10 | 33012 | kbmemfree | 12828 | %user | 52,28 | Last | 0,5 | rxpck/s | 2,97 |
| | | | kbmemused | 931604 | %nice | 0,00 | avg | 0,6 | txpck/s | 0,99 |
| | | | %memused | 98,64 | %system | 0,51 | best | 0,4 | rxkB/s | 0,60 |
| | | | kbbuffers | 11972 | %iowait | 0,00 | wrst | 1,9 | txkB/s | 0,09 |
| | | | kbcached | 67972 | %steal | 0,00 | StDev | 0,4 | rxmcst/s | 2,97 |
| | | | | | %idle | 47,21 | | | | |
| 1 | 50 | 33385 | kbmemfree | 17504 | %user | 79,21 | Last | 0,4 | rxpck/s | 3,96 |
| | | | kbmemused | 926928 | %nice | 0,00 | avg | 0,4 | txpck/s | 2,97 |
| | | | %memused | 98,15 | %system | 6,93 | best | 0,4 | rxkB/s | 0,75 |
| | | | kbbuffers | 13628 | %iowait | 0,00 | wrst | 0,5 | txkB/s | 0,25 |
| | | | kbcached | 69248 | %steal | 0,00 | StDev | 0,0 | rxmcst/s | 2,97 |
| | | | | | %idle | 13,86 | | | | |
| 1 | 100 | 33061 | kbmemfree | 28972 | %user | 56,86 | Last | 0,5 | rxpck/s | 3,92 |
| | | | kbmemused | 915460 | %nice | 0,00 | avg | 0,4 | txpck/s | 1,96 |
| | | | %memused | 96,93 | %system | 0,98 | best | 0,3 | rxkB/s | 0,32 |
| | | | kbbuffers | 15200 | %iowait | 0,00 | wrst | 0,5 | txkB/s | 0,19 |
| | | | kbcached | 70972 | %steal | 0,00 | StDev | 0,0 | rxmcst/s | 2,94 |
| | | | | | %idle | 42,16 | | | | |
| 1 | 200 | 32528 | kbmemfree | 21660 | %user | 53,47 | Last | 0,5 | rxpck/s | 4,00 |
| | | | kbmemused | 922772 | %nice | 0,00 | avg | 0,4 | txpck/s | 2,00 |
| | | | %memused | 97,71 | %system | 0,50 | best | 0,4 | rxkB/s | 0,21 |
| | | | kbbuffers | 15952 | %iowait | 0,00 | wrst | 0,5 | txkB/s | 0,17 |
| | | | kbcached | 72648 | %steal | 0,00 | StDev | 0,0 | rxmcst/s | 3,00 |
| | | | | | %idle | 46,04 | | | | |

**ANEXOS C**

**Cliente: Algoritmo RSA optimizado**

**Tabla 47**

***Variables evaluadas con el algoritmo RSA optimizado: Cliente***

| NÚMERO ITERACIONES | NÚMERO DE LETRAS | TIEMPO DE CIFRADO | CONSUMO DE MEMORIA | | CONSUMO DEL PROCESADOR | | LATENCIA | | REPORTE ESTADÍSTICO DE LA RED | |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | kbmemfree | 1187448 | %user | 15,61 | Last | 0,3 | rxpck/s | 30,00 |
| | | | kbmemused | 847908 | %nice | 0,00 | avg | 0,3 | txpck/s | 28,00 |
| | | | %memused | 41,66 | %system | 1,46 | best | 0,2 | rxkB/s | 3,71 |
| | | | kbbuffers | 25000 | %iowait | 0,00 | wrst | 0,3 | txkB/s | 9,42 |
| | | | kbcached | 327532 | %steal | 0,00 | StDev | 0,0 | rxmcst/s | 5,00 |
| | | | | | %idle | 82,93 | | | | |
| 2 | 2 | 0 | kbmemfree | 1186796 | %user | 9,36 | Last | 0,5 | rxpck/s | 29,00 |
| | | | kbmemused | 848560 | %nice | 0,00 | avg | 1,4 | txpck/s | 28,00 |
| | | | %memused | 41,69 | %system | 0,49 | best | 0,2 | rxkB/s | 3,67 |
| | | | kbbuffers | 26784 | %iowait | 0,00 | wrst | 11,6 | txkB/s | 9,42 |
| | | | kbcached | 328448 | %steal | 0,00 | StDev | 3,6 | rxmcst/s | 0,00 |
| | | | | | %idle | 90,15 | | | | |
| 3 | 3 | 1 | kbmemfree | 1186044 | %user | 8,87 | Last | 0,3 | rxpck/s | 30,00 |
| | | | kbmemused | 849312 | %nice | 0,00 | avg | 0,3 | txpck/s | 29,00 |
| | | | %memused | 41,73 | %system | 1,97 | best | 0,2 | rxkB/s | 3,62 |
| | | | kbbuffers | 27844 | %iowait | 0,00 | wrst | 0,3 | txkB/s | 9,48 |
| | | | kbcached | 328760 | %steal | 0,00 | StDev | 0,0 | rxmcst/s | 0,00 |
| | | | | | %idle | 89,16 | | | | |

**CONTINUA** ⟶

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 4 | 4 | 1 | kbmemfree | 1185752 | %user | 8,82 | Last | 0,2 | rxpck/s | 32,67 |
| | | | kbmemused | 849604 | %nice | 0,00 | avg | 0,3 | txpck/s | 28,71 |
| | | | %memused | 41,74 | %system | 1,47 | best | 0,2 | rxkB/s | 3,94 |
| | | | kbbuffers | 28828 | %iowait | 0,00 | wrst | 0,3 | txkB/s | 9,39 |
| | | | kbcached | 329360 | %steal | 0,00 | StDev | 0,0 | rxmcst/s | 2,97 |
| | | | | | %idle | 89,71 | | | | |
| 5 | 5 | 1 | kbmemfree | 1174828 | %user | 15,50 | Last | 0,3 | rxpck/s | 30,00 |
| | | | kbmemused | 860528 | %nice | 0,00 | avg | 0,2 | txpck/s | 28,00 |
| | | | %memused | 42,28 | %system | 1,50 | best | 0,2 | rxkB/s | 3,99 |
| | | | kbbuffers | 34512 | %iowait | 0,00 | wrst | 0,3 | txkB/s | 9,42 |
| | | | kbcached | 330992 | %steal | 0,00 | StDev | 0,0 | rxmcst/s | 0,00 |
| | | | | | %idle | 83,00 | | | | |
| 6 | 6 | 1 | kbmemfree | 1137352 | %user | 22,77 | Last | 0,3 | rxpck/s | 33,00 |
| | | | kbmemused | 898004 | %nice | 0,00 | avg | 0,3 | txpck/s | 28,00 |
| | | | %memused | 44,12 | %system | 3,47 | best | 0,2 | rxkB/s | 3,92 |
| | | | kbbuffers | 42628 | %iowait | 0,00 | wrst | 0,5 | txkB/s | 9,42 |
| | | | kbcached | 359768 | %steal | 0,00 | StDev | 0,1 | rxmcst/s | 2,00 |
| | | | | | %idle | 73,76 | | | | |
| 7 | 7 | 1 | kbmemfree | 11251444 | %user | 9,95 | Last | 0,3 | rxpck/s | 30,00 |
| | | | kbmemused | 910212 | %nice | 0,00 | avg | 0,3 | txpck/s | 28,00 |
| | | | %memused | 44,72 | %system | 1,00 | best | 0,2 | rxkB/s | 3,61 |
| | | | kbbuffers | 49628 | %iowait | 0,00 | wrst | 0,3 | txkB/s | 9,42 |
| | | | kbcached | 360112 | %steal | 0,00 | StDev | 0,0 | rxmcst/s | 6,00 |
| | | | | | %idle | 89,05 | | | | |
| 8 | 8 | 1 | kbmemfree | 1124873 | %user | 14,49 | Last | 0,2 | rxpck/s | 34,00 |
| | | | kbmemused | 911457 | %nice | 0,00 | avg | 0,2 | txpck/s | 29,00 |
| | | | %memused | 44,79 | %system | 2,90 | best | 0,2 | rxkB/s | 4,07 |
| | | | kbbuffers | 50647 | %iowait | 0,00 | wrst | 0,3 | txkB/s | 9,49 |
| | | | kbcached | 360197 | %steal | 0,00 | StDev | 0,0 | rxmcst/s | 0,00 |
| | | | | | %idle | 82,61 | | | | |

**CONTINUA** →

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 9 | 9 | 1 | kbmemfree | 1123164 | %user | 15,58 | Last | 0,2 | rxpck/s | 34,00 |
| | | | kbmemused | 912192 | %nice | 0,00 | avg | 0,3 | txpck/s | 29,00 |
| | | | %memused | 44,82 | %system | 0,00 | best | 0,2 | rxkB/s | 4,07 |
| | | | kbbuffers | 51044 | %iowait | 0,00 | wrst | 0,5 | txkB/s | 9,49 |
| | | | kbcached | 360284 | %steal | 0,00 | StDev | 0,1 | rxmcst/s | 0,00 |
| | | | | | %idle | 84,42 | | | | |
| 10 | 10 | 1 | kbmemfree | 1121332 | %user | 8,53 | Last | 0,3 | rxpck/s | 28,71 |
| | | | kbmemused | 914024 | %nice | 0,00 | avg | 0,3 | txpck/s | 24,75 |
| | | | %memused | 44,91 | %system | 2,84 | best | 0,2 | rxkB/s | 3,50 |
| | | | kbbuffers | 52224 | %iowait | 0,00 | wrst | 0,3 | txkB/s | 9,11 |
| | | | kbcached | 360612 | %steal | 0,00 | StDev | 0,0 | rxmcst/s | 0,00 |
| | | | | | %idle | 88,63 | | | | |
| 50 | 50 | 3 | kbmemfree | 1100048 | %user | 33,00 | Last | 0,3 | rxpck/s | 30,00 |
| | | | kbmemused | 935308 | %nice | 0,00 | avg | 0,3 | txpck/s | 29,00 |
| | | | %memused | 45,95 | %system | 1,48 | best | 0,2 | rxkB/s | 3,74 |
| | | | kbbuffers | 55108 | %iowait | 0,00 | wrst | 0,3 | txkB/s | 9,53 |
| | | | kbcached | 363800 | %steal | 0,00 | StDev | 0,0 | rxmcst/s | 0,00 |
| | | | | | %idle | 65,52 | | | | |
| 100 | 100 | 6 | kbmemfree | 1098668 | %user | 16,75 | Last | 0,2 | rxpck/s | 34,00 |
| | | | kbmemused | 936688 | %nice | 0,00 | avg | 0,2 | txpck/s | 31,00 |
| | | | %memused | 46,02 | %system | 1,48 | best | 0,2 | rxkB/s | 3,96 |
| | | | kbbuffers | 56284 | %iowait | 0,00 | wrst | 0,3 | txkB/s | 9,71 |
| | | | kbcached | 364104 | %steal | 0,00 | StDev | 0,0 | rxmcst/s | 0,00 |
| | | | | | %idle | 81,77 | | | | |
| 200 | 200 | 11 | kbmemfree | 1089940 | %user | 14,85 | Last | 0,2 | rxpck/s | 32,00 |
| | | | kbmemused | 945416 | %nice | 0,00 | avg | 0,3 | txpck/s | 33,00 |
| | | | %memused | 46,45 | %system | 1,98 | best | 0,2 | rxkB/s | 3,73 |
| | | | kbbuffers | 57164 | %iowait | 0,00 | wrst | 0,7 | txkB/s | 9,93 |
| | | | kbcached | 364392 | %steal | 0,00 | StDev | 0,2 | rxmcst/s | 0,00 |
| | | | | | %idle | 83,17 | | | | |

**ANEXOS D**

**Servidor: Algoritmo RSA optimizado**

**Tabla 48**

***Variables evaluadas con el algoritmo RSA optimizado: Servidor***

| NÚMERO ITERACIONES | NÚMERO DE LETRAS | TIEMPO DE DESCIFRADO | CONSUMO DE MEMORIA | | CONSUMO DEL PROCESADOR | | LATENCIA | | REPORTE ESTADÍSTICO DE LA RED | |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 2 | kbmemfree | 9188 | %user | 6,93 | Last | 0,3 | rxpck/s | 28,71 |
| | | | kbmemused | 932244 | %nice | 0,00 | avg | 0,3 | txpck/s | 28,71 |
| | | | %memused | 99,03 | %system | 0,00 | best | 0,3 | rxkB/s | 3,50 |
| | | | kbbuffers | 5080 | %iowait | 26,24 | wrst | 0,6 | txkB/s | 9,39 |
| | | | kbcached | 75880 | %steal | 0,00 | StDev | 0,1 | rxmcst/s | 0,00 |
| | | | | | %idle | 66,83 | | | | |
| 2 | 2 | 2 | kbmemfree | 18136 | %user | 6,93 | Last | 0,3 | rxpck/s | 29,00 |
| | | | kbmemused | 926296 | %nice | 0,00 | avg | 0,3 | txpck/s | 28,00 |
| | | | %memused | 98,08 | %system | 0,99 | best | 0,3 | rxkB/s | 3,53 |
| | | | kbbuffers | 6268 | %iowait | 1,49 | wrst | 0,6 | txkB/s | 9,42 |
| | | | kbcached | 78048 | %steal | 0,00 | StDev | 0,1 | rxmcst/s | 2,00 |
| | | | | | %idle | 90,59 | | | | |
| 3 | 3 | 3 | kbmemfree | 19144 | %user | 6,93 | Last | 0,3 | rxpck/s | 27,00 |
| | | | kbmemused | 925288 | %nice | 0,00 | avg | 0,3 | txpck/s | 29,00 |
| | | | %memused | 97,97 | %system | 0,50 | best | 0,3 | rxkB/s | 3,41 |
| | | | kbbuffers | 6068 | %iowait | 1,98 | wrst | 0,3 | txkB/s | 9,48 |
| | | | kbcached | 78456 | %steal | 0,00 | StDev | 0,0 | rxmcst/s | 0,00 |
| | | | | | %idle | 90,59 | | | | |

**CONTINUA** ⟶

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 4 | 4 | 2 | kbmemfree | 17932 | %user | 6,93 | Last | 0,2 | rxpck/s | 29,00 |
| | | | kbmemused | 926500 | %nice | 0,00 | avg | 0,3 | txpck/s | 29,00 |
| | | | %memused | 98,1 | %system | 0,00 | best | 0,2 | rxkB/s | 3,53 |
| | | | kbbuffers | 7236 | %iowait | 1,98 | wrst | 0,5 | txkB/s | 9,48 |
| | | | kbcached | 78176 | %steal | 0,00 | StDev | 0,1 | rxmcst/s | 2,00 |
| | | | | | %idle | 91,09 | | | | |
| 5 | 5 | 2 | kbmemfree | 17060 | %user | 8,91 | Last | 0,3 | rxpck/s | 27,45 |
| | | | kbmemused | 927372 | %nice | 0,00 | avg | 0,2 | txpck/s | 27,45 |
| | | | %memused | 98,19 | %system | 1,49 | best | 0,2 | rxkB/s | 3,40 |
| | | | kbbuffers | 12616 | %iowait | 0,00 | wrst | 0,3 | txkB/s | 9,24 |
| | | | kbcached | 78260 | %steal | 0,00 | StDev | 0,0 | rxmcst/s | 1,96 |
| | | | | | %idle | 89,60 | | | | |
| 6 | 6 | 2 | kbmemfree | 19376 | %user | 5,37 | Last | 0,3 | rxpck/s | 28,00 |
| | | | kbmemused | 925056 | %nice | 0,00 | avg | 0,3 | txpck/s | 28,00 |
| | | | %memused | 97,95 | %system | 0,00 | best | 0,3 | rxkB/s | 3,47 |
| | | | kbbuffers | 13540 | %iowait | 1,46 | wrst | 0,3 | txkB/s | 9,42 |
| | | | kbcached | 78516 | %steal | 0,49 | StDev | 0,0 | rxmcst/s | 0,00 |
| | | | | | %idle | 92,68 | | | | |
| 7 | 7 | 2 | kbmemfree | 14436 | %user | 7,43 | Last | 0,3 | rxpck/s | 30,00 |
| | | | kbmemused | 929996 | %nice | 0,00 | avg | 0,5 | txpck/s | 30,00 |
| | | | %memused | 98,47 | %system | 0,99 | best | 0,2 | rxkB/s | 3,62 |
| | | | kbbuffers | 14428 | %iowait | 2,48 | wrst | 1,8 | txkB/s | 9,53 |
| | | | kbcached | 79332 | %steal | 0,00 | StDev | 0,5 | rxmcst/s | 3,00 |
| | | | | | %idle | 89,11 | | | | |
| 8 | 8 | 3 | kbmemfree | 14875 | %user | 8,91 | Last | 0,2 | rxpck/s | 28,00 |
| | | | kbmemused | 929175 | %nice | 0,00 | avg | 0,3 | txpck/s | 28,00 |
| | | | %memused | 98,38 | %system | 1,98 | best | 0,2 | rxkB/s | 3,47 |
| | | | kbbuffers | 14987 | %iowait | 7,43 | wrst | 0,7 | txkB/s | 9,42 |
| | | | kbcached | 79023 | %steal | 0,50 | StDev | 0,2 | rxmcst/s | 0,00 |
| | | | | | %idle | 81,19 | | | | |

**CONTINUA** ⟶

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 9 | 9 | 2 | kbmemfree | 15716 | %user | 2,50 | Last | 0,3 | rxpck/s | 28,00 |
| | | | kbmemused | 928716 | %nice | 0,00 | avg | 0,3 | txpck/s | 28,00 |
| | | | %memused | 98,34 | %system | 0,00 | best | 0,3 | rxkB/s | 3,47 |
| | | | kbbuffers | 15020 | %iowait | 2,00 | wrst | 0,6 | txkB/s | 9,42 |
| | | | kbcached | 78236 | %steal | 0,00 | StDev | 0,1 | rxmcst/s | 2,00 |
| | | | | | %idle | 95,50 | | | | |
| 10 | 10 | 3 | kbmemfree | 13364 | %user | 8,41 | Last | 0,3 | rxpck/s | 29,70 |
| | | | kbmemused | 931068 | %nice | 0,00 | avg | 0,3 | txpck/s | 28,71 |
| | | | %memused | 98,58 | %system | 0,50 | best | 0,2 | rxkB/s | 3,56 |
| | | | kbbuffers | 16448 | %iowait | 1,49 | wrst | 0,3 | txkB/s | 9,40 |
| | | | kbcached | 78284 | %steal | 0,00 | StDev | 0,0 | rxmcst/s | 0,00 |
| | | | | | %idle | 89,11 | | | | |
| 50 | 50 | 6 | kbmemfree | 18280 | %user | 7,92 | Last | 0,3 | rxpck/s | 31,00 |
| | | | kbmemused | 926152 | %nice | 0,00 | avg | 0,3 | txpck/s | 30,00 |
| | | | %memused | 98,06 | %system | 0,00 | best | 0,2 | rxkB/s | 3,66 |
| | | | kbbuffers | 18632 | %iowait | 1,49 | wrst | 0,3 | txkB/s | 9,59 |
| | | | kbcached | 78752 | %steal | 0,00 | StDev | 0,0 | rxmcst/s | 3,00 |
| | | | | | %idle | 90,59 | | | | |
| 100 | 100 | 17 | kbmemfree | 20860 | %user | 6,93 | Last | 0,3 | rxpck/s | 32,00 |
| | | | kbmemused | 923572 | %nice | 0,00 | avg | 0,3 | txpck/s | 30,00 |
| | | | %memused | 97,79 | %system | 0,99 | best | 0,3 | rxkB/s | 3,86 |
| | | | kbbuffers | 17660 | %iowait | 1,49 | wrst | 0,3 | txkB/s | 9,64 |
| | | | kbcached | 78404 | %steal | 0,00 | StDev | 0,0 | rxmcst/s | 2,00 |
| | | | | | %idle | 90,59 | | | | |
| 200 | 200 | 34 | kbmemfree | 33248 | %user | 8,42 | Last | 0,3 | rxpck/s | 31,00 |
| | | | kbmemused | 911184 | %nice | 0,00 | avg | 0,3 | txpck/s | 34,00 |
| | | | %memused | 96,48 | %system | 0,99 | best | 0,3 | rxkB/s | 3,66 |
| | | | kbbuffers | 1680 | %iowait | 1,49 | wrst | 0,3 | txkB/s | 10,00 |
| | | | kbcached | 72032 | %steal | 0,00 | StDev | 0,0 | rxmcst/s | 0,00 |
| | | | | | %idle | 89,11 | | | | |

**ANEXOS E**

**Cliente: Algoritmo RSA optimizado recursivo**

**Tabla 49**

***Variables evaluadas con el algoritmo RSA optimizado recursivo: Cliente***

| NÚMERO ITERACIONES | NÚMERO DE LETRAS | TIEMPO DE DESCIFRADO | CONSUMO DE MEMORIA | | CONSUMO DEL PROCESADOR | | LATENCIA | | REPORTE ESTADÍSTICO DE LA RED | |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | kbmemfree | 406432 | %user | 8,02 | Last | 0,2 | rxpck/s | 28,71 |
| | | | kbmemused | 1628924 | %nice | 0,00 | avg | 0,2 | txpck/s | 28,71 |
| | | | %memused | 80,03 | %system | 1,89 | best | 0,2 | rxkB/s | 3,50 |
| | | | kbbuffers | 746776 | %iowait | 0,00 | wrst | 0,3 | txkB/s | 9,39 |
| | | | kbcached | 339372 | %steal | 0,00 | StDev | 0,0 | rxmcst/s | 0,00 |
| | | | | | %idle | 90,09 | | | | |
| 2 | 2 | 1 | kbmemfree | 403384 | %user | 15,61 | Last | 0,2 | rxpck/s | 29,00 |
| | | | kbmemused | 1631972 | %nice | 0,00 | avg | 0,2 | txpck/s | 28,00 |
| | | | %memused | 80,18 | %system | 1,46 | best | 0,2 | rxkB/s | 3,53 |
| | | | kbbuffers | 747752 | %iowait | 0,00 | wrst | 0,3 | txkB/s | 9,42 |
| | | | kbcached | 339660 | %steal | 0,00 | StDev | 0,0 | rxmcst/s | 2,00 |
| | | | | | %idle | 82,93 | | | | |
| 3 | 3 | 1 | kbmemfree | 402416 | %user | 17,65 | Last | 0,3 | rxpck/s | 27,00 |
| | | | kbmemused | 1632940 | %nice | 0,00 | avg | 0,3 | txpck/s | 29,00 |
| | | | %memused | 80,23 | %system | 0,98 | best | 0,2 | rxkB/s | 3,41 |
| | | | kbbuffers | 747180 | %iowait | 0,00 | wrst | 0,5 | txkB/s | 9,48 |
| | | | kbcached | 340028 | %steal | 0,00 | StDev | 0,1 | rxmcst/s | 0,00 |
| | | | | | %idle | 81,37 | | | | |

**CONTINUA** ⟶

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 4 | 4 | 0 | kbmemfree | 401556 | %user | 8,46 | Last | 0,3 | rxpck/s | 29,00 |
| | | | kbmemused | 1633800 | %nice | 0,00 | avg | 0,3 | txpck/s | 29,00 |
| | | | %memused | 80,27 | %system | 0,00 | best | 0,2 | rxkB/s | 3,53 |
| | | | kbbuffers | 746740 | %iowait | 0,00 | wrst | 0,3 | txkB/s | 9,48 |
| | | | kbcached | 340364 | %steal | 0,00 | StDev | 0,0 | rxmcst/s | 2,00 |
| | | | | | %idle | 91,54 | | | | |
| 5 | 5 | 1 | kbmemfree | 399280 | %user | 16,34 | Last | 0,3 | rxpck/s | 27,45 |
| | | | kbmemused | 1636076 | %nice | 0,00 | avg | 0,2 | txpck/s | 27,45 |
| | | | %memused | 80,38 | %system | 2,97 | best | 0,2 | rxkB/s | 3,40 |
| | | | kbbuffers | 746720 | %iowait | 0,00 | wrst | 0,3 | txkB/s | 9,24 |
| | | | kbcached | 343960 | %steal | 0,00 | StDev | 0,0 | rxmcst/s | 1,96 |
| | | | | | %idle | 80,69 | | | | |
| 6 | 6 | 1 | kbmemfree | 397092 | %user | 9,66 | Last | 0,3 | rxpck/s | 28,00 |
| | | | kbmemused | 1638264 | %nice | 0,00 | avg | 0,2 | txpck/s | 28,00 |
| | | | %memused | 80,49 | %system | 1,45 | best | 0,2 | rxkB/s | 3,47 |
| | | | kbbuffers | 746624 | %iowait | 0,00 | wrst | 0,3 | txkB/s | 9,42 |
| | | | kbcached | 344508 | %steal | 0,00 | StDev | 0,0 | rxmcst/s | 0,00 |
| | | | | | %idle | 88,89 | | | | |
| 7 | 7 | 1 | kbmemfree | 1273444 | %user | 13,00 | Last | 0,3 | rxpck/s | 30,00 |
| | | | kbmemused | 761912 | %nice | 0,00 | avg | 0,3 | txpck/s | 30,00 |
| | | | %memused | 37,43 | %system | 0,50 | best | 0,2 | rxkB/s | 3,62 |
| | | | kbbuffers | 21964 | %iowait | 0,00 | wrst | 0,3 | txkB/s | 9,53 |
| | | | kbcached | 306072 | %steal | 0,00 | StDev | 0,0 | rxmcst/s | 3,00 |
| | | | | | %idle | 86,50 | | | | |
| 8 | 8 | 1 | kbmemfree | 1256368 | %user | 12,81 | Last | 0,3 | rxpck/s | 28,00 |
| | | | kbmemused | 778988 | %nice | 0,00 | avg | 0,3 | txpck/s | 28,00 |
| | | | %memused | 38,27 | %system | 1,48 | best | 0,3 | rxkB/s | 3,47 |
| | | | kbbuffers | 21092 | %iowait | 0,00 | wrst | 0,3 | txkB/s | 9,42 |
| | | | kbcached | 310880 | %steal | 0,00 | StDev | 0,0 | rxmcst/s | 0,00 |
| | | | | | %idle | 85,71 | | | | |

**CONTINUA** ⟶

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 9 | 9 | 1 | kbmemfree | 1263224 | %user | 18,91 | Last | 0,3 | rxpck/s | 28,00 |
| | | | kbmemused | 772132 | %nice | 0,00 | avg | 0,3 | txpck/s | 28,00 |
| | | | %memused | 37,94 | %system | 2,99 | best | 0,3 | rxkB/s | 3,47 |
| | | | kbbuffers | 23168 | %iowait | 0,00 | wrst | 0,3 | txkB/s | 9,42 |
| | | | kbcached | 308012 | %steal | 0,00 | StDev | 0,0 | rxmcst/s | 2,00 |
| | | | | | %idle | 78,11 | | | | |
| 10 | 10 | 1 | kbmemfree | 1262292 | %user | 19,12 | Last | 0,3 | rxpck/s | 29,70 |
| | | | kbmemused | 773064 | %nice | 0,00 | avg | 0,3 | txpck/s | 28,71 |
| | | | %memused | 37,98 | %system | 1,47 | best | 0,2 | rxkB/s | 3,56 |
| | | | kbbuffers | 23888 | %iowait | 0,00 | wrst | 0,3 | txkB/s | 9,40 |
| | | | kbcached | 308296 | %steal | 0,00 | StDev | 0,0 | rxmcst/s | 0,00 |
| | | | | | %idle | 79,41 | | | | |
| 50 | 50 | 2 | kbmemfree | 1239296 | %user | 9,90 | Last | 0,3 | rxpck/s | 31,00 |
| | | | kbmemused | 796060 | %nice | 0,00 | avg | 0,3 | txpck/s | 30,00 |
| | | | %memused | 39,11 | %system | 1,49 | best | 0,2 | rxkB/s | 3,66 |
| | | | kbbuffers | 26068 | %iowait | 0,00 | wrst | 0,3 | txkB/s | 9,59 |
| | | | kbcached | 316724 | %steal | 0,00 | StDev | 0,0 | rxmcst/s | 3,00 |
| | | | | | %idle | 88,61 | | | | |
| 100 | 100 | 3 | kbmemfree | 1236584 | %user | 12,00 | Last | 0,3 | rxpck/s | 32,00 |
| | | | kbmemused | 798772 | %nice | 0,00 | avg | 0,3 | txpck/s | 30,00 |
| | | | %memused | 39,24 | %system | 1,50 | best | 0,2 | rxkB/s | 3,86 |
| | | | kbbuffers | 27740 | %iowait | 0,00 | wrst | 0,3 | txkB/s | 9,64 |
| | | | kbcached | 317316 | %steal | 0,00 | StDev | 0,0 | rxmcst/s | 2,00 |
| | | | | | %idle | 86,50 | | | | |
| 200 | 200 | 6 | kbmemfree | 1234280 | %user | 19,61 | Last | 0,3 | rxpck/s | 31,00 |
| | | | kbmemused | 801076 | %nice | 0,00 | avg | 0,2 | txpck/s | 34,00 |
| | | | %memused | 39,36 | %system | 5,39 | best | 0,2 | rxkB/s | 3,66 |
| | | | kbbuffers | 21992 | %iowait | 0,00 | wrst | 0,3 | txkB/s | 10,00 |
| | | | kbcached | 317828 | %steal | 0,00 | StDev | 0,0 | rxmcst/s | 0,00 |
| | | | | | %idle | 75,00 | | | | |

**ANEXOS F**

**Servidor: Algoritmo RSA optimizado recursivo**

**Tabla 50**

***Variables evaluadas con el algoritmo RSA optimizado recursivo: Servidor***

| NÚMERO ITERACIONES | NÚMERO DE LETRAS | TIEMPO DE DESCIFRADO | CONSUMO DE MEMORIA | | CONSUMO DEL PROCESADOR | | LATENCIA | | REPORTE ESTADÍSTICO DE LA RED | |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 2 | kbmemfree | 41060 | %user | 4,46 | Last | 0,3 | rxpck/s | 29,70 |
| | | | kbmemused | 903372 | %nice | 0,00 | avg | 0,3 | txpck/s | 27,72 |
| | | | %memused | 95,65 | %system | 0,00 | best | 0,3 | rxkB/s | 9,04 |
| | | | kbbuffers | 2044 | %iowait | 1,98 | wrst | 0,3 | txkB/s | 3,44 |
| | | | kbcached | 63976 | %steal | 0,00 | StDev | 0,0 | rxmcst/s | 0,99 |
| | | | | | %idle | 93,56 | | | | |
| 2 | 2 | 4 | kbmemfree | 36748 | %user | 8,46 | Last | 0,3 | rxpck/s | 28,71 |
| | | | kbmemused | 907684 | %nice | 0,00 | avg | 0,3 | txpck/s | 27,72 |
| | | | %memused | 96,11 | %system | 0,00 | best | 0,3 | rxkB/s | 8,99 |
| | | | kbbuffers | 3380 | %iowait | 1,49 | wrst | 0,4 | txkB/s | 3,44 |
| | | | kbcached | 66444 | %steal | 0,00 | StDev | 0,0 | rxmcst/s | 0,99 |
| | | | | | %idle | 90,05 | | | | |
| 3 | 3 | 7 | kbmemfree | 34964 | %user | 1,98 | Last | 0,3 | rxpck/s | 29,70 |
| | | | kbmemused | 909468 | %nice | 0,00 | avg | 0,3 | txpck/s | 26,73 |
| | | | %memused | 96,3 | %system | 0,00 | best | 0,3 | rxkB/s | 9,05 |
| | | | kbbuffers | 4440 | %iowait | 1,49 | wrst | 0,3 | txkB/s | 3,37 |
| | | | kbcached | 66840 | %steal | 0,00 | StDev | 0,0 | rxmcst/s | 0,99 |
| | | | | | %idle | 96,53 | | | | |

**CONTINUA** ⟶

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 4 | 4 | 6 | kbmemfree | 30372 | %user | 5,94 | Last | 0,3 | rxpck/s | 29,70 |
| | | | kbmemused | 914060 | %nice | 0,00 | avg | 0,3 | txpck/s | 27,72 |
| | | | %memused | 96,78 | %system | 0,50 | best | 0,2 | rxkB/s | 9,05 |
| | | | kbbuffers | 5628 | %iowait | 1,49 | wrst | 0,8 | txkB/s | 3,44 |
| | | | kbcached | 67100 | %steal | 0,00 | StDev | 0,2 | rxmcst/s | 0,99 |
| | | | | | %idle | 92,08 | | | | |
| 5 | 5 | 5 | kbmemfree | 20296 | %user | 5,94 | Last | 0,3 | rxpck/s | 27,72 |
| | | | kbmemused | 924136 | %nice | 0,00 | avg | 0,3 | txpck/s | 27,72 |
| | | | %memused | 97,85 | %system | 1,49 | best | 0,2 | rxkB/s | 8,94 |
| | | | kbbuffers | 6968 | %iowait | 1,49 | wrst | 0,3 | txkB/s | 3,44 |
| | | | kbcached | 69228 | %steal | 0,00 | StDev | 0,0 | rxmcst/s | 0,00 |
| | | | | | %idle | 91,09 | | | | |
| 6 | 6 | 7 | kbmemfree | 18284 | %user | 2,00 | Last | 0,3 | rxpck/s | 28,00 |
| | | | kbmemused | 926148 | %nice | 0,00 | avg | 0,4 | txpck/s | 28,00 |
| | | | %memused | 98,06 | %system | 0,00 | best | 0,3 | rxkB/s | 9,04 |
| | | | kbbuffers | 8708 | %iowait | 2,00 | wrst | 0,7 | txkB/s | 3,47 |
| | | | kbcached | 69792 | %steal | 0,00 | StDev | 0,1 | rxmcst/s | 0,00 |
| | | | | | %idle | 96,00 | | | | |
| 7 | 7 | 8 | kbmemfree | 13848 | %user | 7,43 | Last | 0,3 | rxpck/s | 30,69 |
| | | | kbmemused | 930584 | %nice | 0,00 | avg | 0,3 | txpck/s | 28,71 |
| | | | %memused | 98,53 | %system | 0,50 | best | 0,2 | rxkB/s | 9,12 |
| | | | kbbuffers | 3432 | %iowait | 1,98 | wrst | 0,6 | txkB/s | 3,48 |
| | | | kbcached | 78040 | %steal | 0,00 | StDev | 0,1 | rxmcst/s | 0,99 |
| | | | | | %idle | 90,10 | | | | |
| 8 | 8 | 7 | kbmemfree | 13004 | %user | 13,79 | Last | 0,3 | rxpck/s | 29,70 |
| | | | kbmemused | 931428 | %nice | 0,00 | avg | 0,5 | txpck/s | 27,72 |
| | | | %memused | 98,62 | %system | 4,43 | best | 0,2 | rxkB/s | 9,05 |
| | | | kbbuffers | 1796 | %iowait | 13,79 | wrst | 2,3 | txkB/s | 3,44 |
| | | | kbcached | 78428 | %steal | 0,00 | StDev | 0,7 | rxmcst/s | 0,00 |
| | | | | | %idle | 67,98 | | | | |

**CONTINUA** ⟶

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 9 | 9 | 8 | kbmemfree | 9248 | %user | 10,95 | Last | 0,3 | rxpck/s | 27,72 |
| | | | kbmemused | 935184 | %nice | 0,00 | avg | 0,3 | txpck/s | 27,72 |
| | | | %memused | 99,02 | %system | 1,99 | best | 0,2 | rxkB/s | 8,95 |
| | | | kbbuffers | 3168 | %iowait | 5,97 | wrst | 0,3 | txkB/s | 3,44 |
| | | | kbcached | 79344 | %steal | 0,00 | StDev | 0,0 | rxmcst/s | 0,00 |
| | | | | | %idle | 81,09 | | | | |
| 10 | 10 | 9 | kbmemfree | 17372 | %user | 7,80 | Last | 0,3 | rxpck/s | 30,39 |
| | | | kbmemused | 927060 | %nice | 0,00 | avg | 0,3 | txpck/s | 27,45 |
| | | | %memused | 98,16 | %system | 1,46 | best | 0,3 | rxkB/s | 9,01 |
| | | | kbbuffers | 3796 | %iowait | 2,44 | wrst | 0,3 | txkB/s | 3,40 |
| | | | kbcached | 78600 | %steal | 0,00 | StDev | 0,0 | rxmcst/s | 1,96 |
| | | | | | %idle | 88,29 | | | | |
| 50 | 50 | 38 | kbmemfree | 16152 | %user | 8,82 | Last | 0,3 | rxpck/s | 31,68 |
| | | | kbmemused | 928280 | %nice | 0,00 | avg | 0,3 | txpck/s | 28,71 |
| | | | %memused | 98,29 | %system | 0,00 | best | 0,2 | rxkB/s | 9,19 |
| | | | kbbuffers | 5488 | %iowait | 1,96 | wrst | 0,3 | txkB/s | 3,50 |
| | | | kbcached | 76840 | %steal | 0,00 | StDev | 0,0 | rxmcst/s | 1,98 |
| | | | | | %idle | 89,22 | | | | |
| 100 | 100 | 47 | kbmemfree | 18968 | %user | 14,36 | Last | 0,3 | rxpck/s | 30,69 |
| | | | kbmemused | 925464 | %nice | 0,00 | avg | 0,4 | txpck/s | 29,70 |
| | | | %memused | 97,99 | %system | 0,00 | best | 0,3 | rxkB/s | 9,32 |
| | | | kbbuffers | 6924 | %iowait | 1,49 | wrst | 1,2 | txkB/s | 3,56 |
| | | | kbcached | 75160 | %steal | 0,00 | StDev | 0,3 | rxmcst/s | 0,99 |
| | | | | | %idle | 84,16 | | | | |
| 200 | 200 | 71 | kbmemfree | 9344 | %user | 15,42 | Last | 0,3 | rxpck/s | 36,00 |
| | | | kbmemused | 935088 | %nice | 0,00 | avg | 0,2 | txpck/s | 31,00 |
| | | | %memused | 99,01 | %system | 1,00 | best | 0,2 | rxkB/s | 9,63 |
| | | | kbbuffers | 4068 | %iowait | 12,44 | wrst | 0,3 | txkB/s | 3,66 |
| | | | kbcached | 80612 | %steal | 0,00 | StDev | 0,0 | rxmcst/s | 0,00 |
| | | | | | %idle | 71,14 | | | | |

**ANEXOS G**

**Base de datos Alfabeto ASCII extendido y Código (aleatorio)**

## Tabla 51

### *Alfabeto ASCII (extendido) y Código (aleatorio)*

| Clave | Contenido | Si_Alf_Cod | Fecha |
|---|---|---|---|
| **1** | !#$%&\'()*+,./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\\]^_`abcdefghij klmnopqrstuvwxyz{\|}Â€Â□Â,ÂƒÂ„Â…Â†Â‡Â^Â‰ÂŠÂ‹ÂŒÂ□ÂŽÂ□Â□Â'Â'Â"Â"Â•ÂÂÂ˜Â™ÂšÂ›Âœ□ÂžÂŸÂ¡Â¢Â£Â¤Â¥Â¦Â§Â¨Â©ÂªÂ«Â¬ÂÂ®Â¯Â°Â±Â²Â³Â´ÂµÂ¶Â·Â¸Â¹Â°Â»Â¼Â½Â¾Â¿Ã€Ã□Ã,ÃƒÃ„Ã…Ã†Ã‡ÃŠÃ‹ÃŒÃ□ÃŽÃ□Ã□Ã'Ã'Ã"Ã"Ã•ÃÃÃ˜Ã™ÃšÃ›ÃœÃ□ÃžÃŸÃ ¡Ã¢Ã£Ã¤Ã¥Ã¦Ã§Ã¨Ã©ÃªÃ«Ã¬ÃÃ®Ã¯Ã°Ã±Ã²Ã³Ã´ÃµÃ¶Ã·Ã¸Ã¹Ã°Ã»Ã¼Ã½Ã¾ | s | 2015-03-30 |
| **2** | 4Ã§IÃ©<Z0w?yeVÃ•Ã¼uÃ¸ÃšÃÂ^Ã"aqÂ¨Â°@Ã¾1nÂ§Â•Ã²=oÂœÂÂ"(Ã†ÂŸÃ¨cÃ®OLÃ□7Â,AÃ°Ã'Ã˜UÃ□Ã€].Â†DÂ‹WÃÂ*pÃ□Ã‰6Â²ft\|ÃœÂ□Â□Â©Ã·Â"Â„5Ã 3Â¦Ã¯GS)NÃ¶ÂžFÃ…rÂ•Ã½Â¿~Ã´}l E_ÂƒÂ‡Â£Â¥Ã°XkÂ€Ã"mÃ•>QMÂ□RÂŒ!zisÃ^Â´ÃžÂ¾Â¬Ã,&Â˜dÂšh+Ã'ÂÃ„Ã¶Ã¹Â½[ÂµÂ…vÂ¡Ã¥9CÂ·ÃŽ/BÃ¦Ã«Â¸Ã¬Â□ÃªÃ¡Â‰ÃŠÂ£Ã¢JÃŒÃÂ®H,Ãƒ{Ã™ÂªÂŽÃ«Â¢Â¼ÂŠ^xÃ¤Â¹jÂ'Â™Ã‡ÃŸ\\KTÂ¯:Â³8Ã»Ã³Â¤Â'\'bgÂ›Â»;#ÃµÃ±P2Â°%Ã□$Ã‹YÂ±Ã› | n | 2015-03-30 |
| **3** | LÃ«vÂ§dÃœÂ½8Â°Z#ÃšpÃ£>Âž\|hÃ¨Ã™!Ã¹JÃ¾Â‡[}Ã¬Â´ÃžÂ¬tT:Â·Â™Â□Ã°Â«Ã□~ÃŠÂŒÃ¶Â³Ã´Â–NAÂ•]EÂ,Ã□Ã©ÂŸÂ□QÂ°Â¥Ã»)gfÃ¡Â²xl*`+Ã□KÂ‹3Â£q;Ã³nÃ Ã¯5Ã¦Â¯oÃ·VÃ±\'I<Ã‹B ?ÃƒeÂ€Ã¥_Â¼@Ã½Ã°GÂ…Â®Ã‰ÃŸ=PÃ'Â¡Ã¼zÂ"Ã§.Ã›Ã²$Â±Ã"ÃÂ¤Y0X&Ã¢rÂÂšÂ¨cFÂƒUmsMÂ"Ã†/Â˜ÃªÂ€Ã¹Â†ÂŠ1Ã,ÂˆÂµ\\ÃŽÃbjÂ¿k7Ã"(Â6wC^Ã□Ã‡Ã9Ã¤Â□Â¸Ã'WÃˆÂ¾Â'Ã¸Â‰Ã˜2ÃµÂ„Ã®Ã•Ã…Ã□{ÂªÂœSÂŽÃ„Â'ÃŒÂ□4%Â¦OÂ›aDÂ•Â©,HRÂ¢Â»iyu | n | 2015-03-30 |
| **4** | OÃ□RÃ□gÂ´Ã□Â¶Â†Âµy@Ã°QHÃ'hkÃ›0tÃƒÂ,Â□ÃŽÃ½Ã¹ÂŒÂ"()/Â©Â¥WÂ'Â□nMÂƒÂ"Â§Aj\|Ã¸Ã¨Â¡Ã…[Â□Â›sÂ™F{ÃÃ«Â¾Â¢Â¼bÃ¤Â±lÂ□Â¦Â°1LVÃ¾NqÂÂ¹ÂŠCÃÂ«Ã;ÃµÃŸÃ¥ÂžÃ¦Â^~x=Â‡Ã¡Â‹Ã€%ÃªÃ‹Ã'Â®G$2\\ÂªX4cÂ˜DfS<]Ã™Ã†Â^Z\'Ã´d68eÃŠÃ"EÂ¤Â°UÃ¼Âš+ÂB:Ã¯Â„Â¬_ÃšÃ²Ã•Ã±mÃ»>&,Ã¢oÂ¿PÂ…ÂŸu^Ã˜zÃ□*Ã¶.wÂ¸Ã‰3Â»Â£Â•Â£Ã‡Ã¬?iÃ"Â·Â'ÃžˆKÂÂ½Â³v7Â€JrÃ„Âœ ÂŽÃ ÃœÂ¨!Â□Ã©9Ã·ÃŒÂ¯YT#Ã,Ã§Ã³aÃ°5Â‰IpÃœ}Ã® | n | 2015-03-30 |