Conference Paper

Comparative Analysis of File Transfer Performance Between Java and.NET Using a Hybrid Encryption Protocol with AES and RSA

Análisis comparativo del rendimiento en la transferencia de archivos entre Java y.NET utilizando un protocolo de cifrado híbrido con AES y RSA

Paúl Paguay-Soxo[1] and Marco Vivanco[2]

[1]Carrera de Ingeniería en Sistemas, Escuela Superior Politécnica de Chimborazo, Panamericana Sur Km. 1 ½, Riobamba, 593, Ecuador
[2]Registro Civil Oficina Loja, Loja, 593, Ecuador

Corresponding Author:
Paúl Paguay-Soxo
ppaguay@espoch.edu.ec

Received: 4 December 2018
Accepted: 5 December 2018
Published: 27 December 2018

Publishing services provided by Knowledge E

Selection and Peer-review under the responsibility of the SIIPRIN-CITEGC Conference Committee.

Abstract

The purpose of the research project was to carry out the comparative analysis of the performance of a computer system using a hybrid encryption protocol (symmetric AES and asymmetric RSA) and the mechanism of sockets between the JAVA and.NET platforms, applied to the system of information backups for the Loja Technical Office of the Zonal Coordination 7 of the Identification and Certification of the Civil Registry. To determine the performance between both platforms, the analysis of each indicator obtained from the FURPS Model was done with a sample of 335 files distributed among 200 light, 100 medium and 35 heavy, for which the nonparametric analysis algorithm Mann–Whiney was used, obtaining as a result a significant difference in favor of C # in terms of response time, while for the consumption of resources (RAM and CPU) a significant difference was found in favor of Java.

Resumen

El propósito del proyecto de investigación fue realizar el análisis comparativo del rendimiento de un sistema informático utilizando un protocolo de cifrado híbrido (simétrico AES y asimétrico RSA) y el mecanismo de sockets entre las plataformas JAVA y.NET, aplicado al sistema de respaldos de información para la Oficina Técnica Loja de la Coordinación Zonal 7 de la Identificación y Certificación del Registro Civil. Para determinar el rendimiento entre ambas plataformas, el análisis de cada indicador obtenido del Modelo FURPS se realizó con una muestra de 335 archivos distribuidos entre 200 livianos, 100 medianos y 35 pesados, para lo cual se utilizó el algoritmo de análisis no paramétrico Mann-Whiney obteniendo como resultados una diferencia significativa a favor de C# en cuanto al tiempo de respuesta, mientras que para

OPEN ACCESS

los indicadores de consumo de recursos RAM y CPU se encontró una diferencia significativa a favor de Java.

Keywords: cryptography, AES, RSA, Java, C#

Palabras clave: criptografìa, AES, RSA, Java, C#

1. Introducción

El término criptografía proviene de dos vocablos griegos: "cryptos" que significa "escondido" y "graphos" "escritura". Más tarde se añade el sufijo -ía para conferirle el carácter de conocimiento o tratado (Díaz, 1995). De entre varias de las definiciones que algunos autores han dado a esta importante disciplina, se cita la de (Menezes, Ooschot, & Vanstone, 1996) que dice: "Es el estudio de técnicas matemáticas relacionadas con los aspectos de la seguridad de la información tales como la confidencialidad, la integridad de datos, la autenticación de entidad y de origen. No comprende sólo a los medios para proveer seguridad de información, sino un conjunto de técnicas."

La criptografía es una herramienta muy útil cuando se desea tener seguridad informática; puede ser también entendida como un medio para garantizar las propiedades de confidencialidad, integridad y disponibilidad de los recursos de un sistema (Paredes, 2006).

El principio de "Defensa en Profundidad" consiste en el diseño e implementación de varios niveles de seguridad. De modo que, si una "barrera" es franqueada, conviene disponer de medidas de seguridad adicionales, siendo la criptografía el último eslabón de esta cadena. Por otro lado (Gómez, 2011) ubica a la Criptografía en el plano técnico, dentro de un Sistema de Gestión de la Seguridad de la Información (SGSI).

Una categorización de los algoritmos criptográficos puede ser: los que utilizan claves de encriptación Simétricas (privada) y Asimétricos (públicos). En las claves simétricas o secretas, únicamente una clave es usada para cifrar y descifrar. En los algoritmos asimétricos, dos claves son usadas, una pública usada para cifrar y otra privada que sirve para descifrar (Mahajan & Sachdeva, 2013).

Según (García Belmonte, 2016), con la criptografía simétrica la misma llave se comparte entre el emisor y el receptor, lo que hace vulnerable al sistema en el proceso de intercambio de claves, sin embargo, el proceso de cifrado y descifrado se realiza

más rápido y con menores recursos computacionales en comparación con un sistema asimétrico cuya ventaja es el intercambio y distribución de claves.

El sistema criptográfico híbrido es la unión de las ventajas de los dos sistemas criptográficos antes mencionados, "Esta técnica de cifrado combinado se utiliza con muchísima frecuencia. Por ejemplo, se utiliza en Secure Shell (SSH) para proteger las comunicaciones entre el cliente y el servidor y en PGP (Pretty Good Privacy) para enviar correos electrónicos. Además, es el mecanismo básico en Transport Layer Security (TLS), que es el protocolo utilizado en la Web para mantener un canal de comunicación seguro" (Soriano).

El proceso para enviar un archivo usando un sistema criptográfico híbrido es el siguiente:

- Establecer una clave pública y privada para el receptor
- Cifrar un archivo de forma simétrica
- El receptor envía su clave pública
- Se cifra la clave que se usa para cifrar el archivo con la clave pública del receptor
- Se envía el archivo cifrado síncronamente y la clave del archivo cifrada asíncronamente
- El receptor procede a descifrar primero la clave simétrica utilizando su clave privada asimétrica
- Y posterior descifra el archivo utilizando la clave simétrica

AES es un cifrado iterativo que fue propuesto por Joan Daemen y Vincent Rijmen (Rijndael) (Al Hasib & Mahmudul Haque, A Comparative Study of the Performance and Security Issues of AES and RSA Cryptography, 2008). Admite cualquier combinación de datos y longitud de clave de 128, 192 y 256 bits. AES permite una longitud de datos de 128 bits que se puede dividir en cuatro bloques operativos básicos. Estos bloques se consideran como un conjunto de bytes y se organizan como una matriz del orden de 4×4 que también se denomina estado y está sujeto a rondas donde se realizan diversas transformaciones. Para el cifrado completo el número de rondas utilizadas es variable N = 10, 12, 14 para una longitud de clave de 128,192 y 256, respectivamente. Cada ronda de AES usa red de permutaciones y sustituciones, y es adecuado para la implementación de hardware y software (Patil, Narayankar, Narayan, & Meena, 2016).

RSA se fundó en 1977 y es un criptosistema de clave pública. RSA es un algoritmo criptográfico asimétrico llamado así por sus fundadores Rivest, Shamir y Adelman.

Genera dos claves: clave pública para el cifrado y clave privada para descifrar el mensaje. El algoritmo de RSA consta de tres pasos, el paso uno es la generación de la clave que se utilizará como clave para cifrar y descifrar datos, el segundo paso es el cifrado, donde el proceso de conversión de texto plano a texto cifrado se lleva a cabo y el tercer paso es el descifrado, donde el texto cifrado se convierte en texto plano en el lado del receptor. RSA es basado en el problema de factorización de encontrar el producto de dos números primos grandes. El tamaño de clave es de 1024 a 4096 bits (Patil, Narayankar, Narayan, & Meena, 2016). En el Ecuador el algoritmo RSA es el utilizado para el cifrado asimétrico necesario para la firma electrónica, este sistema, parece el más adecuado, para el cumplimiento de los requisitos, y es el más utilizado en la práctica (Vallejo, 2005).

Establecer un canal de comunicación por el cual un proceso puede enviar o recibir datos de un equipo a otro, para ello se utiliza Sockets como un sistema de comunicación entre procesos de diferentes equipos de una red. Un socket generalmente es soportado por lenguajes de programación de alto nivel como (C#, Java, Phyton, etc) el mismo que emplea un protocolo TCP para realizar la conexión y comunicación entre sus terminales.

Mientras se procura tener un sistema lo más seguro posible, el rendimiento de los sistemas informáticos por otro lado, es un requerimiento no funcional que está presente en cualquier proyecto que en la estructura apliquen plataformas que no declive el servicio en el tiempo, tal como lo indica (USA Patente n° US-8972940-B2) lo que denota el impacto que tiene el rendimiento en la calidad del Software (Vidal-Silva, Madariaga, & Solis, 2017).

Según (TIOBE, 2018), entre los lenguajes de programación más utilizados están Java y.NET, los cuales permiten implementar soluciones seguras a través de protocolos criptográficos como los especificados anteriormente, cada lenguaje cuenta con su plataforma de ejecución y características propias. Una de las librerías disponibles para la aplicación de algoritmos criptográficos es la API Bouncy Castle Crypto que implementa algoritmos simétricos, asimétricos, de resumen, firmas, etc., fue desarrollado por la Legión Bouncy Castle y está disponible para los lenguajes de programación Java y C# (bouncycastle, 2018).

Varios trabajos como los de (Al Hasib & Mahmudul), (Padmavathi & Ranjitha, 2013) y (Prajapati, Patel, Nacwan, Kachhiya, & Shah, 2014) analizan los sistemas criptográficos AES y RSA desde el punto de vista de la seguridad, rendimiento en escenarios específicos sin embargo no se analizan implementaciones con lenguajes de programación que se encuentran en el mercado, donde se pueda observar las ventajas y desventajas de cada plataforma, para la utilización en otros escenarios.

El presente trabajo busca realizar un análisis comparativo del rendimiento en la transferencia de archivos entre los lenguajes de programación Java y.NET utilizando un protocolo de cifrado híbrido con los algoritmos AES y RSA aplicado a un sistema de respaldos de información para la Oficina Técnica Loja de la Coordinación Zonal 7 de Registro Civil Identificación y Cedulación. El método utilizado es una secuencia de pasos que empieza por la Identificación de algoritmos criptográficos propuestos pasando luego a la Definición de los índices de medición, Desarrollo de los prototipos de sistemas de respaldos de información, Definición del control de la simulación, Pruebas y validación, Ejecución de la simulación, para terminar con una Interpretación de los datos obtenidos.

Entre los resultados más destacables se obtuvo que en cuanto al indicador de tiempo de respuesta existe una diferencia significativa donde C# ofrece mejores prestaciones, mientras que, para el consumo de recursos, tanto para el uso de CPU y RAM también existe una diferencia significativa donde Java es la mejor opción, por último, se detallan las conclusiones de la investigación, así como la propuesta del trabajo a futuro.

2. Método

El presente trabajo tiene un enfoque cuantitativo ya que se procedió con la recolección de los datos relacionados con las métricas del rendimiento, como son propuestos por (Medina, 2014) así como en el modelo FURPS (Pressman, 2002), para realizar las pruebas de rendimiento, un criterio de evaluación se basa en métricas externas e internas, entre las métricas externas se establece indicadores como: tiempo medio de respuesta, mientras que en las métricas externas aparecen el uso de recursos como CPU y RAM.

Para lograr el objetivo planteado en la investigación, se estableció las siguientes actividades basadas en la propuesta de (Layana Molina, 2017):

2.1. Identificación de algoritmos criptográficos propuestos

En esta parte se realizó una revisión bibliográfica de las características e implementación de cada algoritmo criptográfico que interviene en el proceso de cifrado utilizando un protocolo híbrido. Para el presente caso los algoritmos son AES y RSA, los mismos que en la actualidad se han convertido en estándares de cifrado.

2.2. Definición de los índices de medición

En este apartado se estableció el problema central el cual es la necesidad de un adecuado rendimiento en los sistemas informáticos que implementen protocolos criptográficos como en el presente caso es un sistema de respaldos de información. Al final se establecieron los siguientes indicadores: Tiempo de Respuesta, porcentaje de uso de CPU, porcentaje de uso de RAM.

2.3. Desarrollo de los prototipos de sistemas de respaldos de información

En esta actividad se desarrollaron dos prototipos uno utilizando el lenguaje de programación Java y otro con.NET cada escenario con su respectiva plataforma. Ambos prototipos contaban con dos componentes Servidor y Cliente los cuales realizaban la transferencia de archivos a respaldar, utilizando el protocolo híbrido con los algoritmos AES y RSA. Para la comunicación de los dos componentes se lo realizó con programación en sockets y la incorporación de hilos (programación multithreading) para un servicio con concurrencia.

El escenario se conformó de dos máquinas una con el rol de cliente y otra de servidor comunicados a través de una red LAN.

Las características del hardware utilizado en los escenarios implementados, se detalla en la tabla 1.

Tabla 1: Características hardware.

Característica	Cliente	Servidor
Marca	Dell	Clon
Procesador	Intel i7	Intel Core 2
RAM	8GB	2GB
Disco Duro	2TB	500GB
Internet	10MB	10MB

El software utilizado en los escenarios se detalla en la tabla 2.

2.4. Definición del control de la simulación

Para la simulación se estableció la unidad de análisis que se espera cuando el sistema entre en producción, donde el total de transacciones se espera de 110, tomando en

Tabla 2: Características software.

Característica	Java	C#
Sistema Operativo	Win10	Win7
Framework	Netbeans 8.2	Visual Studio 2015
Plataforma	Java Se 8	.NET Framework 4.6.1
Librerías Criptográficas	JCE	Bouncy Castle
Librerías Sockets	Java.Net	System.Net

cuenta dos respaldos al día para 55 empleados de la Oficina Técnica Loja de la Coordinación Zonal 7 de Registro Civil Identificación y Cedulación. Sin embargo, para mejorar la visibilidad de los resultados se estableció un número mayor al encontrado anteriormente, siendo un total de 335 archivos de los cuales, de acuerdo con las estadísticas analizadas acerca de la cantidad de archivos respaldados en la institución, se escogió 200 archivos livianos, 100 medianos y 35 pesados como se observa en la tabla 3.

Tabla 3: Cantidad de archivos por tamaño.

	Livianos (1MB –2MB)	Medianos (100MB – 200MB)	Pesados (1GB – 2GB)	Total
Cantidad	200	100	35	335
Porcentaje	60%	30%	10%	100%

2.5. Pruebas y validación

Se realizaron las pruebas estableciendo una carpeta con los archivos a transferir en tres momentos para cada grupo de archivos, así también ambos prototipos tanto servidor como cliente fueron instalados en las mismas máquinas, llevándose a cabo sin inconvenientes con una eficacia del 100% (o errores) tanto para Java como para C#.

Para las pruebas de consumo de recurso como son CPU y RAM se utilizaron las herramientas de análisis de rendimiento: para Java se usó VisualVM mientras que para C# se utilizó las Herramientas de diagnóstico en Visual Studio.

2.6. Ejecución de la simulación

En la ejecución de las pruebas se agregaron líneas de código a los prototipos para tomar los tiempos al completar cada tarea del proceso de transferencia de archivos, los cuales se detallan a continuación:

- Inicio del proceso

- Cifrado el archivo
- Cifrado de la clave simétrica
- Envío de la clave simétrica
- Envío del archivo cifrado
- Descifrado de la clave simétrica
- Descifrado del criptograma.

2.7. Interpretación

Se obtuvieron los resultados de los cuales se consolidó los mismos para establecer la comparación del rendimiento entre los dos lenguajes de programación, para esto se utilizó métodos estadísticos para verificar la normalidad de datos y análisis no paramétricos como los de Mann-Whiney de las cuales se obtuvieron las conclusiones correspondientes.

3. Resultados

En este apartado se presentan los resultados obtenidos de las pruebas realizadas en cada escenario con un total de 335 archivos transmitidos con diferentes pesos. Las pruebas están separadas de acuerdo con los indicadores: Tiempo de Respuesta, Uso de Memoria RAM y Uso de CPU.

3.1. Tiempo de Respuesta

El indicador tiempo de respuesta representa el tiempo que transcurre desde que inicia una solicitud de transferencia de archivo hasta que el mismo llega a su destino. Este análisis se lo ha dividido en tres partes en primera instancia se analizará los datos por las fases del proceso de transmisión, posterior se analizará el comportamiento por el tamaño de archivo y al final se realizará un análisis estadístico para verificar si existe o no diferencia significativa en cada uno de los indicadores.

3.1.1. Análisis del tiempo de respuesta por fases del proceso

Para este análisis se han escogido las fases más críticas del proceso de transmisión del archivo como son: Cifrado del archivo, Transmisión del criptograma y Descifrado del criptograma

En la tabla 4 se muestran los datos obtenidos por cada fase crítica del proceso.

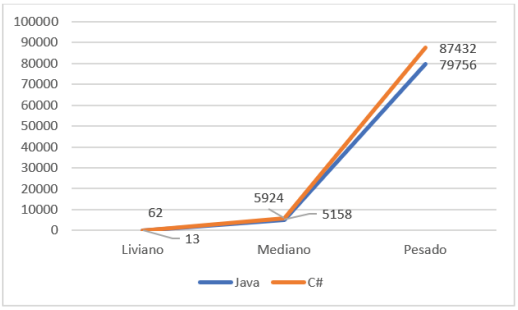
Tabla 4: Media de Tiempos por fases.

	JAVA			C#		
	Livianos (ms)	**Medianos (ms)**	**Pesados (ms)**	**Livianos (ms)**	**Medianos (ms)**	**Pesados (ms)**
Cifrado de criptograma	62	5158	79756	13	5924	87432
Envió de criptograma	194	20697	293931	132	18103	296994
Descifrado de criptograma	96	7890	143059	73	2843	52936

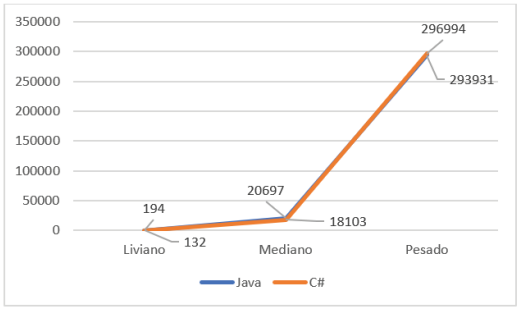
En la Figura 1(a) se observa el comportamiento del tiempo de respuesta para la fase de cifrado del archivo, donde se utilizó la clave pública del servidor para obtener el criptograma resultante, como se evidencia en las pruebas con los tres tipos de archivos C# obtiene un mejor tiempo de respuesta al necesitar menor cantidad de milisegundos para completar el trabajo sin embargo la diferencia no es considerable. En la Figura 1(b) se observa el comportamiento del tiempo de respuesta para la fase de transmisión del archivo, en esta fase se utilizó la programación sockets y multithreading para la atención de peticiones en paralelo, como se evidencia en las pruebas con los tres tipos de archivos C# obtiene un mejor tiempo de respuesta al necesitar menor cantidad de milisegundos para completar el trabajo, sin embargo, la diferencia es menor incluso que en la anterior prueba. En la Figura 1(c) se observa el comportamiento del tiempo de respuesta para la fase de descifrado del archivo, donde se utilizó la clave privada del Servidor para obtener el archivo en texto plano, como se evidencia en las pruebas con los tres tipos de archivos, C# obtiene un mejor tiempo de respuesta al necesitar menor cantidad de milisegundos para completar el trabajo, donde se observa que a medida que cree el archivo la diferencia que existe entre los dos lenguajes es mayor.

3.1.2. Análisis de frecuencias

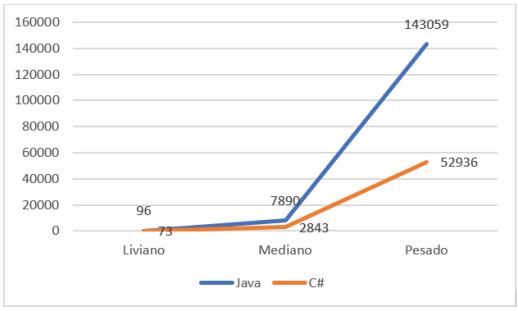
En la tabla 5 se observa el análisis de frecuencias del tiempo de respuesta para el envío de un archivo, cuya información se ha segmentado por tamaño de archivo.



a. Tiempo de Cifrado



b. Tiempo de transmisión del Criptograma



c. Tiempo de descifrado

FIGURA 1: Análisis del tiempo de respuesta por fases del proceso.

TABLA 5: Análisis de frecuencias del tiempo de respuesta.

	JAVA			C#		
	Livianos (ms)	**Medianos (ms)**	**Pesados (ms)**	**Livianos (ms)**	**Medianos (ms)**	**Pesados (ms)**
Promedio	594	38347	630539	255	25749	323234
Mediana	469	26917	611613	236	25141	330202
Desviación estandar	492	57525	170248	87	7334	51684

Con los datos de la tabla 5 se procede con el cálculo de la variación porcentual utilizando la fórmula 1.

$$\text{Valor Porcentual} = (V1 - V2)/V1 * 100. \quad (1)$$

Donde: V1 es el valor mayor y V2 es el segundo valor a comparar.

En la Figura 2 se observa el resultado del cálculo de la variación porcentual que existe entre cada resultado obtenido por tamaño de archivo en los tres casos a favor de C#, denotados por las barras de color naranja.

3.1.3. Pruebas no paramétricas

Utilizando el algoritmo de Kolmogorov-Smirnov, con un nivel de significancia de o para ambos conjuntos de datos, se puede constatar que los mismos no son normales con

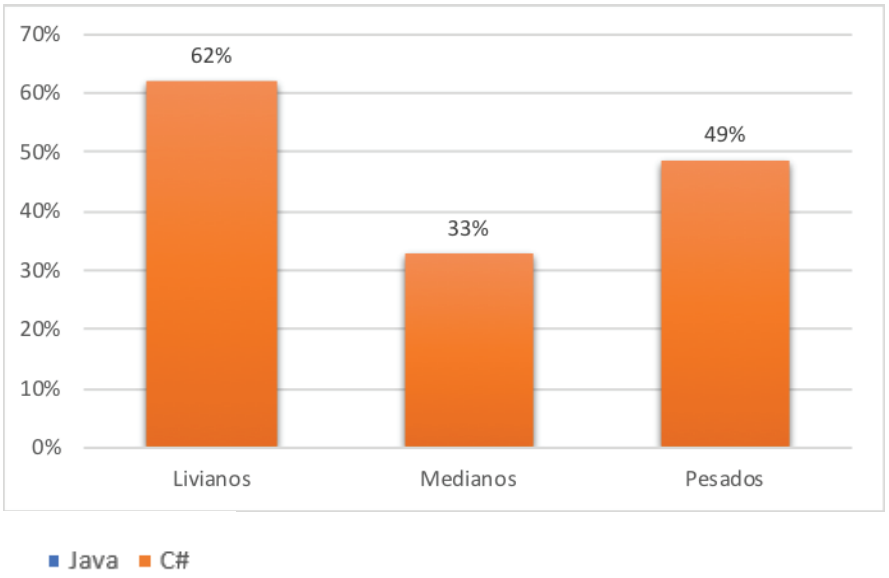


Figura 2: Variación Porcentual del tiempo de respuesta.

un error del 5% y 335 muestras con cada lenguaje de programación, por lo tanto, se procede con la prueba no paramétrica. La tabla 6 muestra la prueba de Mann-Whiney, donde preliminarmente mediante observación se constata que existe diferencia significativa.

Tabla 6: Rangos de prueba no paramétrica con el algoritmo Mann-Whiney para el Tiempo de Respuesta.

	Lenguaje de Programación	N	Rango promedio	Suma de rangos
Tiempo de Respuesta	Java	335	391,20	131050,50
	C#	335	279,80	93734,50
	Total	670		

Mediante la prueba de contraste detallado en la tabla 7, se corrobora el resultado obtenido en el análisis de rangos, con un nivel de significancia asintótica (bilateral) que es 0 (menor que 0.05) se concluye que existe una diferencia significativa a favor de C# ya que el promedio de tiempo de respuesta utilizado para la transmisión de los 335 archivos es menor a la utilizada por Java.

Tabla 7: Pruebas de contraste con el algoritmo Mann–Whitney para el Tiempo de Respuesta.

	Tiempo de Respuesta
U de Mann-Whitney	37454,500
W de Wilcoxon	93734,500
Z	-7,448
Sig. asintótica (bilateral)	0
a. Variable de agrupación: Lenguaje de Programación	

3.2. Uso de Memoria RAM

El indicador uso de RAM corresponde a la medición en intervalos de tiempo mientras el proceso de transferencia de archivos se lleva a cabo.

3.2.1. Análisis de frecuencias

En la tabla 8 se observa el análisis de frecuencias del uso de RAM para el envío de un archivo, la información se ha segmentado por el tamaño de archivo.

Tabla 8: Análisis de frecuencias del Uso de RAM.

	JAVA			C#		
	Livianos (MB)	**Medianos (MB)**	**Pesados (MB)**	**Livianos (MB)**	**Medianos (MB)**	**Pesados (MB)**
Promedio	16	17	20	21	19	18
Mediana	4	16	22	21	19	18
Desviación estandar	9	7	7	1	1	1

En la Figura 3 se observa la variación porcentual que existe entre cada resultado obtenido por tipo de archivo, como se puede visualizar en los dos primeros casos a favor de Java, denotados por las barras de color azul.

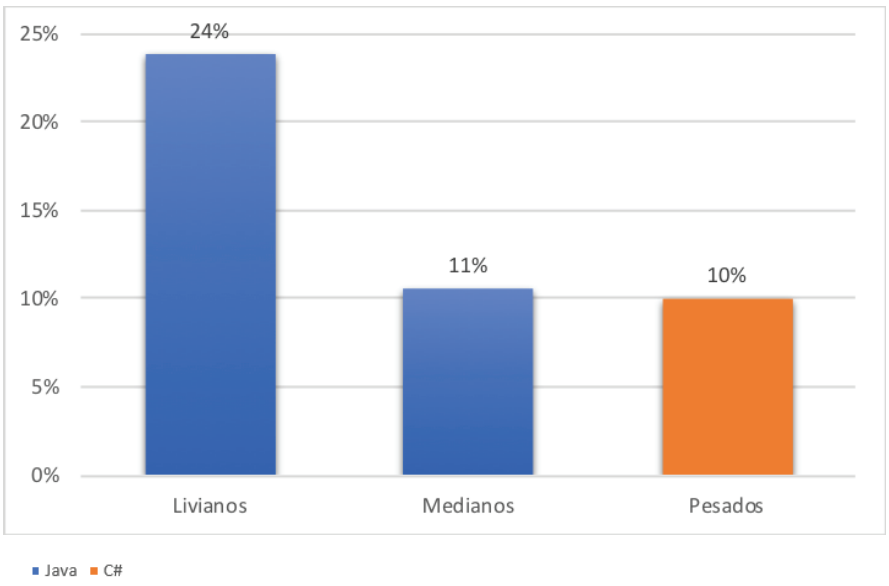


Figura 3: Variación porcentual del uso de RAM.

3.2.2. Pruebas no paramétricas

Utilizando el algoritmo de Kolmogorov–Smirnov se puede constatar con un nivel de significancia de 0,028 y 0 para los conjuntos de datos de Java y C# respectivamente que los mismos no son normales con un error del 5% y 150 muestras, por lo tanto, se procede con la prueba no paramétrica. La tabla 9 muestra la prueba de Mann-Whiney, donde preliminarmente mediante observación se constata que existe diferencia significativa.

Tabla 9: Rangos de prueba no paramétrica con el algoritmo Mann-Whiney para el uso de memoria RAM.

	Lenguaje de Programación	N	Rango promedio	Suma de rangos
Uso de RAM	Java	150	139,38	20906,50
	C#	150	161,62	24243,50
	Total	300		

Mediante la prueba de contraste detallado en la tabla 10, se corrobora el resultado del nivel de significancia asintótica (bilateral) que es 0,026 (menor que 0.05) por lo tanto se concluye que existe una diferencia significativa a favor de Java ya que el promedio de uso de RAM utilizado para la transmisión de los 335 archivos es menor que la que utiliza C#.

Tabla 10: Pruebas de contraste con el algoritmo Mann–Whitney para el uso de memoria RAM.

	Uso de RAM
U de Mann-Whitney	9581,500
W de Wilcoxon	20906,500
Z	-2,222
Sig. asintótica (bilateral)	,026
a. Variable de agrupación: Lenguaje de Programación	

3.3. Uso de CPU

El indicador uso de CPU corresponde a la medición en intervalos de tiempo mientras el proceso de transferencia de archivos se lleva a cabo.

3.3.1. Análisis de frecuencias

En la tabla 11 se observa el análisis de frecuencias del uso de RAM para el envío de un archivo, la información se ha segmentado por el tamaño de archivo.

Tabla 11: Análisis de frecuencias.

	JAVA			C#		
	Livianos (%)	**Medianos (%)**	**Pesados (%)**	**Livianos (%)**	**Medianos (%)**	**Pesados (%)**
Promedio	1	6	5	6	9	7
Mediana	1	3	2	5	8	4
Desviación estandar	1	6	5	3	5	5

En la Figura 4 se observa la variación porcentual que existe entre cada resultado obtenido por tamaño de archivo, como se observa en los tres casos Java obtiene un mejor uso del CPU, denotados por las barras de color azul.

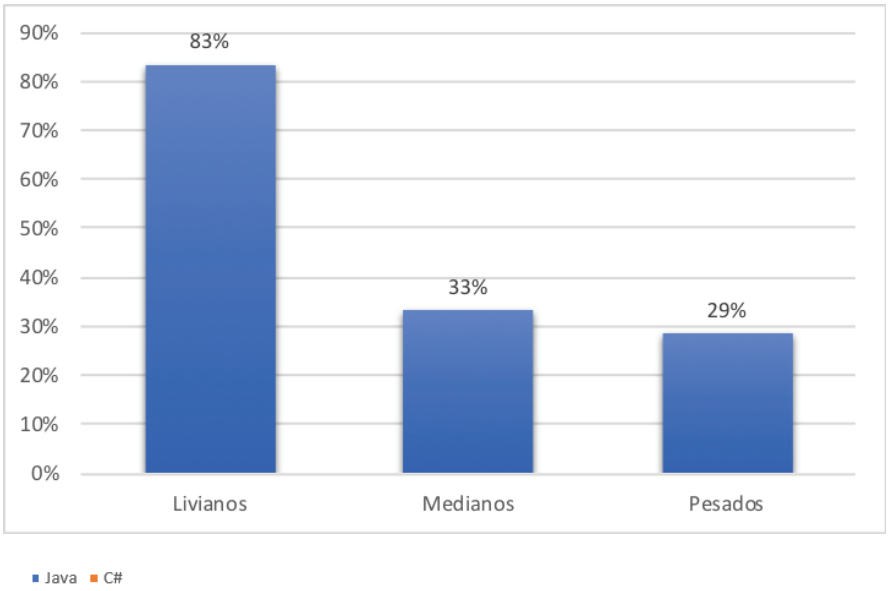


Figura 4: Variación porcentual del uso de CPU.

3.3.2. Pruebas no paramétricas

Utilizando el algoritmo de Kolmogorov–Smirnov se puede constatar con un nivel de significancia de o para ambos conjuntos de datos, que los mismos no son normales con un error del 5% y 150 muestras, por lo tanto, se procede con la prueba no paramétrica. La tabla 12 muestra la prueba de Mann-Whitney, donde preliminarmente mediante observación se evidencia una diferencia significativa.

Mediante la prueba de contraste detallado en la tabla 13, se corrobora el resultado del nivel de significancia asintótica (bilateral) que es o por lo tanto se concluye que existe una diferencia significativa a favor de Java ya que el promedio de uso de CPU utilizado para la transmisión de los archivos es menor que la utilizada por C#.

Tabla 12: Rangos de prueba no paramétrica con el algoritmo Mann-Whitney para el uso del CPU.

	Lenguaje de Programación	N	Rango promedio	Suma de rangos
Uso de CPU	Java	150	106,14	15921,00
	C#	150	194,86	29229,00
	Total	300		

Tabla 13: Pruebas de contraste con el algoritmo Mann-Whiney para el uso del CPU.

	Uso de CPU
U de Mann–Whitney	4596,000
W de Wilcoxon	15921,000
Z	–8,903
Sig. asintótica (bilateral)	0.000
a. Variable de agrupación: Lenguaje de Programación.	

4. Conclusiones y Trabajo Futuro

Una vez realizada las mediciones de cada indicador se obtuvieron los siguientes resultados: para el tiempo de respuesta C# es un 39,8% mejor que Java, para el uso de RAM Java es un 15,9% mejor que C#, así mismo en cuanto al indicador de uso de CPU Java es un 83,5% mejor que C#.

El decidir por uno u otro lenguaje de programación para el desarrollo del sistema informático como es el caso del presente trabajo, donde se aplicó en un sistema de respaldos, depende de la necesidad o prioridad que se le establezca a cada indicador por ejemplo, una empresa que cuente con una buena infraestructura no priorizará el consumo de recursos mientras que a una empresa nueva sin un mayor presupuesto si se espera que asigne un mayor peso al indicador de consumo de recursos (CPU, RAM).

La seguridad informática en la actualidad y en lo posterior será un área crítica de toda organización ya que la infraestructura tecnológica ahora está conectada a la Internet, por lo cual todo trabajo en este campo siempre permitirá blindar en lo más posible los activos informáticos. Por otro lado, llevar a cabo estas medidas de seguridad repercute en el rendimiento de las aplicaciones, es por ello la importancia de estudios como el que se ha realizado que permite seleccionar la mejor alternativa de acuerdo con el escenario en el que se desee implantar la solución.

Los sistemas informáticos han cambiado y seguirán cambiando la manera en la que el ser humano realiza sus actividades, con la aparición de las tecnologías emergentes como es el caso de la Internet de las Cosas cada vez más dispositivos inteligentes se conectarán a la Internet, lo cual requiere de la aplicación de medidas de seguridad que

disminuyan las probabilidades de sufrir un ataque informático todo esto sin mermar el correcto funcionamiento con un adecuado rendimiento, por lo tanto es recomendado realizar un análisis como el realizado en el presente trabajo en el que involucre dispositivos de placa computadora como Raspberry o Arduino.

Bibliografía

[1] Aguirre, J. R. (2006). Libro Electrónico de Seguridad Informática y Criptografía Versión 4.1. Madrid: UPM.

[2] Al Hasib, A., & Mahmudul Haque, A. (2008). A Comparative Study of the Performance and Security Issues of AES and RSA Cryptography. IEEE, 505-510.

[3] Al Hasib, A., & Mahmudul, A. (2008). A Comparative Study of the Performance and Security Issues of AES and RSA Cryptography. IEEE Computer Society, 505-510.

[4] bouncycastle. (30 de Ago de 2018). https://www.bouncycastle.org/. Obtenido de https://www.bouncycastle.org/

[5] Díaz, J. C. (1995). Criptografía: historia de la escritura cifrada. Complutense.

[6] García Belmonte, R. (2016). Firma Digital Basada en Funciones HASH y un Algoritmo Criptográfico Híbrido. CIDETEC, pp. 19.

[7] Gómez, Á. (2011). Enciclopedia de la Seguridad Informática. México: Alfaomega.

[8] Layana Molina, S. (2017). Estudio Comparativo de la Eficiencia de los Algoritmos Criptográficos AES y RSA: Caso de Estudio de una Institución en la Ciudad de Guayaquil. UEES.

[9] Mahajan, P., & Sachdeva, A. (2013). A Study of Encryption Algorithms AES, DES and RSA for Security. Global Journal of Computer Science and Technology Network Web & Security.

[10] Medina, J. (2014). Pruebas de Rendimiento TIC. Murcia: SG6. Obtenido de https://archive.org/details/PruebasDeRendimientoTic

[11] Menezes, A., Ooschot, P., & Vanstone, S. (1996). Handbook of Applied Cryptography. CRC Press, Inc.

[12] Padmavathi, B., & Ranjitha, K. (2013). A Survey on Performance Analysis of DES, AES and RSA Algorithm along with LSB Substitution Technique. International Journal of Science and Research (IJSR), pp. 170-174.

[13] Paredes, G. G. (2006). Introducción a la Criptografía. RU TIC, 55.

[14] Patil, P., Narayankar, P., Narayan, D., & Meena, S. (2016). A Comprehensive Evaluation of Cryptographic Algorithms: DES, 3DES, AES, RSA and Blowfish. ElSevier, pp. 617-624.

[15] Prajapati, P., Patel, N., Nacwan, R., Kachhiya, N., & Shah, P. (2014). Comparative Analysis of DES, AES, RSA Encryption Algorithms. SemanticSholar.

[16] Pressman, R. (2002). Ingeniería de Sofware: Un Enfoque Práctico (Quita Edición ed.). McGraw Hill.

[17] Ramakrishnan, K., Amnazjagam, M., Rajan, R., & Karthik, S. (2015). USA Patente n° US-8972940-B2.

[18] Soriano, M. (s.f.). Seguridad en redes y seguridad en la información. Obtenido de http://improvet.cvut.cz/project/download/C2ES/Seguridad_de_Red_e_Informacion.pdf

[19] TIOBE. (02 de 02 de 2018). www.tiobe.com. Obtenido de https://www.tiobe.com/tiobe-index/

[20] Vallejo, M. C. (24 de Nov de 2005). Firma Electrónica. Obtenido de https://www.derechoecuador.com/firma-electroacutenica

[21] Vidal-Silva, C., Madariaga, E., & Solis, R. (2017). Estudio Piloto de la Importancia del Rendimiento, Seguridad y Fiabilidad en el Proceso de Desarrollo de Software en Chile. Scielo, pp. 95-106. doi:http://dx.doi.org/10.4067/S0718-07642017000300011.