

**Universidad de Mendoza
Facultad de Ingeniería**



MAESTRÍA EN TELEINFORMÁTICA

**“Análisis y comparación de algoritmos
de encriptado y verificación aplicados a
una arquitectura de Fog Computing”**

Ing. Layla M. Martínez Guevara

Director: Dr. Ing. Emmanuel N. Millán

CoDirector: Dr. Ing. Pablo Javier Vidal

San Luis, Argentina – 2023

Dedicatoria

*A mi abuela que siempre
me alentó a estudiar
y se llevó al cielo
la ilusión de verme
finalizar la maestría.*

Agradecimientos

Quiero expresar mi más sincero agradecimiento a mi director, Dr. Ing. Emmanuel Nicolas Millán, por su confianza puesta en mí, su dedicación, y su colaboración para conseguir el equipamiento necesario para realizar este proyecto. Sus conocimientos transmitidos y su ayuda han sido invaluable para mí.

A mi subdirector, Dr. Ing. Pablo Javier Vidal, por sus aportes, sus valiosos consejos y el tiempo dedicado.

Al director de Posgrado de la Facultad de Ingeniería de la Universidad de Mendoza, Mg. Ing. Osvaldo Marianetti, por su predisposición y apoyo para poder llevar a cabo la tesis.

A la empresa Trans Industrias Electrónicas S.A. que me dio la oportunidad de acceder a la maestría.

A mi pareja, por demostrarme que en todo momento cuento con él.

A mis compañeros de trabajo y a todos los que de alguna u otra manera me apoyaron en la realización de este proyecto.

Resumen

Debido a la creciente adopción de Internet de las Cosas (IoT) se ha incrementado exponencialmente el tráfico de los datos que llegan a los servidores en la nube para su análisis afectando la latencia de los sistemas de IoT. Esto puede resultar inaceptable en entornos críticos, principalmente si se deben tomar decisiones en tiempo real. Como solución surge una nueva infraestructura en capas, que introduce los conceptos de Edge y Fog Computing, donde ésta última acerca las capacidades de análisis y procesamiento al lugar en el que se generan los datos, disminuyendo el volumen de información que debe enviarse a la nube y mejorando los tiempos de respuesta, pero intensificando otros problemas, entre ellos, la de seguridad de los datos.

La seguridad es una de las principales preocupaciones en Fog Computing, ya que hay muchos datos confidenciales y al estar disponibles más cerca del usuario se vuelven más vulnerables. Además, los recursos informáticos de los nodos son relativamente pequeños, lo que hace difícil ejecutar un conjunto completo de soluciones de seguridad sin aumentar la latencia o los requerimientos de procesamiento.

En base a eso, en este trabajo de investigación se propone estudiar y comparar diversos algoritmos de encriptación y verificación de datos utilizando sets de datos de diferentes tipos y tamaños, y un software de criptografía como OpenSSL. Las pruebas se realizan sobre Raspberry Pi, que por sus funciones permite implementar un nodo Fog. Mediante la utilización de scripts se evalúan diferentes métricas sobre cada nodo, tales como consumo de CPU, de memoria y energía. Los valores son registrados y tabulados para su procesamiento.

A partir del análisis de los resultados obtenidos se observa que el tipo de datos no tiene incidencia en la elección del algoritmo a utilizar y en base a las métricas estudiadas se pudieron detectar patrones de comportamiento relacionados con el tamaño de los datos que pueden contribuir en la definición de la estrategia de seguridad a implementar.

Índice General

Resumen	1
Índice General	2
I. Introducción	4
Planteamiento del problema	4
Justificación del estudio	7
Objetivos	8
Hipótesis	9
Contexto general de la tesis.....	9
Estructura del documento	9
II. Marco teórico y antecedentes.....	11
Seguridad en la Capa Fog	15
Criptografía.....	17
Trabajos Relacionados	33
III. Metodología.....	47
Primera Etapa	48
Librería de funciones criptográficas.....	48
Algoritmos criptográficos	49
Segunda Etapa	51
Configuración de los Nodos Fog	51
Tercera Etapa	56
Set de datos.....	56
Métricas	58
Scripts.....	59
Cuarta Etapa.....	61
IV. Resultados	63
Nodo 1 – Raspberry Pi 3.....	63
Porcentaje de Uso de Memoria	63

Uso de Memoria Externa.....	69
Porcentaje de Consumo de CPU	83
Tiempo de Ejecución	93
Consumo de Energía.....	97
Síntesis Resultados Nodo 1.....	98
Comparación Encriptación y Desencriptación	99
Comparación Suma De Verificación.....	101
Nodo 2 – Raspberry Pi 1.....	102
Porcentaje de Uso de Memoria	102
Porcentaje de Consumo de CPU	103
Tiempo de Ejecución	106
Síntesis Resultados Nodo 2.....	108
Nodo 3 – Raspberry Pi 4.....	110
Porcentaje de Uso de Memoria	110
Porcentaje de Memoria Externa.....	114
Porcentaje de Consumo de CPU	118
Tiempo de Ejecución	121
Síntesis Resultados Nodo 3.....	124
V. Conclusiones	128
VI. Bibliografía	130
VII. Anexos	133
Scripts	133
Resultados	139
Nodo 1 – Raspberry Pi 3	139
Nodo 3 – Raspberry PI 4.....	152

I. Introducción

Planteamiento del problema

En los últimos años, el uso de internet se ha extendido a todas partes y no se restringe solamente a computadoras y teléfonos inteligentes, sino que cada vez son más las cosas que nos rodean que se conectan a internet, como electrodomésticos, luces, vehículos, semáforos, entre otros dispositivos. El internet actual está migrando rápidamente a *Internet de las Cosas*, (*IoT*, *Internet of Things*) donde los objetos cotidianos se comunican a través de Internet o directamente con otros objetos, permitiendo que se pueda capturar información clave sobre el uso y el rendimiento de los dispositivos para detectar patrones, hacer recomendaciones, mejorar la eficiencia y/o actuar sobre los datos recibidos [1].

Los dispositivos inteligentes tienen generalmente muchas limitaciones relacionadas con los niveles de energía, la cantidad de memoria de almacenamiento y la capacidad de procesamiento. Para solucionar estas últimas dos limitaciones se han creado modelos de computación en la nube que actualmente se utilizan en muchas empresas para numerosos sistemas de IoT. Básicamente, la tecnología de almacenamiento y recuperación de datos desde una ubicación externa se conoce como *Cloud Computing* (*Computación en la nube*) [2]. La computación en la nube se ha convertido en un paradigma para gestionar y prestar servicios bajo demanda a través de Internet. En la computación en la nube, los proveedores de la nube (como Google y Amazon) otorgan a los usuarios recursos para su infraestructura, plataformas y software a partir de un conjunto compartido de recursos por una tarifa. Los usuarios pueden acceder a aplicaciones y datos desde una nube en cualquier lugar y en cualquier momento y, además, les permite escalar sus recursos basados en la demanda. Sin embargo, la mayoría de los centros de datos en la nube están geográficamente centralizados y ubicados en sitios remotos, lejos de la proximidad de los usuarios finales. Hay que tener en cuenta que el despliegue masivo de sensores como parte del IoT, el incremento de transmisiones de video 4K, la realidad virtual aumentada y los avances en otras tecnologías están provocando un incremento en el tráfico procedente de internet que llega a los centros de datos [3]. Si bien la nube nos brinda acceso a cómputo, almacenamiento y conectividad de manera

fácil y rentable, un aumento de tráfico en la red puede provocar que estos recursos centralizados creen retrasos y problemas de rendimiento cuando la fuente que genera los datos está lejos de la infraestructura que los procesa. Los datos recogidos por sensores y dispositivos de IoT se envían a centros de datos en la nube, donde se almacenan y se procesan para obtener información útil que permita tomar decisiones inteligentes. Retardos altos en las respuestas por parte de la nube pueden resultar inaceptables en entornos que necesitan tomar decisiones en tiempo real.

Otra de las limitaciones de Internet de las cosas es que requiere soporte de movilidad y conocimiento de la ubicación de los dispositivos. Por lo tanto, se requiere una nueva plataforma que cumpla con todas estas limitaciones. El nuevo paradigma de computación en la nube sugerido es la *Computación en la Niebla*, o más conocido por su nombre en inglés *Fog Computing*, el cual utilizaremos de ahora en adelante.

Fog Computing surgió debido a algunas limitaciones físicas asociadas con la tecnología de computación en la nube. Este modelo tiene como objetivo eliminar los datos erróneos, darles formato y realizar un pequeño procesamiento de los datos físicamente más cerca de donde se recopilaron para reducir la cantidad de información que se debe enviar a la nube. Esta tecnología se está adoptando en muchos campos debido a la evolución de Internet de las cosas que ha mejorado la conexión inalámbrica cuando los dispositivos físicos transmiten y reciben datos. Compañías como Google, Amazon y Microsoft han visto el potencial que tiene este cambio de paradigma sobre entornos de IoT y han empezado a adaptar el entorno de sus aplicaciones para llevar la capacidad de cómputo y almacenamientos a dispositivos cercanos al usuario [4].

Actualmente, la infraestructura de un sistema IoT está formada por tres capas típicas [5]:

- *Edge o Borde*. Es la capa inferior, formada por dispositivos que funcionan como nodos de una red, por lo general inalámbrica. Su objetivo es la interacción directa con el entorno de aplicación del sistema, ya sea obteniendo información a través de sensores o realizando acciones mediante actuadores.
- *Cloud o Nube*. Es la capa superior donde se encuentran servidores con capacidad computacional para realizar el procesamiento completo de los datos

recogidos por los dispositivos de Edge, y donde se alojan otro tipo de servicios o aplicaciones auxiliares, tales como, de interacción con los usuarios o configuración de los sistemas.

- *Fog o Niebla*. Es el nivel intermedio entre los dos anteriores y que generalmente sirve de mediador, recopilando la información desde el Edge, y si fuera necesario, adaptándola o procesándola para luego enviarla al Cloud.

Debido al uso extendido de los términos, de ahora en más se nombrará a las capas por su nombre en inglés.

Se puede pensar la aplicación de estos sistemas en una planta industrial, donde los dispositivos de Edge serían los sensores desplegados por la planta captando la información sobre el rendimiento de la maquinaria, los cuales podrían enviar los datos a una computadora o servidor de control centralizado que hiciese las funciones de Fog. A su vez, dicho servidor central podría estar enviando los datos al Cloud, hacia servidores externos a la planta, donde se procesa la información para poder tomar decisiones sobre la producción en tiempo real [6].

Con esta estructura en mente, se puede visualizar otra problemática de los sistemas de IoT que es la seguridad. En la mayoría de las implementaciones de la capa de Edge se realiza un despliegue de nodos inalámbricos cubriendo una determinada superficie, sin acceso a ninguna fuente de alimentación fija y haciendo necesario el uso de sistemas de energía más volátiles, como por ejemplo, baterías. La necesidad de aumentar la autonomía individual de los nodos llevó a que el hardware empleado tuviera bajas prestaciones computacionales y de recursos en general, y se descartó el uso de algoritmos criptográficos y protocolos de seguridad debido al alto consumo energético que estos suponen.

Al principio, estas decisiones fueron consideradas como aceptables y óptimas. Realizar un espionaje de la información transmitida entre los nodos no tiene ninguna repercusión dañina, siempre que no sea información crítica, ya que un atacante no puede extraer valor de, por ejemplo, los datos sobre velocidad de una determinada máquina. Por lo tanto, el ahorro de energía obtenido al no implementar medidas de seguridad encontraba su justificación bajo la premisa de no requerir de dicha protección.

Sin embargo, con el paso del tiempo, los sistemas de IoT contruidos a partir de estos diseños comenzaron a ser el objetivo de numerosos ataques. La falta de algoritmos de autenticación permitía que un atacante pudiera hacerse pasar por un integrante de la red y, aunque los datos de los sensores carecían de importancia, esta posición, les permitía escalar privilegios hacia las capas superiores de la infraestructura, alcanzando los servidores del cloud donde sí se puede encontrar información sensible (información personal de los usuarios, contraseñas, etc.).

Fog Computing tiene como ventaja el poder actuar como proxy de los dispositivos con recursos limitados, para actualizar el software o las credenciales de seguridad. Su paradigma se basa en segmentar el tráfico, lo que permite resolver muchos problemas de seguridad, gracias a firewalls adicionales en la red. También la alternativa de procesar los datos confidenciales de forma local, en lugar de en la nube, hace posible prevenir la violación de datos confidenciales.

Justificación del estudio

Como respuesta al gran problema generado por la falta de seguridad en las capas más bajas de los sistemas de IoT, numerosas investigaciones se pusieron en marcha con el fin de encontrar soluciones válidas. Se afirma que los algoritmos de cifrado o encriptación son primordiales para la seguridad. Estos algoritmos cifran la información, es decir, pasan un dato legible a un formato ilegible llamado datos cifrados. Para pasarlo a legible se realiza el descifrado; con esto se aumenta la seguridad y se mantiene la confidencialidad, integridad y disponibilidad (CIA) [4-3]. La mayoría de las investigaciones buscan modificar los protocolos para reducir el consumo energético, o acelerar por hardware los algoritmos criptográficos usuales. La primera opción busca cambiar los pasos a seguir para establecer comunicaciones seguras, reduciendo por ejemplo el número de mensajes intercambiados para completar un proceso de autenticación, mientras que la segunda opción aprovecha la eficiencia ganada al implementar una tarea en hardware al aumentar la potencia y disminuir el tiempo consumido, consiguiendo un consumo energético total menor. Los resultados de dichos desarrollos han dado lugar a múltiples casos de éxito, y se

continúa trabajando en las mencionadas líneas, que resultan realmente prometedoras y viables para proporcionar protección a productos basados en IoT y similares.

En el desarrollo de este trabajo se pretende realizar un aporte sobre la primera línea de investigación y definir estrategias algorítmicas de seguridad para la protección de la información en la capa de Fog Computing.

Objetivos

En este trabajo de tesis se espera cumplir con los siguientes objetivos:

- Implementar en una arquitectura de Fog Computing diversos algoritmos de encriptación de datos (variando parámetros de cada uno, como tamaño de clave, tipos y tamaños de datos a encriptar, entre otros) mediante la utilización de diferente hardware para evaluar métricas de rendimiento.
- Estudiar y evaluar el rendimiento de algoritmos de suma de verificación y su combinación con las técnicas de encriptación implementadas en el objetivo anterior.
- Realizar pruebas sobre hardware real, en condiciones aisladas (sin sobrecarga en el sistema) y pruebas de encriptación y desencriptación de datos con conjuntos de datos provenientes de dispositivos IoT sobre el nodo Fog:

En este trabajo se propone crear diferentes scripts en lenguaje Bash y utilizar software open source para realizar el cifrado/descifrado de datos y la suma de verificación de los mismos, que luego se podrán ejecutar en tres RPI de diferentes modelos, simulando de esta forma los nodos Fog de una plataforma de Fog Computing. Para simular los datos provenientes de los dispositivos IoT se van a utilizar sets de datos de sensores reales. A partir de las evaluaciones de cada ejecución, se va a comparar los distintos algoritmos para poder determinar cuál es la combinación de algoritmos que consume menos recursos de un nodo Fog.

Hipótesis

Para la realización de esta tesis se plantearon las siguientes hipótesis:

“Es posible encontrar un algoritmo de encriptación que utilice pocos recursos y sea eficiente en un entorno Fog computing”

" El tipo de datos utilizado influye en el rendimiento de los algoritmos”

“El tamaño de los datos impacta en forma directamente proporcional en el consumo de recursos de los nodos Fog.

Contexto general de la tesis

Este trabajo se orientó al área de IoT, ya que estas tecnologías hacen un fuerte uso de sensores que pueden comprometer aspectos como, el ancho de banda y la seguridad de los datos de una empresa, o de donde se desarrollan estos sistemas IoT. Por ello se implementa una capa de Fog Computing, la cual no expone los sensores a internet sino a una red local donde están conectados. Esta información se procesa y se cifra para que pueda viajar a la nube de manera segura.

En este proyecto se evalúa el rendimiento de diferentes algoritmos de cifrado sobre nodos Fog, implementados en distintos modelos de Raspberry Pi. Estas pruebas experimentales se desarrollaron sobre equipamiento facilitado durante la duración del trabajo por la Facultad de Ciencias Exactas y Naturales de la Universidad de Cuyo. Los tres nodos RPI utilizados fueron utilizados dentro del cluster Toko de dicha facultad. El equipamiento se encontraba conectado al cluster y la mayoría de las pruebas se pudieron realizar de forma remota".

Estructura del documento

Con la intención de ofrecer una visión general del presente trabajo, a continuación se detalla la estructura de capítulos del mismo:

- **I. Introducción.** Se hace mención sobre las motivaciones que han llevado al desarrollo de este trabajo de tesis. Posteriormente se presentan brevemente los objetivos del trabajo, hipótesis y contexto, para finalizar con la estructura del documento que ofrece al lector una mejor idea del esquema trabajado.
- **II. Estado del Arte.** Este capítulo comienza definiendo brevemente los conceptos bases en el que se desarrolla esta tesis. Se realiza un análisis teórico de la arquitectura de los sistemas de IoT, la relación que tienen con Cloud Computing, los problemas a los que se enfrentan y a los que Fog Computing intenta dar solución. También se hace una revisión de la literatura asociada a la criptografía. Luego se mencionan algunos trabajos actuales relacionados que sirvieron de apoyo y contraste a los resultados obtenidos.
- **III. Metodología.** Se presenta el esquema general de trabajo y se explican todas las tecnologías y herramientas a usar en el desarrollo del mismo. Se realiza la configuración de nodos Fog con capacidades mínimas para evaluar los distintos algoritmos de cifrado.
- **IV. Resultados.** Este capítulo se enfoca en los resultados obtenidos, haciendo una comparativa entre los algoritmos y dispositivos utilizados para la simulación de un nodo Fog.
- **V. Conclusiones.** En este capítulo se exponen las conclusiones del trabajo desarrollado, fundamentándose en los resultados del capítulo predecesor. Adicionalmente, se menciona trabajos a futuro para continuar con la investigación.
- **VI. Bibliografía.** Se detallan las referencias de la bibliografía consultada.
- **VII. Anexos.** En este último capítulo se dejan los scripts utilizados para las pruebas realizadas, y las tablas de resultados que no fueron incluidas en el capítulo IV.

II. Marco teórico y antecedentes

En el mundo actual, prácticamente todas y cada una de las organizaciones de las industrias, tanto de pequeña como gran escala, han estado confiando en la tecnología de computación *en la nube* para almacenar sus datos y utilizar los recursos que la misma ofrece según su requerimiento. El *Instituto Nacional de Estándares y Tecnología (NIST, National Institute of Standards and Technology)*, define *Cloud Computing*, término en inglés para “computación en la nube”, como “Un modelo tecnológico que permite el acceso ubicuo, adaptado y bajo demanda en red a un conjunto compartido de recursos de computación configurables compartidos (por ejemplo: redes, servidores, equipos de almacenamiento, aplicaciones y servicios), que pueden ser rápidamente provisionados y liberados con un esfuerzo de gestión reducido o interacción mínima con el proveedor del servicio.” [7]. Durante la última década, la computación en la nube ha sido la principal plataforma para almacenar y analizar enormes cantidades de datos debido a que tiene una doble propuesta de valor. En primer lugar, la centralización permite reducir costos de administración y operación de los sistemas. En segundo lugar, las organizaciones pueden evitar el gasto de capital de crear un centro de datos consumiendo recursos informáticos a través de Internet de un gran proveedor de servicios. De esta forma, la computación en la nube brinda muchos beneficios a las personas y organizaciones al ofrecer recursos informáticos altamente disponibles, flexibles, escalables y eficientes a un precio, permitiéndoles disminuir los costos sin dejar de ser competitivos.

La computación en la nube se ha infiltrado en una variedad de áreas, que incluyen atención médica, educación, bienes raíces, banca y manufactura. La misma creció de la mano de la utilización cada vez más masiva de dispositivos celulares, tablets, sensores, actuadores, entre otros, en nuestro día a día, proporcionando una gran potencia computacional, y almacenamiento para procesar, filtrar, agregar y almacenar la gran cantidad de datos producidos por los dispositivos. Sin embargo, con tecnologías emergentes como 5G e IoT, la computación en la nube tradicional se está volviendo ineficaz para manejar dificultades como límites de ancho de banda, falta de soporte de movilidad, seguridad y conocimiento de la ubicación, y el retardo en la comunicación en la red, es decir, la *latencia* variable.

La latencia, en términos informáticos la podemos definir como el tiempo que transcurre entre una orden y la respuesta que se produce a esa orden concreta, y depende de la velocidad de la conexión a Internet, del acceso a recursos compartidos entre las máquinas virtuales y la distancia entre el dispositivo que solicita o envía la información, *emisor*, y el nodo remoto, o *receptor*, que nos responde [8]. Muchos servicios en la nube están disponibles en las soluciones comerciales actuales, pero no son adecuados por la latencia, tales como las aplicaciones sensibles a la ubicación y portabilidad, como las relacionadas con *Internet de las Cosas (IoT, Internet of Things)*, *Informática Portátil (Weareable Computing)*, Redes Inteligentes, Vehículos Conectados y Redes Definidas por Software. Además, tales aplicaciones generan grandes volúmenes de datos a alta velocidad, y para cuando los datos lleguen a un sistema en la nube para su análisis, es posible que ya no exista la oportunidad de informar al dispositivo IoT para que tome medidas reactivas, por ejemplo, si se consideran los dispositivos IoT en el dominio médico, donde la latencia para actuar sobre los datos registrados por los sensores podría ser crítica para la vida. En el año 2012, Cisco introduce el concepto de “*Fog Computing*” que amplía y acerca la plataforma en la nube al dispositivo del usuario final para resolver los problemas mencionados [9].

Fog Computing, o “*computación en la niebla*”, es un nuevo paradigma que amplía el modelo de plataforma en la nube al proporcionar recursos informáticos en el borde de una red. Se puede describir como una plataforma similar a la nube que tiene datos, computación, almacenamiento y servicios de aplicaciones, pero es fundamentalmente diferente en el sentido de que está descentralizado. Además, los nodos Fog son capaces de procesar mayor cantidad de datos localmente que los dispositivos IoT, funcionan en las instalaciones, son totalmente portátiles y se pueden instalar en hardware heterogéneo. En pocas palabras, se lleva parte de la computación, el tratamiento de datos y toma de decisiones a un lugar más cercano del sitio donde se genera el dato [9]. Estas características hacen que la plataforma Fog sea muy adecuada para aplicaciones sensibles al tiempo y a la ubicación, pero también se intensifican muchos problemas, entre ellos problemas de seguridad relacionados con datos, virtualización, segregación, red, malware y monitorización.

El objetivo de Fog Computing, es acortar las vías de comunicación entre la nube y los dispositivos y reducir el caudal de datos en redes externas. Los nodos Fog

cumplirían así un papel de capa intermedia en la red, en la que se decide qué datos se procesan localmente y cuáles se envían a la nube o a un centro de datos para ser analizados o procesados.

En la Figura 1 se puede ver el modelo de la infraestructura de Fog Computing [5] donde se pueden observar tres capas claramente definidas:

- *La Capa en el Borde o Capa Edge:* comprende a todos los dispositivos inteligentes (dispositivos en el borde de la red) de una arquitectura de IoT, tales como sensores y actuadores. Los datos que se generan en esta capa se procesan en el mismo terminal, o se envían a un servidor (nodo Fog) de la capa en la niebla.
- *La Capa en la Niebla o Capa Fog:* está compuesta por una serie de servidores de mayor capacidad de procesamiento que los dispositivos de la capa en el borde que reciben los datos de estos, los preparan y los envían a la nube.
- *La Capa en la Nube o Capa Cloud:* incluye poderosas capacidades de almacenamiento y computación para análisis y almacenamiento permanente de grandes volúmenes de datos.

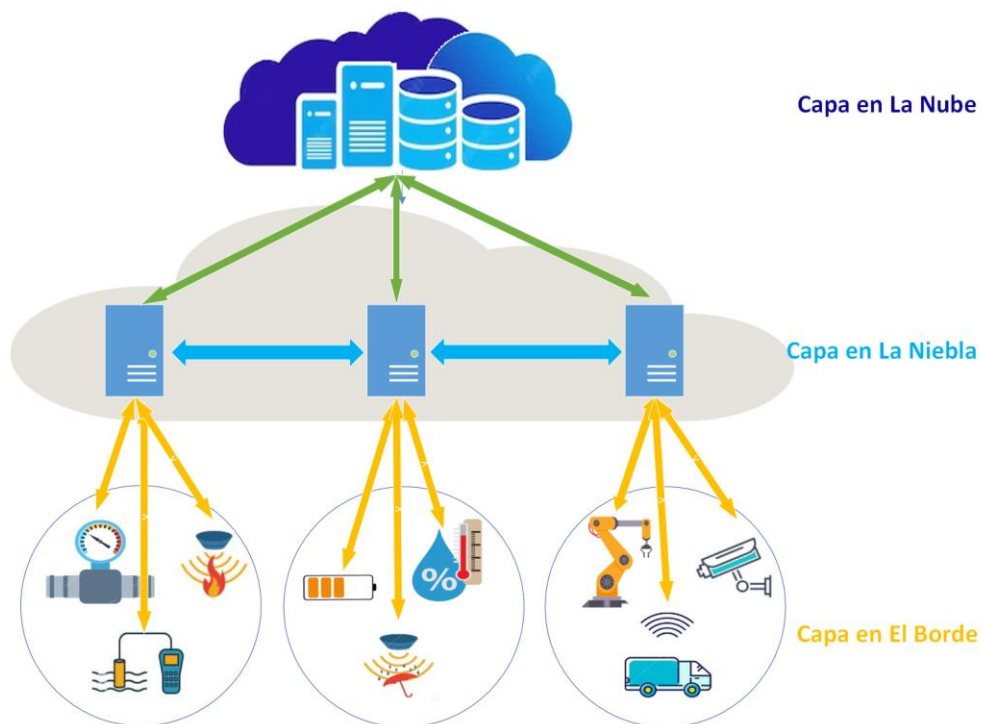


Figura 1– Infraestructura de Fog Computing

Para todas las aplicaciones vinculadas al mundo del IoT es fundamental que los dispositivos de borde de la arquitectura Fog adquieran conocimiento o «conciencia» de su entorno, mediante la colaboración entre ellos, para permitir un funcionamiento autónomo mientras un orquestador en un nivel superior mantiene la lógica de funcionamiento.

Sin embargo, el procesamiento descentralizado en el borde de la red también conlleva desventajas, que resultan sobre todo del trabajo que requiere mantener y administrar un sistema distribuido. Fog computing tiene sus propias limitaciones y tiene algunos desafíos y desventajas sin resolver, como [10] [11]:

- *Control y gestión de los recursos:* Los nodos Fog deben proporcionar los recursos necesarios para el cómputo, la red, el almacenamiento, la latencia y el ancho de banda en el borde de la red y también garantizar la calidad de servicio esperada. En algunos casos, la movilidad inherente asociada a la capa Fog hace que esas métricas cambien dinámicamente e incluso el entorno de la esta capa se virtualiza la mayor parte del tiempo, lo que provoca una latencia adicional, por lo tanto, controlar y administrar recursos es uno de los principales desafíos y problemas.
- *Dispositivos Fog de bajos recursos:* todos estos dispositivos tienen una potencia computacional más baja que la nube, en consecuencia, se necesita una asignación de trabajo eficiente.
- *Limitación de energía:* este aspecto siempre jugará en nuestra contra, si se quiere mayor procesamiento se necesitará más energía. La solución está en recopilar energía del entorno, tanta como sea posible: energía solar y eólica, mediante vibraciones, etc.
- *Conectividad:* la mayoría de los dispositivos Fog en el borde de la red utilizan tecnología inalámbrica para la conectividad. Los dispositivos de borde como los dispositivos de Internet de las cosas (IoT) pueden perder su conectividad con los nodos Fog al aumentar el número de dispositivos conectados a la capa Fog.

- *Heterogeneidad del dispositivo:* en el borde de la red, varios dispositivos son naturalmente heterogéneos. Por lo tanto, el entorno de Fog computing debe tener en cuenta la heterogeneidad al desarrollar aplicaciones para esta capa.
- *Cuestiones de seguridad:* las características de Fog computing, como su carácter distribuido y el hecho de estar más cerca de los usuarios, lo hace más vulnerable que la nube. Incluso, los mecanismos tradicionales de seguridad existentes para la nube no se pueden aplicar al sistema Fog, debido a su naturaleza distribuida y porque tiene un poder de cómputo más bajo que la nube. La seguridad y la privacidad de los datos son dos frenos importantes, tanto por su complejidad técnica como por sus consecuencias legales e impedimentos normativos.

Seguridad en la Capa Fog

La seguridad es una de las principales preocupaciones porque hay muchos datos confidenciales a nuestro alrededor. Podría ser cualquier detalle de precios de la empresa, o incluso podría ser un secreto nacional. Todos los datos deben estar protegidos y se debe asegurar que la arquitectura cuente con todos los métodos necesarios para prevenir un ataque.

En una infraestructura de Fog computing, la seguridad de extremo a extremo debe cubrir todo entre la nube y los dispositivos en el borde de la red. En la arquitectura, la seguridad comienza con el hardware del nodo Fog individual. Si el nodo no está diseñado con la seguridad adecuada para garantizar que es un elemento confiable, no es posible construir una capa Fog confiable de extremo a extremo. Los nodos Fog y la nube en la arquitectura de Fog Computing deben comunicarse de forma segura para evitar ataques pasivos y activos como el “Hombre en el medio” (man-in-the-middle) en el que las puertas de enlace que sirven como dispositivos Fog pueden verse comprometidas o reemplazadas por otras falsas. Los nodos Fog necesitan conectarse a la nube, y en paralelo, la nube debe conectarse a los nodos Fog confiables (no a los falsos o maliciosos). Para llevar seguridad en todas las capas de Fog se necesitan algunos pasos [5]:

- *Seguridad en la capa en la Nube:*
 1. Cada nodo Fog debe descubrirse de forma segura
 2. Realizar la autenticación mutua con la nube.
 3. La nube proporciona claves para los nodos Fog para cifrar y descifrar información de intercambio con la nube, para evitar que los atacantes espíen, modifiquen o eliminen información de intercambio.
 4. La nube actúa como control de acceso
- *Seguridad en la capa Fog:*
 1. Cada dispositivo debe ser descubierto de forma segura por los nodos Fog.
 2. Realizar la autenticación mutua.
 3. Los nodos Fog generan las claves para que los dispositivos de borde puedan cifrar y descifrar información de intercambio.
 4. El nodo Fog actúa como control de acceso.

Una plataforma Fog tiene recursos informáticos relativamente más pequeños debido a su propia naturaleza y, por lo tanto, es difícil ejecutar un conjunto completo de soluciones de seguridad que puedan servir para detectar y prevenir ataques sofisticados, dirigidos y distribuidos sin aumentar la latencia o los requerimientos de procesamiento. A su vez, es un objetivo atractivo para los ciberdelincuentes debido a los altos volúmenes de procesamiento de datos y la probabilidad de poder adquirir datos confidenciales tanto de la nube y dispositivos IoT; y es más accesible en comparación con los sistemas en la nube, dependiendo de la configuración de red y ubicación física, lo que aumenta la probabilidad de un ataque [4].

La búsqueda para asegurar los dispositivos conectados, consumiendo menos energía y poder de cómputo a la vez, es una tarea que parece interminable. Un obstáculo importante en esta búsqueda implica el cifrado, ya que muchos de estos dispositivos no tienen los requisitos de potencia para manejar la criptografía estándar.

Criptografía

Desde siempre ha existido la preocupación por asegurar la privacidad en las comunicaciones, intentando garantizar que un mensaje solo pudiera ser leído por el receptor al que iba destinado. Este reto es aún más difícil en el mundo actual en el que vivimos, en donde se hace un uso masivo e intensivo de las comunicaciones digitales a través de internet. La criptografía juega un papel fundamental para alcanzar este objetivo, ayudándonos a establecer comunicaciones seguras mediante la aplicación de algoritmos y protocolos de seguridad.

Criptografía Simétrica

Cuando hablamos de criptografía simétrica hacemos referencia a todo el conjunto de algoritmos e implementaciones de software que permiten cifrar y descifrar la información utilizando solamente una clave de cifrado.

Así, por ejemplo, podríamos cifrar los datos que quisiéramos, desde un archivo, hasta un fragmento de paquete de red, utilizando una clave, y nuestro destinatario podrá descifrarlo utilizando la misma clave, lo que implica que la clave deberá ser previamente intercambiada entre emisor y receptor. Este intercambio podría ser considerado el punto más débil de un cifrado simétrico.

Entre los algoritmos más conocidos se encuentran:

- *DES*
- *3DES*
- *AES*
- *IDEA*
- *Blowfish*
- *Chacha20*

Los cifradores simétricos se dividen en dos modos de funcionamiento:

- *Cifrado por bloques*
- *Cifrado de flujo*

Los cifradores de flujo cifran caracteres individuales (generalmente dígitos binarios) de un mensaje de texto sin formato, uno a la vez, utilizando una transformación de cifrado que varía con el tiempo. Generalmente son más rápidos que los cifradores de bloque en hardware. También son más apropiados, y en algunos casos obligatorios (por ejemplo, en algunas aplicaciones de telecomunicaciones), cuando el almacenamiento en búfer es limitado o cuando los caracteres deben procesarse individualmente a medida que se reciben. Algunos ejemplos de estos cifradores son RC4 y Chacha-20.

Los cifradores de bloques agrupan el mensaje original en bloques de “X” bytes antes de aplicar el proceso de cifrado. Un bloque pequeño (de 1, 2 o 3 bytes) facilita un ataque por estadísticas de lenguaje o “de diccionario”. Lo típico es utilizar bloques de 8 o 16 bytes, 64 o 128 bits. Actualmente se usan 16 bytes. Este tipo de cifradores brindan confidencialidad y en combinación con otras primitivas se pueden construir generadores de números pseudoaleatorios, códigos de autenticación de mensajes, entre otros. Algunos ejemplos de estos cifradores son DES y AES, que serán explicados más adelante.

Los cifrados de flujo normalmente se ejecutan a una velocidad mayor que los cifrados de bloque y tienen menor complejidad de hardware. Sin embargo, los cifrados de flujo pueden ser susceptibles a graves problemas de seguridad si se utilizan incorrectamente; en particular, el mismo estado inicial (semilla) nunca debe usarse dos veces.

Los cifradores también se diferencian según el modo de cifrado, es decir, en la forma en que se gestionan los bloques del mensaje cifrado. Existen diferentes tipos:

- *ECB*
- *CBC*
- *OFB*
- *CFB*
- *GCM*

ECB (Electronic CodeBook): es el método más simple en el cual el mensaje es dividido en bloques, cada uno de los cuales es cifrado de manera separada. La

desventaja de este modo de cifrado es que bloques idénticos de mensaje sin cifrar producen textos cifrados idénticos. Por esto, no proporciona una auténtica confidencialidad y no es recomendado para protocolos criptográficos. No usa el vector de inicialización.

CBC (Cipher-Block Chaining): este modo de cifrado ha sido ampliamente utilizado junto con una función hash para comprobar la autenticidad de los datos, y hoy en día se sigue utilizando. Consiste en que, a cada bloque de texto plano, se le aplica la operación XOR con el bloque de cifrado anterior. Cada bloque cifrado, depende de lo anterior procesado hasta ese punto. Para realizar esta opción XOR con el primer bloque de texto, se hace uso de un vector de inicialización. Este modo de cifrado se realiza de forma secuencial, no permite ser tratado de forma paralela para aumentar el rendimiento en el cifrado/descifrado de los datos.

Los modos de cifrado OFB y CFB se consideran modos de operación de criptografía de bloques por flujo. Esto no quiere decir que sean lo mismo que la criptografía de flujo, pero simula un proceso similar y se utilizan para distintas aplicaciones con diferentes niveles de seguridad y rapidez.

OFB (Output FeedBack): En primer lugar, se utiliza un vector de inicialización que, en este caso, se genera de forma aleatoria. A continuación, el vector de inicialización se encripta por medio de la clave de cifrado del algoritmo de bloques. Después, se hace una operación XOR entre el resultado del paso anterior y el bloque de texto plano para, de esta forma, generar el primer bloque de texto cifrado. El resultado del segundo paso (que proviene de encriptar el primer vector de inicialización) se utiliza como segundo vector de inicialización para el segundo bloque de texto plano a encriptar. De este modo, el proceso se repite cuantas veces sea necesario para encriptar la totalidad de bloques del mensaje. En este caso también se necesita un vector de inicialización que debe ser único para cada mensaje cifrado. Si no se utiliza un vector de inicialización diferente, se compromete la seguridad del sistema. Tampoco se puede paralelizar.

CFB (Cipher FeedBack): Opera similar a OFB ya que también se genera también un vector de inicialización de forma aleatoria o pseudoaleatoria y después se encripta a través de una clave. Esto produce el primer bloque de texto cifrado. El

resultado del paso anterior se utiliza para hacer una operación XOR con el bloque de texto plano que se desea encriptar. Luego, se realiza un proceso conocido como "registro de desplazamiento", que reorganiza la cadena de datos para generar un nuevo vector de inicialización.

El nuevo vector de inicialización se encripta por medio de los algoritmos de cifrado de bloques. El proceso se repite hasta encriptar la totalidad de los bloques que conforman el mensaje. El cifrado no puede ser paralelizado, sin embargo, el descifrado sí.

GCM (Galois Counter Mode): este modo de cifrado es uno de los mejores en cuanto a seguridad y velocidad. GCM permite el procesamiento en paralelo para acelerar el rendimiento en cifrado/descifrado de los datos. Este método, además de cifrar los datos, también es capaz de autenticarlos y comprobar su integridad, para asegurarnos que no se han modificado. GCM puede aceptar también vectores de inicialización aleatorios.

Criptografía Asimétrica

A diferencia de la criptografía simétrica, la asimétrica hace uso de dos claves, una pública y una privada, y es por ello que también se la llama, junto con los mecanismos de cifrado y autenticación asimétricos, "*Infraestructura de clave pública*", o *PKI* por sus siglas en inglés. Se basa en un par de claves en el cual solo existe una clave pública para una privada, y viceversa, de modo que no sería posible encontrar otra clave pública que se relacione con la privada, ni tampoco otra privada que se relacione con la pública del par. Además de esta condición, otra condición para la generación de claves es que la clave privada nunca puede ser obtenida a partir de la clave pública, y esto es necesario para cualquier algoritmo asimétrico, ya que estos métodos requieren el intercambio, por un medio hostil como Internet, de las claves públicas de los nodos de cifrado.

Para el caso del cifrado asimétrico, cada nodo dispone de las dos claves de cifrado, y pueden intercambiar libremente sus claves públicas, y así un nodo podrá cifrar la información de otro utilizando la clave pública del destinatario, mientras que el destinatario podrá descifrar el contenido mediante su clave privada secreta. Es por

esta razón que la clave privada debe ser protegida de cualquier extraño, puesto que quien la tenga podrá descifrar todo contenido, archivos o tráfico, destinado a su poseedor y, además, podrá autenticarse por su propietario original en cualquier servicio que utilice PKI y en el que haya estado registrado. En este caso, la ventaja es que las claves públicas pueden ser intercambiadas sin mediar ningún resguardo, y pueden intercambiarse directamente, o por medio de certificados digitales x509. La desventaja es que consume más recursos de cómputo, y demora más tiempo en calcularse.

Entre los algoritmos más conocidos están:

- *RSA*
- *DSA*
- *Curva Elíptica*
- *El Gamal*

Cifrado Híbrido

De lo expuesto para los cifrados simétrico y asimétrico concluimos que:

- El cifrado asimétrico en la práctica habitual es inútil para cifrar información, por cuanto solo es apto para cantidades muy pequeñas de información.
- El cifrado simétrico sí es válido, pero es intrínsecamente inseguro para transmitir información por cuanto nos obliga a enviar previamente la clave al otro extremo. Si el atacante ya escucha en este instante previo y captura la clave, todo el cifrado posterior, por bueno que sea el algoritmo con que se realiza, es inútil.

Estas conclusiones derivan en el problema de intercambio de clave, esto es, en resolver cómo intercambiar una clave simétrica de forma segura a través de un medio inseguro. Hay dos alternativas habituales:

- *Intercambio de clave basado en clave pública*

Consiste en que uno de los extremos genera la clave simétrica, la cifra con la clave pública del otro extremo y se la envía cifrada. Al recibirla, este otro extremo la

descifra con su clave privada y obtiene la clave simétrica que usarán ambos durante la comunicación.

- *Intercambio de clave Diffie-Hellman*

Consiste en que ambos extremos acuerdan en claro los parámetros que usarán (el módulo y la base) y generan ad hoc, cada uno por su cuenta, sus claves privadas (x e y). Con estas claves y los parámetros, generan sus claves públicas gx y gy que comunican a la otra parte. Cuando ambos aplican su propia clave privada sobre la clave pública recibida desde la otra parte ($(gx)y$ y $(gy)x$), el resultado en ambas partes coincide y este es la clave simétrica con la que se cifrará la comunicación ($K=(gx)y=(gy)x$). Un atacante que esté escuchando es capaz de capturar ambas claves públicas, pero no dispone de ninguna de las claves privadas, por lo que no podrá calcular la clave de sesión como hacen ambos extremos legítimos.

El cifrado híbrido es la base de todos los protocolos seguros de red como TLS/SSL o SSH.

Criptografía Postcuántica

La criptografía postcuántica es una nueva generación del sistema criptográfico de clave pública que se está sometiendo a evaluación por parte del NIST. No se espera que complete el proceso de estandarización hasta después de 2023.

El algoritmo de criptografía postcuántica utiliza un tamaño de clave mayor en comparación con los algoritmos clásicos de clave pública, por lo tanto, los requisitos de ancho de banda de la red serán más altos. El rendimiento del algoritmo de criptografía postcuántica también puede verse afectado por el perfil de red, la velocidad de la CPU y los índices de llamadas de API.

Cuando se utilizan estos algoritmos criptográficos cuánticos para la comunicación TLS, sólo protege los datos en tránsito, no en reposo y se espera que la seguridad del intercambio de claves públicas entre el cliente y el servidor tenga niveles de seguridad más altos que los algoritmos RSA y ECC actuales.

Estos nuevos algoritmos criptográficos cuánticos se basan en problemas matemáticos difíciles que, según las investigaciones actuales, tan solo las grandes computadoras cuánticas pueden resolver.

A medida que la informática cuántica continúa evolucionando y avanzando, un gran ordenador cuántico podrá ejecutar un algoritmo "SHOR" que puede romper los algoritmos de comunicación TLS actuales (RSA/ECC) en cuestión de minutos. Aunque los grandes ordenadores cuánticos no están disponibles hoy en día, cualquier dato de TLS en tránsito que haya sido espiado y almacenado podrá ser violado cuando estos grandes ordenadores cuánticos estén disponibles.

Criptografía Homomórfica

El cifrado homomórfico (HE) es un método de cifrado que permite realizar cálculos en datos cifrados sin descifrarlos primero con una clave secreta. Los resultados de los cálculos también permanecen cifrados y solo pueden ser descifrados por el propietario de la clave privada.

El propósito del cifrado homomórfico es permitir el cálculo de datos cifrados. Por lo tanto, los datos pueden permanecer confidenciales mientras se procesan, lo que permite realizar tareas útiles con datos que residen en entornos no confiables.

Un criptosistema homomórfico es como otras formas de encriptación pública en el sentido de que utiliza una clave pública para encriptar datos y permite que solo la persona con la clave privada correspondiente acceda a sus datos no encriptados. Sin embargo, lo que lo diferencia de otras formas de encriptación es que utiliza un sistema algebraico que permite realizar una variedad de cálculos u operaciones en los datos encriptados.

En matemáticas, homomórfico describe la transformación de un conjunto de datos en otro conservando las relaciones entre los elementos de ambos conjuntos. El término se deriva de las palabras griegas para «misma estructura». Debido a que los datos en un esquema de cifrado homomórfico conservan la misma estructura, las matemáticas idénticas, ya sea que se realicen en datos cifrados o descifrados, se obtendrán resultados equivalentes.

En la práctica, la mayoría de los esquemas de encriptación homomórfica funcionan mejor con datos representados como números enteros y al usar la suma y la multiplicación como funciones operativas. El cifrado homomórfico requiere pocas rondas de interacciones y utiliza funciones aritméticas en lugar de funciones booleanas como otros métodos de cálculo seguro.

Existen 3 tipos de cifrado homomórfico:

- *Cifrado parcialmente homomórfico*: Este tipo de cifrado homomórfico permite realizar ciertas funciones matemáticas (es decir, suma, multiplicación, etc.) sin descifrar los datos, mientras que para otras funciones es necesario descifrarlos.
- *Cifrado algo homomórfico*: Este tipo de encriptación permite realizar ciertas funciones matemáticas ilimitadamente sobre los datos encriptados. Por el contrario, otras funciones se pueden realizar solo un número fijo de veces, o hasta un cierto nivel de complejidad.
- *Cifrado completamente homomórfico*: permite realizar todas las funciones de forma ilimitada y hasta cualquier nivel de complejidad, sin necesidad de descifrar los datos. Es el tipo de cifrado homomórfico más seguro que se puede obtener hoy.

La primera limitación del cifrado completamente homomórfico (*FHE, Full Homomorphic Encryption*) es que es bastante lento. Es tan lento en este momento que no se le puede dar ningún uso práctico. La sobrecarga computacional de FHE es enorme, en lo que se refiere al tiempo consumido al realizar cálculos en la versión cifrada frente a la versión descifrada de los datos.

La segunda deficiencia de FHE es el soporte limitado para múltiples usuarios. Vale la pena señalar que todos los criptosistemas FHE funcionan mediante el establecimiento de una base de datos que se utiliza para el cálculo y el descifrado de los resultados calculados. Si alguien puede acceder a esta base de datos, también puede encontrar el resultado de texto sin formato descifrado de cualquier cálculo.

Ahora, una forma para que el proveedor del sistema criptográfico homomórfico garantice la seguridad de esta base de datos, es tener una base de datos separada para cada usuario. Sin embargo, si la cantidad de usuarios es enorme, mantener una base de datos separada para todos ellos puede volverse poco práctico e inviable.

Algoritmos Criptográficos

DES

Fue desarrollado en los años 70. Estuvo reconocido como estándar en el *FIPS (Estándar Federal de Procesamiento de la Información)*. Su tamaño de clave es de 56 bits y genera bloques de cifrado de 64 bits.

No se utiliza en aplicaciones dada su pequeña longitud de clave. Una versión mejorada de este algoritmo es 3DES.

3DES

Surge como respuesta ante ataques de fuerza bruta de DES. Utiliza tres (3) claves en su operación. Su tamaño es de clave de 128 bits y genera bloques de cifrado de 64 bits.

AES

Advanced Encryption Standard (AES), también conocido como Rijndael, es un esquema de cifrado por bloques adoptado como un estándar de cifrado por el gobierno de los Estados Unidos, creado en Bélgica. Desde 2006, AES es uno de los algoritmos más populares usados en criptografía simétrica.

El cifrado fue desarrollado por dos criptólogos belgas, Joan Daemen y Vincent Rijmen, ambos estudiantes de la Katholieke Universiteit Leuven, y fue enviado al proceso de selección AES bajo el nombre "Rijndael", como parte de un concurso. Estrictamente hablando, AES no es precisamente Rijndael (aunque en la práctica se los llama de manera indistinta) ya que Rijndael permite un mayor rango de tamaño de bloques y longitud de claves; AES tiene un tamaño de bloque fijo de 128 bits y tamaños de clave de 128, 192 o 256 bits, mientras que Rijndael puede ser

especificado por una clave que sea múltiplo de 32 bits, con un mínimo de 128 bits y un máximo de 256 bits.

El algoritmo AES demanda menor tiempo de ejecución tanto en software como en hardware, es relativamente fácil de implementar, y requiere poca memoria.

El AES tiene 10 rondas para claves de 128 bits, 12 rondas para claves de 192 bits, y 14 rondas para claves de 256 bits. El método más común de ataque hacia un cifrador por bloques consiste en intentar varios ataques sobre versiones del cifrador con un número menor de rondas.

Algunos criptógrafos muestran preocupación sobre la seguridad del AES. Ellos sienten que el margen entre el número de rondas especificado en el cifrador y los mejores ataques conocidos es muy pequeño. El riesgo es que se pueda encontrar alguna manera de mejorar los ataques y de ser así, el cifrado podría ser roto. En el contexto criptográfico se considera "roto" un algoritmo si existe algún ataque más rápido que una búsqueda exhaustiva (ataque por fuerza bruta). De modo que un ataque contra el AES de clave de 128 bits que requiera 120 operaciones sería considerado como un ataque que "rompe" el AES, aun tomando en cuenta que por ahora sería un ataque irrealizable. Hasta el momento, tales preocupaciones pueden ser ignoradas.

Otra preocupación es la estructura matemática de AES. A diferencia de la mayoría de cifradores de bloques, AES tiene una descripción matemática muy ordenada. Esto no ha llevado todavía a ningún ataque, pero algunos investigadores están preocupados de que futuros ataques quizá encuentren una manera de explotar esta estructura.

Chacha20

El algoritmo Chacha20 es un algoritmo de cifrado simétrico de alta velocidad creado por Bernstein en 2008 que soporta claves de 128 y 256 bits y es un cifrado de flujo a diferencia de AES que es un cifrado por bloques. Su código es mucho más eficiente y rápido que AES, en implementaciones de software, por lo que rápidamente se ha convertido en uno de los algoritmos más usados en la actualidad.

El algoritmo recibe como entrada una llave de 256 bits y un nonce o “semilla” de 96 bits. Al ser un cifrador de flujo utiliza estos datos para generar un flujo de llaves del mismo tamaño que los datos a encriptar/desencriptar. Una vez generado el flujo lo une con los datos mediante la operación XOR para obtener el resultado. Como la operación XOR es reversible el proceso de encriptación y desencriptación es exactamente el mismo.

Tiene características similares a su predecesor Salsa20 y al igual que este, Chacha trabaja con unidades de 32 bits o palabras. Esto quiere decir que, cada 32 bits se considera un solo elemento y se realizan las operaciones tomándose como uno. También utiliza operaciones de suma, rotación y XOR sobre 4 bytes.

El salto de popularidad se produce cuando, tras su puesta en marcha en la versión de escritorio de Chrome, llega a Android el nuevo Chacha20 para la encriptación. Además de ser un algoritmo simétrico que ofrece más seguridad, y que es inmune a varios tipos de ataques, lo más destacable es que consigue un rendimiento tres veces superior a protocolos algo más antiguos como puede ser AES, de esta manera, también se aprovechan mejor las capacidades de la CPU y se hace notar una reducción del 16% en el ancho de banda utilizado, lo que hace que se pueda aprovechar aún más la conexión.

Chacha20 se utiliza ampliamente en las conexiones HTTPS, actualmente si queremos tener la mejor seguridad podemos elegir entre AES o Chacha20, no obstante, este último nos proporciona un mayor rendimiento, aunque tengamos aceleración de cifrado por hardware. Este protocolo también se utiliza en las conexiones SSH para administrar servidores, en transferencias por SFTP, y otro protocolo muy popular que usa Chacha20 es VPN WireGuard, lo que lo hace, entre otros motivos, mucho más rápido que OpenVPN o IPsec, aunque tengamos aceleración de cifrado por hardware.

Blowfish

Blowfish es un codificador de bloques simétricos, diseñado por Bruce Schneier en 1993 e incluido en un gran número de conjuntos de codificadores y productos de cifrado. No se han encontrado técnicas de criptoanálisis efectivas contra el Blowfish.

Sin embargo, se ha dado más atención a la decodificación de bloques con bloques más grandes, como AES y Twofish.

Schneier diseñó Blowfish como un algoritmo de uso general, sin patente, que intentaba reemplazar al antiguo DES y evitar los problemas asociados con otros algoritmos. Además, muchos otros diseños eran propiedad privada, patentados o los guardaba el gobierno.

TwoFish

Es un cifrado de clave simétrica, que dispone de un tamaño de bloque de 128 bits, lo que puede verse a priori, como muy seguro ante ataques de fuerza bruta, al requerir grandes capacidades de procesamiento para poder descifrarlo. La longitud de su clave puede variar entre los 128, 192 o 256 bits. Es de código abierto, y su uso es totalmente gratuito.

Está pensado y optimizado para usarse en unidades de procesamiento de 32 bits tanto en entornos de hardware y software. Surgió en el año 1998, y se estableció como uno de los mejores algoritmos de cifrado, que entraría en sustitución de DES, pero fue descartado porque a pesar de ser muy seguro, resultaba muy lento en comparación con otros algoritmos de la competencia.

TwoFish utiliza una clave para cifrar y descifrar todos los datos e información, y soporta las claves junto con la información sin ningún tipo de formato. Una vez recibida toda la información, la cifra, por lo cual ya no es visible sin pasar por el proceso de decodificación. Todos estos datos, se enviarán al usuario o sistema final, acompañados de la clave de cifrado, para que se pueda ver la misma.

Otro punto interesante de TwoFish, son las cajas de sustitución (S-Boxes), las cuales son calculadas previamente y dependen de la clave. Estas son un componente básico del algoritmo, y se utilizan para hacer menos visible la relación entre un texto plano y uno cifrado. Pero estas pueden ser el talón de Aquiles del algoritmo. El riesgo se minimiza en gran medida, al hacer que las S-Boxes dependan de la clave transmitida.

RSA

El nombre RSA proviene de las iniciales de sus tres creadores, Rivest, Shamir y Adleman, allá por 1977. RSA sirve para cifrar y descifrar información, y por ello también provee servicios de autenticidad y de integridad, mediante lo que se conoce como *Infraestructura de clave pública (PKI)*.

Se trata de uno de los algoritmos más utilizados en la actualidad. De hecho, la mayor parte de los sitios web hoy corren sobre SSL/TLS, y permiten la autenticación mediante cifrado asimétrico basado en RSA.

RSA es un algoritmo perfectamente seguro en la actualidad, es decir, no existe una computadora ni clúster de alto rendimiento que pueda romper el algoritmo en un tiempo aceptable. No obstante, no es un algoritmo que pueda resistir al criptoanálisis cuántico, es decir, a un ataque realizado implementando un algoritmo cuántico como el de Shor en una computadora cuántica con cientos o miles de qubits, que además disponga de corrección de errores y un control aceptable del ruido.

DSA

El *Algoritmo de Firma Digital (Digital Signature Algorithm)* fue patentado en USA y reconocido como un estándar FIPS del NIST (USA). Sólo puede ser utilizado en firma electrónica (no cifrado). El tamaño de clave que utiliza es de 1024 bits.

Curva Elíptica

Es el siguiente algoritmo asimétrico más reconocido después de RSA. Surge a mediados de los 80 y es útil en aplicaciones donde la memoria, el ancho de banda y el poder computacional son limitados.

Utiliza tamaños de claves más pequeños que RSA: una clave de 160 bits de curva elíptica equivale a 1024 bits en RSA. Es más rápido que RSA para firmar y descifrar, pero más lento para verificar firma y cifrar.

Diffie-Hellman

No es un algoritmo asimétrico propiamente dicho, es un protocolo de establecimiento de claves, se usa para generar una clave privada a ambos extremos de un canal de comunicación inseguro. Se emplea para obtener la clave privada con la que posteriormente se cifrará la información junto con un algoritmo de cifrado simétrico. El punto fuerte del Diffie-Hellman es que, su seguridad radica en la dificultad de calcular el logaritmo discreto de números grandes (Diffie-Hellmann también permite el uso de curvas elípticas). La desventaja de este protocolo es que no proporciona autenticación, no puede validar la identidad de los usuarios, por tanto, si un tercer usuario se pone en medio de la comunicación, también accedería a las claves y, en consecuencia, podría establecer comunicaciones con el emisor y el receptor suplantando las identidades. Para evitar esto existen varias soluciones que mitigan y solucionan el problema, cómo hacer uso de certificados digitales.

Funciones de Una Vía o Funciones Hash

La criptografía asimétrica permite identificar al emisor y al receptor del mensaje. Para identificar el mensaje propiamente dicho se utilizan las llamadas funciones hash. Una función hash es un algoritmo que transforma un conjunto arbitrario de elementos de datos, como puede ser un fichero de texto, en un único valor de longitud fija (el "hash"). El valor hash calculado puede ser utilizado para la verificación de la integridad de copias de un dato original sin la necesidad de proveer el dato original. Esta irreversibilidad significa que un valor hash puede ser libremente distribuido o almacenado, ya que solo se utiliza para fines de comparación.

La seguridad proporcionada por un algoritmo hash es sumamente dependiente de su capacidad de producir un único valor para un conjunto de datos dados. Cuando una función hash produce el mismo valor para dos conjuntos de datos distintos, entonces se dice que se ha producido una colisión. Una colisión aumenta la posibilidad de que un atacante pueda elaborar computacionalmente conjuntos de datos que proporcionen acceso a información segura o para alterar ficheros de datos informáticos de tal forma que no cambiará el valor hash resultante y así eludir la detección. Una función hash fuerte es aquella que es resistente a este tipo de ataques computacionales, mientras que, una función hash débil es aquella donde existe una

creencia casi certera de que se pueden producir colisiones. Finalmente, una función hash quebrantada es aquella sobre la que se conoce métodos computacionales para producir colisiones.

A continuación se describen los algoritmos hash más utilizados:

MD5

En criptografía, *MD5* (abreviatura de *Message-Digest Algorithm 5, Algoritmo de Resumen del Mensaje 5*) es un algoritmo de reducción criptográfica de 128 bits ampliamente usado. Uno de sus usos es el de comprobar que algún archivo no haya sido modificado.

MD5 fue desarrollado en 1991 por el profesor Ronald Rivest del *MIT* (*Massachusetts Institute of Technology, Instituto Tecnológico de Massachusetts*) como reemplazo del algoritmo MD4 después de que Hans Dobbertin descubriese su debilidad.

La codificación del MD5 de 128 bits es representada típicamente como un número de 32 símbolos hexadecimales.

Los resúmenes MD5 se utilizan extensamente en el mundo del software para proporcionar la seguridad de que un archivo descargado de Internet no se haya alterado. Comparando una suma MD5 publicada con la suma de comprobación del archivo descargado, un usuario puede tener la confianza suficiente de que el archivo es igual que el publicado por los desarrolladores. Esto protege al usuario contra los “Troyanos” y virus que algún otro usuario malicioso pudiera incluir en el software. También permite reconocer una descarga corrupta o incompleta.

Para comprobar la integridad de un archivo descargado de Internet se puede utilizar una herramienta para comparar la suma MD5 de dicho archivo con un archivo MD5SUM con el resumen MD5 del archivo original. En los sistemas UNIX, el comando de md5sum es un ejemplo de tal herramienta.

En sistemas UNIX y GNU/Linux se utiliza el algoritmo MD5 para calcular el hash de las claves de los usuarios. En el disco se guarda el resultado del MD5 de la clave que se introduce al dar de alta un usuario, y cuando este quiere entrar en el

sistema se compara el hash MD5 de la clave introducida con el hash que hay guardado en el disco duro. Si coinciden, es la misma clave y el usuario será autenticado. Los sistemas actuales GNU/Linux utilizan funciones de hash más seguras, como pueden ser SHA-2 o SHA-3.

El MD5 también se puede usar para comprobar que los correos electrónicos no han sido alterados usando claves públicas y privadas.

A pesar de haber sido considerado criptográficamente seguro en un principio, ciertas investigaciones han revelado vulnerabilidades que hacen cuestionable el uso futuro del MD5. En agosto de 2004, Xiaoyun Wang, Dengguo Feng, Xuejia Lai y Hongbo Yu anunciaron el descubrimiento de colisiones de hash para MD5. Aunque dicho ataque era analítico, el tamaño del hash (128 bits) es lo suficientemente pequeño como para que resulte vulnerable frente a ataques de “fuerza bruta”.

Debido al descubrimiento de métodos sencillos para generar colisiones de hash, muchos investigadores recomiendan su sustitución por algoritmos alternativos tales como SHA-256 o RIPEMD-160.

SHA-2

SHA-2 es un conjunto de funciones hash criptográficas diseñadas por la *Agencia de Seguridad Nacional (NSA)* y publicada en 2001 por el NIST como un FIPS.

SHA significa algoritmo de hash seguro. Tanto SHA-0 como SHA-1 generan resúmenes de 160 bits. Ambas se consideran inseguras y se desaconseja su uso. Como paliativo, se desarrolló SHA-2, que implementa funciones hash de 224, 256, 384 y 512 bits. Se las conoce habitualmente como SHA-224, SHA-256, SHA-384 y SHA-512. SHA-3, por su parte, cambia internamente los algoritmos, pero está pensada para generar resúmenes de la misma longitud que SHA-2.

En 2005, se identificaron fallas de seguridad en el SHA-1, permitiendo que existiera una debilidad matemática y evidenciando así la necesidad de elaborar una función hash más fuerte. Aunque el SHA-2 se comporta de forma parecida al algoritmo SHA-1, estos ataques no han sido extendidos satisfactoriamente a SHA-2.

La competición de funciones hash de la NIST seleccionó una nueva función hash, el SHA-3, en 2012. A diferencia de SHA-2 con SHA-1, el algoritmo SHA-3 no es derivación del SHA-2.

RIPEMD-160

RIPEMD-160 (acrónimo de *RACE Integrity Primitives Evaluation Message Digest*, primitivas de integridad del resumen del mensaje) es un algoritmo de resumen del mensaje de 160 bits desarrollado en Europa por Hans Dobbertin, Antoon Bosselaers y Bart Preneel, y publicado primeramente en 1996. Es una versión mejorada de RIPEMD, que estaba basado sobre los principios del diseño del algoritmo MD4, y es similar en seguridad y funcionamiento a SHA-1.

También existen versiones de 128, 256 y 320 bits de este algoritmo, llamadas RIPEMD-128, RIPEMD-256 y RIPEMD-320 respectivamente. La versión 128 bits fue pensada solamente como un reemplazo para el RIPEMD original, que también era de 128 bits. Las versiones de 256 y 320 bits solamente disminuyen la posibilidad de colisiones hash accidentales, y no tienen niveles más altos de seguridad con respecto a RIPEMD-128 y RIPEMD-160.

RIPEMD-160 fue diseñado en la comunidad académica abierta, en contraste con el algoritmo SHA-1, diseñado por la Agencia de Seguridad Nacional estadounidense (NSA). Por otra parte, RIPEMD-160 es un diseño menos popular y ninguna patente está asociada al RIPEMD-160.

RIPEMD160 se ha reemplazado por los algoritmos hash seguros SHA-256 y SHA-512 y sus clases derivadas. SHA-256 y SHA-512 ofrecen una mejor seguridad y rendimiento que RIPEMD160.

Trabajos Relacionados

Debido al hecho de que los sensores están limitados en sus recursos, una técnica criptográfica de computación pesada no puede desplegarse en ellos. Así que,

asegurar el sistema es ahora responsabilidad de los nodos Fog para no propagar paquetes maliciosos a la nube.

Vishwanath et. al [12] implementan el proceso de encriptación sobre tres conjuntos de datos de diferentes tipos para comprobar cómo funciona en la capa Fog usando el algoritmo AES. En la validación de las pruebas realizadas, los tres conjuntos de datos se encriptan y desencriptan con precisión. Se utilizó un celular Android como dispositivo de borde y se implementó la encriptación de los conjuntos de datos en él. Además, se evaluó el tiempo de encriptado y descifrado, la carga del usuario, el tiempo de respuesta, la utilización de memoria sobre el tamaño del archivo. En base a ello concluyen en que, el algoritmo de análisis AES es el proceso de cifrado más seguro.

El algoritmo AES utiliza 128, 192 y 256 bits de tamaño de clave. A medida que aumenta el tamaño de la clave, es más difícil descifrar el texto encriptado. Por lo tanto, Patel et. al [13] amplían a 512 bits el tamaño de la entrada y la clave, lo que garantiza una mayor seguridad. El número de rondas para el procesamiento del texto sin formato se incrementa a 22, donde las primeras 21 rondas realizan las cuatro operaciones del algoritmo AES, y la ronda final implementa tres de las cuatro operaciones de encriptación. El descifrado se realiza utilizando las operaciones inversas del algoritmo de cifrado. AES-S será adecuado para aplicaciones con requisitos de alta seguridad y rendimiento, sin aumentar el diseño general en comparación con los bits AES-128 originales.

Otros autores realizan comparaciones técnicas para seleccionar la tecnología y el algoritmo de cifrado más adecuado. Por ejemplo, Montoya [14] utiliza un sistema de puntaje para valorar las pruebas que realiza en las que compara los algoritmos de cifrado RSA, *DHEC (Diffie-Hellman Elliptic Curve)*, *HJP (Homomórfico JPAILLIER)* y AES por un lado, y las tecnologías de comunicación por el otro, para determinar cuál es la mejor combinación para la preservación de la privacidad de la información en dispositivos IoT portátiles. La evaluación sobre los algoritmos de cifrado, que midió variables como el tiempo de ejecución, nivel de riesgo, consumo de procesamiento, energía, y ancho de banda, determinó que DHEC es el algoritmo de cifrado, que cuenta con mejores características para proteger la información entre la capa Fog y la nube, permitiendo eficiencia y escalabilidad. Según las pruebas realizadas para

cada una de las tecnologías de comunicación, en las cuales se midió compatibilidad, consumo de energía, procesamiento y seguridad (referida a la de posibilidad de sufrir ataques de interceptación de información), se determinó que NFC es la tecnología que mejor se adapta para el transporte de la información de los dispositivos hasta la capa Fog.

Recientemente, en el año 2022, un artículo científico publicó un estudio comparativo de los algoritmos de criptografía simétrica AES, 3DES y ChaCha20. Los autores, Centella Claros et. al [15] realizaron un estudio empírico en el cual compararon el rendimiento de ChaCha20 con otros dos algoritmos de encriptación AES, y 3DES. Se evaluó el tiempo de ejecución, consumo de CPU y consumo de energía que toma cada uno de los algoritmos para encriptar y desencriptar información. Se desarrolló un experimento con 150 archivos de distintos formatos y tamaños para probar cada algoritmo. Se determinó que el algoritmo ChaCha20 es el más rápido, AES es el que menos CPU consume y no existe diferencia significativa en el consumo de energía de ninguno de los algoritmos.

En Ali et. al [16] se compara la eficacia de transferir información de salud de un paciente (PHI) usando dos estándares diferentes de cifrado, AES y RSA. La mayor diferencia entre RSA y AES es que RSA se basa en cifrado asimétrico mientras que, AES se basa en cifrado simétrico. Los algoritmos AES y RSA se codificaron dentro del hardware de un Raspberry PI utilizando el software Python de código abierto. Además, la Raspberry PI fue utilizada como un dispositivo de Fog Computing para cifrar los datos PHI. Los resultados demuestran que AES supera a RSA en el proceso de encriptación ya que requiere menor tiempo de ejecución y hace un uso más eficiente de la memoria.

Froiz et. Al [17] analizó el efecto sobre el consumo de corriente en un sistema IoT de automatización de un hogar al hacer uso de encriptación al comunicar un nodo ZiWi con un gateway local. En el caso de los módulos basados en NodeMCU, se utilizó TLSv1/SSLv3 con ECDHE-RSA-AES128-GCM-SHA256 como conjunto de cifrado, mientras que, en los nodos basados en ZigBee, se habilitó el *cifrado de subcapa de soporte de aplicaciones (APS)*. Las medidas de corriente se obtuvieron utilizando un Arduino y un Adafruit INA219. El uso de *Secure Sockets Layer (SSL)* genera una carga computacional y de comunicaciones adicional que aumenta el

consumo medio a lo largo del tiempo. Sin embargo, el uso de un mecanismo de cifrado más ligero (y menos robusto) como el APS de ZigBee apenas aumenta el consumo medio, pero aumenta la privacidad y la seguridad respecto a no utilizar ningún tipo de encriptación en los nodos.

En el 2014 se presentó el *Cifrado Basado en Atributos (ABE)*, una solución común para proporcionar seguridad de transmisión de datos, uso compartido y almacenamiento en el entorno extendido como IoT. Pero ABE se basa en un emparejamiento bilineal que lo hace inadecuado para dispositivos con recursos limitados en IoT. Los autores del cifrado (Xuanxia Yaoa, 2014) propusieron que ABE sin emparejamiento confíe en la Criptografía de Curva Elíptica (ECC) para etiquetar los problemas de seguridad y privacidad en IoT.

ECC requiere menos cómputo y tamaño de memoria y también proporciona más seguridad, lo que lo convierte en un excelente criptosistema para un entorno de IoT y computación en la nube. Los autores Bai T Daisy Premila, 2015 propusieron un nuevo marco de seguridad basado en ECC para IoT y Cloud Computing. El ECC de 160 bits ofrece una seguridad equivalente al RSA de 1024 bits. También Barraza [18] sugiere utilizar ECC para asegurar Fog Computing, ya que proporciona longitudes de clave más cortas, tamaños de mensaje más pequeños y menor uso de recursos. La combinación de 84 esquemas ligeros permite una mejor escalabilidad y menos gastos de recursos que los esquemas basados en RSA empleados en SSL y TLS.

En 2016, M.R. Balasubramaniam, propuso EECC (criptografía de curva elíptica ElGamal) como un nuevo método eficiente para cifrar y descifrar un texto utilizando el valor de ASCII hexadecimal para cada carácter. A pesar de que ECC tiene un tamaño de clave corto, brinda la misma seguridad en comparación con otros sistemas criptográficos como RSA. Estas características han hecho que la criptografía de curva elíptica sea efectiva en algunas aplicaciones, como dispositivos móviles, tarjetas inteligentes, tarjetas con chip y comercio electrónico.

Con el fin de saber qué factores afectan el rendimiento de ABE, Krinah et. al [19] realizaron pruebas en dos dispositivos que actuaron como nodos Fog, a saber, un teléfono inteligente Samsung S3 (CPU ST Ericsson U8420 de 1 GHz, 1,4 GB de RAM) y una Raspberry Pi Zero (CPU de un solo núcleo de 1 GHz, 512 MB de RAM). Las pruebas se llevaron a cabo (cifrado CP-ABE) en conjuntos de datos médicos que

consisten en registros de pacientes con diabetes 1. En ambos entornos, se varió el cifrado, tamaño de la clave, el número de atributos y el tamaño de los datos a cifrar. Se midió el tamaño del texto cifrado resultante, el tiempo de ejecución del cifrado, así como el consumo de CPU y RAM. Los resultados obtenidos fueron casi similares en ambos ambientes, detectando que los dos factores más influyentes son el tamaño de la clave y el número de atributos. De hecho, cuanto más se aumenta el tamaño de las claves de cifrado y el número de atributos, más CPU, RAM y tiempo durará la operación de cifrado. Por otro lado, el tamaño de los datos de entrada se comprobó que no tiene influencia sobre las actuaciones.

Algunos autores proponen nuevos protocolos de encriptación, tales como Boakye-Boateng et. al [20] quienes investigaron la posibilidad de crear un protocolo de cifrado basado en “*Una Sola Vez*” (*One-Time Pad*) sin pérdida de paquetes; menos gastos generales de tiempo y energía en comparación con los protocolos que han sido propuestos por la investigación existente. El OTP es una clave de cifrado con una longitud igual a la del texto plano y se utiliza una sola vez. El texto cifrado, también de la misma longitud que el texto sin formato, es generado por un módulo de adición (XOR) de la clave y el texto plano; y es difícil descifrar por el adversario mediante análisis estadístico debido a la frecuencia uniforme de distribución de la clave. Esta investigación indaga en la posibilidad de generar las OTP utilizando generadores de números aleatorios. El protocolo se probó en nodos de sensores inalámbricos, que tienen recursos limitados. Se generaron las OTP utilizando un *RGN (Generador de Números Aleatorios – Random Generator Number)* dentro de los nodos. Se consideraron en la investigación un RNG de *registro de desplazamiento de retroalimentación lineal (LFSR)* de 16 bits y otro *congruencial lineal multiplicativo (MLC)* de 32 bits. Los dos generadores de números aleatorios eran ideales porque se diseñaron teniendo en cuenta las limitaciones de recursos. El LFSR fue el más rápido de los dos según la documentación, pero su principal defecto fue su espacio de claves limitado.

Un RNG debe ser determinista entre nodos y "verdaderamente aleatorio" para los adversarios. Sin embargo, debe tenerse en cuenta que cuanto más complejo sea el algoritmo del RNG, mayor es su efecto sobre la energía y sobrecarga de tiempo. Pruebas lineales y diferenciales existentes para OTP se aplicaron a este protocolo y se determinó que la fuerza de este protocolo depende del tipo de RNG utilizado.

Aunque se usaron RNG idénticos en el cifrado y descifrado, se recomienda usar dos RNG diferentes y que el RNG para el descifrado de un nodo sea el RNG utilizado para el cifrado del otro nodo y viceversa.

Otra área de investigación se basa en la criptografía post-cuántica [6] cuya metodología consiste en cifrar los mensajes añadiéndoles errores, de manera que el proceso de eliminación de estos esté asociado a cada clave privada. El sistema criptográfico más popular en esta categoría es el sistema McEliece, basado en códigos Goppa binarios, cuyo inconveniente más característico es que solamente sirve para la transmisión cifrada de mensaje y no tiene ningún procedimiento para generar firmas digitales. Algunas alternativas para la implementación de firmas basadas en códigos de corrección son Niederreiter y CFS (Courtois, Finiasz, Sendrier).

Otros trabajos buscan emplear nuevas técnicas como Blockchain o Machine Learning para solucionar los problemas de seguridad. Chacon [21] integra Machine Learning a una red IoT para monitorear en tiempo real y poder responder ante atacantes que quieran penetrar en dicha red. El objetivo final es la implementación de una técnica que en poco tiempo pueda detectar de forma oportuna y correcta la presencia de algunos malwares en la red. Para ello se implementó y comparó un algoritmo K-NN y una Red Neuronal para ver su comportamiento frente ataques de DoS y Exploits. La comparación entre ambas implementaciones arrojó que la red neuronal por la composición del conjunto de datos y su funcionamiento tiene un mejor desempeño en cuanto a la detección de trazas malignas DoS y Exploits. La red neuronal tuvo siempre un menor tiempo que osciló en tiempos cercanos a los 15 segundos, tanto para entrenar como para evaluar. Por el contrario, el algoritmo de K-NN estuvo por encima de los 135 segundos, una diferencia que puede ser fundamental en el momento de detectar un ataque en tiempo real. Este trabajo encontró que la implementación de red neuronal tuvo un desempeño del 0.98 de precisión en tiempos cercanos a los 5 segundos analizando conjuntos de datos de 100.000 trazas de red.

Una implementación ligera de Blockchain que incluye elementos mínimos de tecnología Blockchain como el libro mayor distribuido y la criptografía puede cumplir con la mayoría de los requisitos de seguridad de la capa Fog. Dicha implementación, además, realizará las tareas de cifrar y descifrar datos en el hardware con un

protocolo inalámbrico seguro (por ejemplo, Zigbee) para lograr comunicación segura. El rendimiento de una de estas implementaciones ligeras se ha informado en Shreshth et. al [22]. Los autores investigan el uso de la tecnología Blockchain en el contexto de Fog Computing para el caso de uso de atención médica inteligente. Se observa que el consumo de energía y el requisito de latencia son aceptables para el caso de uso de atención médica, incluso cuando se implementa en el entorno de integración Cloud+Fog. Sin embargo, el rendimiento de la integración de Blockchain y Fog Computing con varios parámetros de ajuste necesita ser evaluado en otros dominios de aplicación de IoT.

La encriptación es necesaria para garantizar la seguridad y privacidad de los datos transmitidos al canal público y almacenados en nodos Fog o en un servidor en la nube, pero además de ellos es necesario que cada usuario o nodo Fog en el sistema se identifique y autentique de manera única con el fin de establecer la confianza y evitar la suplantación de identidad. Sin embargo, no es óptimo precompartir claves de sesión entre los nodos Fog y los dispositivos finales debido a la naturaleza dinámica y de movilidad de los dispositivos finales y los nodos Fog. La autenticación de dispositivos con restricción de memoria presenta grandes problemas, ya que el consumo de memoria es elevado, cuando se aplica a otros modelos que tienen como finalidad la autenticación mutua. Por todo ello, algunos autores investigaron esquemas que brinden una solución para autenticar usuarios o nodos y producir claves de sesión comunes.

Ibrahim [23] propone un esquema de autenticación mutua Edge-Fog eficiente y seguro, para permitir que cualquier usuario Fog y cualquier servidor Fog pueda autenticarse mutuamente, sin depender de ninguna PKI (*Infraestructura de Clave Pública, Public key Infrastructure*). Se requiere que el usuario Fog almacene sólo una clave maestra secreta de larga duración, y esta clave le permite moverse aleatoriamente en la red y autenticarse mutuamente con cualquier servidor Fog bajo la autoridad de un CSP (*Proveedor de Servicio en la Nube, Cloud Service Provider*) en particular. Además, el usuario Fog debe poder autenticarse con nuevos servidores que se unen a la capa Fog sin la necesidad de volver a registrarse y sin gastos generales adicionales. El esquema utiliza herramientas criptográficas elementales (funciones hash y cifrado simétrico). Los cálculos que requiere el usuario Fog son solo unas pocas invocaciones de hash y cifrados/descifrados simétricos. Esto hace

que el esquema sea muy eficiente para la implementación en tarjetas inteligentes y dispositivos con recursos muy limitados, como sensores y teléfonos inteligentes.

Jia et. al [24] proponen un protocolo AKA (*Acuerdo de Clave Autenticado, Agreement Key Authenticated*) de tres partes con emparejamientos bilineales para un sistema de atención médica de IoT impulsado por Fog Computing, centrándose solo en la autenticación y el acuerdo de clave. Establecen un mecanismo basado en verificadores para autenticar al usuario y los nodos Fog. Un esquema AKA basado en verificador es una variante de un esquema AKA basado en contraseña, en el que el servidor solo conserva un verificador en lugar de una imagen de la clave. Cada nodo Fog o dispositivo final tiene una identidad única y debe registrarse en el servidor de la nube. El servidor de la nube mantendrá un verificador de ese nodo o dispositivo para el futuro. El verificador se deriva de la identidad y la clave maestra del CSP. El protocolo genera una clave común de sesión compartida por un dispositivo final, el nodo Fog y el servidor en la nube. El nodo Fog no preserva la información privada (contraseña, verificador) de cualquier usuario. Si hay un nodo Fog comprometido, entonces los datos del usuario en otros nodos no serán afectados incluso si utilizan la misma contraseña y tarjeta inteligente.

Loffi [25] introdujo un nuevo modelo de autenticación mutua en entornos IoT para el contexto de Fog Computing. El protocolo propuesto fue adecuado y optimizado junto con algunos protocolos existentes (RSA, AES-192 y SHA-512). La autenticación es realizada durante el handshake, más específicamente durante la verificación de factores de autenticación: función desafío-respuesta, tiempo de respuesta y nonce. A través de una aplicación final, se pudo probar la factibilidad técnica utilizando los lenguajes de programación C++ y Java. Se destacó la velocidad de autenticación del modelo, ya que la infraestructura está cerca de los sensores/actuadores.

Morishima y Nishi [26] proponen una plataforma IoT basada en Fog Computing denominada Door (Inspección profunda de paquetes en el Router, Deep packet inspectOr On Router) que permite a los nodos Fog anonimizar datos confidenciales, responder a un mensaje con baja latencia y reducir el ancho de banda requerido mediante la agregación de múltiples transacciones de datos. Implementaron la modificación de la carga y la generación de paquetes de reconocimiento de forma transparente para la red con funciones de DooR, que recalculaban la *suma de*

verificación (*checksum*), el número de secuencia, y el campo *ACK* (*Reconocimiento, Acknowledgement*) de la cabecera de los paquetes. Las operaciones correctas de estas funciones fueron verificadas. Los resultados mostraron que DooR envió un paquete de reconocimiento a los dispositivos IoT de forma transparente con latencia inferior a 10 ms, y DooR redujo el uso de ancho de banda hasta en un 41% al concatenar los contenidos, siempre que la suma de las longitudes de los contenidos no excediera el MTU.

Basha et. al [27] propone un sistema de control de acceso FCSSI (Fog-based Self-Sovereign) + RSA + IoMT (Internet of Medical Things) de PHR (Registro de Salud del Paciente, Patient's Health Record) que protege los detalles de identidad del paciente en PHR de ataques externos. Utiliza RSA para almacenar los datos en formato distribuido de forma segura y eficaz. Requiere un consumo de tiempo mínimo en comparación con el almacenamiento centralizado existente de datos PHR. Además de eso, minimiza el retraso en la sincronización y la propagación. En el futuro podría ser aplicado a varios dominios como agricultura, logística, educación, entre otros.

Como se mencionó anteriormente, se pueden aprovechar las técnicas de cifrado para cifrar el canal después de la autenticación para brindar privacidad de los datos transmitidos. Sin embargo, la autenticación y el cifrado por sí solo no pueden garantizar la seguridad.

TLS (*Seguridad de la capa de Transporte, Transport Layer Security*), por ejemplo, es uno de los protocolos criptográficos más populares. Sin embargo, todavía hay trabajos que explotan vulnerabilidades en sus conjuntos de cifrado y en el propio protocolo. El atacante puede comprometer un enlace TLS, por ejemplo, instalando sigilosamente un certificado de cliente. Li et. al [28] investiga las amenazas potenciales de los ataques "*Hombre en el Medio*" (*Man-in-the-Middle*) en el canal de control OpenFlow. Propone una contramedida ligera utilizando filtros Bloom. Implementa un prototipo de este método para monitorear modificaciones sigilosas de paquetes. El resultado de la evaluación muestra que el sistema de monitoreo de filtros Bloom es eficiente y consume pocos recursos.

Los paradigmas de seguridad tradicionales centrados en proporcionar protecciones basadas en el perímetro y los protocolos punto a punto cliente/servidor,

como TLS, ya no son las mejores opciones para abordar los nuevos desafíos de seguridad en dispositivos finales de Fog Computing, donde la energía y los recursos computacionales están limitados. Venčkauskas et. al [29] presenta un *Protocolo de Transmisión Ligero y Seguro (LSSP)* para la capa Fog. Este protocolo es liviano, sin conexión, admite operaciones de transmisión y multidifusión, y puede proporcionar autenticación de fuentes de datos, integridad de datos y confidencialidad. El protocolo se basa en métodos criptográficos simples y energéticamente eficientes, como *Códigos de Autenticación de Mensajes Hash (HMAC)* y cifrados simétricos, y utiliza paquetes de *Protocolo de Datagramas de Usuario (UDP)* modificados para incorporar datos de autenticación en la transmisión de datos. Se podría agregar redundancia de datos para mejorar la confiabilidad en redes con pérdidas. Los resultados experimentales resumidos en este documento confirman que el método propuesto utiliza eficientemente la energía y los recursos computacionales y al mismo tiempo proporciona propiedades de seguridad a la par con el estándar *Datagram TLS (DTLS)*. Este protocolo está diseñado para ser utilizado por dispositivos con recursos limitados para la transmisión de datos, donde es posible un cierto grado de pérdida de paquetes (por ejemplo, transmisión de video). Los resultados también muestran que la modificación del Modo 1 del protocolo propuesto tiene casi las mismas características de rendimiento, sobrecarga de datos y consumo de energía que el protocolo UDP simple, mientras que, al mismo tiempo proporciona autenticación de fuente de datos. Las modificaciones de Modo 2 y Modo 3, cuando se usan con paquetes ECC redundantes adicionales, brindan una mayor confiabilidad de transferencia de datos en comparación con el protocolo DTLS, con propiedades de seguridad similares. Por otro lado, si el escenario de uso requiere el establecimiento frecuente de nuevas conexiones, la sobrecarga de ancho de banda utilizada en el protocolo DTLS, así como el complejo procedimiento de negociación, hacen que DTLS sea menos eficiente en ancho de banda, incluso en los casos en que se utilizan paquetes ECC adicionales para el protocolo LSSP.

Abbas et. al [30] presentaron un nuevo *Servicio de Seguridad de Niebla (FSS)*, que proporciona autenticación basada en identidad, la integridad de datos y el no repudio de nodos de conexión a través de un *Generador de Clave Privada (PKG)* en la capa Fog. FSS mejora la seguridad de extremo a extremo entre la capa de IoT y la capa Fog. La implementación y evaluación se han realizado en forma de simulación

utilizando OPNET, para demostrar la idoneidad del mecanismo propuesto. En primer lugar, los valores de referencia se obtuvieron en un entorno real donde se consideraron los dispositivos IoT (con diferentes capacidades y recursos) conectados de forma inalámbrica a un nodo Fog. El desempeño se midió en términos del tiempo de respuesta, utilizando varias operaciones del método planteado, como la función de generación de claves secretas aleatorias, la función de cifrado y descifrado de claves privadas, la función de generación aleatoria de nonce y el cifrado de claves públicas y función de descifrado. Estos valores de referencia se utilizaron luego en el simulador y se aplicaron a nodos y al servidor Fog. Se ejecutó y midió el desempeño del servicio propuesto en diferentes cargas de tráfico, y resultó ser mejor en comparación con el servicio APaaS y el método heredado.

Noura et. al [31] proponen una solución criptográfica novedosa para proteger los datos en Fog Computing. La solución combina el modo de operación AES-GMAC con dispersión de información para proporcionar confidencialidad de datos, integridad y disponibilidad junto con la autenticación de origen. La solución se basa en el enfoque dinámico dependiente de la clave, que permite un buen compromiso entre el nivel de seguridad y la complejidad computacional. En la solución propuesta, los datos recopilados en un nodo Fog se cifran, autentican y dispersan de forma pseudoaleatoria a sus n nodos Fog vecinos. Para la recuperación de datos originales, es necesario cualquier k de los n fragmentos junto con la clave dinámica correspondiente. Esto complica al atacante que necesita comprometer al menos k nodos Fog para revelar los datos encriptados. Además, debe buscar la clave dinámica, que es diferente para cada dato de entrada. Por otro lado, los fragmentos redundantes protegen los datos almacenados contra hasta $(n - k)$ fallas de los nodos Fog o indisponibilidad. Las pruebas de análisis de seguridad y desempeño muestran que el esquema de seguridad propuesto exhibe un alto nivel de eficiencia y robustez.

Pasha et. al [32] presentó un modelo de protección de datos para Fog Computing para proteger los datos y manejar la movilidad que utiliza cifrado simétrico y clave pública. El mismo se trata de un modelo *RBTA (Region-Based Trust-Aware)* para traducir la confianza entre los nodos Fog de regiones, un *Control de Acceso Basado en Roles basado en la privacidad basado en la niebla (FPRBAC)* para el control de acceso en los nodos Fog, un servicio de gestión de movilidad para gestionar los cambios de usuarios y las ubicaciones de los dispositivos Fog. Para implementar

nuestro marco en la práctica, es necesario proporcionar alta disponibilidad de servicios y recursos Fog y servicios de ubicación actualizados para abordar el requisito de respuesta sensible. Los resultados experimentales demostraron la viabilidad y eficiencia del modelo.

La verificación de la integridad de los datos en Fog Computing es otro aspecto más a tener en cuenta para aumentar la seguridad. Existe la posibilidad de que los sistemas puedan ser infiltrados por usuarios maliciosos y, como resultado, los datos podrían corromperse, lo cual es causa de inquietud. Una vez que un atacante altera un conjunto de elementos de datos, el daño puede propagarse por la base de datos. Cuando las transacciones válidas leen datos dañados, pueden actualizar otros elementos de datos en función del valor leído. Dada la naturaleza sensible de los datos, por ejemplo, en el área de la salud, y la necesidad crítica de proporcionar acceso en tiempo real para la toma de decisiones, es vital que cualquier daño causado por una transacción maliciosa y la propagación por transacciones válidas sea corregido inmediatamente y con precisión. Un enfoque para garantizar la integridad de los datos es la posesión de datos comprobables (*PDP, Prueba de Posesión*), como se describe en Ateniese et al., (2007) [33]. Este método permite a un cliente verificar que el servidor posee sus datos, sin tener que recuperar todos los datos. El modelo implementa un protocolo de desafío/respuesta entre cliente y servidor, en el cual el servidor tiene que entregar una llamada Prueba de Posesión. En lugar de una prueba determinista, esta prueba es probabilística, ya que en este protocolo no es posible verificar cada bit de datos, esto requeriría tener que descargar todo el conjunto de datos. En su lugar, se verifican muestras aleatorias de bloques. El objetivo del PDP es, por lo tanto, detectar el mal comportamiento de un servidor cuando ya no posee el archivo completo. A diferencia de otros métodos, PDP requiere que el cliente almacene una cantidad pequeña y constante de metadatos localmente, para minimizar la comunicación de red. Esto permite verificar grandes conjuntos de datos, distribuidos en muchos dispositivos de almacenamiento.

MANET (Redes Ad hoc Móviles, Mobile Ad hoc NETWORKS) es un paradigma novedoso de una red ad hoc móvil con las ventajas tanto de la movilidad como de Fog Computing. El protocolo de enrutamiento de AODV (*Vector de Distancia Bajo demanda Ad hoc, Ad hoc On-demand Distance Vector*) se ha aplicado ampliamente en MANET. Fang et. al [34] proponen un protocolo de enrutamiento AODV ligero y

seguro SAL-SAODV (*Source Anonymity-based Lightweight Secure AODV*) basado en el anonimato de la fuente. El esquema consta de las siguientes tres partes: el algoritmo de anonimato de origen se emplea para alcanzar el nodo de origen, sin ser rastreado ni localizado; se aplica el esquema seguro mejorado basado en el polinomio de CRC-4 para sustituir la firma digital RSA de SAODV y garantizar la integridad de los datos, además de reducir el consumo de energía (en un 35% según las pruebas realizadas) y de cómputo; el esquema *RDTM (Transmisión Retardada Aleatoria, Random Delayed Transmitting Scheme)* se implementa para separar el código de verificación y los datos transmitidos, y lograr resultados a prueba de manipulaciones. Los resultados de la simulación muestran que el rendimiento integral del SAL-SAODV propuesto es una compensación del rendimiento de la transmisión, la eficiencia energética y la seguridad, y mejor que AODV y SAODV. Cabe destacar que la latencia de un extremo a otro aumentará debido al cambio aleatorio del esquema de transmisión retardada. Por lo tanto, el protocolo SAL-SAODV es adecuado para los escenarios, que son insensibles a los retrasos, pero tienen un alto requisito de seguridad.

Mubarakali et. al [35] presenta el algoritmo *FDSDT (Transmisión de Datos Sensibles al Retraso basado en Fog, Fog-based Delay-Sensitive Data Transmission)*, cuyo objetivo es acelerar el proceso de acceso a los datos y reducir el retraso de las aplicaciones sensibles al tiempo. Los nodos Fog en el modelo propuesto son nodos de almacenamiento efímero, es decir, están diseñados para almacenar sólo datos recientes, no datos históricos. Por lo tanto, se ha establecido un valor de umbral en el sistema. En el extremo del remitente, la técnica primero calcula el valor de la suma de comprobación del archivo que se transferirá. Es crucial para las aplicaciones urgentes en las que los datos deben llegar al cliente en el menor tiempo posible. En este estudio, el sistema se ha construido e implementado físicamente en un prototipo basado en Fog Computing y se ha probado con éxito en aplicaciones sensibles al tiempo que puede reducir significativamente el retraso. Los resultados experimentales del algoritmo FDSDT propuesto muestran que cuando la nube está relativamente lejos del usuario, incurre en una importante cantidad de demora, y eso podría ser perjudicial para aplicaciones sensibles a la demora, como la atención médica. El algoritmo FDSDT minimiza 15,67 ms el *RTT (Tiempo de Viaje, Round Trip Time)* y 2 s *AD (Retardo Promedio, Average Delay)* para conjuntos de datos de 10 KB, 100 KB y 1

MB en las regiones de los datacenter de Amazon de India, Singapur y Japón comparadas con metodologías convencionales.

Johney et. al [36] propone que el servidor Fog recopile y transfiera los datos cifrados del usuario, y aplique el código Reed Solomon para la detección de errores y hash MD5 para la detección de mensajes maliciosos, logrando así garantizar una mayor seguridad e integridad. Estos datos se almacenarán en diferentes servidores en la nube y en los respectivos dispositivos Fog del usuario. Por lo tanto, la combinación de la nube con Fog Computing garantizará la integridad, disponibilidad y confidencialidad de los datos de los usuarios.

III. Metodología

A continuación, se describen las etapas que se llevaron a cabo para alcanzar los objetivos definidos.

1. En la primera etapa se realizó un estudio de los algoritmos criptográficos que mejor se adecuan para brindar confidencialidad e integridad en la capa Fog, donde se destacaron las ventajas y desventajas de cada uno de ellos. Al realizar la comparación entre ellos, se seleccionaron los más adecuados en términos de velocidad de lectura/escritura y consumo de recursos.
2. En la segunda etapa se configuraron los nodos Fog sobre los cuales se analizaron los algoritmos seleccionados en el punto 1.
3. En la tercera etapa se crearon los scripts integrando esquemas criptográficos, haciendo uso de algoritmos orientados a la confidencialidad y otros algoritmos dirigidos a la integridad de los datos. Se evaluó el desempeño de cada uno de ellos, aplicándolos sobre conjuntos de datos de distinto tamaño y tipo (texto, imágenes y video).
4. Por último, en la cuarta etapa se realizaron las pruebas sobre los nodos Fog utilizando los scripts generados y se presentaron los resultados obtenidos para las métricas de tiempo de ejecución, consumo de CPU, memoria y memoria externa. También se midió el consumo de energía en uno de los nodos, primero en reposo y luego para algunas de las combinaciones de algoritmos seleccionados.

A continuación se explican la primera, segunda y tercera etapa. De la cuarta etapa solo se realiza una síntesis de las pruebas realizadas y se explican luego detalladamente en el capítulo Resultados.

Primera Etapa

A partir de la información obtenida durante la confección del marco teórico y para poder cumplir con los objetivos propuestos durante este trabajo de investigación, se decidió comenzar seleccionando una serie de algoritmos para analizar teniendo en cuenta cuales eran los más utilizados, los más rápidos, seguros y que empleaban menos recursos. También se investigó qué librerías de código abierto existen y que algoritmos tienen disponibles.

Librería de funciones criptográficas

OpenSSL

OpenSSL es un proyecto open-source o de código abierto, que implementa funciones criptográficas dentro de una librería y que provee un programa que permite realizar por línea de comandos diversas tareas, como encriptar y desencriptar un archivo, calcular su hash, etc. Su librería permite implementar los protocolos de seguridad web SSL, TLS y otra serie de operaciones como la criptografía simétrica. Esta serie de operaciones y protocolos tienen como objetivo proteger la seguridad de la información entre dos partes. El código fuente puede ser descargado en la página oficial de Openssl en www.openssl.org.

Entre las funciones que nos ofrece OpenSSL se encuentran las siguientes:

- Creación de claves con el sistema de cifrado asimétrico.
- Intercambio de claves con el algoritmo Diffie-Hellman.
- Creación de firmas con DSA, un algoritmo derivado del RSA, pero especializado en generar firmas digitales.
- Creación de certificados para páginas web
- Calcular funciones hash o funciones de resumen.
- Cifrado simétrico y asimétrico

- Prueba de clientes y servidores SSL/TLS, con el fin de autenticar dichos certificados en páginas web.
- Gestión de correo electrónico firmado digitalmente para comprobar la autenticidad y garantizar el parámetro de no-repudio de los mensajes.
- Una alternativa a OpenSSL es GNUPG.

GNUPG

GnuPG (GNU Privacy Guard) es una implementación completa y gratuita del estándar OpenPGP definido por RFC4880 (también conocido como PGP). GnuPG es un software libre que permite encriptar y firmar datos y comunicaciones; cuenta con un versátil sistema de gestión de claves, junto con módulos de acceso a todo tipo de directorios de claves públicas. GnuPG, también conocido como GPG, es una herramienta de línea de comandos con funciones para una fácil integración con otras aplicaciones. Hay disponible una gran cantidad de aplicaciones y bibliotecas frontend. GnuPG también brinda soporte para S/MIME y Secure Shell (ssh).

Para este trabajo se optó por usar Openssl ya que es de fácil instalación y utilización y dispone de una amplia variedad de algoritmos entre los que se incluye la mayoría de los que se nombraron durante el marco teórico.

Algoritmos criptográficos

Los algoritmos seleccionados para las pruebas experimentales son los siguientes:

- *Algoritmos simétricos*
 - AES-256-CBC (cifrado de bloque)
 - 3DES (cifrado de bloque)
 - CHACHA20 (cifrado de flujo)

En base en los resultados de investigaciones anteriores se seleccionaron estos tres algoritmos que están disponibles en la librería Openssl por ser más rápidos, y que requieren menor consumo de CPU. Respecto al modo de cifrado se eligió la modalidad CBC, ya que es más segura que ECB y está disponible tanto para AES como para 3DES, lo que permite su comparación.

- *Algoritmos asimétricos*

- RSA
- Curvas Elípticas (X25519) (ECDH)
- Curvas Elípticas (secp256k1) (ECDSA)

Se eligió RSA por ser el más popular y utilizado actualmente, y dos algoritmos de curvas elípticas que en algunas investigaciones, tales como [14] y [18] recomiendan su uso para Fog Computing por consumir menos memoria y CPU.

- *Protocolo de establecimiento de clave*

- Diffie-Hellman

Se eligió un algoritmo de cifrado híbrido ya que se consideran más seguros que los algoritmos simétricos y asimétricos y a partir de las pruebas en los nodos se pretendía poder determinar si es significativa la diferencia de rendimiento al incrementar la seguridad. Diffie-Hellman es el más conocido y está disponible en OpenSSL.

- *Resumen o Suma de Verificación*

- SHA-1
- SHA-512
- SHA3-512
- Ripemd-160

Ripemd-160 se seleccionó a modo de comparación con SHA-1, ya que ambos ofrecen niveles de seguridad similares, pero tienen diferente origen, (RIPEMD-160 fue diseñado en la comunidad académica abierta, mientras que SHA-1 fue diseñado por la NSA), y el primero ha sido menos estudiado, tal como se mencionó en el capítulo anterior. Ambos fueron reemplazados por las versiones de SHA-2 (como SHA-256 y SHA-512) y SHA-3. Para este trabajo de investigación se eligió a SHA-512 que es una versión mejorada y es más segura que SHA-1, y a SHA3-512 que es aún más robusto, pero no deriva de SHA-1 y SHA-2.

Segunda Etapa

Durante esta etapa se evaluó que equipos eran de fácil acceso o estaban disponibles en la universidad y que permitieran por sus características técnicas simular nodos Fog. Se consiguieron tres Raspberry Pi que se pusieron en red y se configuraron para poder realizar las pruebas necesarias.

Configuración de los Nodos Fog

Para simular los nodos Fog se decidió utilizar tres modelos de Raspberry Pi con diferentes características y especificaciones técnicas, principalmente de CPU y memoria con el objetivo de poder comparar los tiempos de ejecución de los algoritmos y el consumo de los diferentes recursos.

Raspberry Pi

Raspberry Pi es una computadora de tamaño reducido (placa de 85 x 54 mm) y bajo coste desarrollado en Reino Unido por la fundación Raspberry Pi, con el objetivo principal de estimular la enseñanza de las ciencias de la computación a niños en sus colegios y a personas de países subdesarrollados.

Todo modelo contiene al menos: un puerto de salida de video HDMI, otro de tipo RCA, minijack para conectar audio y un puerto USB 2.0 (la versión 3 Modelo B dispone de 4 USB) donde se puede conectar un monitor y un mouse, además de un puerto Ethernet.

Dentro de Raspberry Pi se pueden encontrar diferentes modelos:

Modelo	RPI 1 Model B+	RPI 2 Model B	RPI 3 Model B	RPI Zero W	RPI 4
CPU	ARM1176JZF-S (700 MHz)	ARM Cortex A7 Quad Core (900 MHz)	ARMv8 64-bit Quad-Core (1.2 GHz)	ARM11 (1GHz)	ARM Cortex-A72 (1,5 GHz)
GPU	Broadcom VideoCore IV	Broadcom VideoCore IV	Broadcom VideoCore IV	No	VideoCore VI
Memoria RAM	512 MB	1 GB	1 GB	512 MB	1 GB/2 GB/4 GB LPDDR4 SDRAM
USB	4 USB 2.0	4 USB 2.0	4 USB 2.0	1 Micro USB	2 x USB 2.0 2 x USB 3.0
Ethernet	Sí	Sí	Sí	Sí	Sí
Salida de vídeo	HDMI, RCA	HDMI, RCA	HDMI, RCA	HDMI	HDMI
Salida de audio	3.5 mm Jack, HDMI	3.5 mm Jack, HDMI	3.5 mm Jack, HDMI	HDMI	Jack, Micro HDMI
Almacenamiento	Micro SD	Micro SD	Micro SD	Micro SD	Micro SD
Tamaño	85.6 x 56 mm	85.6 x 56 mm	85.6 x 56 mm	65 x 30 mm	88 x 58 mm
Consumo energético					
Boot (Max)	0.26	0.4	0.75	0.20	0.85
Boot (Avg)	0.22	0.22	0.35	0.15	0.7
Idle	0.20	0.22	0.30	0.10	0.6
Stress	0.35	0.82	1.34	0.35	1.25

Comparación entre los modelos de Raspberry Pi.

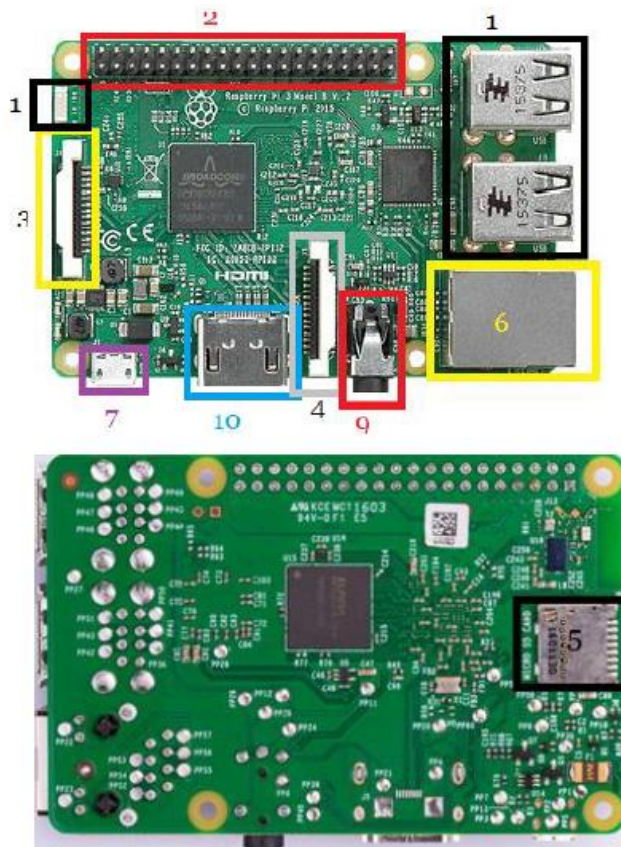
Cabe mencionar que al momento de finalización de esta tesis salió al mercado la RPI 5 (Octubre 2023), dicho modelo no está incluido en la anterior tabla.

Como se puede comprobar, la Raspberry Pi 4 es la que mejores prestaciones ofrece de entre los modelos comparados, seguida por la Raspberry Pi 3. Sobre esta última se realizaron todas las pruebas, ya que se disponía de acceso completo a la misma durante el desarrollo de la investigación. Sobre la Raspberry Pi 4 solo se pudieron realizar las ejecuciones de los programas, pero no las mediciones de consumo, ya que solo se consiguió a préstamo por unos días.

A continuación, se detallan los componentes existentes en la RPI 3 modelo B:

1. *Puertos USB*: posee cuatro puertos USB del estándar 2.0 a los que se suelen conectar dispositivos externos para manejar la Raspberry, como un teclado o monitor.

2. *Pines GPIO*: estos pines sirven de interfaz entre la Raspberry Pi y el exterior. Contiene 40 pines, entre los que se encuentran pines del tipo UART, I2C o SPI para manejar dispositivos como luces LED, sensores, etc.
3. *Conector DSI*: permite la conexión de displays o pequeñas pantallas LCD al dispositivo.
4. *Conector CSI*: conector tipo bus de 15 pines que permite añadir un dispositivo compatible con la interfaz CSI-2, y mediante el cual se conecta la cámara.
5. *Almacenamiento*: el aparato contiene un lector de tarjetas de memoria.
6. *microSD*. El arranque del sistema se hará desde la propia tarjeta, por lo que es conveniente que su tamaño sea de al menos 16GB de capacidad.
7. *Puerto Ethernet*: dispone de un puerto Ethernet para enchufar un cable RJ-45 directamente al router, aunque en la última versión no es necesario ya que tiene integrada conexión WiFi 802.11n y Bluetooth 4.1.
8. *Alimentación*: integra un conector micro USB, con el que la mayoría de los cargadores para smartphone son compatibles (5V, mayor de 700mA).
9. *Salidas de audio*: posee un conector de audio Jack de 3,5mm, además de la posibilidad de usar el propio HDMI.
10. *Conector HDMI*: permite la conexión de dispositivos como monitores para la extracción de vídeo y ser usado como salida de audio.
11. *Broadcom BCM2837*: el diseño incluye un procesador central ARMv8 de 64 bit a 1.2GHz, una GPU VideoCore IV y 1 GB de memoria RAM.



Nº	Identificador
1	Salida UDB 2.0
2	Pines GPIO
3	Conector DSI Display Module
4	Conector CSI Camera Module
5	Ranura para tarjeta Micro SD
6	Puerto Ethernet (rj45)
7	Alimentación micro USB
8	Leds Indicadores
9	Salida de audio
10	Salida HDMI

Figura 2 - Raspberry Pi 3 Modelo B. Recuperada de: <https://comohacer.eu/wp-content/uploads/2016/05/partes-raspberry-pi-830x524.jpg>

Terminales GPIO

La principal diferencia entre la Raspberry Pi y los demás SBC (computadoras de placa reducida) es la incorporación de los pines GPIO, que actúan como una interfaz física entre ella y el mundo exterior, y que pueden configurarse como entradas o salidas para controlar sensores, dispositivos externos, etc. Entre los terminales GPIO se encuentran otros dos tipos de pines: los SPI y los I²C.

El bus SPI es un estándar de comunicaciones usado para la transferencia de información entre circuitos integrados en equipos electrónicos. Tiene una arquitectura de tipo maestro-esclavo. Incluye una línea de reloj, dato entrante, dato saliente y un pin CS, que conecta o desconecta la operación del dispositivo con el que uno desea comunicarse.

Por otra parte, los I²C son buses de comunicación en serie diseñados para conectar diferentes partes de un circuito, por ejemplo, entre un controlador y circuitos periféricos integrados.

Su principal característica es que utiliza dos líneas para transmitir la información, una para los datos (SDA) y otra para la señal de reloj (SCL), además de una tercera línea de referencia (GND).

Pin#	NAME		NAME	Pin#
01	3.3v DC Power		DC Power 5v	02
03	GPIO02 (SDA1 , I ² C)		DC Power 5v	04
05	GPIO03 (SCL1 , I ² C)		Ground	06
07	GPIO04 (GPIO_GCLK)		(TXD0) GPIO14	08
09	Ground		(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)		(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)		Ground	14
15	GPIO22 (GPIO_GEN3)		(GPIO_GEN4) GPIO23	16
17	3.3v DC Power		(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)		Ground	20
21	GPIO09 (SPI_MISO)		(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)		(SPI_CE0_N) GPIO08	24
25	Ground		(SPI_CE1_N) GPIO07	26
27	ID_SD (I ² C ID EEPROM)		(I ² C ID EEPROM) ID_SC	28
29	GPIO05		Ground	30
31	GPIO06		GPIO12	32
33	GPIO13		Ground	34
35	GPIO19		GPIO16	36
37	GPIO26		GPIO20	38
39	Ground		GPIO21	40

Figura 3 - Pines Raspberry Pi 3 Modelo B. Recuperada de: <https://www.element14.com/community/docs/DOC-73950//raspberry-pi-3- model-b-gpio-40-pin-block-pinout>

Configuración inicial

La Raspberry Pi es una computadora pequeña que necesita un sistema operativo para arrancar todos los procesos necesarios, para que las capacidades de hardware y de computación estén disponibles para las aplicaciones que se deseen ejecutar. Como no posee unidad de almacenamiento interna, se debe grabar ese sistema operativo en una tarjeta micro-SD. El sistema operativo escogido para este proyecto es Debian, cuya versión para Raspberry Pi se conoce por Raspbian, es la oficial y más utilizada y se puede descargar en forma gratuita.

Para las pruebas se utilizó la Raspberry Pi 1 (poco usada actualmente), la Raspberry Pi 3 (ampliamente usada en nuestro país por una relación costo - recursos ofrecidos) y la Raspberry Pi 4.

OpenSSL

En los nodos se instaló el paquete OpenSSL. La última versión disponible de este software se puede consultar en el sitio oficial y realizar desde allí la descarga. Para ello se puede utilizar el comando *wget* desde la terminal de la RPI.

```
wget https://www.openssl.org/source/openssl-3.0.7.tar.gz
```

Una vez finalizada la descarga, se descomprimen los archivos.

```
tar -zxvf openssl-3.0.7.tar.gz
```

El siguiente paso es cambiarse de directorio a la nueva carpeta

```
cd openssl-3.0.7
```

Para asegurarse de contar con todas las dependencias para compilar los archivos, utilizar

```
sudo apt install build-essential
```

Por último, correr los siguientes comandos para la instalación

```
./Configure
```

```
make
```

```
sudo make install
```

Para verificar la versión instalada se puede utilizar el siguiente comando

```
openssl version
```

Tercera Etapa

En esta etapa el enfoque estuvo en definir los datos a utilizar, las métricas a medir y generar los scripts necesarios para realizar las pruebas en los nodos configurados en la etapa anterior.

Set de datos

En base a la literatura y a las investigaciones encontradas durante la confección del marco teórico, se pudo observar que las pruebas para comparar los

diferentes algoritmos utilizaban en la mayoría de los casos archivos en texto plano menores a 10 MB, como Patel et. al [13], que utilizó un archivo .csv con información médica de pacientes, mientras que, muy pocos papers emplearon un conjunto de datos no estructurados, como Vishwanath et. al [12] que utilizaron un conjunto de datos con imágenes, caracteres, números y representaciones pictóricas de 10 MB. Para el presente trabajo se eligieron tres tipos de datos para realizar las pruebas: texto, imágenes y video. Teniendo en cuenta de que en IoT la mayoría de los datos enviados entre los nodos no son de gran tamaño, no se seleccionaron archivos superiores a los 100 MB para las comparaciones.

A continuación se detallan los archivos seleccionados. Los datos de texto plano están en formato .csv y corresponden a bases de datos extraídos de sensores IoT que se encuentran disponibles en internet. En el caso de las imágenes, se usaron tres formatos (.jpg, .csv y .png) y son fotos captadas por cámaras IP, mientras que los videos, en formato .mp4 corresponden a filmaciones realizadas por drones.

- *Texto CSV*
 - transporte (1.1MB)
 - lot-telemetry1 (50 MB)
 - home (100 MB)
- *Video MP4*
 - oveja_70858 (1.1MB)
 - campo (50 MB)
 - girasoles (100 MB)
- *Imagen*
 - *JPG: temp_nubes2 (1.1MB)*
 - *PNG: shelter_ingreso_50 (50 MB)*
 - *BMP: peaje_100 (100 MB)*

Métricas

Teniendo en cuenta que los nodos Fogs están limitados en recursos (de almacenamiento, cómputo, memoria y energía) se tuvieron en cuenta para las pruebas a realizar las siguientes métricas:

- Consumo de CPU: mide el porcentaje de la CPU mediante el uso de la herramienta perf.
- IOWait: mide el porcentaje de tiempo que la CPU permanece en estado IOWait (a la espera de la finalización de operaciones de Entrada/Salida).
- Uso de Memoria: mide el porcentaje de uso de la memoria RAM con la herramienta perf y ps.
- Accesos a memoria externa: contabiliza la cantidad de accesos a la memoria externa utilizando la herramienta perf.
- Tiempo de ejecución: se mide en segundos con el comando time.
- Consumo de energía: se utiliza un medidor de consumo usb para determinar el consumo del nodo Fog durante las pruebas. El medidor utilizado es de la marca KEWEISI que permite medir el nivel de tensión en Volts y el consumo de corriente en Amperes. Antes de comenzar con las mediciones, se determinó el consumo en estado de reposo de la RPI y sin ningún sensor conectado. Por cuestiones de disponibilidad, esta métrica solo se pudo medir en la RPI 3, arrojando un consumo de 250 mA en estado de reposo.

Las métricas se midieron sobre los conjuntos de datos anteriormente mencionados. En el caso de la desenscriptación, las mediciones se realizaron sobre el archivo encriptado, con lo cual el tamaño del archivo era mayor al original, pero a fines de una mejor representación de los resultados, se tabularon con el tamaño original.

Para obtener una estadística, de cada prueba se realizaron 5 ejecuciones para obtener un promedio de cada métrica. Cada script guarda en un archivo de salida tipo csv (comma-separated values) los resultados de cada prueba, de la siguiente forma:

Archivo, algoritmo simétrico, tipo de datos, tamaño, prueba, n° repetición, algoritmo suma de verificación, tiempo en segundos, % cpu utilizado, % memoria, %memoria externa % llamadas de lectura, tiempo de lectura (seg), %llamadas de escritura, tiempo de escritura (seg), tiempo empleado en llamadas al sistema (seg).

Parámetros y métricas almacenadas en cada prueba

Se muestra a continuación un ejemplo de los resultados obtenidos para un archivo:

**shelter_ingreso_50.png, -chacha20, imagen, 50M, Encriptacion, 1, sha3-512, 27.036329199, 0.259, 1.35, 0, 1.052156, 71.96, 0.409948, 28.04, 1.462104
shelter_ingreso_50.png, -chacha20, imagen, 50M, Suma de Verificacion, 1, sha3-512, 37.188495071, 0.456, 1.25, 0, 0.088823, 86.84, 0.004756, 4.65, 0.102289
shelter_ingreso_50.png.encrypted, -chacha20, imagen, 50M, Desencriptacion, 1, sha3-512, 31.246443301, 0.316, 1, 0, 0.761493, 51.09, 0.720623, 48.34, 1.490592**

Ejemplo de datos almacenados en cada prueba

Scripts

Los scripts fueron escritos en lenguaje Bash y las funciones de encriptación, desencriptación y suma de verificación se usaron de la librería OpenSSL.

Los scripts utilizados tienen la siguiente estructura:

1. Se genera clave privada.
 - 1.1. Para cifrar utilizando el algoritmo asimétrico se debe crear una clave pública y una clave privada. Primero se genera la clave privada con el algoritmo asimétrico seleccionado.
2. Se genera clave pública
 - 2.1. A partir de la clave privada generada en el punto 1 se genera la clave pública.

3. Encriptación

- 3.1. Se encripta con clave.
- 3.2. Se encripta un archivo del set de datos utilizando un algoritmo simétrico y una contraseña con la función *“openssl enc”*.
- 3.3. Se repite la encriptación para poder obtener las métricas aplicando las herramientas *ps*, *perf* y *strace* incluidas en la distribución Linux raspbian.
- 3.4. Se extraen las métricas seleccionadas y se guardan en un archivo.

4. Suma de verificación y Hmac

- 4.1. Se aplica la función *digest* y *sign* o *hmac*.

Un algoritmo de resumen de mensajes toma una única entrada (un mensaje) y produce un "resumen de mensaje" o hash que le permite verificar la integridad del mensaje. Este hash se puede generar mediante la función *digest*. Cualquier cambio en el mensaje (idealmente) dará como resultado un hash diferente. Un atacante no solo debe reemplazar el mensaje sino también el resumen por un nuevo par válido.

Un algoritmo MAC (Message Authentication Code, código de autenticación del mensaje) toma dos entradas, un mensaje y una clave secreta, y produce un MAC que le permite verificar la integridad y autenticidad del mensaje: cualquier cambio en el mensaje o en la clave secreta (idealmente) resultará en un MAC diferente. Nadie sin acceso a la clave debería poder generar un cálculo de MAC que lo verifique.

La función HMAC o “hash con clave” combina el MAC con el resultado de una función hash. Es una forma de ampliar el uso de las funciones hash ya que el resultado no depende solo del hash del mensaje sino de otro parámetro que puede ser la clave secreta del usuario.

- 4.2. Se calcula la suma de verificación del archivo encriptado utilizando *openssl dgst* y se firma con la clave privada utilizando *sign* para enviarlo al nodo Fog destino. Para los algoritmos que no soportan firma, se

calcula sobre el archivo encriptado *hmac* con contraseña para autenticar el mensaje.

4.3. Se repite el comando para poder obtener las métricas aplicando las herramientas *ps*, *perf* y *strace*.

4.4. Se extraen las métricas seleccionadas y se guardan en un archivo.

5. Descriptación

5.1. Se verifica la integridad del archivo con la clave pública. Para los algoritmos que no soportan firma, se calcula *hmac* sobre el archivo encriptado y se compara con el recibido.

5.2. Se descripta usando el mismo algoritmo simétrico y la misma contraseña usada para la encriptación utilizando la función “*openssl enc -d*”.

5.3. Se repite la descriptación para poder obtener las métricas aplicando las herramientas *ps*, *perf* y *strace*.

5.4. Se extraen las métricas seleccionadas y se guardan en un archivo.

Cuarta Etapa

Durante esta etapa se realizaron las pruebas y mediciones sobre los nodos Fog configurados en la etapa 2. Para ello se utilizaron los conjuntos de datos, métricas y scripts mencionados en la etapa 3.

Se ejecutaron en el nodo 1 y 3 cinco repeticiones de cada script para cada combinación de algoritmo asimétrico, simétrico y de resumen seleccionados en la etapa 1. Los resultados obtenidos se promediaron y en base ellos se realizaron tablas comparativas para determinar qué combinación ofrecía mejores resultados.

De acuerdo con los resultados tabulados en la RPI 3, se seleccionaron dos combinaciones de algoritmos que se aplicaron un tipo y tamaño de archivo para realizar las mediciones de consumo de energía. El archivo utilizado fue *girasoles.mp4* de 100MB.

Se seleccionaron también algunas combinaciones para ejecutar los scripts en una Raspberry Pi 1, puesto que al ser un nodo con especificaciones técnicas inferiores a las Raspberry Pi 3 y 4, no se consideró necesario realizar todas las pruebas.

Todos los resultados obtenidos en la cuarta etapa se muestran y analizan en el siguiente capítulo.

IV. Resultados

En esta sección se analizan los resultados obtenidos para cada una de las pruebas realizadas con las diferentes configuraciones utilizadas, indicando:

- Nodo utilizado
- Algoritmo asimétrico
- Algoritmo simétrico
- Métrica
- Función: Encriptación, Desencriptación o Suma de Verificación. (Cabe aclarar que, si bien en los scripts no están declaradas tales funciones, se va a utilizar en lo sucesivo este término para hacer referencia al conjunto de instrucciones del código asociadas a realizar y almacenar los resultados de la ejecución de estas tareas).
- Tipo de datos
- Tamaño de datos

En cada tabla se resalta el valor mínimo (en color celeste) y el valor máximo (en color rosa) obtenido de cada métrica.

Nodo 1 – Raspberry Pi 3

Porcentaje de Uso de Memoria

En esta primera parte de la sección se presentan las Tablas 1 a 4 con los resultados experimentales obtenidos con respecto al porcentaje de memoria utilizado por cada función según el algoritmo empleado, y el tipo y tamaño de archivo analizado. Las tablas han sido ordenadas en base al algoritmo asimétrico empleado.

Cabe recordar que el modelo Raspberry Pi 3 cuenta con 1GB de memoria RAM disponible.

Algoritmo de Verificación SHA3-512

Se reflejan en las cuatro tablas siguientes los resultados obtenidos de consumo de memoria RAM por cada una de las funciones utilizando el algoritmo de verificación SHA3-512.

En color celeste se resaltan aquellos valores que han obtenido los menores valores de consumo de memoria y, por ende, requieren menos recursos hardware a la hora de ejecutar las funciones. En color rosa los valores que indican el mayor consumo de recursos de memoria. Este formato se mantendrá a lo largo de todas las tablas.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	0.44 %	0.44 %	0.45 %	0.16 %	0.44 %	0.45 %	0.43 %	0.45 %	0.45 %
	Encriptación	0.45 %	0.38 %	0.45 %	0.45 %	0.45 %	0.45 %	0.47 %	0.45 %	0.45 %
	Suma de Verificación	0.29 %	0.35 %	0.35 %	0.27 %	0.25 %	0.27 %	0.29 %	0.29 %	0.24 %
chacha20	Desencriptación	0.45 %	0.44 %	0.45 %	0.27 %	0.45 %	0.45 %	0.17 %	0.44 %	0.45 %
	Encriptación	0.41 %	0.28 %	0.45 %	0.45 %	0.38 %	0.45 %	0.45 %	0.45 %	0.22 %
	Suma de Verificación	0.27 %	0.40 %	0.43 %	0.38 %	0.25 %	0.25 %	0.27 %	0.29 %	0.27 %
des3	Desencriptación	0.43 %	0.45 %	0.45 %	0.27 %	0.45 %	0.45 %	0.43 %	0.45 %	0.45 %
	Encriptación	0.37 %	0.38 %	0.45 %	0.45 %	0.45 %	0.45 %	0.45 %	0.45 %	0.35 %
	Suma de Verificación	0.29 %	0.32 %	0.40 %	0.25 %	0.26 %	0.25 %	0.25 %	0.25 %	0.25 %

Tabla 1 – Porcentaje de uso de memoria RAM total utilizando algoritmo ECDSA y SHA3-512

En la Tabla 1 podemos observar que en la combinación del algoritmo asimétrico de Curvas Elípticas DSA (ECDSA) con el algoritmo simétrico AES-256-CBC el proceso de desencriptación de un archivo de texto de 1MB es el que obtiene el menor valor de consumo de memoria RAM, mientras que la encriptación de un archivo de video de 1MB es el resultado más alto. En el conjunto de datos de imágenes, la suma de verificación obtiene los menores valores de consumo seguidos, por la encriptación y la desencriptación. Este mismo comportamiento se repite para los datos de texto y video.

Ahora bien, si evaluamos Chacha20 y DES3, podemos observar que la suma de verificación consume menos memoria RAM que las funciones de encriptación y desencriptación. En cada uno de los conjuntos de datos se observa ocupación de memoria similar, no existiendo diferencias significativas. También, se observa que en ninguno de los casos el sistema hace un uso mayor al 0.47% de los recursos de memoria del dispositivo de cómputo.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	0.45 %	0.45 %	0.45 %	0.44 %	0.45 %	0.45 %	0.38 %	0.43 %	0.45 %
	Encriptación	0.45 %	0.45 %	0.45 %	0.47 %	0.45 %	0.44 %	0.51 %	0.45 %	0.45 %
	Suma de Verificación	0.35 %	0.43 %	0.41 %	0.34 %	0.27 %	0.25 %	0.27 %	0.27 %	0.33 %
chacha20	Desencriptación	0.45 %	0.45 %	0.43 %	0.45 %	0.45 %	0.45 %	0.43 %	0.45 %	0.45 %
	Encriptación	0.41 %	0.45 %	0.37 %	0.45 %	0.45 %	0.45 %	0.39 %	0.45 %	0.45 %
	Suma de Verificación	0.31 %	0.46 %	0.41 %	0.28 %	0.29 %	0.29 %	0.29 %	0.27 %	0.42 %
des3	Desencriptación	0.44 %	0.45 %	0.44 %	0.46 %	0.45 %	0.45 %	0.45 %	0.45 %	0.45 %
	Encriptación	0.46 %	0.41 %	0.45 %	0.45 %	0.45 %	0.45 %	0.47 %	0.45 %	0.43 %
	Suma de Verificación	0.31 %	0.33 %	0.36 %	0.34 %	0.25 %	0.27 %	0.25 %	0.27 %	0.33 %

Tabla 2 - Porcentaje de uso de memoria RAM total utilizando algoritmo RSA y SHA3-512

En los resultados de la Tabla 2 utilizando el algoritmo RSA podemos notar que para el algoritmo AES-256-CBC el proceso de encriptación de un archivo de 1 MB es el valor más alto. En el conjunto de datos de imágenes, la encriptación y desencriptación con dicho algoritmo requieren el mismo porcentaje de memoria para los tres tamaños de archivo. La suma de verificación obtiene menor valor de consumo para los tres conjuntos de datos, aunque en el conjunto de imágenes, el consumo es ligeramente inferior al de la encriptación y desencriptación.

En el caso de Chacha20, el conjunto de datos de texto requiere el mismo porcentaje de memoria tanto para encriptación como desencriptación, mientras que el de suma de verificación es menor. Para los conjuntos de datos de imágenes y videos los porcentajes obtenidos en las tres funciones son similares.

Al evaluar DES3, podemos observar el mismo comportamiento que Chacha20 en el caso del conjunto de datos de texto. Para los archivos de imágenes y video, la suma de verificación consume menos memoria que las funciones de encriptación y desencriptación.

Además, se puede destacar que en ninguna de las pruebas el sistema hace un uso mayor al 0.51% de los recursos de memoria de la RPI 3.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	0.47 %	0.47 %	0.47 %	0.37 %	0.47 %	0.47 %	0.40 %	0.47 %	0.47 %
	Encriptación	0.43 %	0.41 %	0.47 %	0.47 %	0.47 %	0.47 %	0.47 %	0.47 %	0.46 %
	Suma de Verificación	0.45 %	0.44 %	0.45 %	0.43 %	0.44 %	0.44 %	0.44 %	0.43 %	0.47 %
chacha20	Desencriptación	0.47 %	0.47 %	0.47 %	0.42 %	0.47 %	0.47 %	0.42 %	0.47 %	0.47 %
	Encriptación	0.47 %	0.42 %	0.47 %	0.42 %	0.47 %	0.47 %	0.47 %	0.47 %	0.47 %
	Suma de Verificación	0.43 %	0.46 %	0.47 %	0.43 %	0.43 %	0.44 %	0.44 %	0.44 %	0.42 %
des3	Desencriptación	0.47 %	0.47 %	0.47 %	0.40 %	0.47 %	0.47 %	0.38 %	0.47 %	0.47 %
	Encriptación	0.43 %	0.41 %	0.47 %	0.43 %	0.47 %	0.47 %	0.46 %	0.47 %	0.47 %
	Suma de Verificación	0.43 %	0.42 %	0.47 %	0.42 %	0.44 %	0.43 %	0.44 %	0.44 %	0.45 %

Tabla 3 - Porcentaje de uso de memoria RAM total utilizando algoritmo ECC y SHA3-512

En la Tabla 3 se puede resaltar que empleando el algoritmo de Curvas Elípticas (ECC) el porcentaje de consumo de la desencriptación de archivos de 50 y 100 MB es el mismo para todos los conjuntos de datos utilizando cualquiera de los tres algoritmos simétricos. En el caso de los archivos de 1 MB se puede observar que el resultado más bajo ocurre utilizando el algoritmo AES-256-CBC.

De los tres algoritmos simétricos analizados la suma de verificación consume menos memoria que la encriptación y desencriptación, salvo para los archivos de 1 MB de texto y video.

En general, no se aprecian diferencias significativas en los resultados obtenidos, pero en promedio, DES3 requiere un consumo ligeramente menor si se quisiera utilizar un solo algoritmo para todas las ejecuciones. También se puede enfatizar en el hecho de que ninguno de los resultados supera el 0.47% de la memoria disponible en la RPI3.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	0.45 %	0.44 %	0.45 %	0.19 %	0.45 %	0.45 %	0.20 %	0.45 %	0.45 %
	Encriptación	0.26 %	0.32 %	0.45 %	0.44 %	0.46 %	0.44 %	0.45 %	0.45 %	0.26 %
	Suma de Verificación	0.33 %	0.24 %	0.28 %	0.27 %	0.29 %	0.29 %	0.30 %	0.28 %	0.22 %
chacha20	Desencriptación	0.44 %	0.45 %	0.45 %	0.23 %	0.45 %	0.45 %	0.10 %	0.40 %	0.45 %
	Encriptación	0.37 %	0.40 %	0.45 %	0.44 %	0.41 %	0.44 %	0.36 %	0.45 %	0.38 %
	Suma de Verificación	0.28 %	0.27 %	0.29 %	0.37 %	0.29 %	0.28 %	0.28 %	0.28 %	0.26 %
des3	Desencriptación	0.45 %	0.43 %	0.45 %	0.38 %	0.44 %	0.45 %	0.13 %	0.45 %	0.45 %
	Encriptación	0.41 %	0.46 %	0.43 %	0.45 %	0.47 %	0.45 %	0.49 %	0.43 %	0.45 %
	Suma de Verificación	0.33 %	0.27 %	0.22 %	0.32 %	0.28 %	0.29 %	0.30 %	0.28 %	0.24 %

Tabla 4 - Porcentaje de uso de memoria RAM total utilizando algoritmo DH y SHA3-512

Se puede observar en la Tabla 4 los resultados obtenidos para el algoritmo Diffie-Hellman (DH) en donde la suma de verificación aplicada a archivos encriptados con el algoritmo AES-256-CBC consume menos memoria que la encriptación y desencriptación, salvo para los archivos de 1 MB.

Para Chacha20 y DES3 se repite este comportamiento de que la suma de verificación requiere menos consumo de memoria que la encriptación y desencriptación en los archivos de tamaño de 50 y 100 MB.

El menor porcentaje de consumo de memoria se obtiene con el algoritmo Chacha20 en la desencriptación de un archivo de video de 1 MB, mientras que el porcentaje más alto, se da para la encriptación, aplicado al mismo archivo, pero utilizando el algoritmo DES3.

Resumen resultados métrica porcentaje de uso de memoria:

En base a los resultados registrados en las Tablas 1, 2, 3 y 4, no se puede establecer una relación entre el porcentaje de uso de memoria y el tipo y tamaño de archivo ya que se obtuvieron valores muy similares.

En general se observa que la función de encriptación es la que más recursos de memoria consume, y que el porcentaje de uso de memoria no supera el 0.51% en ninguna de las tres funciones de encriptación, desencriptación y suma de verificación. Si bien el porcentaje más alto corresponde a la encriptación con el algoritmo RSA en combinación con AES-256-CBC, aplicado a un archivo de video de 1MB (Tabla 2), se

observa que el algoritmo ECC es en general el que más memoria utiliza en las tres funciones.

Para la función de encriptación, el algoritmo de ECDSA ofrece mejores resultados combinado con AES, mientras que DH combinado con DES3 consume menos memoria que con otros algoritmos.

En el caso de la descriptación, los mejores resultados son obtenidos, en su mayoría, en los archivos de video y texto de 1 MB y con la combinación de algoritmos DH/CHACHA20.

En la suma de verificación, en general, DH es el que menos memoria consume y ECC el que más memoria necesita.

Las pruebas anteriores se repitieron para los algoritmos de suma de verificación SHA-512, SHA-1 y RIPEMD-160, pero dado que no se observaron diferencias significativas en los resultados, se dejan en el Anexo en la sección “Resultados Nodo RPI 3 – Porcentaje de Uso de Memoria” las Tablas A1 a A11 para un análisis más detallado. Como conclusión de los resultados obtenidos sobre el consumo de memoria, no se puede establecer una relación con el tipo y tamaño de datos utilizados en las pruebas realizadas. Otra característica que comparten es que la suma de verificación es la función que menos recursos de memoria necesita, mientras que la encriptación y descriptación requieren valores muy similares. Se puede destacar que en ninguno de los casos el sistema hace un uso mayor al 0.51% de los recursos de memoria del dispositivo de cómputo.

A partir del análisis de los resultados se puede observar que ninguno de los algoritmos consume un porcentaje significativo de memoria, que pueda afectar otras tareas ejecutadas en el nodo. DH es el óptimo en el caso de que se requiera trabajar solo con un algoritmo asimétrico para todas las funciones ya que, en general, es el que consume menos memoria para todas las funciones, tipos y tamaños de archivos.

Uso de Memoria Externa

En esta sección se presentan las Tablas 5 al 19 con los resultados experimentales obtenidos con respecto a la cantidad de accesos a la memoria externa requeridos por cada función según el algoritmo empleado, y el tipo y tamaño de archivo analizado. Las tablas han sido organizadas según el algoritmo de suma de verificación utilizado.

Algoritmo de Verificación SHA3-512

En las Tablas 5, 6, 7 y 8 se pueden analizar los resultados registrados en las pruebas realizadas en la RPI 3 con el algoritmo SHA3-512 de suma de verificación.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	732,784	9,015,009	17,939,106	738,022	8,863,180	17,226,269	745,050	9,011,509	17,097,383
	Encriptación	681,093	6,901,081	13,103,709	686,414	6,768,556	13,521,154	692,895	6,757,586	13,073,084
	Suma de Verificación	148,935	1,284,654	2,585,428	146,240	1,326,792	2,598,727	148,388	1,327,624	2,545,555
chacha20	Desencriptación	749,593	9,708,141	19,173,453	759,205	9,607,945	19,022,199	758,087	9,767,486	19,414,342
	Encriptación	704,272	7,516,397	14,269,207	700,073	7,372,925	14,096,470	706,866	7,167,676	14,146,765
	Suma de Verificación	150,103	1,299,784	2,501,593	144,091	1,261,013	2,633,406	147,411	1,312,160	2,635,486
des3	Desencriptación	599,560	13,671,396	27,201,890	603,661	13,364,614	26,394,325	611,230	13,353,265	26,690,299
	Encriptación	529,515	9,859,845	19,428,805	535,570	9,884,154	20,116,797	545,299	10,063,260	19,829,523
	Suma de Verificación	147,322	1,368,417	2,463,640	148,009	1,278,326	2,532,197	147,044	1,358,086	2,438,068

Tabla 5 – Cantidad de accesos a memoria externa utilizando Algoritmo ECDSA y SHA3-512

En la Tabla 5 se puede notar que utilizando el algoritmo ECDSA, para los tres conjuntos de datos, a medida que se aumenta el tamaño del archivo, aumenta la cantidad de accesos a la memoria externa y que la suma de verificación requiere menos accesos que la encriptación, seguida por la desencriptación.

Los valores mínimos de encriptación y desencriptación se obtienen para los archivos de imagen de 1 MB.

AES-256-CBC requiere menos accesos que los otros dos algoritmos para la encriptación y desencriptación de archivos de 50 MB y 100 MB.

Chacha20 es el que menos memoria externa necesita en la suma de verificación, obteniéndose el mínimo para el archivo de texto de 1 MB.

DES3 es el indicado, en cuanto accesos a memoria externa se refiere, para la encriptación y desencriptación de archivos de 1 MB.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	722,600	8,702,746	17,145,651	725,331	8,664,649	16,913,539	735,107	8,691,705	17,296,604
	Encriptación	687,765	6,686,715	13,881,236	685,342	6,721,188	12,947,542	689,459	6,886,406	12,978,619
	Suma de Verificación	145,601	1,319,437	2,605,329	144,712	1,384,934	2,497,960	145,030	1,332,722	2,461,000
chacha20	Desencriptación	707,884	7,584,815	14,629,925	707,455	7,429,103	14,903,629	714,141	7,692,393	14,770,840
	Encriptación	694,118	7,330,969	14,098,493	697,344	7,435,074	14,467,101	706,961	7,804,698	14,226,164
	Suma de Verificación	144,262	1,365,333	2,602,781	145,262	1,331,731	2,514,353	145,624	1,331,512	2,618,993
des3	Desencriptación	598,412	12,417,663	25,259,080	589,505	12,292,692	24,411,575	601,082	13,201,132	25,737,418
	Encriptación	531,887	9,811,867	19,459,689	536,052	9,955,237	19,512,547	530,567	9,545,722	19,514,143
	Suma de Verificación	145,035	1,428,660	2,462,603	144,759	1,317,993	2,621,617	144,785	1,425,023	2,559,454

Tabla 6 - Cantidad de accesos a memoria externa utilizando algoritmo RSA y SHA3-512

En la Tabla 6 se puede observar que, la menor cantidad de accesos de memoria externa utilizando RSA se obtiene en la encriptación y desencriptación de los archivos de tamaño 1 MB. Sin importar el tipo de datos, a medida que se aumenta el tamaño del archivo, aumenta la cantidad de accesos a la memoria externa y la suma de verificación requiere menos accesos que la encriptación, seguida por la desencriptación.

AES-256-CBC es el algoritmo que requiere menos accesos a memoria externa para la encriptación de archivos de 50 MB y 100 MB, mientras que, para los mismos datos, Chacha20 es el más eficiente para la desencriptación.

Con Chacha20 se obtiene el mínimo necesario de memoria externa en la suma de verificación, obteniéndose para el archivo de texto de 1 MB, no obstante, en general la suma de verificación requiere de recursos similares independientemente del algoritmo simétrico utilizado.

DES3 es el que menos memoria externa utiliza en la encriptación y desencriptación de archivos de 1 MB, pero utiliza el máximo para la desencriptación de un archivo de video de 100 MB.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	723,058	8,651,746	17,492,117	724,267	8,609,851	17,593,934	746,569	9,175,468	17,155,304
	Encriptación	680,649	6,438,326	13,266,075	683,442	6,475,369	13,147,636	689,501	6,641,803	13,279,823
	Suma de Verificación	130,627	1,325,390	2,577,115	132,430	1,344,232	2,624,805	131,353	1,403,336	2,421,739
chacha20	Desencriptación	752,214	9,608,768	18,078,114	740,372	9,412,669	18,516,997	757,101	9,291,699	18,867,553
	Encriptación	687,833	7,437,526	14,050,054	701,794	7,237,523	14,038,932	706,743	7,112,741	14,130,301
	Suma de Verificación	132,138	1,284,672	2,646,194	131,156	1,287,323	2,621,910	132,102	1,287,550	2,594,358
des3	Desencriptación	586,594	13,109,635	25,604,671	598,224	12,608,748	26,667,617	619,510	12,872,105	26,098,104
	Encriptación	531,809	9,798,918	19,266,979	522,057	9,698,399	19,198,609	537,290	9,932,172	18,831,598
	Suma de Verificación	132,235	1,296,185	2,622,366	130,729	1,265,774	2,763,188	133,679	1,259,003	2,667,102

Tabla 7 - Cantidad de accesos a memoria externa utilizando algoritmo ECC y SHA3-512

En los resultados obtenidos con el algoritmo ECC (Tabla 7), se puede apreciar que, para los tres conjuntos de datos, a medida que se incrementa el tamaño del archivo, se requiere mayor cantidad de accesos a la memoria externa, obteniéndose de esta forma los valores mínimos de las tres funciones para los archivos de tamaño 1 MB. También se observa que la suma de verificación requiere menos accesos que la encriptación, seguida por la desencriptación.

AES-256-CBC requiere menos accesos que los otros dos algoritmos para la encriptación de archivos y desencriptación de 50 MB y 100 MB, seguido por Chacha20.

Con AES-256-CBC también se obtiene el mínimo necesario de memoria externa en la suma de verificación, obteniéndose para el archivo de imagen de 1 MB.

DES3 es el que menos memoria externa utiliza en la encriptación y desencriptación de archivos de 1 MB, pero utiliza el máximo para la desencriptación de un archivo de texto de 100 MB.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	737,243	9,138,165	17,732,961	732,042	8,833,776	17,703,465	738,998	8,903,661	18,117,694
	Encriptación	690,184	6,766,681	12,873,157	686,993	6,718,182	13,217,024	701,767	6,800,889	12,845,511
	Suma de Verificación	124,991	1,061,022	1,991,211	124,812	1,021,525	2,084,916	127,125	1,094,093	2,003,504
chacha20	Desencriptación	757,396	9,915,957	19,565,660	739,592	9,633,427	19,326,505	756,767	9,605,991	19,593,717
	Encriptación	702,651	7,383,785	13,804,999	700,451	7,582,683	14,275,327	708,121	7,249,120	14,234,042
	Suma de Verificación	125,259	1,046,956	1,939,007	126,539	1,055,529	2,093,968	126,586	1,152,489	1,980,165
des3	Desencriptación	595,659	13,019,787	26,349,860	601,157	13,185,256	26,054,438	614,810	13,052,357	25,783,411
	Encriptación	526,974	10,074,504	19,345,472	530,966	9,964,953	19,625,789	538,683	9,809,132	19,919,280
	Suma de Verificación	125,939	1,059,622	2,021,089	125,111	1,035,069	2,065,734	126,754	1,134,012	2,121,101

Tabla 8 - Cantidad de accesos a memoria externa utilizando algoritmo DH y SHA3-512

Al igual que con los otros algoritmos asimétricos, se puede notar en la Tabla 8 que, para los tres conjuntos de datos, a mayor tamaño de archivo, mayor cantidad de accesos a la memoria externa y que la suma de verificación requiere menos accesos que la encriptación, seguida por la desencriptación.

AES-256-CBC requiere menos accesos que los otros dos algoritmos para la encriptación de archivos y desencriptación de 50 MB y 100 MB, seguido por Chacha20.

Para la encriptación y desencriptación de archivos de 1 MB, DES3 es el que menos memoria externa utiliza, pero utiliza el máximo para la desencriptación de un archivo de imagen de 100 MB.

En el caso de la suma de verificación, con AES-256-CBC se obtiene el mínimo necesario de memoria externa, y se registra con el archivo de texto de 1 MB.

Resumen resultados accesos a memoria externa -algoritmo SHA3-512:

Como resumen de las pruebas realizadas con el algoritmo de suma de verificación SHA3-512, se puede resaltar que, para los tres conjuntos de datos, a medida que se aumenta el tamaño del archivo, aumenta la cantidad de accesos a la memoria externa, obteniéndose de esta forma los valores mínimos de las tres funciones para los archivos de tamaño 1 MB. También se observa que la suma de verificación requiere menos accesos que la encriptación, seguida por la desencriptación.

En las Tablas 5, 6, 7 y 8, AES-256-CBC requiere menos accesos a la memoria externa que los otros dos algoritmos para la encriptación y desencriptación de archivos de 50 MB y 100 MB, mientras que DES3 es el que menos memoria externa utiliza en la encriptación y desencriptación de archivos de 1 MB. En la Tabla 6, AES-256-CBC requiere menos accesos a la memoria externa que los otros dos algoritmos para la encriptación de archivos de 50 MB y 100 MB, pero para la desencriptación resulta óptimo Chacha20. Para los archivos de 1 MB, al igual que en las otras tablas, DES3 es el que menos memoria externa necesita.

En cuanto a la suma de verificación, en las Tablas 5 y 6 se puede concluir que esta función aplicada sobre archivos encriptados con Chacha20 es la que menos

memoria externa necesita, mientras que en las Tablas 7 y 8 requiere menos memoria externa sobre archivos encriptados con AES-256-CBC.

Algoritmo de Verificación SHA-512

En las Tablas 9, 10, 11 y 12 se presentan los resultados obtenidos de las pruebas realizadas en la RPI 3 con el algoritmo SHA-512 de suma de verificación.

En todas ellas se puede observar que se repite el mismo comportamiento que para el algoritmo SHA3-512, donde sin importar el tipo de datos, a medida que se aumenta el tamaño del archivo, aumenta la cantidad de accesos a la memoria externa y la suma de verificación requiere menos accesos que la encriptación, seguida por la desencriptación.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	733,513	9,204,934	17,141,778	738,060	9,053,106	16,725,230	732,235	8,803,366	17,993,409
	Encriptación	686,387	6,784,634	12,907,555	691,516	6,854,693	12,758,840	696,464	7,000,865	13,933,777
	Suma de Verificación	154,582	1,649,266	3,270,397	156,679	1,631,306	3,216,579	155,165	1,689,956	3,333,839
chacha20	Desencriptación	760,297	9,734,152	18,748,583	757,291	9,671,132	18,547,656	759,008	10,155,930	19,096,947
	Encriptación	697,738	7,219,339	14,351,538	693,407	7,184,853	13,754,052	710,576	7,458,239	14,328,412
	Suma de Verificación	153,077	1,681,200	3,216,342	156,987	1,607,103	3,408,035	157,246	1,641,073	3,367,748
des3	Desencriptación	597,071	12,957,342	27,115,506	583,806	13,488,462	25,922,542	610,410	13,148,863	27,356,801
	Encriptación	535,967	10,079,166	19,205,601	535,752	9,783,489	19,569,450	543,488	10,260,773	19,381,364
	Suma de Verificación	156,819	1,674,440	3,311,248	157,028	1,767,333	3,418,340	155,049	1,665,804	3,370,484

Tabla 9 - Cantidad de accesos a memoria externa utilizando algoritmo ECDSA y SHA-512

En la Tabla 9 se puede observar que con AES-256-CBC los valores mínimos de encriptación y desencriptación se obtienen para los archivos de imagen de 1 MB. Este algoritmo, combinado con ECDSA, requiere menos accesos a la memoria externa que los otros dos algoritmos para la encriptación y desencriptación de archivos de 50 MB y 100 MB.

Chacha20 es el que menos memoria externa necesita en la suma de verificación, obteniéndose el mínimo para el archivo de texto de 1 MB.

DES3 es el que menos memoria externa utiliza en la encriptación y desencriptación de archivos de 1 MB. No obstante, es el que más consume en la desencriptación de datos de video de 100 MB.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	730,060	8,922,860	17,242,376	728,894	8,644,987	16,847,196	737,070	8,745,620	17,154,768
	Encriptación	689,948	6,610,338	13,521,578	683,996	6,913,496	13,879,122	692,572	6,830,177	13,121,840
	Suma de Verificación	150,512	1,799,398	3,126,784	152,669	1,720,360	3,188,324	154,035	1,680,225	3,211,626
chacha20	Desencriptación	702,754	7,509,342	14,634,397	703,375	7,605,278	15,013,402	709,984	7,669,391	14,690,651
	Encriptación	697,658	7,329,687	14,457,419	707,839	7,445,329	14,061,181	705,684	7,428,141	14,066,596
	Suma de Verificación	151,367	1,651,162	3,151,527	153,553	1,722,694	3,304,794	153,171	1,641,404	3,268,779
des3	Desencriptación	590,599	12,392,176	25,279,610	580,683	12,489,463	25,515,656	616,780	12,473,130	26,318,591
	Encriptación	531,096	9,812,971	19,598,210	529,507	9,463,286	19,543,755	537,907	10,236,257	19,465,663
	Suma de Verificación	153,593	1,806,469	3,480,162	152,052	1,723,539	3,441,550	153,720	1,747,509	3,270,700

Tabla 10 - Cantidad de accesos a memoria externa utilizando algoritmo RSA y SHA-512

En los resultados obtenidos con RSA (Tabla 10) se puede notar que, AES-256-CBC requiere menos accesos que los otros dos algoritmos para la encriptación de archivos de 50 MB y 100 MB, mientras que, para los mismos datos, Chacha20 requiere menos accesos que los otros dos algoritmos para la desencriptación.

El mínimo necesario de memoria externa en la suma de verificación se obtiene para el archivo de imagen de 1 MB encriptado con AES-256-CBC, no obstante, en general la suma de verificación requiere de recursos similares si se utiliza Chacha20.

DES3 es el que menos memoria externa utiliza en la encriptación y desencriptación de archivos de 1 MB, pero utiliza el máximo para la desencriptación de un archivo de video de 100 MB.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	725,734	8,866,478	17,069,580	737,201	8,938,840	17,113,383	734,194	9,041,759	16,709,458
	Encriptación	679,552	6,637,251	12,987,853	687,564	6,389,875	12,875,727	692,218	6,688,489	12,622,241
	Suma de Verificación	138,224	1,717,362	3,289,385	141,033	1,720,600	3,323,266	139,587	1,647,186	3,434,397
chacha20	Desencriptación	746,639	9,812,646	19,536,368	741,079	9,612,983	19,016,840	748,194	9,881,915	18,381,987
	Encriptación	690,901	7,086,420	14,101,611	697,507	7,066,785	13,720,872	703,580	7,722,864	14,113,411
	Suma de Verificación	136,900	1,634,986	3,331,175	139,107	1,610,224	3,298,486	143,585	1,674,865	3,197,032
des3	Desencriptación	589,942	13,210,345	26,260,208	588,176	13,228,505	24,788,094	599,676	13,073,397	26,312,331
	Encriptación	517,931	9,675,765	19,273,286	525,076	9,485,998	19,164,786	539,486	9,411,843	19,221,108
	Suma de Verificación	138,880	1,733,491	3,309,900	139,946	1,590,143	3,207,960	141,310	1,777,736	3,345,267

Tabla 11 - Cantidad de accesos a memoria externa utilizando algoritmo ECC y SHA-512

Analizando los valores registrados en las pruebas realizadas con ECC (Tabla 11), se puede establecer que, AES-256-CBC requiere menos accesos que los otros

dos algoritmos para la encriptación de archivos y desenscriptación de 50 MB y 100 MB, seguido por Chacha20.

Con Chacha20 se obtiene el mínimo necesario de memoria externa en la suma de verificación, obteniéndose para el archivo de imagen de 1 MB.

DES3 es el que menos memoria externa utiliza en la encriptación y desenscriptación de archivos de 1 MB, pero utiliza el máximo para la desenscriptación de un archivo de video de 100 MB.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desenscriptación	732,597	9,102,945	18,397,192	732,525	8,826,896	17,750,368	738,433	8,987,286	17,818,959
	Encriptación	692,452	6,958,714	13,228,756	689,167	6,862,796	13,248,772	704,070	6,860,380	12,929,190
	Suma de Verificación	130,217	1,272,143	2,632,508	131,846	1,266,881	2,374,129	133,099	1,305,713	2,606,284
chacha20	Desenscriptación	736,616	9,636,354	20,254,164	744,787	9,877,889	19,124,466	773,321	9,619,521	19,323,065
	Encriptación	700,442	7,332,781	14,572,869	701,247	7,460,753	14,273,614	705,497	7,305,601	14,392,999
	Suma de Verificación	134,417	1,273,274	2,542,316	131,303	1,321,847	2,497,635	132,773	1,355,126	2,469,893
des3	Desenscriptación	600,984	13,278,012	28,144,929	597,194	13,200,615	27,002,203	608,274	13,107,965	25,975,107
	Encriptación	527,318	10,228,972	19,203,416	532,043	9,829,353	19,254,274	541,588	10,236,515	19,560,445
	Suma de Verificación	131,920	1,251,750	2,560,324	133,127	1,297,994	2,476,555	131,105	1,284,652	2,423,851

Tabla 12 - Cantidad de accesos a memoria externa utilizando algoritmo DH y SHA-512

Al igual que con los otros algoritmos asimétricos, se puede notar que combinado con DH (Tabla 12), AES-256-CBC también requiere menos accesos que los otros dos algoritmos para la encriptación de archivos y desenscriptación de 50 MB y 100 MB, seguido por Chacha20. Con este algoritmo también se obtiene el mínimo necesario de memoria externa en la suma de verificación, registrándose para el archivo de imagen de 1 MB.

DES3 es el que menos memoria externa utiliza en la encriptación y desenscriptación de archivos de 1 MB, pero utiliza el máximo para la desenscriptación de un archivo de imagen de 100 MB.

Resumen resultados accesos a memoria externa -algoritmo SHA-512:

A modo de síntesis, de los resultados analizados en las Tablas 9, 10, 11, 12 con el algoritmo de suma de verificación SHA-512, se observa un comportamiento similar al del algoritmo SHA3-512, del que se puede resaltar que, para los tres conjuntos de datos, a medida que se incrementa el tamaño del archivo, aumenta la

cantidad de accesos a la memoria externa, obteniéndose de esta forma los valores mínimos de las tres funciones para los archivos de tamaño 1 MB. También se observa que la suma de verificación requiere menos accesos que la encriptación, seguida por la desenscriptación.

En las Tablas 9, 10 y 12, AES-256-CBC requiere menos accesos a la memoria externa que los otros dos algoritmos para la encriptación y desenscriptación de archivos de 50 MB y 100 MB, mientras que DES3 es el que menos memoria externa utiliza en la encriptación y desenscriptación de archivos de 1 MB. En la Tabla 11, AES-256-CBC requiere menos accesos a la memoria externa que los otros dos algoritmos para la encriptación de archivos de 50 MB y 100 MB, pero para la desenscriptación resulta óptimo Chacha 20. Para los archivos de 1 MB, al igual que en las otras tablas, DES3 es el que menos memoria externa necesita.

En cuanto a la suma de verificación, en las Tablas 9 y 11 se puede concluir que esta función aplicada sobre archivos encriptados con Chacha20 es la que menos memoria externa necesita, mientras que en las Tablas 10 y 12 requiere menos memoria externa sobre archivos encriptados con AES-256-CBC.

Si se comparan los resultados con los valores obtenidos con el algoritmo SHA3-512, SHA-512 requiere de 0,5% a 6,8% más cantidad de accesos a memoria externa.

Algoritmo de Verificación SHA1

En las Tablas 13, 14, 15 y 16 se muestran los resultados registrados en las pruebas realizadas en la RPI 3 con el algoritmo SHA-1 de suma de verificación.

En los resultados obtenidos se observa que se repite el mismo comportamiento obtenido en las ejecuciones realizadas con los algoritmos SHA3-512 y SHA-512, donde para los tres conjuntos de datos, a medida que se aumenta el tamaño del archivo, aumenta la cantidad de accesos a la memoria externa y de las tres funciones, la suma de verificación requiere menos accesos que la encriptación, seguida por la desenscriptación.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	749,814	9,242,903	17,099,314	730,513	8,881,208	17,475,862	740,231	8,889,870	16,823,858
	Encriptación	685,557	6,898,101	12,695,259	680,734	6,559,794	13,411,818	686,529	6,880,273	13,140,521
	Suma de Verificación	147,957	1,525,974	2,943,065	149,548	1,483,088	2,772,601	153,555	1,439,044	2,800,588
chacha20	Desencriptación	756,290	9,769,575	19,294,290	751,554	10,062,850	19,486,409	762,682	9,733,394	19,314,295
	Encriptación	693,921	7,326,879	14,092,226	704,431	7,411,414	15,127,904	703,974	7,706,196	13,628,290
	Suma de Verificación	150,154	1,535,983	2,723,041	152,889	1,458,753	2,834,522	152,625	1,529,908	3,051,990
des3	Desencriptación	601,098	13,234,697	26,066,528	597,094	13,524,473	26,514,871	614,892	13,791,135	26,062,488
	Encriptación	537,857	9,823,169	19,966,261	530,540	9,639,803	20,124,304	544,071	9,966,057	18,837,739
	Suma de Verificación	149,262	1,464,457	2,862,797	150,567	1,480,745	2,801,725	152,916	1,464,281	2,933,423

Tabla 13 - Cantidad de accesos a memoria externa utilizando algoritmo ECDSA y SHA1

En la Tabla 13 se puede observar que, en los resultados obtenidos con ECDSA, AES-256-CBC es el algoritmo que requiere menor cantidad de accesos a memoria externa para la encriptación y desencriptación de archivos de 50 MB y 100 MB.

El mínimo necesario de memoria externa en la suma de verificación se obtiene para el archivo de imagen de 1 MB encriptado con AES-256-CBC, no obstante, en general, la suma de verificación requiere de recursos similares si se utiliza Chacha20.

DES3 es el que menos memoria externa utiliza en la encriptación y desencriptación de archivos de 1 MB. Sin embargo, es el que más consume en la desencriptación de datos de texto de 100 MB.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	729,299	8,710,294	16,736,800	729,245	8,422,755	16,920,924	730,667	8,833,944	17,071,141
	Encriptación	686,327	6,711,700	13,190,385	684,067	6,680,994	13,015,514	687,590	6,654,518	12,871,183
	Suma de Verificación	147,814	1,492,612	2,930,074	146,931	1,414,300	2,828,747	147,866	1,421,024	2,871,114
chacha20	Desencriptación	703,985	7,537,044	14,924,091	707,069	7,622,415	14,868,925	708,810	7,662,502	14,672,764
	Encriptación	695,690	7,631,941	14,838,516	691,945	7,276,459	14,329,040	716,580	7,353,206	13,955,529
	Suma de Verificación	150,553	1,477,441	2,671,158	146,318	1,453,070	2,723,379	146,247	1,495,046	2,812,760
des3	Desencriptación	578,540	13,607,844	25,439,069	592,980	13,006,595	24,203,327	610,532	12,559,916	24,441,624
	Encriptación	529,782	10,138,634	19,360,865	531,874	9,985,524	19,810,426	543,483	10,044,430	19,556,716
	Suma de Verificación	145,427	1,469,883	2,848,000	146,414	1,420,626	2,892,106	147,374	1,530,821	2,953,690

Tabla 14 - Cantidad de accesos a memoria externa utilizando algoritmo RSA y SHA1

En la Tabla 14 se puede notar que, AES-256-CBC requiere menos accesos que los otros dos algoritmos para la encriptación de archivos de 50 MB y 100 MB,

mientras que, para los mismos datos, Chacha20 requiere menos accesos que los otros dos algoritmos para la descriptación.

DES3 es el que menos memoria externa utiliza en la encriptación y descriptación de archivos de 1 MB, pero utiliza el máximo para la descriptación de un archivo de imagen de 100 MB. También con este algoritmo se obtiene el mínimo necesario de memoria externa en la suma de verificación y se produce para el archivo de imagen de 1MB encriptado con DES3.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Descriptación	724,968	8,504,343	17,484,759	724,979	8,701,120	16,824,818	743,796	9,238,836	17,688,792
	Encriptación	685,319	6,595,955	12,607,385	686,045	6,708,703	12,923,071	679,736	6,978,172	13,094,965
	Suma de Verificación	135,800	1,512,661	2,752,267	133,133	1,511,511	2,903,475	136,575	1,457,933	2,817,238
chacha20	Descriptación	739,309	9,596,640	19,372,290	737,222	9,387,329	18,958,102	748,258	10,098,678	18,646,473
	Encriptación	691,899	7,239,558	14,141,643	697,174	7,139,135	14,301,334	698,799	7,279,414	14,462,888
	Suma de Verificación	133,259	1,463,120	2,798,978	134,242	1,494,679	2,818,514	134,446	1,482,552	2,898,572
des3	Descriptación	590,266	12,350,916	25,415,900	595,484	12,909,657	28,093,765	615,247	12,910,376	25,234,638
	Encriptación	525,102	9,514,377	19,471,930	527,603	9,871,239	19,322,218	536,463	10,144,324	19,270,782
	Suma de Verificación	137,433	1,451,385	2,863,139	133,555	1,508,479	2,933,123	135,674	1,487,556	3,014,489

Tabla 15 - Cantidad de accesos a memoria externa utilizando algoritmo ECC y SHA1

En el caso del algoritmo ECC (Tabla 15) se observa un comportamiento similar a las dos tablas anteriores donde AES-256-CBC es el algoritmo simétrico que requiere menos accesos a memoria externa para la encriptación de archivos y descriptación de 50 MB y 100 MB, seguido por Chacha20.

Con AES-256-CBC también se obtiene el mínimo necesario de memoria externa en la suma de verificación, y se registra con el archivo de texto de 1 MB.

DES3 resulta óptimo en la encriptación y descriptación de archivos de 1 MB, pero utiliza el máximo para la descriptación de un archivo de texto de 100 MB.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	743,613	9,135,376	17,585,118	732,000	9,171,934	17,439,600	747,256	8,965,694	17,363,731
	Encriptación	688,116	7,018,349	12,875,144	681,999	6,693,690	13,193,695	690,084	7,195,367	13,287,197
	Suma de Verificación	137,124	1,546,381	3,027,860	135,799	1,553,617	2,975,536	139,179	1,633,001	2,923,585
chacha20	Desencriptación	748,517	9,863,142	20,165,794	738,823	9,739,081	19,084,627	758,045	10,187,695	18,752,584
	Encriptación	695,943	7,632,610	13,906,941	695,218	7,465,726	14,727,141	703,718	7,213,652	14,526,671
	Suma de Verificación	135,092	1,524,108	3,012,273	135,939	1,494,349	3,059,722	138,935	1,512,255	3,063,085
des3	Desencriptación	601,662	13,194,400	26,095,351	595,846	13,017,157	26,170,075	606,403	12,914,449	25,685,244
	Encriptación	526,666	9,835,360	20,088,227	524,551	10,039,156	19,437,218	545,807	10,243,528	19,098,904
	Suma de Verificación	128,129	1,176,716	2,105,528	127,112	1,111,919	2,220,014	127,615	1,245,261	2,340,900

Tabla 16 - Cantidad de accesos a memoria externa utilizando algoritmo DH y SHA1

Tal como sucede con los otros algoritmos asimétricos, con DH (Tabla 16) continúa AES-256-CBC siendo el algoritmo que requiere menos accesos a memoria externa para la encriptación de archivos y desencriptación de 50 MB y 100 MB, seguido por Chacha20.

DES3 es el que menos memoria externa utiliza en la encriptación y desencriptación de archivos de 1 MB, pero utiliza el máximo para la desencriptación de un archivo de texto de 100 MB.

En el caso de la suma de verificación, con DES3 se obtiene el mínimo necesario de memoria externa en la suma de verificación, obteniéndose para el archivo de texto de 1 MB.

Resumen resultados accesos a memoria externa -algoritmo SHA1:

Como resumen de los resultados analizados en las Tablas 13, 14, 15 y 16 de las pruebas realizadas con el algoritmo de suma de verificación SHA-1, se observa un comportamiento similar al de los algoritmos SHA3-512 y SHA-512, del que se puede destacar que, para los tres conjuntos de datos, a medida que se incrementa el tamaño del archivo, aumenta la cantidad de accesos a la memoria externa, obteniéndose de esta forma los valores mínimos de las tres funciones para los archivos de tamaño 1 MB. También se observa que la suma de verificación requiere menos accesos que la encriptación, seguida por la desencriptación.

En las cuatro tablas mencionadas, AES-256-CBC requiere menos accesos a la memoria externa que los otros dos algoritmos para la encriptación y desencriptación de archivos de 50 MB y 100 MB, mientras que DES3 es el que menos memoria

externa utiliza en la encriptación y desencriptación de archivos de 1 MB. En cuanto a la suma de verificación, en las Tablas 14 y 16 se puede concluir que esta función aplicada sobre archivos encriptados con DES3 es la que menos memoria externa necesita, mientras que en las Tablas 13 y 15 requiere menos memoria externa sobre archivos encriptados con AES-256-CBC.

Algoritmo de Verificación RIPEMD-160

En las Tablas 17, 18 y 19, se muestran los resultados registrados para las pruebas realizadas con el algoritmo RIPEMD-160. El mismo no es compatible con ECDSA, por esta razón no se incluye en el informe los resultados de este algoritmo asimétrico.

Al igual que con los algoritmos SHA3-512, Sha-512 y SHA-1 se observa el mismo patrón en los resultados para los tres conjuntos de datos, es decir, la cantidad de accesos a la memoria externa se incrementa con el tamaño del archivo y la suma de verificación requiere menos accesos que la función encriptación, seguida por la desencriptación.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	729,806	8,771,765	17,067,193	726,396	8,651,743	17,365,081	732,363	9,054,071	16,953,136
	Encriptación	683,817	6,896,929	12,711,314	694,168	6,726,841	13,095,179	692,151	7,057,569	13,390,450
	Suma de Verificación	143,672	1,285,342	2,557,580	145,488	1,392,473	2,525,786	146,467	1,398,726	2,594,962
chacha20	Desencriptación	702,051	7,558,826	14,579,496	708,708	7,858,795	14,769,859	715,167	7,523,328	14,558,163
	Encriptación	698,574	7,324,849	14,351,964	694,049	7,698,780	14,123,917	708,723	7,586,092	14,445,708
	Suma de Verificación	145,469	1,310,095	2,551,685	145,049	1,252,253	2,598,130	146,268	1,340,413	2,620,508
des3	Desencriptación	616,872	12,682,665	25,711,814	576,490	12,977,140	26,646,678	609,299	13,073,667	26,342,990
	Encriptación	533,077	10,011,037	19,559,229	534,798	10,161,463	19,113,204	539,273	9,740,566	19,852,995
	Suma de Verificación	141,589	1,234,346	2,635,562	145,419	1,243,877	2,489,490	146,153	1,347,526	2,661,863

Tabla 17 - Cantidad de accesos a memoria externa utilizando algoritmo RSA y RIPEMD-160

En la Tabla 17 se puede notar que, AES-256-CBC requiere menos accesos que los otros dos algoritmos para la encriptación de archivos de 50 MB y 100 MB, mientras que, para los mismos datos, Chacha20 requiere menos accesos que los otros dos algoritmos para la desencriptación.

DES3 es el que menos memoria externa utiliza en la encriptación y desenscriptación de archivos de 1 MB, pero utiliza el máximo para la desenscriptación de un archivo de texto de 100 MB.

El mínimo necesario de memoria externa en la suma de verificación se obtiene para el archivo de imagen de 1 MB encriptado con DES3.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desenscriptación	724,477	9,243,285	17,120,240	739,353	8,666,775	17,205,675	741,819	8,935,378	17,164,526
	Encriptación	691,598	6,628,939	13,039,228	682,912	6,486,306	12,858,182	687,891	6,858,594	12,629,624
	Suma de Verificación	131,611	1,320,200	2,522,377	131,581	1,271,791	2,573,831	133,934	1,316,551	2,530,892
chacha20	Desenscriptación	743,863	9,843,894	18,407,887	743,197	9,577,905	18,665,024	756,529	9,532,797	18,573,215
	Encriptación	697,835	7,247,429	13,911,989	709,193	7,183,044	14,590,624	707,092	7,519,651	14,370,187
	Suma de Verificación	131,525	1,335,874	2,860,098	132,752	1,373,653	2,589,425	133,501	1,350,303	2,557,094
des3	Desenscriptación	595,066	13,315,076	26,557,973	590,018	13,095,290	26,117,931	617,711	12,897,692	26,169,521
	Encriptación	534,533	9,635,277	20,350,890	525,290	9,668,407	19,704,140	534,541	9,670,777	18,790,594
	Suma de Verificación	130,585	1,375,406	2,507,965	131,702	1,249,178	2,515,936	132,362	1,350,767	2,655,959

Tabla 18 - Cantidad de accesos a memoria externa utilizando algoritmo ECC y RIPEMD-160

De la Tabla 18 se puede establecer que, AES-256-CBC requiere menos accesos que los otros dos algoritmos para la encriptación de archivos y desenscriptación de 50 MB y 100 MB, seguido por Chacha20.

DES3 es el que menos memoria externa utiliza en la encriptación y desenscriptación de archivos de 1 MB, pero utiliza el máximo para la desenscriptación de un archivo de texto de 100 MB. También se obtiene con DES3 el mínimo necesario de memoria externa en la suma de verificación y ocurre para el archivo de imagen de 1 MB. Sin embargo, no se aprecian diferencias considerables en la cantidad de accesos utilizando otro algoritmo simétrico.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	737,243	9,138,165	17,732,961	732,042	8,833,776	17,703,465	738,998	8,903,661	18,117,694
	Encriptación	690,184	6,766,681	12,873,157	686,993	6,718,182	13,217,024	701,767	6,800,889	12,845,511
	Suma de Verificación	124,991	1,061,022	1,991,211	124,812	1,021,525	2,084,916	127,125	1,094,093	2,003,504
chacha20	Desencriptación	750,260	9,995,123	18,506,345	744,667	10,342,562	18,884,786	754,321	9,993,386	19,115,865
	Encriptación	698,981	7,443,633	14,351,913	702,837	7,394,539	14,607,312	712,557	7,632,274	14,186,556
	Suma de Verificación	132,525	1,382,191	2,672,998	131,762	1,385,448	2,718,087	133,140	1,355,752	2,829,161
des3	Desencriptación	598,019	12,939,102	27,559,457	600,924	13,058,251	25,919,836	625,317	13,654,495	27,070,897
	Encriptación	528,393	9,800,303	19,271,004	526,298	9,858,166	19,213,405	553,670	9,972,572	20,466,567
	Suma de Verificación	131,693	1,393,687	2,589,136	134,738	1,413,370	2,659,815	134,197	1,405,148	2,759,576

Tabla 19 - Cantidad de accesos a memoria externa utilizando algoritmo DH y RIPEMD-160

Tal como sucede con los otros algoritmos asimétricos, AES-256-CBC requiere, en combinación con DH (Tabla 19), menos accesos que los otros dos algoritmos para la encriptación y desencriptación de archivos 50 MB y 100 MB, seguido por Chacha20.

Con AES-256-CBC se obtiene el mínimo necesario de memoria externa en la suma de verificación, obteniéndose para el archivo de texto de 1 MB.

DES3 es el que menos memoria externa utiliza en la encriptación y desencriptación de archivos de 1 MB, pero utiliza el máximo para la desencriptación de un archivo de imagen de 100 MB.

Resumen resultados accesos a memoria externa -algoritmo RIPEMD-160:

Se puede observar en los resultados obtenidos en las Tablas 17, 18 y 19 que los accesos a la memoria externa utilizando el algoritmo RIPEMD-160 se incrementan con el aumento del tamaño del archivo sin importar el tipo de datos que se analice. También se puede apreciar que la desencriptación es la función que necesita más memoria externa para ejecutarse, mientras que la suma de verificación es la que menos consume.

En las tres tablas citadas, AES-256-CBC requiere menos accesos a la memoria externa que los otros dos algoritmos para la encriptación y desencriptación de archivos de 50 MB y 100 MB, mientras que DES3 es el que menos memoria externa utiliza en la encriptación y desencriptación de archivos de 1 MB. En cuanto a la suma de verificación, en las Tablas 18 y 19 se puede concluir que esta función aplicada sobre archivos encriptados con AES-256-CBC es la que menos memoria externa

necesita, mientras que en la Tabla 17 requiere menos memoria externa sobre archivos encriptados con DES3.

Resumen resultados métrica cantidad de accesos a memoria externa:

Como conclusión de los resultados obtenidos en la sección Uso de Memoria Externa, se puede observar en los resultados obtenidos en las Tablas 5 al 19 que los accesos a la memoria externa se incrementan con el aumento del tamaño del archivo sin importar el tipo de datos que se analice. También se puede apreciar que la descriptación es la función que necesita más memoria externa para ejecutarse, mientras que la suma de verificación es la que menos consume.

En general, la mejor combinación es DH, AES-256-CBC y SHA3-512, mientras que la mayor cantidad de accesos se da en la combinación DH, DES3 y SHA-512.

Para los archivos más pequeños de 1 MB, DES3 requiere de menos recursos. Para los archivos de tamaño 50 MB y 100 MB conviene AES-256-CBC o Chacha20.

Porcentaje de Consumo de CPU

En esta sección se presentan las Tablas 20 a 23 con los resultados experimentales obtenidos con respecto al porcentaje de consumo de CPU requerido por cada función según el algoritmo empleado, y el tipo y tamaño de archivo analizado. Recordemos que la RPI 3 cuenta con una CPU ARMv8 64-bit Quad-Core de 1.2 GHz.

Algoritmo de Verificación SHA3-512

En las Tablas 20, 21, 22 y 23 se pueden observar los resultados obtenidos de las pruebas realizadas en la RPI 3 con el algoritmo SHA3-512 de suma de verificación.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	95.98 %	77.20 %	70.14 %	97.56 %	90.58 %	66.94 %	97.84 %	92.66 %	72.84 %
	Encriptación	97.02 %	44.46 %	30.12 %	96.92 %	64.98 %	28.66 %	32.66 %	59.56 %	30.90 %
	Suma de Verificación	95.00 %	99.38 %	99.90 %	73.16 %	99.80 %	99.82 %	95.00 %	99.56 %	92.10 %
chacha20	Desencriptación	97.90 %	92.18 %	57.98 %	98.00 %	67.68 %	66.72 %	97.10 %	68.64 %	58.72 %
	Encriptación	72.24 %	19.48 %	21.36 %	72.48 %	37.18 %	16.66 %	57.20 %	45.12 %	15.52 %
	Suma de Verificación	94.94 %	99.78 %	99.68 %	65.66 %	99.74 %	99.84 %	95.26 %	99.66 %	99.66 %
des3	Desencriptación	98.72 %	94.60 %	84.26 %	97.74 %	87.58 %	88.16 %	98.42 %	93.98 %	89.72 %
	Encriptación	97.14 %	62.22 %	61.84 %	97.68 %	91.40 %	60.58 %	51.52 %	88.54 %	61.40 %
	Suma de Verificación	95.04 %	92.72 %	99.80 %	93.92 %	99.74 %	99.80 %	95.10 %	99.60 %	86.24 %

Tabla 20 - Porcentaje de uso de CPU utilizando algoritmo ECDSA y SHA3-512

En la Tabla 20 el máximo porcentaje de uso de CPU con el algoritmo ECDSA se produce en la suma de verificación de una imagen de 100 MB encriptada con AES-256-CBC. En el caso de la encriptación, este algoritmo consume más que Chacha20.

El mínimo porcentaje de consumo del procesador ocurre en la encriptación con Chacha20 de un archivo de video de 100 MB. Este algoritmo requiere menos porcentaje de uso de CPU que los otros dos algoritmos simétricos.

DES3 es el algoritmo que más porcentaje de CPU requiere en la encriptación, y en la desencriptación también, salvo para un archivo de texto de 1 MB.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	98.20 %	87.26 %	75.82 %	98.22 %	85.98 %	77.00 %	98.14 %	83.88 %	77.70 %
	Encriptación	95.24 %	64.06 %	34.16 %	95.72 %	74.30 %	35.94 %	78.24 %	66.84 %	33.16 %
	Suma de Verificación	94.88 %	86.36 %	91.72 %	69.82 %	99.80 %	99.82 %	95.70 %	99.80 %	84.64 %
chacha20	Desencriptación	97.84 %	93.16 %	55.28 %	95.96 %	81.42 %	71.76 %	97.24 %	98.80 %	54.92 %
	Encriptación	89.18 %	39.60 %	21.76 %	88.62 %	47.18 %	21.66 %	71.92 %	40.34 %	21.38 %
	Suma de Verificación	95.86 %	78.20 %	99.86 %	65.88 %	99.80 %	99.86 %	95.86 %	99.40 %	99.88 %
des3	Desencriptación	98.82 %	93.70 %	86.10 %	98.84 %	92.44 %	86.10 %	97.52 %	92.92 %	88.60 %
	Encriptación	95.88 %	86.96 %	66.18 %	97.82 %	88.38 %	63.02 %	86.88 %	85.78 %	67.48 %
	Suma de Verificación	95.68 %	72.24 %	99.90 %	74.66 %	99.74 %	99.88 %	95.64 %	99.80 %	99.80 %

Tabla 21 - Porcentaje de uso de CPU utilizando algoritmo RSA y SHA3-512

Analizando la Tabla 21 se puede observar que con RSA se ocupa casi la totalidad del procesador con la suma de verificación de una imagen de 100 MB

encriptada con DES3. Además, este algoritmo es el que más CPU consume para encriptación y desencriptación.

El mínimo porcentaje de uso de CPU ocurre en la encriptación con Chacha20 de un archivo de video de 100 MB.

Chacha20 es el algoritmo que menos porcentaje de CPU requiere en la encriptación, y el valor mínimo se obtiene para esta función aplicada a un archivo de video de 100 MB.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	98.20 %	88.32 %	77.88 %	98.14 %	77.68 %	75.78 %	98.30 %	71.12 %	78.02 %
	Encriptación	94.40 %	43.08 %	29.58 %	93.24 %	73.74 %	27.84 %	52.70 %	58.44 %	26.82 %
	Suma de Verificación	94.80 %	88.24 %	99.88 %	94.74 %	99.80 %	99.74 %	95.00 %	99.78 %	99.86 %
chacha20	Desencriptación	97.88 %	94.12 %	59.36 %	97.60 %	71.64 %	60.34 %	93.70 %	78.56 %	46.56 %
	Encriptación	81.46 %	25.00 %	21.92 %	85.52 %	34.38 %	15.92 %	65.98 %	37.00 %	16.56 %
	Suma de Verificación	94.90 %	65.82 %	99.90 %	38.98 %	99.78 %	99.90 %	94.82 %	99.70 %	96.32 %
des3	Desencriptación	98.76 %	94.92 %	82.88 %	97.38 %	85.44 %	84.10 %	98.26 %	84.38 %	85.74 %
	Encriptación	96.48 %	63.82 %	59.66 %	95.98 %	85.20 %	52.40 %	55.64 %	89.94 %	59.26 %
	Suma de Verificación	94.86 %	86.62 %	99.88 %	75.84 %	99.80 %	99.86 %	95.06 %	99.52 %	99.82 %

Tabla 22 - Porcentaje de uso de CPU utilizando algoritmo ECC y SHA3-512

A partir de la Tabla 22 se puede determinar que, el máximo porcentaje de uso de CPU se produce en la suma de verificación de una imagen o texto de 100 MB encriptado con Chacha20.

Chacha20 es el algoritmo que menos uso requiere del procesador en la encriptación, y el valor mínimo se obtiene para esta función aplicada a un archivo de texto de 100 MB.

DES3 es el algoritmo que más consume en encriptación y desencriptación para el conjunto de datos de imagen y video, seguido por AES-256-CBC.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	98.20 %	99.56 %	73.48 %	97.70 %	83.44 %	76.42 %	97.24 %	83.58 %	71.58 %
	Encriptación	96.56 %	58.96 %	31.62 %	96.68 %	66.54 %	33.06 %	84.86 %	67.38 %	30.94 %
	Suma de Verificación	92.42 %	99.68 %	99.86 %	93.46 %	99.78 %	99.86 %	90.50 %	99.70 %	89.00 %
chacha20	Desencriptación	98.02 %	91.44 %	69.10 %	97.98 %	82.12 %	42.80 %	93.06 %	88.32 %	64.30 %
	Encriptación	89.74 %	45.88 %	19.46 %	89.72 %	42.32 %	20.56 %	38.94 %	40.12 %	21.16 %
	Suma de Verificación	92.40 %	87.30 %	99.78 %	91.92 %	99.74 %	99.76 %	91.90 %	99.74 %	86.14 %
des3	Desencriptación	98.64 %	97.64 %	86.70 %	97.86 %	83.84 %	85.10 %	97.86 %	86.30 %	82.52 %
	Encriptación	98.10 %	73.82 %	60.44 %	97.96 %	90.12 %	63.92 %	97.20 %	85.82 %	61.20 %
	Suma de Verificación	87.48 %	84.86 %	88.92 %	93.14 %	99.74 %	99.78 %	91.48 %	99.70 %	60.44 %

Tabla 23 - Porcentaje de uso de CPU utilizando algoritmo DH y SHA3-512

De los resultados obtenidos en las pruebas realizadas con el algoritmo DH (Tabla 23) se puede apreciar que, el máximo porcentaje de consumo de CPU se produce en la suma de verificación de una imagen o texto de 100 MB encriptado con AES-256-CBC.

Chacha20 es el algoritmo requiere el menor porcentaje de uso del procesador en la encriptación, y el valor mínimo se obtiene para esta función aplicada a un archivo de imagen de 100 MB.

Respecto de la suma de verificación, para el conjunto de datos de imagen, DES3 es el algoritmo que menos consume. En el caso del conjunto de datos de texto, Chacha20 es el que menos uso del procesador realiza, y para los videos, AES-256-CBC para tamaño de 1 MB, AES y DES3 para tamaño 50 MB y DES3 para datos de 100 MB.

Resumen resultados métrica porcentaje de uso de CPU:

Analizando los resultados registrados en las Tablas 20, 21, 22 y 23 se puede observar que el porcentaje de uso de CPU disminuye con el aumento del tamaño del archivo sin importar el tipo de datos que se analice, pero en el caso de la suma de verificación, no se puede establecer una relación con el tipo o tamaño de datos. También se puede determinar que la suma de verificación es la función que necesita más recursos de la CPU para ejecutarse, mientras que la encriptación es la que menos consume.

Dado que este comportamiento no se corresponde con lo esperado, es decir, se estimaba que el consumo de la CPU aumentaría con el tamaño del archivo, se recurrió al análisis de otras variables para poder explicar estos resultados.

Teniendo en cuenta las tablas obtenidas en la sección anterior con la cantidad de accesos a memoria externa, se observó que a medida que aumenta el tamaño del archivo, aumenta la cantidad de accesos, y esto puede ocasionar que la CPU esté en reposo mientras el sistema realiza tareas de lectura y escritura de datos.

Para comprobar esto se utilizaron los comandos *iostat* y *sar* y se repitieron algunas de las pruebas para archivos de 1MB y 100MB.

El comando *iostat* se utiliza para monitorear la carga de dispositivos de entrada/salida del sistema, observando el tiempo que los dispositivos están activos en relación con sus tasas de transferencia promedio. Entrega un informe que consta de una fila de encabezado de CPU seguida de una fila de estadísticas de CPU. En sistemas multiprocesador, las estadísticas de CPU se calculan en todo el sistema como promedios entre todos los procesadores.

El parámetro de intervalo especifica la cantidad de tiempo en segundos entre cada informe. En este caso se utilizó 1 segundo que es el mínimo permitido.

El comando *iostat* genera dos tipos de informes, el informe de utilización de CPU y el informe de utilización del dispositivo. La información que necesitamos se va a extraer del primer informe que tiene el siguiente formato:

%user: Muestra el porcentaje de utilización de CPU que se produjo durante la ejecución a nivel de usuario (aplicación).

%nice: Muestra el porcentaje de utilización de la CPU que se produjo durante la ejecución a nivel de usuario con prioridad inicial.

%system: Muestra el porcentaje de utilización de la CPU que se produjo durante la ejecución a nivel del sistema (kernel).

%iowait: Muestra el porcentaje de tiempo que la CPU o CPUs estuvieron inactivas durante el cual el sistema tuvo una solicitud de E/S pendiente del dispositivo de almacenamiento.

%steal: Muestra el porcentaje de tiempo que la CPU o CPUs virtuales pasaron en espera involuntaria mientras el hipervisor prestaba servicio a otro procesador virtual.

%idle: Muestra el porcentaje de tiempo que la CPU o CPUs estuvieron inactivas y el sistema no tuvo una solicitud de E/S pendiente del dispositivo de almacenamiento.

El segundo informe generado por el comando *iostat* es el Informe de utilización del dispositivo. El informe del dispositivo proporciona estadísticas por dispositivo físico o partición. Las tasas de transferencia se muestran en bloques de 1K de forma predeterminada.

Se muestra como ejemplo a continuación la salida del comando *iostat* para la RPI 3 cuando está en reposo:

```

rpi3@rpi6402:~ $ iostat
Linux 5.15.76-v8+ (rpi6402)      24/10/23      _aarch64_      (4 CPU)

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           0.01    0.00    0.01    0.01    0.00   99.97

Device            tps    kB_read/s    kB_wrtn/s    kB_dscd/s    kB_read    kB_wrtn    kB_dscd
mmcblk0           0.12         0.26         2.90         4.07    2322460    26420956    37058686

```

Figura 4 – Captura tomada de la ejecución del comando iostat

La información que brinda el primer informe se puede obtener también con el comando *sar*, el cual permite visualizarlo mejor en tiempo real, imprimiendo el tiempo en el que se tomó la muestra y arrojando los valores promedios de las muestras. Se deja a continuación una captura tomada de la RPI 3 en estado de reposo con el comando *sar*.

```

Linux 5.15.76-v8+ (rpi6402)      29/10/23      _aarch64_      (4 CPU)

16:42:26      CPU      %user   %nice   %system %iowait  %steal   %idle
16:42:27      all       0.25    0.00    0.00    0.00    0.00   99.75
16:42:28      all       0.00    0.00    0.00    0.25    0.00   99.75
16:42:29      all       0.00    0.00    0.00    0.00    0.00  100.00
Average:      all       0.08    0.00    0.00    0.08    0.00   99.83

```

Figura 5 – Captura tomada de la ejecución del comando sar

Utilizando estos comandos se pudo determinar que en los archivos de 1MB la CPU se está utilizando durante todo el proceso, prácticamente nunca está en estado

Idle o en IOWait, como se puede ver en los siguientes histogramas realizados en base a los resultados obtenidos para un archivo de video de 1MB. La prueba fue realizada utilizando los mismos scripts comentados anteriormente, y se reiteró 5 veces para las 3 funciones empleando los algoritmos ECDSA/Chacha20/Sha3-512 de donde se obtuvieron 23 muestras.

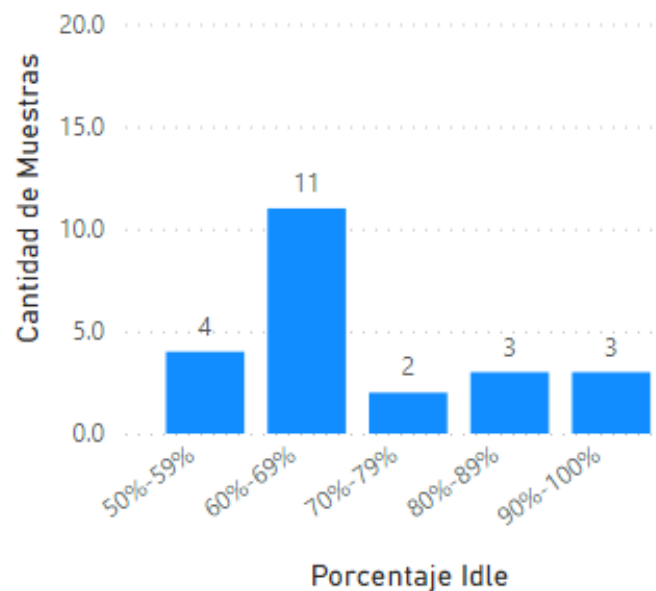


Figura 6 – Histogramas de porcentaje Idle en una RPI 3 para un archivo de 1 MB

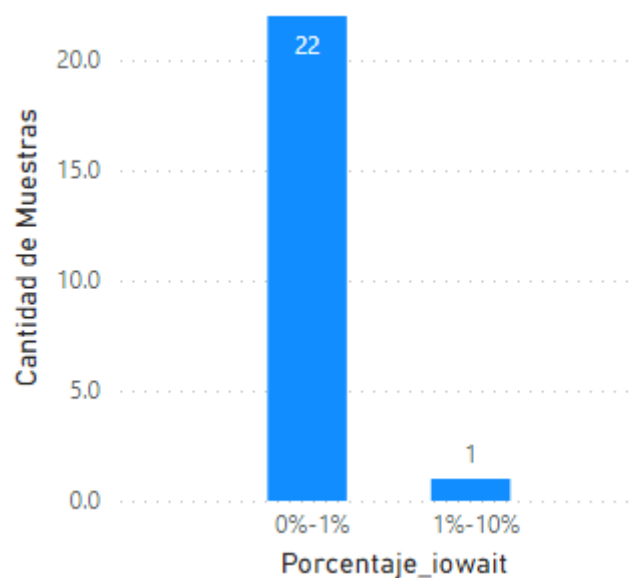


Figura 7 – Histogramas de porcentaje IOWait en una RPI 3 para un archivo de 1 MB

La misma prueba se repitió para un archivo de video de 100 MB y se pudo observar que el porcentaje de las muestras en estado Idle e IOWait aumentan considerablemente.

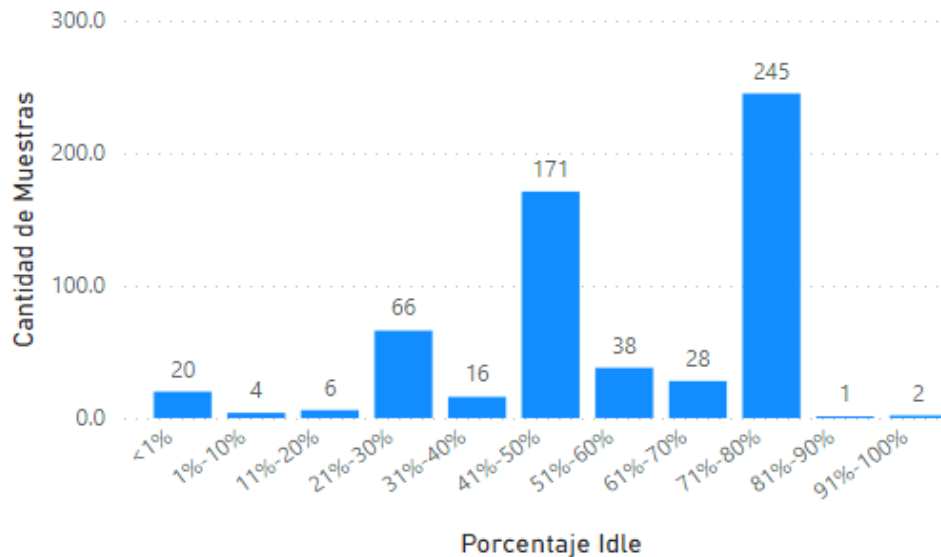


Figura 8 - Histograma de porcentaje Idle en una RPI 3 para un archivo de 100 MB

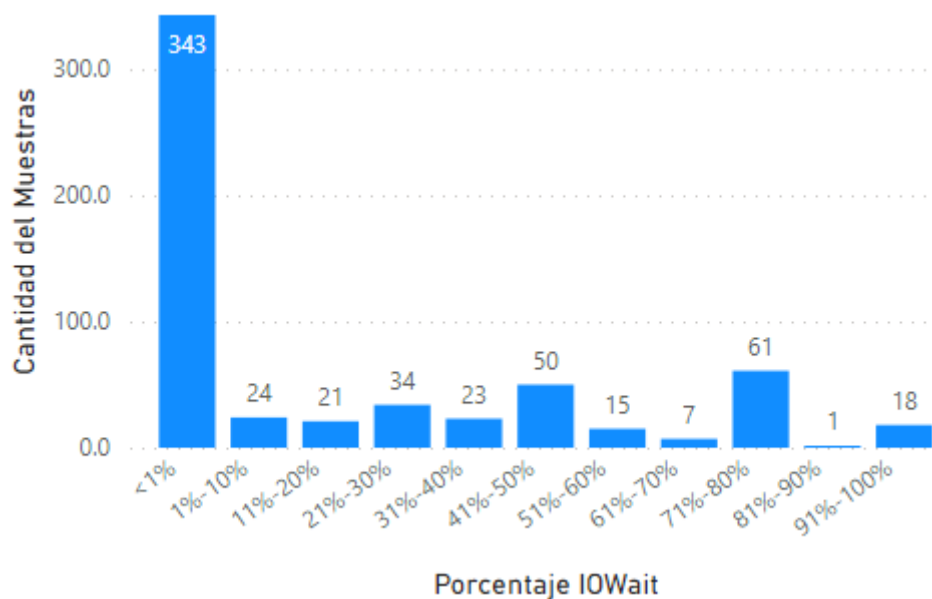


Figura 9 - Histograma de porcentaje IOWait en una RPI 3 para un archivo de 100 MB

Sobre un total de 597 muestras, el promedio del valor de IOWait es del 20,35%, y el 42,55% de las muestras da un valor mayor al 1%. Para un mejor análisis, se repitieron las pruebas, pero discriminando los resultados por función. Con ese fin se

ejecutaron 10 repeticiones de cada función con un archivo de 100 MB para poder obtener más muestras.

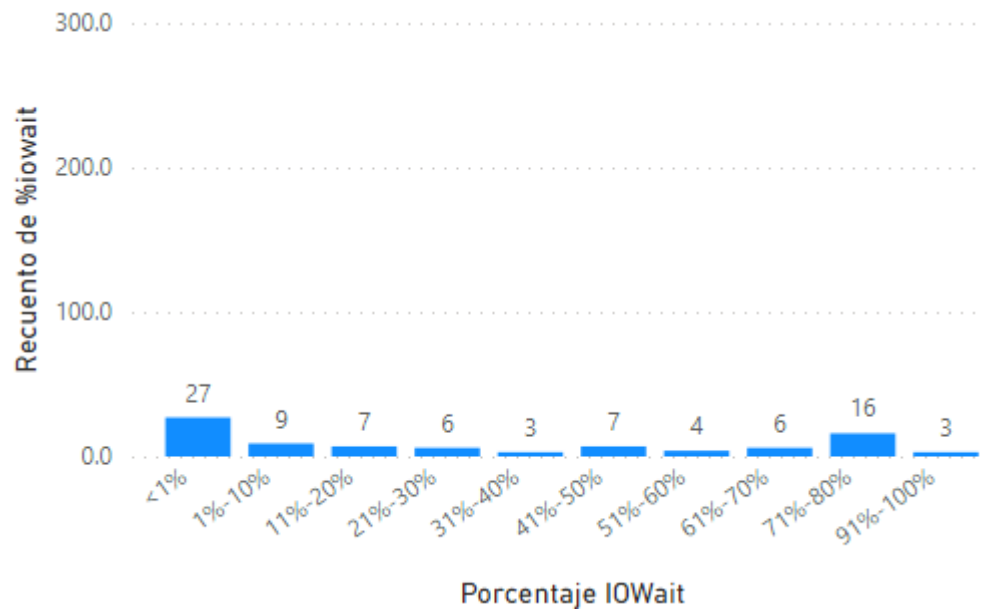


Figura 10 - Histograma de porcentaje IOWait en RPI 3 durante la encriptación de un archivo de 100 MB

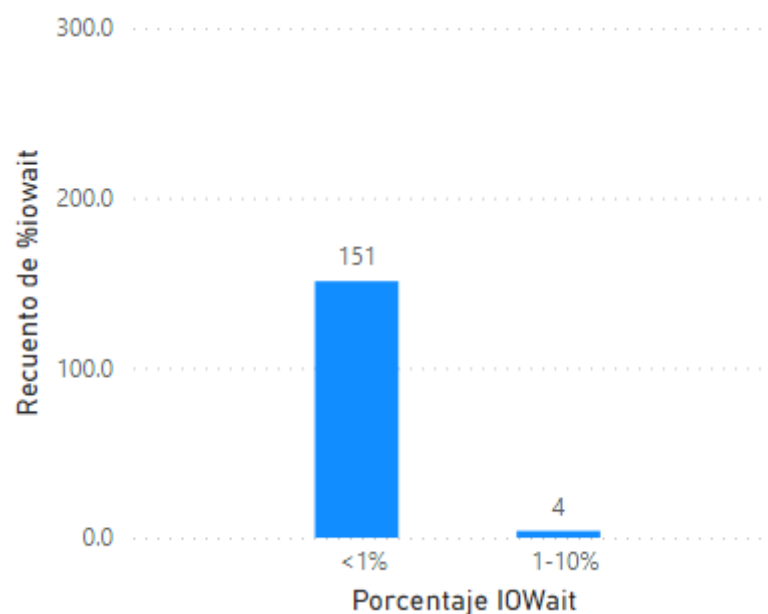


Figura 11 - Histograma de porcentaje IOWait en RPI 3 durante la suma de verificación de un archivo de 100 MB

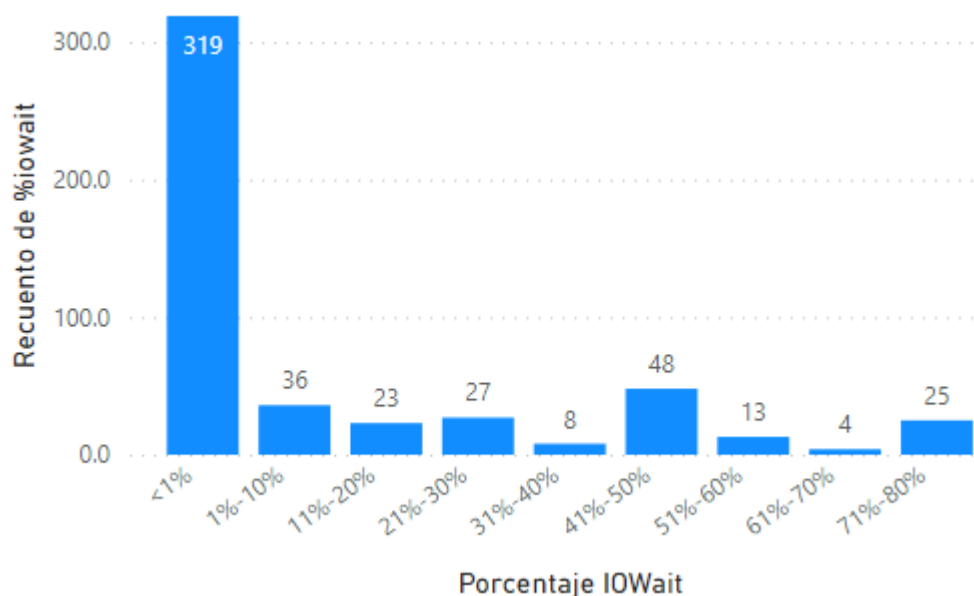


Figura 12 - Histograma de porcentaje IOWait en RPI 3 durante la descriptación de un archivo de 100 MB

A partir de los histogramas anteriores se puede observar que en la encriptación el 69.32% de las muestras corresponde a un valor de IOWait superior al 1%. El promedio obtenido es de 32.23%. En el caso de la suma de verificación el valor promedio de IOWait es muy bajo, 0.09%, y el de la descriptación 36,58%. Esto explica los resultados obtenidos en las Tablas 20 a 23, donde la suma de verificación es la que más porcentaje de CPU consume, seguida por la descriptación y por último la encriptación. Es decir, en la encriptación la CPU está la mayor parte del tiempo en estado ocioso esperando datos para encriptar, algunos de ellos provenientes de la memoria externa. En las Tablas 5 a la 19 se observa que el acceso a la memoria externa aumenta a medida que aumenta el tamaño del archivo, y que la suma de verificación es la que menos memoria externa requiere.

Retomando el análisis de las Tablas 20 a 23, Chacha20 requiere menos porcentaje de CPU que los otros dos algoritmos para la encriptación mientras que DES3 es el que más consume en encriptación y descriptación. En cuanto a la suma de verificación, no hay diferencias significativas en las distintas combinaciones con los algoritmos asimétricos y simétricos.

Estas pruebas se repitieron para los algoritmos de suma de verificación SHA-512, SHA-1 y RIPEMD-160, donde se observó un comportamiento similar en los resultados y por esta razón no se incluyeron las tablas en el informe, pero están

disponibles en el Anexo en la sección “Resultados – Nodo RPI 3 - Porcentaje de Consumo de CPU” en las Tablas A12 a A22 para un análisis más minucioso.

A partir de los resultados obtenidos utilizando todas las combinaciones posibles de algoritmos, se puede establecer que, para los tres conjuntos de datos, a medida que se aumenta el tamaño del archivo, disminuye el porcentaje de uso de la CPU en el caso de la encriptación y desencriptación, ya que se incrementa el tiempo que la CPU se encuentra en estado *IOWait*, es decir, en reposo, en espera de poder hacer una operación de IO, (leer o escribir en el dispositivo de almacenamiento). En el caso de la suma de verificación no se puede establecer una relación con el tipo o tamaño de datos.

En general se puede observar que la función de encriptación es la que menos uso de CPU requiere, mientras que la suma de verificación es la que más consume. También se puede apreciar que el algoritmo Chacha20 necesita disponer de un porcentaje menor del total de la CPU, y la combinación de éste con ECDSA y SHA3-512 en la función de encriptación alcanzan el porcentaje de uso mínimo (Tabla 20).

Por otro lado, el porcentaje de uso máximo ronda entre 90,7% y 90,9% para la suma de verificación en archivos de 100 MB. No obstante, no disminuye significativamente el consumo de CPU utilizado en archivos de menor tamaño.

Tiempo de Ejecución

En esta sección se presentan las Tablas 24 a 27 con los resultados experimentales obtenidos con respecto al tiempo de ejecución que demanda cada función según el algoritmo empleado, y el tipo y tamaño de archivo analizado. Solo se muestran los resultados obtenidos utilizando el algoritmo de suma de verificación SHA3-512, y el resto de los resultados se dejan en el Anexo en las Tablas A23 a A33 para su comparación.

Analizando los valores obtenidos para esta métrica, se puede observar que, para los tres conjuntos de datos, a medida que aumenta el tamaño del archivo, aumenta el tiempo de ejecución de la prueba.

Algoritmo de Verificación SHA3-512

En las Tablas 24 a 27 se pueden observar los resultados obtenidos de las pruebas realizadas en la RPI 3 con el algoritmo SHA3-512 de suma de verificación.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	0.18	5.03	10.67	0.17	4.19	11.11	0.17	4.05	9.95
	Encriptación	0.15	6.43	18.15	0.15	4.48	19.60	1.67	5.41	18.19
	Suma de Verificación	0.05	1.55	3.07	0.49	1.53	3.09	0.06	1.54	3.48
chacha20	Desencriptación	0.15	2.96	9.19	0.15	4.70	7.85	0.15	5.15	9.19
	Encriptación	1.42	8.92	14.65	1.35	5.08	19.49	0.57	3.91	20.45
	Suma de Verificación	0.06	1.54	3.08	0.09	1.53	3.07	0.06	1.55	3.09
des3	Desencriptación	0.25	9.86	22.11	0.25	11.02	21.14	0.26	10.10	20.81
	Encriptación	0.23	14.78	27.09	0.23	9.03	27.48	1.30	9.41	27.03
	Suma de Verificación	0.06	1.70	3.07	0.06	1.53	3.10	0.06	1.55	3.72

Tabla 24 - Tiempo de ejecución utilizando algoritmo ECDSA y SHA3-512

En la Tabla 24 se puede observar que la función que más tiempo tarda en ejecutarse con el algoritmo ECDSA es la encriptación de un texto de 100 MB utilizando el algoritmo Chacha20. En general la encriptación es la función que demora más tiempo en realizarse.

La función que menos tiempo demora es, salvo para un archivo de texto de 1 MB, la suma de verificación, siendo la más rápida, la suma de verificación de una imagen de 1 MB con AES-256-CBC.

En el caso de la desencriptación, para el conjunto de datos de imágenes, Chacha20 es el que menos tiempo demora. Para los archivos de 1 MB y 100 MB de texto y video, también es el óptimo Chacha20, pero para los archivos de texto y video de 50 MB, el algoritmo AES-256-CBC es más rápido.

DES3 es el algoritmo que más demora en encriptación y desencriptación.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	0.17	4.26	9.56	0.18	4.44	9.37	0.18	4.52	9.35
	Encriptación	0.15	4.60	15.72	0.15	3.74	14.88	0.42	4.32	16.44
	Suma de Verificación	0.06	2.20	3.43	0.77	1.54	3.09	0.07	1.55	4.03
chacha20	Desencriptación	0.15	2.91	10.05	0.15	3.42	7.27	0.16	2.70	9.72
	Encriptación	0.14	4.48	14.59	0.14	3.52	14.42	0.30	4.71	14.72
	Suma de Verificación	0.06	2.40	3.09	0.74	1.54	3.08	0.07	1.56	3.09
des3	Desencriptación	0.25	9.99	21.66	0.25	10.06	21.62	0.26	10.12	21.02
	Encriptación	0.23	9.58	24.94	0.23	9.40	26.12	0.30	9.90	24.53
	Suma de Verificación	0.06	2.94	3.09	1.39	1.54	3.08	0.07	1.55	3.11

Tabla 25 - Tiempo de ejecución utilizando algoritmo RSA y SHA3-512

La función que más tiempo tarda en ejecutarse en la Tabla 25 es la encriptación de un texto de 100 MB utilizando el algoritmo DES3. Al igual que con ECDSA, la encriptación es la función que demora más tiempo en realizarse con RSA.

La suma de verificación es, en general, la función que menos tiempo demora, obteniéndose el mejor tiempo para una imagen de 1 MB.

En el caso de la desencriptación, para el conjunto de datos de texto Chacha20 es el que menos tiempo demora, lo mismo sucede con los archivos de 1 MB y 50 MB de imagen y video, pero para los archivos de imagen y video de 100 MB, el algoritmo AES-256-CBC es más rápido.

DES3 es el algoritmo que más demora en encriptación y desencriptación, pero en la suma de verificación no hay diferencias significativas con los otros algoritmos.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	0.17	4.25	9.27	0.17	5.50	9.81	0.17	6.19	9.32
	Encriptación	0.16	8.31	18.31	0.16	3.85	19.91	0.41	5.06	20.58
	Suma de Verificación	0.05	1.94	3.06	0.05	1.52	3.07	0.05	1.54	3.08
chacha20	Desencriptación	0.15	2.86	9.07	0.15	4.13	8.80	0.16	3.63	11.38
	Encriptación	0.16	7.11	14.22	0.15	5.26	19.70	0.82	5.03	19.67
	Suma de Verificación	0.05	2.76	3.07	2.62	1.52	3.06	0.05	1.54	3.21
des3	Desencriptación	0.25	9.90	22.64	0.25	11.33	22.13	0.26	11.38	21.78
	Encriptación	0.24	13.71	27.81	0.23	9.80	31.81	1.25	9.25	28.35
	Suma de Verificación	0.05	2.03	3.07	0.41	1.52	3.06	0.05	1.54	3.07

Tabla 26 - Tiempo de ejecución utilizando algoritmo ECC y SHA3-512

En la Tabla 26 se puede observar que, la función que más tiempo tarda en ejecutarse con ECC es la encriptación de un texto de 100 MB utilizando el algoritmo DES3.

La suma de verificación es la función que menos tiempo demora en general, siendo la más rápida la suma de verificación de archivos de 1 MB de imagen y video. En el caso de archivos de texto, es más rápido para un archivo de 1 MB, pero si se utiliza sobre archivos encriptados con AES-256-CBC.

Chacha20 es el que menos tiempo demora para la desenscriptación, con excepción de la función aplicada a un archivo de video de 100 MB.

DES3 es el algoritmo que más demora en encriptación y desenscriptación.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desenscriptación	0.17	3.70	10.16	0.17	4.72	9.50	0.17	4.89	10.60
	Encriptación	0.15	5.78	17.18	0.15	4.58	15.88	0.20	4.43	17.51
	Suma de Verificación	0.04	1.15	2.27	0.04	1.14	2.28	0.05	1.14	2.83
chacha20	Desenscriptación	0.15	2.99	7.60	0.15	3.42	12.26	0.16	3.09	8.29
	Encriptación	0.14	4.21	16.39	0.14	4.23	14.96	1.92	4.47	14.86
	Suma de Verificación	0.04	1.52	2.28	0.04	1.13	2.28	0.04	1.14	3.23
des3	Desenscriptación	0.25	9.55	21.77	0.25	11.28	21.83	0.26	11.12	22.80
	Encriptación	0.23	11.41	27.10	0.23	9.16	25.75	0.24	9.78	26.86
	Suma de Verificación	0.05	1.75	2.82	0.04	1.13	2.27	0.05	1.14	4.07

Tabla 27 - Tiempo de ejecución utilizando algoritmo DH y SHA3-512

Analizando los resultados obtenidos con el algoritmo DH (Tabla 27) se puede observar que, se repite el mismo comportamiento observado con el algoritmo ECC, con la diferencia de que el máximo tiempo se registra en el archivo de imagen de 100 MB, y este valor es aproximadamente un 18% inferior.

En el caso de la desenscriptación, para el conjunto de datos de imagen y video Chacha20 es el que menos tiempo demora y DES3 continúa siendo el algoritmo que más tiempo requiere.

La suma de verificación es nuevamente la función más rápida en ejecutarse, y el mínimo se obtiene en los archivos de 1 MB sin importar el algoritmo simétrico utilizado.

Resumen resultados métrica tiempo de ejecución:

Las pruebas se repitieron en el RPI 3 para los algoritmos de suma de verificación SHA-512, SHA-1 y RIPEMD-160 y se puede determinar, a partir de los resultados obtenidos, que el tiempo de ejecución aumenta al incrementarse el tamaño de los archivos, pero no depende del tipo de datos.

La función encriptación es la que demora más tiempo en ejecutarse para archivos de 50 MB y 100 MB, mientras que para cualquier tipo y tamaño de archivo la suma de verificación es la función que menos demora.

El tiempo de ejecución más alto fue registrado en la encriptación de un video de 100 MB con la combinación de algoritmos ECDSA/DES3. En general se puede observar que DES3 es el algoritmo más tiempo demanda, mientras que Chacha20 es el algoritmo más rápido. En el caso de la suma de verificación, el algoritmo SHA1, seguido por SHA-512 son los más eficientes para archivos de 50 MB y 100MB. En el caso de los archivos de 1 MB no se aprecia diferencia entre los distintos algoritmos.

Consumo de Energía

Se realizaron mediciones de consumo utilizando el algoritmo de curvas Elípticas y RSA para un archivo de video de tamaño 100 MB. Como algoritmos simétricos se seleccionaron Chacha20 y AES-256-CBC que eran más rápidos para los archivos de 100 MB de acuerdo con los resultados obtenidos anteriormente.



Figura 13 – Fotografía tomada de las mediciones de consumo sobre la RPI 3

La Raspberry Pi 3 en reposo consume 0,25 A.

Se muestra a continuación el promedio de consumo en mA obtenido en cinco repeticiones de cada combinación de algoritmos, y el pico máximo de consumo también en mA.

Algoritmo Simétrico	Función	ECDSA		RSA	
		Consumo	Máx. Consumo	Consumo	Máx. Consumo
aes256cbc	Desencriptación	41.98	50	41.72	50
	Encriptación	52.00	99	45.68	72
chacha20	Desencriptación	39.94	50	40.84	50
	Encriptación	48.71	91	44.76	70

Tabla 28 – Resultados de mediciones de consumo en encriptación y desencriptación en Raspberry Pi3

Se puede observar en la Tabla 28 que chacha20 consume menos energía, tanto en encriptación como en desencriptación, que AES-256-CBC, pero el pico de consumo en la desencriptación es de 0,5 A para todas las combinaciones.

Repitiendo las pruebas, pero esta vez para la función suma de verificación, se puede observar en la Tabla 29 que el algoritmo SHA3-512 consume menos que SHA512, y combinado con ECDSA disminuye aún más el consumo.

Algoritmo Resumen	Función	ECDSA		RSA	
		Consumo	Máx.Consumo	Consumo	Máx.Consumo
sha512	Suma de Verificación			49.14	66
sha3-512	Suma de Verificación	34.12	42	47.19	62

Tabla 29 – Resultado de mediciones de consumo de la función suma de verificación en Raspberry Pi3

Síntesis Resultados Nodo 1

A partir del análisis de las pruebas realizadas en el Nodo 1 (RPI 3), se pueden resumir los resultados obtenidos de la siguiente forma:

- Funciones de Encriptación y Desencriptación

En porcentaje de uso de memoria no hay diferencias significativas entre los diferentes algoritmos.

En accesos a la memoria externa, AES requiere menos memoria para archivos de mayor tamaño (50MB y 100MB). Para archivos pequeños, DES3 requiere menos accesos a la memoria.

Chacha20 es el algoritmo que consume menos recursos de CPU y el más rápido.

- Función de Suma de Verificación

No se pueden establecer relaciones o patrones en los resultados. Los valores obtenidos son similares, se encuentran dentro del mismo orden. En las tablas que se dejan a continuación se anotaron los algoritmos que ligeramente sobresalen, pero en general no se puede establecer un patrón por tipo o tamaño de archivos.

Comparación Encriptación y Desencriptación

EC	Sha3-512	Sha512	Sha1	Ripemd-160
CPU	Chacha20	Chacha20 (Encriptación) AES y chacha20 (desencriptación)	Chacha20	Chacha20 (Encriptación) AES y chacha20 (desencriptación)
Tiempo	Chacha20	Chacha20	Chacha20	Chacha20
Memoria externa	AES (archivos 50 MB y 100MB) DES3 (archivos de 1,1 MB)	AES (archivos 50 MB y 100MB) DES3 (archivos de 1,1 MB)	AES (archivos 50 MB y 100MB) DES3 (archivos de 1,1 MB)	AES (archivos 50 MB y 100MB) DES3 (archivos de 1,1 MB)
Memoria	-	-	-	DES3

ECDSA	Sha3-512	Sha512	Sha1	Ripemd-160
CPU	Chacha20	Chacha encriptación	Chacha	Algoritmo no soportado
Tiempo	AES	Chacha	Chacha	Algoritmo no soportado
Memoria externa	AES (archivos 50MB y 100MB) DES3 (archivos de 1,1 MB)	AES (archivos 50MB y 100MB) DES3 (archivos de 1,1 MB)	AES (archivos 50MB y 100MB) DES3 (archivos de 1,1 MB)	Algoritmo no soportado
Memoria	-	-	-	Algoritmo no soportado

DH	Sha3-512	Sha512	Sha1	Ripemd-160
CPU	Chacha20	Chacha20	Chacha20	Chacha20
Tiempo	Chacha20	Chacha20	Chacha20 (encriptación) Chacha20 y AES (desencriptación)	Chacha20
Memoria externa	AES (archivos 50MB y 100MB) DES3 (archivos de 1,1 MB)	AES (archivos 50MB y 100MB) DES3 (archivos de 1,1 MB)	AES (archivos 50MB y 100MB) DES3 (archivos de 1,1 MB)	AES (archivos 50MB y 100MB) DES3 (archivos de 1,1 MB)
Memoria	-	-	-	-

RSA	Sha3-512	Sha512	Sha1	Ripemd-160
CPU	Chacha20	Chacha20	Chacha20	Chacha20
Tiempo	Chacha20	Chacha20	Chacha20	Chacha20
Memoria externa	*Archivos de 50MB y 100MB AES (Encriptación) Chacha20 (Desencriptación) *Archivo de 1,1MB DES 3	*Archivos de 50MB y 100MB AES (Encriptación) Chacha20 (Desencriptación) *Archivo de 1,1MB DES 3	*Archivos de 50MB y 100MB AES (Encriptación) Chacha20 (Desencriptación) *Archivo de 1,1MB DES 3	*Archivos de 50MB y 100MB AES (Encriptación) Chacha20 (Desencriptación) *Archivo de 1,1MB DES 3
Memoria	-	-	-	AES y DES3

Comparación Suma De Verificación

EC	Sha3-512	Sha512	Sha1	Ripemd-160
CPU	Chacha20	-	-	DES
Tiempo	AES	-	-	AES y Chacha20
Memoria externa	-	-	-	AES y DES3
Memoria	-	-	AES en los archivos de imagen	Chacha20 y DES3

ECDSA	sha3-512	Sha512	Sha1	Ripemd-160
CPU	-	DES3	-	Algoritmo no soportado
Tiempo	Chacha20	Chacha20	Chacha20	Algoritmo no soportado
Memoria externa	-	-	-	Algoritmo no soportado
Memoria	DES3	-	-	Algoritmo no soportado

DH	sha3-512	Sha512	Sha1	Ripemd-160
CPU	DES3	DES3 (archivos de video e imagen) Chacha y DES (archivos de texto)	-	-
Tiempo	AES	-	DES3	Chacha20
Memoria externa	-	DES3 (archivos de video)	DES3	AES
Memoria	-	DES3 (archivos de video)	AES	DES3 (archivos de texto)

RSA	sha3-512	Sha512	Sha1	Ripemd-160
CPU	-	-	-	-
Tiempo	-	Chacha20 (archivos de imagen)	-	Chacha20
Memoria externa	-	-	AES	DES3
Memoria	DES3	Chacha20	-	DES3 y AES

Nodo 2 – Raspberry Pi 1

Teniendo en cuenta los resultados obtenidos con la Raspberry Pi 3, se decidió realizar con la Raspberry Pi 1 las pruebas solo para el algoritmo simétrico Chacha20 y el algoritmo de verificación Sha3-512 que fue la combinación que ofrecía mejor rendimiento en el nodo 1.

Se realizaron las comparaciones entre todos los algoritmos asimétricos seleccionados anteriormente y para los tres tipos de datos y tamaños.

La versión de la herramienta Perf instalada en la RPI 1 no permitía tomar las métricas de cantidad de accesos a memoria externa.

Porcentaje de Uso de Memoria

En la siguiente tabla se muestran los resultados de las pruebas realizadas tomando como métrica el consumo de memoria de la RPI 1. Este modelo de RPI cuenta con 500MB de memoria RAM, es decir, la mitad de lo que dispone el RPI 3.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
Diffie-Hellman	Desencriptación	1.37 %	1.32 %	1.04 %	1.25 %	1.29 %	0.99 %	1.33 %	1.29 %	1.10 %
	Encriptación	1.36 %	1.26 %	1.37 %	1.36 %	1.34 %	1.17 %	1.29 %	1.36 %	1.35 %
	Suma de Verificación	1.30 %	1.23 %	1.21 %	1.32 %	1.18 %	1.21 %	1.30 %	1.02 %	1.24 %
ECC	Desencriptación	1.35 %	1.27 %	1.04 %	1.13 %	1.30 %	1.17 %	1.26 %	1.25 %	1.04 %
	Encriptación	1.35 %	1.37 %	1.40 %	1.24 %	1.86 %	1.38 %	1.12 %	1.62 %	1.64 %
	Suma de Verificación	1.33 %	1.20 %	1.51 %	1.31 %	1.30 %	1.33 %	1.10 %	1.23 %	1.28 %
ECDSA	Desencriptación	1.37 %	1.26 %	0.97 %	1.26 %	1.31 %	1.07 %	1.34 %	1.27 %	1.11 %
	Encriptación	1.35 %	1.34 %	1.34 %	1.25 %	1.35 %	1.11 %	1.30 %	1.36 %	1.33 %
	Suma de Verificación	1.34 %	1.24 %	1.20 %	1.30 %	1.22 %	1.29 %	1.21 %	1.33 %	1.30 %
RSA	Desencriptación	1.37 %	1.25 %	1.02 %	1.27 %	1.30 %	1.10 %	1.36 %	1.29 %	1.09 %
	Encriptación	1.35 %	1.35 %	1.38 %	1.26 %	1.35 %	1.21 %	1.30 %	1.45 %	1.64 %
	Suma de Verificación	1.31 %	1.15 %	1.30 %	1.23 %	1.28 %	1.32 %	1.24 %	1.24 %	1.31 %

Tabla 30 - Comparación de los resultados obtenidos para la métrica Uso de Memoria

Como se puede observar en la Tabla 30, no se puede establecer una relación entre el tipo de datos, tamaño de archivos y algoritmos utilizados en base a los resultados obtenidos.

El máximo consumo de memoria se produce en la encriptación de un archivo de texto de 50 MB utilizando el algoritmo ECC, mientras que el valor mínimo se obtiene en la desenscriptación un archivo de imagen de 100 MB con el algoritmo ECDSA. No obstante, no se aprecian diferencias significativas en el porcentaje de uso de memoria para todos los resultados.

Porcentaje de Consumo de CPU

A continuación, se presentan los resultados de las pruebas realizadas tomando como métrica el consumo de CPU de la RPI 1. A diferencia de la RPI 3, el Nodo 2 cuenta con un solo núcleo, y un procesador de 32 bits.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
Diffie-Hellman	Desenscriptación	42.00 %	35.28 %	29.12 %	41.42 %	33.04 %	29.52 %	40.78 %	30.08 %	29.02 %
	Encriptación	40.92 %	26.16 %	24.70 %	37.82 %	25.74 %	24.80 %	41.22 %	25.30 %	24.85 %
	Suma de Verificación	40.96 %	45.58 %	45.52 %	40.58 %	45.58 %	45.48 %	41.18 %	45.44 %	45.52 %
ECC	Desenscriptación	40.52 %	31.90 %	29.08 %	38.63 %	32.60 %	29.92 %	40.08 %	30.56 %	29.38 %
	Encriptación	38.62 %	25.48 %	25.18 %	39.88 %	25.86 %	25.24 %	38.83 %	25.70 %	24.62 %
	Suma de Verificación	41.08 %	45.72 %	45.48 %	40.58 %	45.58 %	45.48 %	40.74 %	45.78 %	45.54 %
ECDSA	Desenscriptación	41.58 %	34.34 %	30.20 %	39.60 %	32.80 %	30.10 %	40.43 %	32.44 %	30.02 %
	Encriptación	40.70 %	26.04 %	25.52 %	36.13 %	26.08 %	25.28 %	40.28 %	26.54 %	25.70 %
	Suma de Verificación	41.32 %	45.58 %	45.44 %	40.30 %	45.50 %	45.32 %	39.30 %	45.68 %	45.38 %
RSA	Desenscriptación	40.52 %	34.62 %	29.56 %	39.18 %	30.76 %	29.38 %	39.90 %	32.68 %	29.66 %
	Encriptación	39.98 %	25.92 %	24.94 %	40.76 %	26.14 %	24.72 %	40.90 %	25.24 %	25.12 %
	Suma de Verificación	41.88 %	45.60 %	45.36 %	40.27 %	45.52 %	45.32 %	41.20 %	45.64 %	45.42 %

Tabla 31 - Comparación de los resultados obtenidos para la métrica Consumo de CPU

En la Tabla 31 se puede observar que no se aprecia diferencia significativa entre los diferentes algoritmos asimétricos. El máximo consumo de CPU se produce en la suma de verificación de un archivo de video de 50 MB encriptado con la combinación ECC y Chacha20, mientras que la encriptación para un archivo de video de 100 MB utilizando la misma combinación, es la que menos consume.

La encriptación y desenscriptación consumen más CPU para los archivos de menor tamaño. Este mismo comportamiento se observó en el nodo 1, por lo cual se decidió utilizar nuevamente el comando sar para ver el porcentaje de tiempo en el que la CPU permanecía en estado IOWait.

Para la prueba se seleccionó la misma combinación de algoritmos y archivos que en el Nodo 1, es decir, ECDSA/Chacha20/SHA3-512 aplicado a un archivo de video de 1 MB y 100 MB. Se realizaron 10 repeticiones de las 3 funciones y se muestran a continuación los histogramas obtenidos.

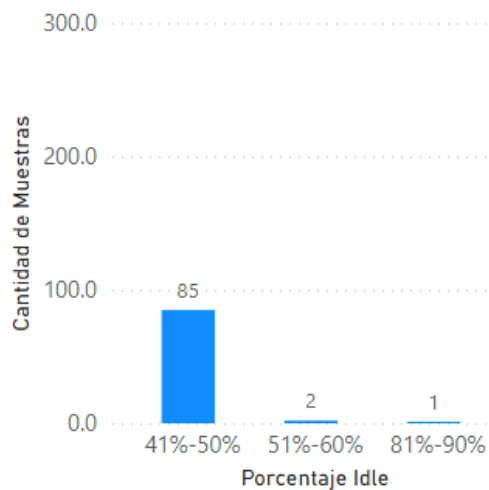


Figura 14 – Histogramas de porcentaje en estado Idle en una RPI 1 para un archivo de 1 MB

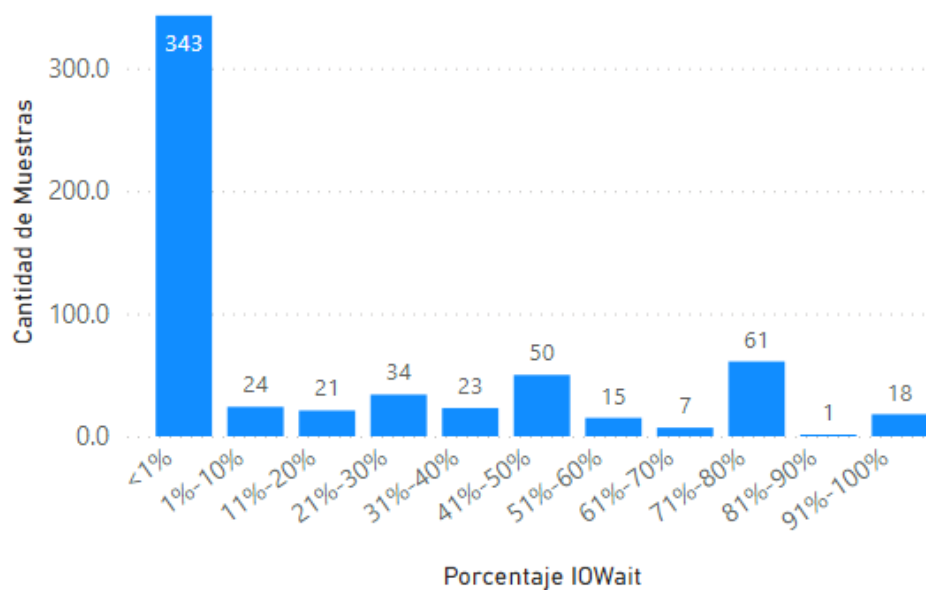


Figura 15 – Histogramas de porcentaje en estado IOWait en una RPI 1 para un archivo de 1 MB

En base a los gráficos se puede observar que para las tres funciones aplicadas a un archivo de 1MB la CPU en ningún momento está en estado Idle, y que el porcentaje de IOWait es del 0.09%

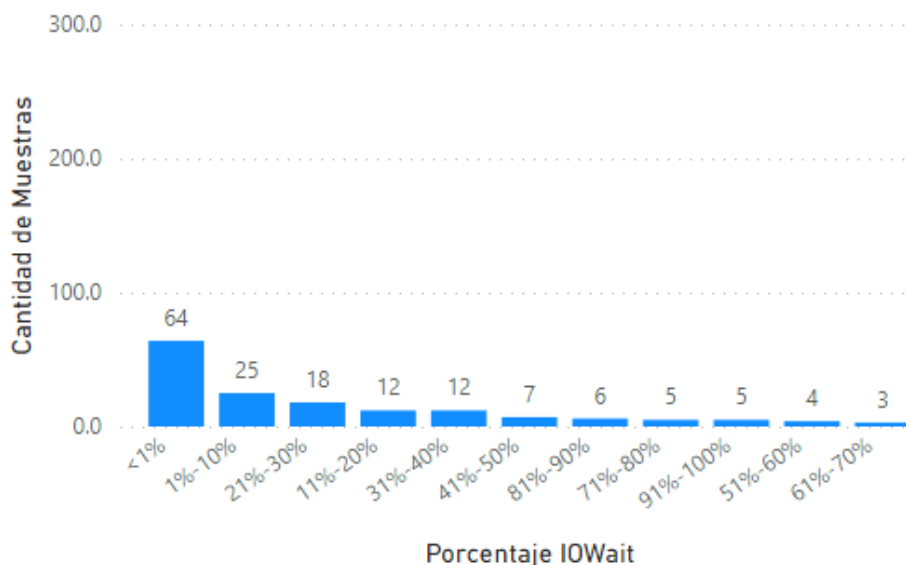


Figura 16 - Histograma de porcentaje IOWait en RPI 1 durante la encriptación de un archivo de 1 MB

A diferencia del archivo de 1 MB, durante la encriptación de un archivo de 100 MB, el porcentaje de IOWait es distinto de cero en la mayoría de las muestras, arrojando el promedio de IOWait un valor de 20.13%.

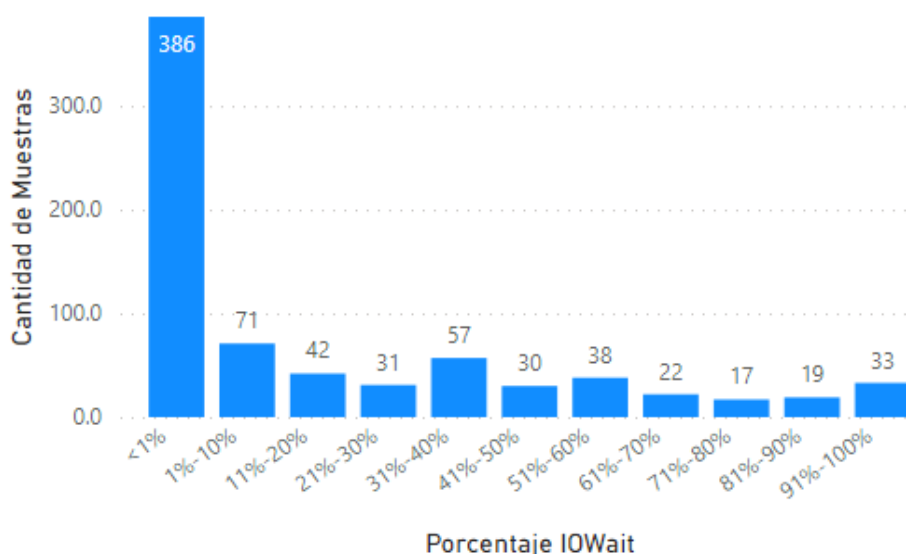


Figura 17 - Histograma de porcentaje IOWait en RPI 1 durante la descriptación de un archivo de 1 MB

En el caso de la descriptación, el tiempo que el procesador pasa en estado IOWait es en promedio menor que en la encriptación, un 19,5 % y por esta razón en la Tabla 31 se ve un porcentaje ligeramente mayor de consumo de CPU para esta función.

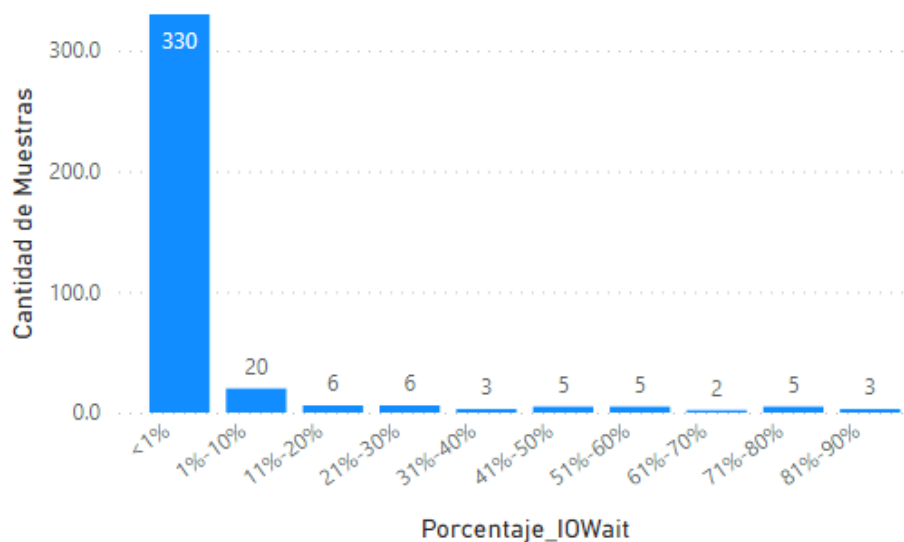


Figura 18 - Histograma de porcentaje IOWait en RPI 1 durante la suma de verificación de un archivo de 100 MB

En la suma de verificación el porcentaje de IOWait es significativamente más bajo, en comparación a las dos funciones anteriores y esta diferencia es clara en el consumo de la CPU en la Tabla 31.

Tiempo de Ejecución

En la siguiente tabla se muestran los resultados de las pruebas realizadas tomando como métrica el tiempo de ejecución de las funciones de encriptación, desencriptación y suma de verificación en una RPI 1. Este modelo de RPI, no solo cuenta con menos memoria RAM, sino que el procesador de un solo núcleo es más lento que el del Nodo1 (700 MHz vs. 1.2 GHz del RPI 3).

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
Diffie-Hellman	Desencriptación	1.30	27.77	68.10	1.27	29.86	67.27	1.36	33.45	68.42
	Encriptación	1.17	27.01	56.50	1.38	27.35	55.97	1.21	27.92	56.28
	Suma de Verificación	1.01	37.01	74.78	1.03	36.73	74.60	1.05	37.18	75.15
ECC	Desencriptación	1.35	31.24	68.28	1.41	30.81	66.56	1.37	33.25	67.77
	Encriptación	1.26	27.76	55.66	1.20	27.42	55.60	1.23	27.85	47.73
	Suma de Verificación	1.01	36.99	74.91	1.03	36.84	74.85	1.06	36.90	75.12
ECDSA	Desencriptación	1.33	28.87	65.77	1.40	30.81	65.70	1.37	30.88	66.46
	Encriptación	1.21	27.13	54.92	1.40	27.02	55.19	1.20	26.92	54.62
	Suma de Verificación	1.06	37.06	74.89	1.10	36.91	75.09	1.10	37.14	75.16
RSA	Desencriptación	1.36	28.36	66.92	1.46	32.73	67.40	1.38	30.77	67.33
	Encriptación	1.21	27.42	55.98	1.22	27.13	56.14	1.21	28.18	56.08
	Suma de Verificación	1.22	37.25	75.29	1.29	37.05	75.23	1.26	37.23	75.35

Tabla 32 - Comparación de los resultados obtenidos para la métrica Tiempo de Ejecución

A medida que aumenta el tamaño del archivo aumenta el tiempo de ejecución de cada función, pero no tiene influencia de qué tipo de datos es el archivo.

La suma de verificación para archivos de 1 MB es la función que menos demora, con excepción de cuando se aplica sobre archivos encriptados con RSA. Por el contrario, para tamaños de archivos de 50 MB y 100 MB, es la que más tiempo demora.

La desencriptación requiere más tiempo que la encriptación sin importar el algoritmo utilizado.

La función que requiere menor tiempo es la suma de verificación de una imagen de 1 MB encriptada con DH o ECC, mientras que la que mayor tiempo requiere es la suma de verificación de un archivo de video 100 MB encriptado con RSA.

No se aprecia diferencia significativa entre los diferentes algoritmos asimétricos.

Síntesis Resultados Nodo 2

Dado que el dispositivo seleccionado como Nodo 2 es una RPI 1, fabricada en el año 2012 con recursos considerablemente inferiores a la RPI 3 (fabricada en el 2016) y a la RPI 4 (del año 2019), tal como se comparó en el capítulo anterior, casi sin disponibilidad en el mercado y poco utilizada en nuevos proyectos de desarrollo, sólo se consideró necesario realizar las pruebas sobre un algoritmo simétrico y uno de suma de verificación con fines comparativos. En base al análisis de los resultados del Nodo 1 se seleccionó la combinación Chacha20/SHA3-512 que fue la que obtuvo mejor rendimiento.

A continuación se resumen los principales resultados obtenidos en el Nodo 2.

- Porcentaje de Uso de Memoria

Respecto de esta métrica no se observan diferencias significativas entre los diferentes algoritmos ni se puede establecer una relación entre el tipo de datos, tamaño de archivos y algoritmos utilizados en base a los resultados obtenidos.

En comparación con la RPI 3, los porcentajes obtenidos de uso de memoria son de 2 a 5 veces mayores que los registrados en el nodo 1, pero cabe recordar, que la RPI 1 dispone de la mitad de la memoria RAM que la RPI 3.

- Porcentaje de Uso de CPU

No se pueden establecer relaciones o patrones en los resultados, pero se puede resaltar que la suma de verificación es la función que más porcentaje de CPU consume independientemente del tipo y tamaño de archivo, seguida por la descriptación.

A través del uso de la herramienta sar se encontró la explicación a este comportamiento, donde se ve que la CPU está todo el tiempo en uso mientras se ejecuta esta función, a diferencia de la encriptación y la descriptación que debe esperar los datos de entrada/salida.

En la mayoría de los resultados se puede observar que el consumo es de aproximadamente la mitad de lo que requiere la RPI 3, pero esto se puede explicar en base a la siguiente métrica de tiempo de ejecución.

- Tiempo de Ejecución

A medida que aumenta el tamaño del archivo aumenta el tiempo de ejecución de cada función, pero no tiene influencia de qué tipo de datos es el archivo.

En base a los resultados obtenidos también se puede determinar que la encriptación es la función más rápida en ejecutarse para los archivos de 50 y 100 MB sin importar el algoritmo utilizado, mientras que la suma de verificación es la más rápida para los archivos de 1 MB.

En contraste con la RPI 3, los tiempos de ejecución aumentan entre 3 y 10 veces, llegando incluso a ser 20 veces más lento en algunos casos puntuales.

A modo de ejemplo, podemos tomar el caso de la desenscriptación de un archivo de 100 MB de video y aplicar la combinación de algoritmos ECDSA/Chacha20/Sha3-512 para comparar las métricas.

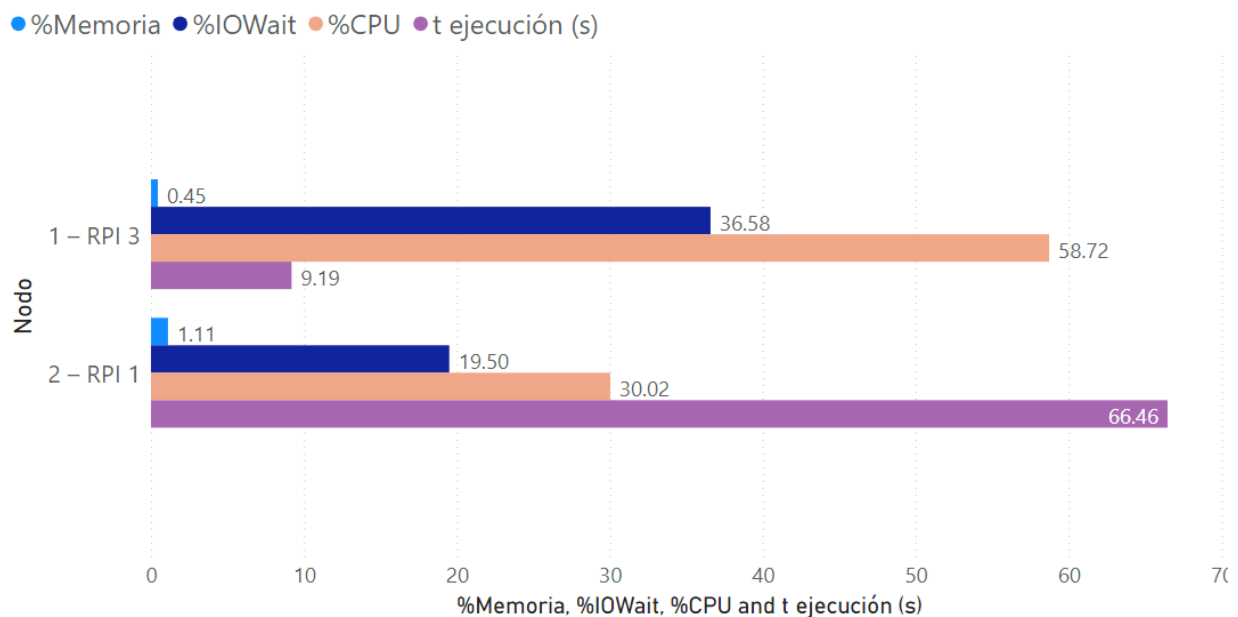


Figura 19 – Comparación de métricas entre RPI 1 y RPI3

Nodo 3 – Raspberry Pi 4

Las pruebas realizadas en el Nodo RPI 3 se repitieron en el Nodo RPI 4, que tal como se puede observar en la tabla comparativa de la sección Metodología, este nodo dispone de mayor capacidad de memoria y CPU que los dos nodos anteriores.

Porcentaje de Uso de Memoria

A continuación, se presentan los resultados experimentales obtenidos en la RPI 4 con respecto al porcentaje de memoria utilizado por cada función empleando el algoritmo de suma de verificación SHA-512. El resto de los algoritmos demostraron un comportamiento muy similar en los resultados obtenidos para esta métrica en el nodo 3, y se dejan en el anexo las tablas para un análisis más detallado.

Algoritmo de Verificación SHA-512

En las Tablas 33, 34, 35 y 36 se muestran los resultados obtenidos de consumo de memoria por cada una de las funciones teniendo en cuenta el algoritmo de verificación SHA-512.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	0.20 %	0.20 %	0.19 %	0.20 %	0.18 %	0.18 %	0.19 %	0.19 %	0.19 %
	Encriptación	0.19 %	0.19 %	0.19 %	0.19 %	0.20 %	0.19 %	0.20 %	0.19 %	0.20 %
	Suma de Verificación	0.10 %	0.10 %	0.10 %	0.10 %	0.10 %	0.10 %	0.10 %	0.10 %	0.10 %
chacha20	Desencriptación	0.20 %	0.19 %	0.19 %	0.18 %	0.19 %	0.20 %	0.18 %	0.19 %	0.19 %
	Encriptación	0.19 %	0.20 %	0.20 %	0.19 %	0.20 %	0.19 %	0.20 %	0.19 %	0.20 %
	Suma de Verificación	0.10 %	0.10 %	0.11 %	0.10 %	0.10 %	0.10 %	0.10 %	0.11 %	0.10 %
des3	Desencriptación	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.24 %	0.20 %	0.24 %
	Encriptación	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.24 %	0.20 %	0.24 %
	Suma de Verificación	0.17 %	0.17 %	0.17 %	0.17 %	0.17 %	0.17 %	0.23 %	0.17 %	0.22 %

Tabla 33 - Porcentaje de uso de memoria RAM total utilizando algoritmo ECDSA y SHA-512

Tabla 33 podemos notar que para el algoritmo DES3 la encriptación y desencriptación requieren el mismo porcentaje para los conjuntos de datos de texto

e imagen y es igual al 0,20% de memoria. En el caso de los archivos de video de 1 MB y 100 MB este porcentaje se incrementa a 0,24%, obteniéndose el máximo valor de todas las pruebas realizadas. Respecto de la suma de verificación, se obtiene que el consumo es menor que para las otras funciones con los tres conjuntos de datos.

En el caso de AES-256-CBC y Chacha20, los porcentajes obtenidos en encriptación y desencriptación son muy similares en todas las pruebas variando entre el 0,18% y el 20%, y la suma de verificación varía entre 0,10% y 0,11%.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.19 %	0.20 %	0.19 %
	Encriptación	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %
	Suma de Verificación	0.17 %	0.17 %	0.17 %	0.17 %	0.17 %	0.17 %	0.10 %	0.14 %	0.10 %
chacha20	Desencriptación	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.23 %	0.20 %	0.23 %
	Encriptación	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.23 %	0.20 %	0.23 %
	Suma de Verificación	0.17 %	0.17 %	0.17 %	0.17 %	0.17 %	0.17 %	0.20 %	0.17 %	0.20 %
des3	Desencriptación	0.20 %	0.20 %	0.19 %	0.20 %	0.19 %	0.20 %	0.20 %	0.20 %	0.19 %
	Encriptación	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.19 %	0.20 %
	Suma de Verificación	0.10 %	0.10 %	0.10 %	0.10 %	0.10 %	0.10 %	0.10 %	0.10 %	0.10 %

Tabla 34 - Porcentaje de uso de memoria RAM total utilizando algoritmo RSA y SHA-512

En la Tabla 34 se observan los resultados obtenidos con el algoritmo RSA y, se puede notar que, para el algoritmo AES-256-CBC, la encriptación y desencriptación requieren el mismo porcentaje de memoria para los conjuntos de datos de imagen y texto, y en el caso del conjunto de datos de video la diferencia no es significativa. La suma de verificación obtiene menor valor de consumo para los tres conjuntos de datos, obteniéndose el mínimo para los archivos de video de 1 MB y 100 MB.

En el caso de Chacha20, el conjunto de datos de imagen y texto requiere el mismo porcentaje de memoria de 0,20% tanto para encriptación como desencriptación, mientras que el de suma de verificación es menor. Para el conjunto de datos de video, los porcentajes obtenidos en encriptación y desencriptación de archivos de 1 MB y 100 MB son iguales a 0,23% y son los valores máximos obtenidos en todas las pruebas.

Al evaluar DES3, podemos observar un consumo ligeramente menor en algunas de las pruebas realizadas en encriptación y desencriptación, pero no representan variaciones significativas. La suma de verificación consume menos memoria que las funciones de encriptación y desencriptación y es el mismo porcentaje para todos los tipos y tamaños de archivo. Este valor de 0,10% es el mínimo porcentaje obtenido en todas las pruebas, y coincide con el que se obtuvo con AES-256-CBC para archivos de video de 1 MB y 100 MB.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	0.20 %	0.20 %	0.19 %	0.20 %	0.19 %	0.20 %	0.19 %	0.20 %	0.19 %
	Encriptación	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %
	Suma de Verificación	0.12 %	0.10 %	0.11 %	0.10 %	0.10 %	0.10 %	0.10 %	0.13 %	0.10 %
chacha20	Desencriptación	0.18 %	0.20 %	0.19 %	0.19 %	0.20 %	0.18 %	0.23 %	0.20 %	0.23 %
	Encriptación	0.27 %	0.32 %	0.20 %	0.19 %	0.20 %	0.20 %	0.23 %	0.24 %	0.23 %
	Suma de Verificación	0.12 %	0.12 %	0.10 %	0.10 %	0.10 %	0.10 %	0.20 %	0.10 %	0.20 %
des3	Desencriptación	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.19 %
	Encriptación	0.20 %	0.20 %	0.21 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %
	Suma de Verificación	0.17 %	0.18 %	0.17 %	0.17 %	0.17 %	0.17 %	0.10 %	0.17 %	0.10 %

Tabla 35 - Porcentaje de uso de memoria RAM total utilizando algoritmo ECC y SHA-512

Al realizar las pruebas con el algoritmo ECC, podemos resaltar que la combinación con el algoritmo AES-256-CBC consume la misma cantidad de memoria en el proceso de encriptación para todos los tamaños y tipos de archivos, tal como se muestra en la Tabla 35. La desencriptación ofrece un consumo muy similar, mientras que la suma de verificación es la que menos memoria necesita. Con DES3 ocurre un comportamiento similar.

En el caso de Chacha20, la encriptación consume más que la desencriptación, y el valor máximo sucede en la encriptación de una imagen de 50 MB.

En los tres algoritmos asimétricos analizados se puede observar que la suma de verificación consume menos memoria que la encriptación y desencriptación, y que los valores mínimos se obtienen para los conjuntos de datos de texto encriptados con AES-256-CBC y Chacha20. Este valor se repite para los archivos de 1 MB y 100 MB de video encriptados con AES-256-CBC y DES3, y el archivo de video de 50 MB y de

imagen de 100 MB encriptado con Chacha20. También ocurre para un archivo de 50 MB de imagen encriptado con AES-256-CBC.

Además, se puede destacar que en ninguno de los casos el sistema hace un uso mayor al 0.32% de los recursos de memoria del dispositivo de cómputo.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	0.20 %	0.19 %	0.20 %	0.20 %	0.20 %	0.19 %	0.19 %	0.20 %	0.18 %
	Encriptación	0.20 %	0.20 %	0.19 %	0.20 %	0.20 %	0.18 %	0.20 %	0.19 %	0.20 %
	Suma de Verificación	0.10 %	0.10 %	0.10 %	0.10 %	0.10 %	0.10 %	0.10 %	0.10 %	0.10 %
chacha20	Desencriptación	0.19 %	0.20 %	0.19 %	0.19 %	0.19 %	0.19 %	0.18 %	0.20 %	0.20 %
	Encriptación	0.20 %	0.19 %	0.20 %	0.20 %	0.20 %	0.19 %	0.20 %	0.19 %	0.19 %
	Suma de Verificación	0.10 %	0.10 %	0.06 %	0.10 %	0.10 %	0.10 %	0.10 %	0.09 %	0.10 %
des3	Desencriptación	0.19 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.18 %	0.20 %	0.20 %
	Encriptación	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %
	Suma de Verificación	0.16 %	0.16 %	0.16 %	0.16 %	0.16 %	0.16 %	0.09 %	0.16 %	0.10 %

Tabla 36 - Porcentaje de uso de memoria RAM total utilizando algoritmo DH y SHA-512

En la Tabla 36 se puede observar para el algoritmo AES-256-CBC que la suma de verificación consume menos memoria que la encriptación y desencriptación, obteniéndose el valor de 0,10% para todos los tipos y tamaños de archivo. El valor mínimo para desencriptación se obtiene con un archivo de texto y de video de 100 MB.

Con el algoritmo Chacha20 se obtiene el menor porcentaje de consumo de memoria en la desencriptación de un archivo de video de 1 MB. No obstante, los porcentajes de uso de memoria de encriptación y desencriptación son muy similares. La suma de verificación es la función que menos consume, obteniéndose el mínimo de 0,06% para un archivo de imagen de 100 MB.

En el caso de DES3 se puede observar resultados muy similares en todos los conjuntos de datos, y ligeramente superiores a los de los otros dos algoritmos.

En general, se registraron valores muy similares entre los tres algoritmos asimétricos y se puede resaltar que en ninguno de los casos el sistema hace un uso mayor al 0,20% de los recursos de memoria del dispositivo de cómputo.

Resumen resultados métrica porcentaje de uso de memoria:

En base a los resultados registrados en las Tablas 33, 34, 35 y 36, no se puede establecer una relación entre el porcentaje de uso de memoria y el tipo y tamaño de archivo ya que se obtuvieron valores muy similares.

En general se observa que la función de encriptación es la que más recursos de memoria consume, pero no se diferencia casi del consumo en la desencriptación, y que el porcentaje de uso de memoria no supera el 0.32% en ninguna de las tres funciones analizadas. El porcentaje más alto corresponde a la encriptación con el algoritmo ECC en combinación con Chacha (Tabla 35), aplicado a un archivo de imagen de 50 MB, y se observa que este algoritmo es, en general, el que más memoria utiliza. Analizando los resultados obtenidos con el resto de los algoritmos de suma de verificación este comportamiento se mantiene y el valor máximo se registró con el algoritmo ECC como se puede apreciar en la Tabla A40 del anexo con un valor de 0,32%. Por el contrario, el mínimo consumo es requerido por el algoritmo DH con un valor de 0,16% en combinación con DES3 y SHA1 (Tabla A41), y con ECC con las combinaciones AES con SHA1 y AES con Ripemd-160 (Tablas A40 y A43 respectivamente).

Por otro lado, el valor mínimo de consumo para la suma de verificación se registró en la combinación DH/Chacha20 y SHA-512 con un porcentaje de 0,06% (Tabla 36).

A partir de los resultados obtenidos se puede observar que ninguno de los algoritmos consume un porcentaje significativo de memoria, que pueda afectar otras tareas ejecutadas en el nodo. DH es el óptimo en el caso de que se requiriera trabajar solo con un algoritmo para todas las funciones ya que, en general, es el que consume menos memoria para todas las funciones, tipos y tamaños de archivos, seguido por RSA.

Porcentaje de Memoria Externa

En esta sección se presentan las tablas con los resultados experimentales obtenidos en la RPI 4 con respecto al porcentaje de memoria utilizado por cada función según el algoritmo empleado, y el tipo y tamaño de archivo analizado.

Algoritmo de Verificación SHA1

En las tablas 37 a 40 se muestran los resultados obtenidos utilizando el algoritmo de suma de verificación SHA-1. Los algoritmos restantes ofrecieron un comportamiento similar y se dejan en el anexo para un análisis más minucioso.

En todas las pruebas se pudo determinar cómo patrón que, para los tres conjuntos de datos, a medida que se aumenta el tamaño del archivo, aumenta la cantidad de accesos a la memoria externa y que la suma de verificación requiere menos accesos que la encriptación, seguida por la desenscriptación.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desenscriptación	35,917,526	531,892,884	1,043,903,141	35,978,418	529,635,903	1,041,912,464	36,483,380	533,546,740	1,046,235,927
	Encriptación	33,188,903	399,541,212	777,546,734	33,229,065	397,765,537	776,227,279	33,605,544	400,727,136	779,106,405
	Suma de Verificación	4,620,153	23,324,204	42,667,911	4,621,629	23,282,274	42,621,084	4,640,395	23,364,571	42,810,489
chacha20	Desenscriptación	32,081,653	345,977,756	669,909,544	32,120,254	344,398,945	668,742,431	32,438,863	347,014,268	671,108,645
	Encriptación	30,364,426	262,154,094	501,244,737	30,380,048	261,055,502	500,391,017	30,619,358	262,971,004	502,360,174
	Suma de Verificación	4,623,325	23,313,071	42,625,585	4,622,732	23,269,605	42,582,908	4,644,359	23,398,958	42,948,017
des3	Desenscriptación	125,631,718	5,408,584,018	10,862,280,616	126,258,505	5,384,808,758	10,841,552,130	131,643,445	5,427,067,637	10,888,240,213
	Encriptación	96,553,834	3,997,795,167	8,025,082,796	97,018,221	3,980,139,643	8,009,811,392	100,988,713	4,011,496,091	8,044,101,018
	Suma de Verificación	4,619,687	23,303,357	42,641,021	4,622,162	23,270,847	42,589,076	4,640,220	23,447,860	42,755,677

Tabla 37 - Cantidad de accesos a memoria externa utilizando algoritmo ECDSA y SHA1

Los resultados de la Tabla 37 muestran este patrón de comportamiento, en el que la mínima cantidad de accesos de memoria externa es requerida para los archivos de 1MB y el máximo para la función desenscriptación. Ambos valores extremos ocurren utilizando el algoritmo DES3.

Chacha20 es el algoritmo que menos memoria externa utiliza para encriptación y desenscriptación, mientras que en la suma de verificación no hay una clara diferencia entre los tres algoritmos simétricos.

Algorit...	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desenscriptación	36,311,502	552,107,981	1,084,786,446	36,370,885	549,678,290	1,082,606,804	36,893,455	553,846,184	1,086,927,660
	Encriptación	33,189,059	399,467,165	777,549,238	33,230,084	397,840,833	776,047,706	33,602,890	400,699,852	779,283,858
	Suma de Verificación	5,975,789	24,625,202	43,950,071	5,952,843	24,655,380	43,931,639	5,963,303	24,772,978	43,979,017
chacha20	Desenscriptación	35,044,025	490,721,297	961,434,438	35,098,732	488,761,563	959,396,185	35,559,797	492,330,924	963,349,092
	Encriptación	30,363,557	262,117,037	501,433,795	30,384,826	261,033,599	500,360,755	30,637,061	262,950,490	502,571,162
	Suma de Verificación	5,956,647	24,656,611	44,073,745	5,953,854	24,627,800	43,951,874	5,954,463	24,728,780	44,250,381
des3	Desenscriptación	101,487,297	4,238,616,400	8,509,489,641	101,977,930	4,219,918,888	8,492,739,522	106,194,907	4,252,974,691	8,529,657,857
	Encriptación	96,551,157	3,997,894,087	8,025,356,588	97,017,126	3,980,218,260	8,009,811,124	100,994,890	4,011,555,625	8,044,087,004
	Suma de Verificación	5,960,862	24,689,074	43,996,930	5,951,806	24,656,431	43,990,558	5,983,662	24,745,562	44,068,949

Tabla 38 - Cantidad de accesos a memoria externa utilizando algoritmo RSA y SHA1

En la Tabla 38 es posible notar que, los valores mínimos de las tres funciones se obtienen para los archivos de tamaño 1 MB.

Chacha20 requiere menos accesos que los otros dos algoritmos para la encriptación y desencriptación. DES3 utiliza el máximo para la desencriptación de un archivo de video de 100 MB y aumenta considerablemente la cantidad de accesos respecto de los otros dos algoritmos (8 veces).

El mínimo necesario de memoria externa en la suma de verificación se obtiene para el archivo de texto de 1 MB encriptado con DES3, no obstante, en general la suma de verificación requiere de recursos similares con los tres algoritmos.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	35,920,003	531,907,540	1,044,064,428	35,974,114	529,598,162	1,041,660,834	36,480,641	533,569,497	1,046,259,845
	Encriptación	27,133,261	399,361,026	777,525,173	33,230,489	397,775,876	776,074,928	33,606,595	400,765,238	779,437,085
	Suma de Verificación	3,236,785	22,331,387	41,809,029	3,322,950	22,224,215	41,733,498	3,341,184	22,372,825	41,921,038
chacha20	Desencriptación	32,084,053	346,047,150	669,968,896	32,118,729	344,462,450	668,477,298	32,439,869	347,037,879	671,507,274
	Encriptación	30,358,652	262,030,601	501,243,644	30,382,603	261,156,486	500,455,844	30,624,151	262,960,409	502,292,739
	Suma de Verificación	3,320,021	22,280,813	42,050,093	3,321,215	22,234,289	41,768,644	3,341,517	22,348,606	41,970,862
des3	Desencriptación	125,630,891	5,408,698,591	10,862,198,758	126,262,810	5,384,732,537	10,841,279,609	131,644,061	5,427,135,671	10,888,043,066
	Encriptación	96,548,556	3,997,740,865	8,025,365,546	97,015,885	3,980,226,129	8,009,641,008	100,991,469	4,011,493,456	8,044,137,297
	Suma de Verificación	3,317,511	22,293,463	41,768,143	3,320,868	22,284,871	41,872,135	3,338,204	22,338,753	42,028,549

Tabla 39 - Cantidad de accesos a memoria externa utilizando algoritmo ECC Y SHA1

En la Tabla 39 se puede observar que, los valores mínimos de las tres funciones se obtienen para los archivos de tamaño 1 MB.

Chacha20 requiere menos accesos que los otros dos algoritmos para la encriptación y desencriptación. DES3 utiliza el máximo para la desencriptación de un archivo de video de 100 MB y aumenta considerablemente la cantidad de accesos respecto de los otros dos algoritmos (más de 10 veces).

El mínimo necesario de memoria externa en la suma de verificación se obtiene para el archivo de imagen de 1 MB encriptado con AES-256-CBC, no obstante, en general la suma de verificación requiere de recursos similares con los tres algoritmos.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	35,924,641	531,809,915	1,043,651,321	35,973,598	529,625,664	1,042,097,336	36,483,978	533,762,636	1,046,249,293
	Encriptación	33,186,528	399,420,256	777,831,918	33,229,980	397,803,021	776,123,835	33,657,687	400,767,054	779,022,444
	Suma de Verificación	3,316,141	22,346,188	41,853,616	3,320,844	22,232,217	41,767,449	3,339,940	22,369,297	42,026,834
chacha20	Desencriptación	32,084,331	346,212,314	670,030,310	32,118,672	344,457,925	668,752,853	32,447,803	346,903,806	671,711,988
	Encriptación	30,356,798	262,127,239	501,527,444	30,384,314	261,155,443	500,549,709	30,623,447	263,006,101	502,602,363
	Suma de Verificación	3,319,755	22,288,824	41,890,755	3,321,225	22,208,364	41,931,041	3,343,066	22,379,872	41,873,634
des3	Desencriptación	125,635,598	5,408,557,612	10,862,116,712	126,259,274	5,384,631,335	10,841,444,450	131,641,749	5,427,120,526	10,887,814,820
	Encriptación	96,552,449	3,997,949,014	8,025,089,676	97,017,003	3,980,238,984	8,009,777,309	100,985,446	4,011,449,350	8,044,290,018
	Suma de Verificación	3,319,230	22,360,097	41,892,007	3,323,012	22,288,026	41,801,462	3,340,160	22,389,096	42,097,579

Tabla 40 - Cantidad de accesos a memoria externa utilizando algoritmo DH y SHA1

En la Tabla 40 se puede observar que, Chacha20 requiere menos accesos que los otros dos algoritmos para la encriptación y desencriptación. AES-256-CBC requiere 1,5 veces más accesos de memoria que Chacha 20, y DES3 10 veces más que AES-256-CBC.

El mínimo necesario de memoria externa en la suma de verificación se obtiene para el archivo de imagen de 1 MB encriptado con AES-256-CBC, no obstante, en general la suma de verificación requiere de recursos similares con los tres algoritmos.

Resumen resultados métrica accesos a memoria externa:

Analizando el resto de los algoritmos de suma de verificación, tal como se mencionó al inicio de la sección, para los tres conjuntos de datos, a medida que se incrementa el tamaño del archivo, aumenta la cantidad de accesos a la memoria externa y la suma de verificación requiere menos accesos que la encriptación, seguida por la desencriptación.

Para encriptación y desencriptación la combinación óptima se puede observar en la Tabla 37 y 39 de los algoritmos ECDSA/Chacha20 y ECC/Chacha20, mientras que la peor combinación es ECDSA/DES3.

En el caso de la suma de verificación ECC y SHA-1 requieren la menor cantidad de accesos de memoria combinados con cualquiera de los tres algoritmos simétricos, ya que no son significativas las diferencias observadas. También ofrece resultados muy similares DH/SHA1.

En general, DES3 en combinación con cualquiera de los algoritmos asimétricos, es el que más accesos a la memoria externa requiere para todos los archivos y funciones.

Porcentaje de Consumo de CPU

Algoritmo de Verificación SHA3-512

Se presentan en las Tablas 41, 42, 43 y 44 los resultados obtenidos de consumo de CPU por cada una de las funciones teniendo en cuenta el algoritmo de verificación SHA3-512.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	97.48 %	95.88 %	31.80 %	98.06 %	99.80 %	17.86 %	76.26 %	99.28 %	37.28 %
	Encriptación	62.72 %	20.90 %	15.82 %	98.38 %	38.42 %	16.32 %	97.48 %	27.78 %	32.44 %
	Suma de Verificación	95.00 %	48.90 %	87.82 %	95.84 %	99.86 %	99.90 %	91.80 %	99.62 %	99.90 %
chacha20	Desencriptación	97.20 %	88.58 %	22.90 %	91.96 %	97.46 %	12.64 %	95.88 %	99.56 %	24.22 %
	Encriptación	61.54 %	4.82 %	5.18 %	59.26 %	11.36 %	6.88 %	92.52 %	16.38 %	15.66 %
	Suma de Verificación	95.64 %	82.82 %	74.46 %	95.74 %	99.86 %	99.90 %	95.58 %	84.50 %	99.78 %
des3	Desencriptación	98.50 %	99.58 %	59.42 %	98.82 %	97.90 %	36.08 %	83.20 %	86.18 %	63.54 %
	Encriptación	98.24 %	27.12 %	29.72 %	98.66 %	71.38 %	25.00 %	97.06 %	33.36 %	51.72 %
	Suma de Verificación	95.90 %	83.58 %	81.36 %	95.80 %	99.64 %	97.24 %	95.96 %	99.48 %	96.30 %

Tabla 41 - Porcentaje de uso de CPU utilizando algoritmo ECDSA y SHA3-512

Tal como se visualiza en la Tabla 41, el consumo de CPU para los casos de la desencriptación y encriptación decrece a medida que incrementa el tamaño de archivo, a excepción de casos específicos; como en la encriptación del archivo de video del algoritmo AES-256-CBC de 50 MB, en la encriptación de archivo de imagen del algoritmo chacha20 de 50 MB y en la encriptación de los archivos del algoritmo DES3 de video.

En la suma de verificación, se ven los comportamientos más críticos en cuanto al uso de recursos, con más del 90% del uso de CPU en la mayoría de los datos procesados, salvo excepciones aisladas. En donde más notorio es este detalle, es en la suma de verificación de 100 MB del archivo de imagen, donde se observa un comportamiento extremo de este uso.

En la encriptación usando los diferentes algoritmos, es donde se observan promedios considerablemente menores en comparación al resto de funciones. Siendo el caso del 15.52% para los archivos de video de 100 MB, el resultado más bajo.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	79.96 %	99.80 %	26.52 %	98.24 %	99.24 %	36.94 %	78.84 %	99.80 %	42.78 %
	Encriptación	98.48 %	58.78 %	30.70 %	74.12 %	41.42 %	30.80 %	60.54 %	62.39 %	37.82 %
	Suma de Verificación	96.52 %	99.56 %	77.48 %	96.46 %	99.88 %	99.90 %	96.34 %	69.89 %	99.90 %
chacha20	Desencriptación	97.72 %	99.70 %	22.72 %	97.64 %	99.20 %	22.68 %	93.16 %	99.72 %	25.28 %
	Encriptación	97.14 %	37.44 %	13.50 %	95.66 %	20.28 %	13.08 %	92.94 %	30.60 %	14.14 %
	Suma de Verificación	96.16 %	68.80 %	81.96 %	96.34 %	94.52 %	99.78 %	77.06 %	99.50 %	99.90 %
des3	Desencriptación	98.88 %	99.74 %	52.96 %	79.54 %	98.10 %	52.50 %	98.36 %	97.79 %	66.42 %
	Encriptación	98.86 %	77.86 %	47.30 %	96.54 %	80.32 %	44.28 %	83.96 %	85.25 %	54.14 %
	Suma de Verificación	92.40 %	50.98 %	98.88 %	96.36 %	99.84 %	99.90 %	96.28 %	85.55 %	99.52 %

Tabla 42 - Porcentaje de uso de CPU utilizando algoritmo RSA y SHA3-512

Para los datos que se plasmaron en la Tabla 42, los resultados más destacados se encuentran en las encriptaciones y desencriptaciones de los archivos de 100 MB, donde se pueden apreciar los menores porcentajes de consumo, a excepción de casos aislados.

En las encriptaciones de los datos usando el algoritmo Chacha20, se observan los valores más bajos, siendo el límite menor, el 13.08% mostrado por el archivo de texto de 100 MB.

Los comportamientos de uso más elevado de procesador se ven en las sumas de verificación, mostrando los resultados límites de 99.90% en varios casos, como en los archivos de texto y video de 100 MB.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	97.58 %	99.62 %	27.74 %	98.12 %	99.34 %	16.40 %	75.13 %	99.84 %	35.56 %
	Encriptación	97.86 %	14.68 %	23.78 %	79.20 %	67.78 %	18.06 %	97.16 %	38.12 %	30.66 %
	Suma de Verificación	81.06 %	69.24 %	99.78 %	95.84 %	99.86 %	99.90 %	62.28 %	66.74 %	99.90 %
chacha20	Desencriptación	97.18 %	98.28 %	21.02 %	97.80 %	93.88 %	11.18 %	93.16 %	99.68 %	25.28 %
	Encriptación	96.80 %	42.08 %	11.00 %	78.46 %	21.02 %	7.64 %	92.94 %	28.42 %	14.14 %
	Suma de Verificación	77.10 %	83.60 %	90.96 %	95.66 %	99.88 %	99.90 %	77.06 %	99.86 %	99.90 %
des3	Desencriptación	98.44 %	98.26 %	49.88 %	98.76 %	99.10 %	38.42 %	81.64 %	99.68 %	65.66 %
	Encriptación	62.80 %	24.56 %	46.22 %	84.36 %	75.00 %	29.66 %	98.74 %	43.08 %	47.62 %
	Suma de Verificación	94.90 %	99.74 %	74.18 %	90.94 %	99.90 %	99.90 %	95.58 %	99.74 %	99.90 %

Tabla 43 - Porcentaje de uso de CPU utilizando algoritmo Curvas Elípticas (ECC) y SHA3-512

En la Tabla 43 se ve resaltado en rosa el punto clave: la suma de verificación de los datos en la combinación ECC/SHA3-512 es lo que más estresa al procesador,

principalmente en los archivos de mayor tamaño de imagen y video. Lo contrario ocurre para los casos de encriptación y desencriptación de los datos; a mayor tamaño de archivo, menor uso del procesador. Donde se produce el resultado más notorio, destacado en color celeste, es en la encriptación del archivo de texto de 100MB con Chacha20, arrojando un uso de solo un 7.64%.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	97.64 %	85.90 %	39.72 %	98.00 %	99.42 %	15.12 %	63.14 %	95.32 %	33.60 %
	Encriptación	96.94 %	30.72 %	15.86 %	63.06 %	47.04 %	18.96 %	97.24 %	34.32 %	31.32 %
	Suma de Verificación	58.54 %	99.70 %	94.26 %	95.58 %	99.86 %	96.56 %	76.80 %	66.12 %	99.90 %
chacha20	Desencriptación	97.42 %	99.76 %	24.94 %	96.96 %	98.68 %	10.54 %	28.40 %	99.06 %	22.48 %
	Encriptación	96.80 %	6.96 %	5.48 %	96.84 %	23.90 %	8.34 %	96.26 %	10.34 %	14.24 %
	Suma de Verificación	56.74 %	85.40 %	90.98 %	90.40 %	92.72 %	93.26 %	72.76 %	99.46 %	99.90 %
des3	Desencriptación	98.64 %	99.74 %	56.02 %	98.84 %	99.48 %	33.28 %	81.22 %	99.58 %	66.84 %
	Encriptación	98.04 %	25.46 %	27.76 %	98.80 %	77.54 %	27.94 %	97.14 %	33.42 %	51.50 %
	Suma de Verificación	57.92 %	81.38 %	99.86 %	95.38 %	99.86 %	99.62 %	77.16 %	99.20 %	99.80 %

Tabla 44 - Porcentaje de uso de CPU utilizando algoritmo DH y SHA3-512

La encriptación de los datos es la función que menos recursos consume para el caso de DH/SHA3-512, como puede verse en la Tabla 44. El uso del CPU es inversamente proporcional al tamaño del archivo: a mayor tamaño, menor uso.

Caso contrario a lo mencionado anteriormente, sucede para la suma de verificación, es decir, a mayor tamaño de archivo, mayor consumo de recursos, registrándose los picos altos, en los datos de video de 100 MB.

En los archivos de 50 MB y 100 MB se ve una clara diferencia de consumo de CPU entre las funciones de encriptación y desencriptación, donde esta última utiliza más recursos del procesador. La diferencia es más notable si se utiliza el algoritmo Chacha20.

Resumen resultados métrica porcentaje de uso de CPU:

Viendo en conjunto y analizando en perspectiva las Tablas 41 a 44 y las ejecuciones realizadas con los otros algoritmos de suma de verificación (las Tablas A45 a A55 se adjuntan en el Anexo en la sección Nodo RPI4 – Resultados – Porcentaje de Uso de CPU), se puede concluir que, salvo casos puntuales, el menor porcentaje de uso de CPU corresponde a las encriptaciones y desencriptaciones de

los archivos de mayor tamaño. En el caso de las sumas de verificación si se obtienen resultados un tanto más acorde a lo esperado: a mayor tamaño de archivo, mayor uso de recursos. Este comportamiento es el mismo que se observó en los Nodos 1 y 2, y aunque no se pudieron realizar las pruebas ya que no se disponía de la RPI 4 para las pruebas, se estima que la explicación está en el aumento del tiempo en que la CPU está en estado IOWait.

Otra similitud que se observa en los resultados del Nodo 3 es en la encriptación cuando se utiliza el algoritmo Chacha20. En la mayoría de los casos, el uso de recursos de la encriptación es significativamente menor en comparación con la desenscriptación (aproximadamente 4 veces menor) y la suma de verificación (en ocasiones hasta 18 veces menor). El mínimo porcentaje de uso de CPU se obtiene con DH, mientras que el máximo ocurre con RSA.

En el caso de la desenscriptación, también ofrece un menor consumo Chacha20, siendo la mejor opción la combinación con DH, mientras que AES-256-CBC es el algoritmo que más CPU requiere.

En cuanto a la suma de verificación, los mejores resultados se obtienen con SHA1, sobre archivos encriptados con DH.

Tiempo de Ejecución

Algoritmo de Verificación SHA-512

Las Tablas 45 a 48 muestran los tiempos de ejecución que demora cada función empleando el algoritmo de suma de verificación SHA-512.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	0.08	1.88	15.05	0.08	1.87	28.94	1.58	1.90	9.94
	Encriptación	0.36	23.49	34.66	0.25	5.25	28.99	0.10	20.24	15.94
	Suma de Verificación	0.03	0.77	2.69	0.02	0.40	0.94	0.02	1.11	0.81
chacha20	Desencriptación	0.07	1.28	11.08	0.07	1.28	26.25	0.14	1.30	9.31
	Encriptación	0.22	16.40	29.32	0.19	4.75	20.74	0.06	16.96	12.24
	Suma de Verificación	0.02	1.18	0.99	0.02	0.42	0.80	0.02	0.43	0.82
des3	Desencriptación	0.14	5.32	20.18	0.14	5.57	32.55	0.47	6.07	17.45
	Encriptación	0.13	19.30	31.48	0.13	7.72	33.52	0.13	15.32	20.38
	Suma de Verificación	0.02	1.46	1.31	0.02	0.42	0.80	0.03	1.77	0.81

Tabla 45 - Tiempo de ejecución utilizando algoritmo ECDSA y SHA-512

Tal como se visualiza en la Tabla 45, el tiempo de ejecución para las funciones de desencriptación y encriptación se incrementa a medida que se incrementa el tamaño de archivo, a excepción de casos específicos; como en la encriptación de archivo de video de 50 MB con el algoritmo AES-256-CBC y Chacha20.

La suma de verificación es la función más rápida en ejecutarse y se obtienen los tiempos mínimos para los archivos de datos de 1 MB.

La encriptación demora más que la desencriptación y los mejores tiempos se obtienen utilizando el algoritmo Chacha20. Si bien el mayor tiempo se registra en la encriptación de un archivo de imagen de 100 MB, DES3 es en general el que más tiempo demanda para la ejecución de las tres funciones.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	0.13	1.94	11.47	0.08	1.86	11.32	0.35	1.89	11.14
	Encriptación	0.09	3.74	16.30	0.25	6.60	17.76	0.37	3.99	13.87
	Suma de Verificación	0.03	1.17	1.83	0.03	0.62	1.26	0.06	1.20	1.24
chacha20	Desencriptación	0.07	1.27	10.98	0.07	2.22	7.67	0.09	1.28	8.68
	Encriptación	0.06	2.34	13.48	0.07	6.91	12.55	0.06	4.51	11.20
	Suma de Verificación	0.03	1.71	0.82	0.03	0.41	0.80	0.03	0.41	0.81
des3	Desencriptación	0.14	5.50	19.12	0.14	5.91	20.49	1.57	5.50	20.78
	Encriptación	0.13	8.00	21.81	0.13	9.61	22.05	0.13	7.13	18.19
	Suma de Verificación	0.03	0.41	2.30	0.03	0.57	0.81	0.03	1.01	0.81

Tabla 46 - Tiempo de ejecución utilizando algoritmo RSA y SHA-512

Con el algoritmo RSA se observa un comportamiento similar a ECDSA, donde se observa mayor demanda de tiempo a medida que aumenta el tamaño del archivo, para las tres funciones y donde la suma de verificación es la más rápida en ejecutarse. A diferencia del algoritmo anterior, en la Tabla 46 se observa que el máximo se registra en la encriptación con DES3 de un archivo de texto de 100 MB y este algoritmo es el más lento de los tres.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	0.08	1.96	12.04	0.08	1.86	25.32	1.58	1.89	9.94
	Encriptación	0.56	16.72	21.92	0.09	7.45	28.83	0.10	9.55	15.94
	Suma de Verificación	0.02	0.41	2.53	0.02	0.40	0.80	0.02	0.40	0.81
chacha20	Desencriptación	0.07	1.29	13.26	0.07	1.28	24.23	0.09	1.28	8.68
	Encriptación	0.35	13.93	17.29	0.06	6.40	29.31	0.06	12.19	11.20
	Suma de Verificación	0.03	1.04	1.32	0.02	0.41	0.80	0.03	0.41	0.81
des3	Desencriptación	0.14	5.32	20.65	0.14	5.31	30.54	0.25	5.43	15.56
	Encriptación	0.13	23.50	32.94	0.13	8.54	35.05	0.14	12.72	21.34
	Suma de Verificación	0.02	0.52	0.80	0.02	0.41	0.80	0.02	0.41	0.83

Tabla 47 - Tiempo de ejecución utilizando algoritmo Curvas Elípticas (ECC) y SHA-512

En la Tabla 47, se puede ver que el algoritmo ECC tiene el mismo comportamiento que el algoritmo RSA con la única diferencia de que en todas las funciones demanda un poco más de tiempo de ejecución en los archivos de 50 y 100 MB.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	0.08	1.91	13.19	0.08	1.85	22.71	0.37	1.89	12.85
	Encriptación	0.09	27.47	27.33	0.10	7.35	33.54	0.10	7.68	15.47
	Suma de Verificación	0.02	0.40	0.81	0.02	0.40	0.80	0.16	0.41	0.83
chacha20	Desencriptación	0.07	1.29	9.21	0.07	1.28	17.21	0.28	1.29	9.45
	Encriptación	0.06	15.14	23.21	0.10	7.89	24.68	0.06	16.94	11.58
	Suma de Verificación	0.28	1.07	1.67	0.26	0.41	0.80	1.36	0.42	0.82
des3	Desencriptación	0.14	5.31	18.40	0.14	5.30	30.80	0.45	5.33	15.84
	Encriptación	0.29	18.52	34.22	0.13	9.63	33.60	0.14	13.03	19.41
	Suma de Verificación	0.50	0.40	1.27	0.03	0.40	0.83	0.02	0.41	0.80

Tabla 48 - Tiempo de ejecución utilizando algoritmo DH y SHA-512

En el caso del algoritmo DH, se puede observar que, en algunos casos, la suma de verificación de archivos de 1 MB demanda más tiempo de ejecución que la encriptación y la desencriptación. Para los tamaños de archivos más grandes, demora mucho menos tiempo, al igual que sucedía con los algoritmos anteriores.

La encriptación también requiere más tiempo que la desencriptación y DES3 es el algoritmo más lento. El tiempo más alto se registra en la encriptación de una imagen de 100 MB.

Resumen resultados métrica tiempo de ejecución:

En base a los resultados obtenidos en las Tablas 45 a 48 y las ejecuciones realizadas con los otros algoritmos de suma de verificación (los resultados se muestran en las tablas del Anexo en la sección Nodo RPI 4 – Resultados – Tiempo de Ejecución), se puede concluir que, salvo excepciones, el tiempo de ejecución es menor, mientras más pequeño es el archivo. Por otro lado, la suma de verificación es la función que menos tiempo requiere para ejecutarse, seguida por la desencriptación. Para esta función, SHA3-512 es el algoritmo más lento.

Otra similitud que se observa en los resultados es en la encriptación cuando se utiliza el algoritmo Chacha20. En la mayoría de los casos, el tiempo de ejecución es menor con este algoritmo, obteniéndose el mínimo con RSA mientras que el máximo ocurre con ECDSA y DES3.

Síntesis Resultados Nodo 3

Dada la falta de disponibilidad de tiempo con la que se contó con la RPI 4, no se pudieron realizar las pruebas de consumo de energía, ni se pudieron repetir las pruebas de consumo de CPU y determinar los tiempos de acceso a memoria. No obstante, se pudieron ejecutar las 5 repeticiones de todos los scripts para los distintos algoritmos y archivos.

A continuación se realiza una síntesis de los principales resultados.

- Funciones de Encriptación y Desencriptación

En porcentaje de uso de memoria no hay diferencias significativas entre los diferentes algoritmos. La encriptación es la función que más memoria consume, pero no se diferencia casi de la desencriptación. En general, tal como sucede en el Nodo 1, ECC requiere más memoria, mientras que DH es el algoritmo que menos porcentaje utiliza.

En accesos a la memoria externa, a mayor tamaño de archivos, mayor cantidad de accesos y Chacha20 resulta ser el mejor en esta métrica. Por el contrario, DES3 es el que requiere más accesos a la memoria externa.

Al igual que en el nodo 1, Chacha20 es el algoritmo que consume menos recursos de CPU y el más rápido.

- Función de Suma de Verificación

Es la función que menos memoria y accesos a la memoria externa requiere, también es la más rápida en ejecutarse. Por el contrario, es la que mayor porcentaje de CPU consume. La explicación a este resultado se estima que, al igual que en los nodos 1 y 2, se debe a que la CPU pasa menor tiempo en estado IDLE o IOWait que en las otras funciones.

En cuanto a los algoritmos, SHA 1 es el que menos memoria y CPU consume, y SHA3-512 es el más lento.

Discusión de los resultados Nodo 1, 2 y 3

De acuerdo con las pruebas llevadas a cabo durante este trabajo, se puede realizar un análisis por separado de los resultados obtenidos de las funciones de encriptación y desencriptación por un lado, y de la suma de verificación por el otro.

En el Nodo 1 (RPI 3) no se pudieron apreciar diferencias significativas en el porcentaje de uso de memoria entre los diferentes algoritmos asimétricos y simétricos, solo se observa un consumo ligeramente superior utilizando el algoritmo ECC, y un uso más conservador del recurso si se emplea DH. No obstante, en cuanto a la cantidad de accesos a la memoria externa sí se pudieron registrar diferencias si se

modifica el algoritmo simétrico. Para las funciones de encriptación y desencriptación. AES es el algoritmo que requiere menos memoria externa para archivos de mayor tamaño (50 MB y 100 MB). Para archivos pequeños, DES3 requiere menos accesos a la memoria. Respecto al porcentaje de consumo de CPU y al tiempo de ejecución de los algoritmos, se pudo determinar que Chacha20 es el algoritmo que consume menos recursos de CPU y el más rápido. Respecto de las funciones de suma de verificación, no se pueden establecer relaciones o patrones en los resultados. Los valores obtenidos son similares para todos los algoritmos, se encuentran dentro del mismo orden.

En el nodo 2 (RPI 1) solo se evaluó el comportamiento de los algoritmos asimétricos combinados con Chacha20, como algoritmo simétrico, y Sha3-512, como algoritmo de suma de verificación. Se obtuvo el mismo resultado que con el nodo 1 en cuanto al porcentaje de uso de memoria se refiere, es decir, no hay diferencias en el consumo de memoria entre los distintos algoritmos. Los accesos a memoria externa no pudieron contabilizarse en la RPI 1. En lo que se refiere al consumo de CPU y al tiempo de ejecución, no pudieron tampoco detectarse diferencias significativas en el uso de un algoritmo a otro. Solamente en el tiempo de ejecución se puede resaltar que para los archivos de 1 MB es ligeramente más rápido DH, mientras que ECDSA lo es para los de 100 MB.

En el caso del nodo 3 (RPI 4), en el porcentaje de uso de memoria no hay diferencias significativas entre los diferentes algoritmos. Al igual que en el nodo 1, la encriptación es la función que más memoria consume, pero no se diferencia casi de la desencriptación. También coinciden en que ECC requiere más memoria, mientras que DH es el algoritmo que menos porcentaje utiliza. En accesos a la memoria externa, también se observa coincidencia con el nodo 1, en que a mayor tamaño de archivo, mayor cantidad de accesos, pero en este caso Chacha20 resulta ser el mejor en esta métrica. Por el contrario, DES3 es el que requiere más accesos a la memoria externa. En cuanto a consumo de CPU y tiempo de ejecución, Chacha20 es el algoritmo que consume menos recursos y el más rápido, tal como se observó en el nodo 1. En lo que respecta a la suma de verificación, es la función que menos memoria y accesos a la memoria externa requiere, también es la más rápida en ejecutarse. Por el contrario, es la que mayor porcentaje de CPU consume. La explicación a este resultado se estima que, al igual que en los nodos 1 y 2, se debe a

que la CPU pasa menor tiempo en estado IDLE o IOWait que en las otras funciones, ya que realiza menos operaciones de lectura y escritura en la memoria externa, y por esta razón también demora menos tiempo que las otras funciones. En cuanto a los algoritmos empleados, SHA 1 es el que menos memoria y CPU consume, y SHA3-512 es el más lento.

Analizando todas las combinaciones posibles de algoritmos en los tres nodos, se puede establecer que, para los tres conjuntos de datos, ninguno de los algoritmos consume un porcentaje significativo de memoria, que pueda afectar otras tareas ejecutadas en los nodos y no se puede establecer una relación entre el tipo y tamaño de datos. Respecto de la cantidad de accesos a la memoria externa, estos se incrementan si aumenta el tamaño de los archivos, la descriptación es la función que más accesos realiza, seguida por la encriptación y por último la suma de verificación. En cuanto al porcentaje de uso de la CPU, a medida que se aumenta el tamaño del archivo, disminuye el porcentaje de uso del procesador para el caso de la encriptación y descriptación, ya que se incrementa el tiempo que la CPU se encuentra en estado IOWait, es decir, en reposo, en espera de poder hacer una operación de IO, (leer o escribir en el dispositivo de almacenamiento). En el caso de la suma de verificación no se puede establecer una relación con el tipo o tamaño de datos. Por último, se puede determinar que el tiempo de ejecución aumenta al incrementarse el tamaño de los archivos, pero no depende del tipo de datos.

Respecto al consumo de energía, solo se realizaron mediciones en el nodo 1 utilizando el algoritmo de curvas Elípticas y RSA para un archivo de video de tamaño 100 MB. Como algoritmos simétricos se seleccionaron Chacha20 y AES-256-CBC que eran más rápidos para los archivos de 100 MB de acuerdo con los resultados obtenidos anteriormente. Se puede observar que Chacha20 consume menos tanto en encriptación como en descriptación que AES-256-CBC, pero el pico de consumo en la descriptación es de 0,5 A para todas las combinaciones.

V. Conclusiones

El presente trabajo de tesis ha propuesto un análisis y comparación de algoritmos de cifrado y verificación de mensajes en el entorno de una arquitectura de Fog Computing con el fin de encontrar una combinación óptima que haga un uso eficiente de los recursos de los nodos Fog, no genere retardos en la comunicación, y no comprometa la seguridad de los datos.

Para el desarrollo de este proyecto se recurrió primeramente a la investigación bibliográfica con el objetivo de estudiar sobre los algoritmos de cifrado y verificación existentes y conocer las pruebas que se han llevado a cabo aplicadas a una arquitectura de Fog Computing. La información recolectada permitió determinar qué análisis o estudios se habían realizado y qué debilidades presentaban, como por ejemplo, se observó que en varios de ellos se realizaban las pruebas de cifrado sobre archivos de texto plano y menores a 10 MB, o que se limitaban solo a técnicas de cifrado y no agregaban autenticación o verificación de la integridad del mensaje.

Como siguiente paso, se seleccionaron los algoritmos que estos estudios consideraban como los más rápidos y eficientes en cuanto a consumo de recursos, y que según la bibliografía son los más utilizados.

Posteriormente, se simulaban nodos Fog con tres modelos de Raspberry Pi para hacer pruebas en hardware real. Para el desarrollo de las mismas, se crearon scripts que facilitaron la repetición y variación de parámetros para evaluar diferentes métricas relacionados a los recursos que son críticos en un nodo Fog, tales como consumo de memoria, CPU y consumo de energía. Otra variable que se consideró importante cuantificar, fue el tiempo que demanda agregar seguridad a la información que se está transmitiendo. Finalmente, se analizaron y compararon los resultados para determinar qué algoritmos son adecuados para cada conjunto de datos.

La metodología de trabajo propuesta anteriormente permitió cumplir con los objetivos definidos al comienzo de este documento y en base a los resultados obtenidos se ha logrado analizar qué algoritmos son más eficientes para emplear en el entorno de Fog Computing, aunque solo se pudo comprobar en forma parcial la hipótesis planteada ya que no se pudo encontrar una única combinación de algoritmos

que se diferencie claramente de las demás y que pueda ser aplicada en todas las situaciones. Respecto de las otras dos hipótesis, en las pruebas realizadas en los tres nodos se pudo determinar que no tiene influencia el tipo de datos que se procesan, lo cual permite avizorar que aplicar algoritmos de seguridad sobre plataformas de Fog Computing es factible para una amplia variedad de sistemas IoT. De la misma forma, el aumento del tamaño de los datos no influye de forma proporcional en todas las métricas, y en algunas no refleja incidencia. En relación a los recursos consumidos por las tareas de cifrado y verificación, se observa claramente que agregar seguridad a los datos compromete el uso de la CPU y demanda un tiempo que puede ser crítico en aplicaciones en tiempo real, no así en sistemas IoT donde los sensores realizan transmisiones esporádicas.

Como conclusión general de este trabajo se pueden destacar los conocimientos adquiridos durante el desarrollo del mismo sobre dos temáticas principales: Fog Computing y Criptografía. Respecto del primer tema se profundizó en los problemas que aún presenta la implementación de la arquitectura de Fog Computing. Con referencia al segundo tema, se incorporaron conocimientos sobre las ventajas y desventajas, o vulnerabilidades que presentan los diferentes algoritmos. A su vez, se desarrollaron habilidades en herramientas como OpenSSL, un completo software de criptografía, y en lenguaje Bash, que se utilizó en la creación de scripts para la ejecución de las distintas pruebas de rendimiento y la recolección de datos. Para la obtención de gráficos y tratamiento de datos en el post procesamiento se adquirieron conocimientos en Power Bi de Microsoft.

Como trabajo futuro se plantea el desarrollo de pruebas sobre una arquitectura de Fog Computing que utilice otro tipo de nodos Fog en reemplazo de las RPIs, como por ejemplo, placas Arduino. A su vez, se podría incorporar sensores IoT donde se realice adquisición y envío de datos en tiempo real. También se propone continuar la búsqueda y evaluación de algoritmos que sean más rápidos para aplicaciones críticas, tales como las relacionadas con la salud. Como aporte para futuras investigaciones sobre la problemática planteada en esta tesis de maestría, los resultados más destacados se están escribiendo en un trabajo de investigación para enviar a una revista científica internacional en inglés.

VI. Bibliografía

- [1] Atlam H.F., Walters R.J. & Wills G.B. (2020). Fog Computing and the Internet of Things: A Review Big Data and Cognitive Computing, vol. 2, no. 2, p. 10, doi:10.3390/bdcc2020010
- [2] Habibi P., Farhoudi M., Kazemian S., Khrosandi S., Leon-Garcia A. (2020). Fog Computing: A comprehensive Architectural Survey, In *IEEE Access - Open Access Journal*, 8, doi: 10.1109/ACCESS.2020.2983253.
- [3] Medina Barroso M. D. (2019). Edge Computing para IoT, Tesis de Maestría en Ingeniería Informática.
- [4] Khan, S., Parkinson, S., & Qin, Y. (2017). Fog computing security: a review of current applications and security solutions. *Journal of Cloud Computing*, 6(1), pp. 1-22.
- [5] Bouachir, O., Aloqaily, M., Tesng, L., Boukerche, A. (2020). Blockchain and Fog Computing for Cyber-Physical Systems: Case of Smart Industry. In *Computer*, 53(9), pp. 36–45. doi:10.1109/mc.2020.2996212
- [6] Sánchez, Jaime (2021). Aplicación de sistemas post-cuánticos a la seguridad en nodos de internet of things. Tesis (Maestría), In *E.T.S.I. Industriales (UPM)*.
- [7] Mell, Peter; Grance, Tim (2011). The NIST Definition of Cloud Computing in *National Institute of Standards and Technology* (Special Publication (SP),800-145). <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf>
- [8] Kim JY, Schulzrine H (2013). Cloud support for latency-sensitive telephony application. In *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference*. vol.1. IEEE. pp. 421-426
- [9] De Giusti, A. E., Naiouf, M., Medina, S., Antueno, J. D., De Giusti, L. C., Lanciotti, J., & Costanzo, M. (2021). Arquitecturas Edge-Fog-Cloud en Procesamiento Distribuido. In *XXIII Workshop de Investigadores en Ciencias de la Computación (WICC 2021, Chilecito, La Rioja)*.
- [10] Kahvazadeh S. (2019). Security architecture for Fog-To-Cloud continuum system. In *Universitat Politècnica de Catalunya Departament d' Arquitectura de Computadors*.
- [11] Perera, C., Qin, Y., Estrella, J. C., Reiff-Marganiec, S., & Vasilakos, A. V. (2017). Fog computing for sustainable smart cities: A survey. *ACM Computing Surveys (CSUR)*, 50(3), pp.1-43.
- [12] Vishwanath, A., Peruri, R., & Jing H. (2016). Security in Fog computing through encryption. Kennesaw, Georgia, USA.
- [13] Patel, V., & Bhavsar, S. Development of Security Platform in Fog Computing. *GIT-Journal of Engineering and Technology* (Thirteenth Volume, 2021, ISSN 2249 – 6157)
- [14] Montoya Ponce, R. J., (2018) Método Para La Preservación De La Privacidad En Dispositivos lot Vestibles Extendiendo La Seguridad Usando Fog Computing, *Instituto Tecnológico Metropolitano Facultad de ingenierías Medellín*, Colombia
- [15] Centellas Claros, L. S., Blanco Coca, L., & Sandoval Alcocer, J. P. (2022). Estudio comparativo de los algoritmos de criptografía simétrica AES, 3DES y ChaCha20. *Acta Nova*, 10(3), 283-302. Recuperado en 17 de noviembre de 2022, de http://www.scielo.org.bo/scielo.php?script=sci_arttext&pid=S1683-07892022000100283&lng=es&tlng=es.
- [16] Ali, A. B. R., Alagumariappan, P., Vijayalakshmi, S., Gughan, N. M., & Bhaskar, K. B. (2020, September). Analysis of Secure Transfer of Healthcare Data using Fog Computing. In *2020*

International Conference on Smart Electronics and Communication (ICOSEC), pp. 529-531. IEEE. ISBN: 978-1-7281-5461-9

[17] Froiz-Míguez, T. Fernández-Caramés, P. Fraga-Lamas, & L. Castedo (2018). Design, Implementation and Practical. Evaluation of an IoT Home Automation System for Fog Computing Applications Based on MQTT and ZigBee-WiFi Sensor Nodes, *Sensors*, vol. 18, no. 8, p. 2660, doi: 10.3390/s18082660.

[18] Barraza, A. de J. C. (2019). Modelo De Implementación De Ciberseguridad Para Sistemas IoT En El Marco De Redes 5G, *Universidad Tecnológica de Pereira, Facultad de Ingenierías Pereira*, Colombia.

[19] Krinah, A., Challal, Y., Omar, M. and Nouali, O. (2019) Using ABE for Medical Data Protection in Fog Computing. In *Proceedings of the 21st International Conference on Enterprise Information Systems (ICEIS 2019)*, pages 155-161– Science and Technology Publications, Lda. A. DOI: 10.5220/0007424601550161

[20] Boakye-Boateng, K., Kuada, E., Antwi-Boasiako, E., & Djaba, E. (2019). Encryption Protocol for Resource-Constrained Devices in Fog-Based IoT using One-Time Pads. *IEEE Internet of Things Journal*, 1–1. doi:10.1109/jiot.2019.2893172

[21] Chacon Acosta, D. S. (2021), Detección de amenazas en una red IoT usando técnicas de Machine Learning. In *Universidad de los Andes Departamento de Ingeniería de Sistemas y Computación Ingeniería de Sistemas y Computación, Bogotá, Colombia*

[22] Shreshth Tuli, Redowan Mahmud, Shikhar Tuli, and Rajkumar Buyya. Fogbus (2019), A blockchain-based lightweight framework for edge and fog computing. In *Journal of Systems and Software*.

[23] Ibrahim, M. H. (2016). OCTOPUS: An edge-fog mutual authentication scheme. *Int. J. Netw. Secur.*, 18(6), 1089-1101.

[24] Jia, X., He, D., Kumar, N., & Choo, K.-K. R. (2018). Authenticated key agreement scheme for fog-driven Mubarakali IoT healthcare system. *Wireless Networks*. doi:10.1007/s11276-018-1759-3.

[25] Leandro Loffi (2019) Protocolo Otimizado Para Autenticação Mútua Em Iot No Contexto De Fog Computing, In *Universidad Federal de Santa Catarina*, Florianópolis, Tesis de Maestría en Ciencia de la Computación.

[26] R. Morishima and H. Nishi, (2019). Network Transparent Fog-based IoT Platform for Industrial IoT. In *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, pp. 920-925, doi: 10.1109/INDIN41052.2019.8972178.

[27] Basha, A. J., Rajkumar, N., AlZain, M. A., Masud, M., & Abouhawwash, M. Fog-based Self-Sovereign Identity with RSA in Securing IoMT Data.

[28] Li, C., Qin, Z., Novak, E., & Li, Q. (2017). Securing SDN Infrastructure of IoT–Fog Networks From MitM Attacks. *IEEE Internet of Things Journal*, 4(5), 1156–1164. doi:10.1109/jiot.2017.2685596.

[29] A. Venčkauskas, N. Morkevicius, K. Bagdonas, R. Damaševičius, and R. Maskeliūnas, “A Lightweight Protocol for Secure Video Streaming,” *Sensors*, vol. 18, no. 5, p. 1554, May 2018, doi: 10.3390/s18051554.

[30] N. Abbas, M. Asim, N. Tariq, T. Baker, and S. Abbas (2019). A Mechanism for Securing IoT-enabled Applications at the Fog Layer. In *Journal of Sensor and Actuator Networks*, vol. 8, no. 1, p. 16, doi: 10.3390/jsan8010016.

[31] Hassan Noura, Ola Salman, Ali Chehab, Raphaël Couturier, (2019) Preserving data security in distributed fog computing, *Ad Hoc Networks*, Volume 94, 101937, ISSN 1570-8705, <https://doi.org/10.1016/j.adhoc.2019.101937>.(<https://www.sciencedirect.com/science/article/pii/S1570870519303634>)

- [32] Pasha, H., Scholar, P. G., & Suma, S. A, 2022. Practical Approach towards Encryption model for Fog Computing. *International Journal of Research in Engineering and Science (IJRES)* ISSN (Online): 2320-9364, ISSN (Print): 2320-9356 www.ijres.org Volume 10 Issue 5, (PP. 30-38)
- [33] Ateniese, G., Burns, R., Curtmola, R., Herring, J., Kissner, L., Peterson, Z., & Song, D. (2007, October). Provable data possession at untrusted stores. In *Proceedings of the 14th ACM conference on Computer and communications security*, pp. 598-609.
- [34] Fang, W., Zhang, W., Xiao, J. Yang, Y. & Chen, W. (2017). A Source Anonymity-Based Lightweight Secure AODV Protocol for Fog-Based MANET, *Sensors*, vol. 17, no. 6, p. 1421, doi: 10.3390/s17061421.
- [35] Mubarakali, A., Durai, A. D., Alshehri, M., AlFarraj, O., Ramakrishnan, J., & Mavaluru, D. (2020). Fog-Based Delay-Sensitive Data Transmission Algorithm for Data Forwarding and Storage in Cloud Environment for Multimedia Applications. *Big Data*. doi:10.1089/big.2020.0090
- [36] Johnney, R. K., Shelly E. & Remesh Babu, K. R. (2019). Enhanced Security through Cloud-Fog Integration, In *International Conference on Communication and Electronics Systems (ICCES)*, pp. 1530-1535, doi: 10.1109/ICCES45898.2019.9002058.
- [37] Alazeb, A., & Panda, B. (2019). Ensuring data integrity in Fog computing based health-care systems. In *International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage* (pp. 63-77). Springer, Cham.
- [38] Haouari, F., Faraj, R., & AlJa'am, J. M. (2018). Fog computing potentials, applications, and challenges. In *2018 International Conference on Computer and Applications (ICCA)* pp. 399-406. IEEE. doi:10.1109/comapp.2018.8460182

VII. Anexos

En este capítulo se incluyen dos secciones principales: Scripts y Resultados. En la primera sección se dejan los scripts en lenguaje Bash para poder analizar cómo se obtuvieron y recolectaron los datos, y así poder repetir las pruebas en otros nodos. En la segunda sección se presentan los resultados que no fueron incluidos en el capítulo Resultados para poder hacer un análisis más minucioso de los mismos.

Scripts

A continuación se deja el script utilizado para poder modificar los parámetros utilizados en cada prueba y la cantidad de repeticiones de cada proceso.

```
#!/bin/bash

source metricasXXXX

rm perf*
rm strace*
rm resultados.txt
rm tiempo.txt

echo "Archivo, Algoritmo, Datos, Tamaño, Funcion, N°_iteracion,
Algoritmo_verificacion, Tiempo, %CPU, %Memoria, t_escritura, %_escritura,
t_lectura, %_lectura, t_llamadas_sistema" >>resultados.txt

for i in {1..5};
do
    rm ./setdatos_mp4/*.mp4.*
    archivo="*.mp4"
    tipo="video"
    #algoritmo="-des3"
    #algoritmo="-aes-256-cbc"
    algoritmo="-chacha20"
    suma="ripemd160"
    time (metricas ./setdatos_mp4 $algoritmo $archivo $suma $i $tipo)
```

```

rm ./setdatos_jpg/*.jpg.*
archivo="*.jpg"
tipo="imagen"
#algoritmo="-des3"
#algoritmo="-aes-256-cbc"
algoritmo="-chacha20"
suma="ripemd160"

time (metricas ./setdatos_jpg $algoritmo $archivo $suma $i $tipo)

rm ./setdatos_jpg/*.bmp.*
archivo="*.bmp"
tipo="imagen"
#algoritmo="-des3"
#algoritmo="-aes-256-cbc"
algoritmo="-chacha20"
suma="ripemd160"

time (metricas ./setdatos_jpg $algoritmo $archivo $suma $i $tipo)

rm ./setdatos_jpg/*.png.*
archivo="*.png"
tipo="imagen"
#algoritmo="-des3"
#algoritmo="-aes-256-cbc"
algoritmo="-chacha20"
suma="ripemd160"
time (metricas ./setdatos_jpg $algoritmo $archivo $suma $i $tipo)

rm ./setdatos_csv/*.csv.*
archivo="*.csv"
tipo="texto"
#algoritmo="-des3"
#algoritmo="-aes-256-cbc"
algoritmo="-chacha20"
suma="ripemd160"
time (metricas ./setdatos_csv $algoritmo $archivo $suma $i $tipo)

```

done

A partir de este programa principal se puede definir qué algoritmo asimétrico utilizar cambiando la función “métricasXXXX” por el algoritmo deseado.

- metricasECDSA
- metricasRSA
- metricasECC
- metricasDH

A continuación se deja el código completo de métricas ECDSA y para de los otros algoritmos sólo se copian las líneas correspondientes a la generación del par de claves públicas y privadas que es lo único que se modifica dependiendo del algoritmo seleccionado.

```
#METRICAS ECDSA

#!/bin/bash

metricas () {

#Se generan las claves públicas y privadas utilizando el algoritmo asimétrico DH
openssl genpkey -genparam -algorithm DH -out dh.params.pem
openssl genpkey -aes128 -paramfile dh.params.pem -out privateDH.pem -pass
pass:MTI2023
openssl pkey -in privateDH.pem -passin pass:MTI2023 -pubout -out publicDH.pem

#Se genera una clave aleatoria para cifrar el archivo mediante algoritmo
simétrico
openssl rand -base64 -out Qkey.txt 64
echo $2

#Función Encriptación
for i in `find $1 -type f -name $3`
do
    echo "Encriptando $1/$i"
    fichero="$1/$(basename "$i")"
    size=$(du -h $fichero | awk '{print $1}')

```

```

perf_5.10 stat -o perf.txt openssl enc $2 -a -pass file:Qkey.txt -in "$i" -
out $1/$(basename "$i".encrypted) -pbkdf2 &

ps -axo pid,pmem,cmd | grep openssl | awk '{print $2}' >>mem.txt

strace -c -e trace=read,write -o strace.txt openssl enc $2 -a -pass
file:Qkey.txt -in "$i" -out $1/$(basename "$i".encrypted) -pbkdf2

perf_5.10 stat -e ext_mem_req,L1-dcache-loads,cache-misses,cache-references
-o perf2.txt openssl enc $2 -a -pass file:Qkey.txt -in "$i" -out
$1/$(basename "$i".encrypted) -pbkdf2


Tread=$(head -n4 strace.txt | awk '{print $2}' | tail -n1)
Pread=$(head -n4 strace.txt | awk '{print $1}' | tail -n1)
Twrite=$(head -n3 strace.txt | awk '{print $2}' | tail -n1)
Pwrite=$(head -n3 strace.txt | awk '{print $1}' | tail -n1)
Tsyscall=$(tail -n1 strace.txt | awk '{print $2}')
Tiempo=$(tail -n6 perf.txt | awk '{print $1}' | head -n1)
CPU=$(cut -d '#' -f2 perf.txt | awk '{print $1}' | head -n6 |tail -n1)
Mem=$(awk '{total += $1; count ++} END {print total / count}' mem.txt)
Mem_ext=$(cut -d '#' -f2 perf2.txt | awk '{print $1}' | head -n6 |tail -
n1|awk '{gsub(",",""); print $1; print $2; print $3}')


rm mem.txt

echo "$(basename "$i"), $2, $6, $size, "Encriptacion", $5, $4, $Tiempo, $CPU,
$Mem, $Mem_ext, $Twrite, $Pwrite, $Tread, $Pread, $Tsyscall"
>>resultados.txt

done

#Función Suma de Verificación

for i in `find $1 -type f -name $3`
do
    echo "Suma de Verificacion $1/$i"
    fichero="$1/$(basename "$i")"

    perf_5.10 stat -o perfchecksum.txt openssl dgst $4 -hmac "1234" $1/$(basename
"$i".encrypted) > $1/$(basename "$i".encrypted.hmac1) & ps -axo pid,pmem,cmd
| grep openssl | awk '{print $2}' >>mem_CS.txt &

    strace -c -o stracechecksum.txt openssl dgst $4 -hmac "1234" $1/$(basename
"$i".encrypted) > $1/$(basename "$i".encrypted.hmac1)

```

```

perf_5.10 stat -e ext_mem_req,L1-dcache-loads,cache-misses,cache-references
-o perf2checksum.txt openssl dgst $4 -hmac "1234" $1/$(basename
"$i".encrypted) > $1/$(basename "$i".encrypted.hmac1)

size=$(du -h $fichero | awk '{print $1}')
Tread_CS=$(head -n4 stracechecksum.txt | awk '{print $2}' | tail -n1)
Pread_CS=$(head -n4 stracechecksum.txt | awk '{print $1}' | tail -n1)
Twrite_CS=$(head -n3 stracechecksum.txt | awk '{print $2}' | tail -n1)
Pwrite_CS=$(head -n3 stracechecksum.txt | awk '{print $1}' | tail -n1)
Tsycall_CS=$(tail -n1 stracechecksum.txt | awk '{print $2}')
Tiempo_CS=$(tail -n6 perfchecksum.txt | awk '{print $1}' | head -n1)
CPU_CS=$(cut -d '#' -f2 perfchecksum.txt | awk '{print $1}' | head -n6 | tail
-n1)
Mem_CS=$(awk '{total += $1; count ++} END {print total / count}' mem_CS.txt)
Mem_ext_CS=$(cut -d '#' -f2 perf2checksum.txt | awk '{print $1}' | head -n6
|tail -n1 | awk '{gsub(",",""); print $1; print $2; print $3}')

rm mem_CS.txt

echo "$(basename "$i"), $2, $6, $size, "Suma de Verificacion", $5, $4,
$Tiempo_CS, $CPU_CS, $Mem_CS, $Mem_ext_CS, $Twrite_CS, $Pwrite_CS,
$Tread_CS, $Pread_CS, $Tsycall_CS" >>resultados.txt
done

#Función Desencriptación

for i in `find $1 -type f -name $3"*.encrypted"`
do
    echo "Desencriptando $1/$i"
    fichero="$1/$(basename "$i")"

    perf_5.10 stat -o perfdesenc.txt openssl enc $2 -d -a -pass file:Qkey.txt -
in "$i" -out $1/$(basename "$i".desenc) -pbkdf2 &
    ps -axo pid,pmem,cmd | grep openssl | awk '{print $2}' >>mem_D.txt
    strace -c -o stracedesenc.txt openssl enc $2 -d -a -pass file:Qkey.txt -in
"$i" -out $1/$(basename "$i".desenc) -pbkdf2
    perf_5.10 stat -e ext_mem_req,L1-dcache-loads,cache-misses,cache-references
-o perf2desenc.txt openssl enc $2 -a -pass file:Qkey.txt -in "$i" -out
$1/$(basename "$i".encrypted) -pbkdf2

    Tread_D=$(head -n4 stracedesenc.txt | awk '{print $2}' | tail -n1)

```

```

Pread_D=$(head -n4 stracedesenc.txt | awk '{print $1}' | tail -n1)
Twrite_D=$(head -n3 stracedesenc.txt | awk '{print $2}' | tail -n1)
Pwrite_D=$(head -n3 stracedesenc.txt | awk '{print $1}' | tail -n1)
Tsycall_D=$(tail -n1 stracedesenc.txt | awk '{print $2}')
Tiempo_D=$(tail -n6 perfdesenc.txt | awk '{print $1}' | head -n1)
CPU_D=$(cut -d '#' -f2 perfdesenc.txt | awk '{print $1}' | head -n6 | tail
n1)
Mem_D=$(awk '{total += 1; count ++} END {print total / count}' mem_D.txt)
Mem_ext_D=$(cut -d '#' -f2 perf2desenc.txt | awk '{print $1}' | head -n6
|tail -n1 | awk '{gsub(",",""); print $1; print $2; print $3}')
size=$(du -h $1/$(basename "$i".desenc) | awk '{print $1}')
rm mem_D.txt

echo "$(basename "$i"), $2, $6, $size, "Desencryption", $5, $4, $Tiempo_D,
$CPU_D, $Mem_D, $Mem_ext_D, $Twrite_D, $Pwrite_D, $Tread_D, $Pread_D,
$Tsycall_D" >>resultados.txt

done
exit
}

```

```

#METRICAS RSA
#Se generan las claves públicas y privadas utilizando el algoritmo asimétrico
RSA
openssl genrsa -out clave.pem 2048
openssl rsa -in clave.pem -out clave.pub.pem -outform PEM -pubout

```

```

#METRICAS ECC
#Se generan las claves públicas y privadas utilizando el algoritmo asimétrico de
Curvas Elípticas (Curva X25519)

openssl genpkey -algorithm X25519 -out priv1.pem

openssl pkey -in priv1.pem -pubout -out pub1.pem

openssl genpkey -algorithm X25519 -out priv2.pem

openssl pkey -in priv2.pem -pubout -out pub2.pem

openssl pkeyutl -derive -inkey priv1.pem -peerkey pub2.pem -out secreto1

```

```
#METRICAS DH

#Se generan las claves públicas y privadas utilizando el protocolo de intercambio
de claves de Diffie Hellman

openssl genpkey -genparam -algorithm DH -out dh.params.pem

openssl genpkey -aes128 -paramfile dh.params.pem -out privateDH.pem -pass
pass:MTI2023

openssl pkey -in privateDH.pem -passin pass:MTI2023 -pubout -out publicDH.pem
```

Resultados

A continuación se dejan los resultados de los nodos 1 y 3 de los algoritmos de suma de verificación que no se incluyeron en el capítulo Resultados. Se encuentran ordenados por métricas analizadas y algoritmos empleados.

Nodo 1 – Raspberry Pi 3

Porcentaje de Uso de Memoria

Algoritmo de Verificación SHA-512

En las Tablas A1 a A4 se muestra el porcentaje de uso de memoria utilizado en las pruebas realizadas en la RPI 3 con el algoritmo SHA-512 de suma de verificación.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	0.45 %	0.45 %	0.45 %	0.25 %	0.47 %	0.45 %	0.45 %	0.45 %	0.45 %
	Encriptación	0.45 %	0.40 %	0.45 %	0.45 %	0.45 %	0.45 %	0.45 %	0.45 %	0.40 %
	Suma de Verificación	0.31 %	0.33 %	0.45 %	0.39 %	0.29 %	0.27 %	0.25 %	0.28 %	0.33 %
chacha20	Desencriptación	0.45 %	0.45 %	0.45 %	0.19 %	0.45 %	0.45 %	0.40 %	0.45 %	0.45 %
	Encriptación	0.45 %	0.44 %	0.44 %	0.45 %	0.44 %	0.45 %	0.45 %	0.45 %	0.32 %
	Suma de Verificación	0.33 %	0.30 %	0.34 %	0.33 %	0.29 %	0.29 %	0.31 %	0.27 %	0.25 %
des3	Desencriptación	0.45 %	0.45 %	0.45 %	0.06 %	0.45 %	0.45 %	0.45 %	0.45 %	0.45 %
	Encriptación	0.45 %	0.33 %	0.44 %	0.45 %	0.45 %	0.45 %	0.47 %	0.45 %	0.43 %
	Suma de Verificación	0.31 %	0.31 %	0.45 %	0.29 %	0.25 %	0.34 %	0.27 %	0.27 %	0.26 %

A1 - Porcentaje de uso de memoria RAM total utilizando algoritmo ECDSA y SHA-512

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	0.45 %	0.44 %	0.45 %	0.40 %	0.45 %	0.45 %	0.45 %	0.40 %	0.45 %
	Encriptación	0.45 %	0.46 %	0.44 %	0.38 %	0.45 %	0.45 %	0.45 %	0.44 %	0.45 %
	Suma de Verificación	0.34 %	0.36 %	0.35 %	0.37 %	0.27 %	0.27 %	0.29 %	0.29 %	0.45 %
chacha20	Desencriptación	0.44 %	0.43 %	0.44 %	0.45 %	0.44 %	0.45 %	0.45 %	0.45 %	0.44 %
	Encriptación	0.45 %	0.45 %	0.45 %	0.45 %	0.47 %	0.45 %	0.47 %	0.45 %	0.45 %
	Suma de Verificación	0.31 %	0.31 %	0.34 %	0.28 %	0.27 %	0.25 %	0.29 %	0.25 %	0.30 %
des3	Desencriptación	0.45 %	0.45 %	0.45 %	0.31 %	0.45 %	0.45 %	0.45 %	0.45 %	0.45 %
	Encriptación	0.45 %	0.45 %	0.45 %	0.45 %	0.45 %	0.41 %	0.47 %	0.45 %	0.48 %
	Suma de Verificación	0.35 %	0.29 %	0.42 %	0.30 %	0.27 %	0.28 %	0.27 %	0.25 %	0.37 %

A2 - Porcentaje de uso de memoria RAM total utilizando algoritmo RSA y SHA-512

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	0.47 %	0.47 %	0.47 %	0.44 %	0.47 %	0.47 %	0.42 %	0.46 %	0.47 %
	Encriptación	0.42 %	0.42 %	0.47 %	0.43 %	0.47 %	0.47 %	0.47 %	0.47 %	0.47 %
	Suma de Verificación	0.44 %	0.43 %	0.47 %	0.43 %	0.43 %	0.45 %	0.45 %	0.45 %	0.45 %
chacha20	Desencriptación	0.47 %	0.47 %	0.47 %	0.43 %	0.47 %	0.47 %	0.38 %	0.47 %	0.47 %
	Encriptación	0.42 %	0.42 %	0.44 %	0.45 %	0.47 %	0.47 %	0.47 %	0.47 %	0.47 %
	Suma de Verificación	0.44 %	0.43 %	0.47 %	0.44 %	0.44 %	0.44 %	0.44 %	0.44 %	0.44 %
des3	Desencriptación	0.48 %	0.48 %	0.48 %	0.48 %	0.48 %	0.48 %	0.37 %	0.48 %	0.48 %
	Encriptación	0.48 %	0.43 %	0.48 %	0.48 %	0.48 %	0.48 %	0.49 %	0.49 %	0.48 %
	Suma de Verificación	0.46 %	0.46 %	0.48 %	0.44 %	0.44 %	0.46 %	0.46 %	0.46 %	0.46 %

A3 - Porcentaje de uso de memoria RAM total utilizando algoritmo ECC y SHA-512

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	0.44 %	0.45 %	0.45 %	0.34 %	0.45 %	0.45 %	0.08 %	0.39 %	0.44 %
	Encriptación	0.29 %	0.20 %	0.45 %	0.45 %	0.47 %	0.45 %	0.45 %	0.45 %	0.29 %
	Suma de Verificación	0.30 %	0.25 %	0.29 %	0.31 %	0.29 %	0.28 %	0.27 %	0.28 %	0.27 %
chacha20	Desencriptación	0.44 %	0.45 %	0.45 %	0.27 %	0.45 %	0.45 %	0.24 %	0.45 %	0.45 %
	Encriptación	0.27 %	0.46 %	0.45 %	0.45 %	0.47 %	0.45 %	0.44 %	0.44 %	0.40 %
	Suma de Verificación	0.28 %	0.22 %	0.29 %	0.27 %	0.28 %	0.29 %	0.28 %	0.30 %	0.29 %
des3	Desencriptación	0.45 %	0.45 %	0.45 %	0.44 %	0.45 %	0.45 %	0.24 %	0.45 %	0.45 %
	Encriptación	0.19 %	0.32 %	0.42 %	0.44 %	0.45 %	0.44 %	0.51 %	0.45 %	0.31 %
	Suma de Verificación	0.28 %	0.26 %	0.28 %	0.28 %	0.27 %	0.28 %	0.28 %	0.28 %	0.28 %

A4 - Porcentaje de uso de memoria RAM total utilizando algoritmo DH y SHA-512

Algoritmo de Verificación SHA-1

En las Tablas A5 a A8 se pueden observar los resultados obtenidos de las pruebas realizadas en la RPI 3 con el algoritmo SHA-1 de suma de verificación.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	0.45 %	0.45 %	0.45 %	0.21 %	0.45 %	0.45 %	0.36 %	0.45 %	0.45 %
	Encriptación	0.38 %	0.36 %	0.45 %	0.44 %	0.45 %	0.45 %	0.47 %	0.45 %	0.34 %
	Suma de Verificación	0.25 %	0.29 %	0.43 %	0.32 %	0.27 %	0.27 %	0.29 %	0.27 %	0.27 %
chacha20	Desencriptación	0.45 %	0.45 %	0.45 %	0.23 %	0.45 %	0.45 %	0.33 %	0.45 %	0.45 %
	Encriptación	0.45 %	0.35 %	0.44 %	0.45 %	0.45 %	0.45 %	0.45 %	0.45 %	0.28 %
	Suma de Verificación	0.29 %	0.36 %	0.36 %	0.28 %	0.22 %	0.29 %	0.29 %	0.29 %	0.26 %
des3	Desencriptación	0.45 %	0.45 %	0.44 %	0.16 %	0.44 %	0.45 %	0.38 %	0.45 %	0.45 %
	Encriptación	0.45 %	0.44 %	0.45 %	0.45 %	0.45 %	0.45 %	0.45 %	0.45 %	0.33 %
	Suma de Verificación	0.37 %	0.35 %	0.38 %	0.31 %	0.25 %	0.29 %	0.28 %	0.25 %	0.28 %

A5 - Porcentaje de uso de memoria RAM total utilizando algoritmo ECDSA y SHA1

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	0.45 %	0.44 %	0.45 %	0.40 %	0.43 %	0.45 %	0.38 %	0.45 %	0.45 %
	Encriptación	0.45 %	0.44 %	0.45 %	0.46 %	0.45 %	0.45 %	0.39 %	0.45 %	0.40 %
	Suma de Verificación	0.33 %	0.29 %	0.33 %	0.31 %	0.31 %	0.29 %	0.29 %	0.25 %	0.36 %
chacha20	Desencriptación	0.45 %	0.45 %	0.45 %	0.45 %	0.44 %	0.45 %	0.32 %	0.45 %	0.45 %
	Encriptación	0.45 %	0.45 %	0.44 %	0.43 %	0.43 %	0.41 %	0.41 %	0.45 %	0.38 %
	Suma de Verificación	0.29 %	0.33 %	0.36 %	0.35 %	0.29 %	0.33 %	0.27 %	0.31 %	0.35 %
des3	Desencriptación	0.45 %	0.43 %	0.44 %	0.45 %	0.45 %	0.45 %	0.45 %	0.45 %	0.45 %
	Encriptación	0.45 %	0.43 %	0.45 %	0.46 %	0.46 %	0.45 %	0.46 %	0.43 %	0.45 %
	Suma de Verificación	0.28 %	0.35 %	0.37 %	0.36 %	0.29 %	0.31 %	0.34 %	0.27 %	0.23 %

A6 - Porcentaje de uso de memoria RAM total utilizando algoritmo RSA y SHA1

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	0.45 %	0.45 %	0.45 %	0.34 %	0.45 %	0.44 %	0.24 %	0.45 %	0.45 %
	Encriptación	0.43 %	0.25 %	0.45 %	0.41 %	0.44 %	0.45 %	0.45 %	0.45 %	0.45 %
	Suma de Verificación	0.34 %	0.30 %	0.38 %	0.28 %	0.31 %	0.34 %	0.35 %	0.29 %	0.30 %
chacha20	Desencriptación	0.43 %	0.45 %	0.45 %	0.45 %	0.45 %	0.45 %	0.28 %	0.44 %	0.45 %
	Encriptación	0.30 %	0.49 %	0.45 %	0.45 %	0.45 %	0.45 %	0.45 %	0.45 %	0.45 %
	Suma de Verificación	0.36 %	0.38 %	0.39 %	0.30 %	0.33 %	0.32 %	0.32 %	0.33 %	0.27 %
des3	Desencriptación	0.45 %	0.45 %	0.45 %	0.41 %	0.45 %	0.45 %	0.32 %	0.43 %	0.45 %
	Encriptación	0.29 %	0.43 %	0.45 %	0.44 %	0.45 %	0.45 %	0.45 %	0.44 %	0.45 %
	Suma de Verificación	0.36 %	0.35 %	0.43 %	0.31 %	0.31 %	0.27 %	0.31 %	0.37 %	0.30 %

A7 - Porcentaje de uso de memoria RAM total utilizando algoritmo ECC y SHA1

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	0.43 %	0.45 %	0.45 %	0.12 %	0.48 %	0.44 %	0.30 %	0.45 %	0.41 %
	Encriptación	0.41 %	0.34 %	0.45 %	0.45 %	0.44 %	0.45 %	0.45 %	0.43 %	0.39 %
	Suma de Verificación	0.23 %	0.14 %	0.32 %	0.20 %	0.24 %	0.23 %	0.24 %	0.25 %	0.24 %
chacha20	Desencriptación	0.44 %	0.45 %	0.45 %	0.08 %	0.44 %	0.45 %	0.35 %	0.45 %	0.45 %
	Encriptación	0.45 %	0.45 %	0.45 %	0.44 %	0.45 %	0.45 %	0.47 %	0.45 %	0.45 %
	Suma de Verificación	0.23 %	0.27 %	0.39 %	0.24 %	0.24 %	0.24 %	0.27 %	0.23 %	0.26 %
des3	Desencriptación	0.45 %	0.45 %	0.45 %	0.24 %	0.45 %	0.45 %	0.15 %	0.43 %	0.45 %
	Encriptación	0.31 %	0.44 %	0.44 %	0.44 %	0.46 %	0.45 %	0.45 %	0.43 %	0.38 %
	Suma de Verificación	0.27 %	0.29 %	0.26 %	0.24 %	0.27 %	0.28 %	0.28 %	0.29 %	0.28 %

A8 - Porcentaje de uso de memoria RAM total utilizando algoritmo DH y SHA1

Algoritmo de Verificación RIPEMD-160

Las Tablas A9 a A11 muestran los resultados obtenidos en las pruebas realizadas en la RPI 3 con el algoritmo SHA1 de suma de verificación.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	0.45 %	0.45 %	0.44 %	0.45 %	0.45 %	0.45 %	0.45 %	0.44 %	0.45 %
	Encriptación	0.45 %	0.46 %	0.45 %	0.45 %	0.46 %	0.45 %	0.46 %	0.41 %	0.45 %
	Suma de Verificación	0.31 %	0.37 %	0.37 %	0.36 %	0.29 %	0.31 %	0.25 %	0.25 %	0.37 %
chacha20	Desencriptación	0.47 %	0.47 %	0.47 %	0.47 %	0.47 %	0.47 %	0.45 %	0.47 %	0.47 %
	Encriptación	0.45 %	0.47 %	0.47 %	0.47 %	0.47 %	0.47 %	0.47 %	0.47 %	0.47 %
	Suma de Verificación	0.45 %	0.46 %	0.44 %	0.44 %	0.45 %	0.45 %	0.44 %	0.45 %	0.45 %
des3	Desencriptación	0.45 %	0.45 %	0.45 %	0.45 %	0.45 %	0.43 %	0.45 %	0.45 %	0.44 %
	Encriptación	0.44 %	0.45 %	0.45 %	0.45 %	0.45 %	0.45 %	0.47 %	0.44 %	0.47 %
	Suma de Verificación	0.31 %	0.28 %	0.37 %	0.35 %	0.25 %	0.27 %	0.25 %	0.31 %	0.43 %

A9 - Porcentaje de uso de memoria RAM total utilizando algoritmo RSA y RIPEMD-160

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	0.47 %	0.46 %	0.47 %	0.39 %	0.47 %	0.47 %	0.44 %	0.47 %	0.47 %
	Encriptación	0.41 %	0.46 %	0.47 %	0.45 %	0.47 %	0.47 %	0.46 %	0.47 %	0.47 %
	Suma de Verificación	0.44 %	0.44 %	0.45 %	0.44 %	0.42 %	0.44 %	0.44 %	0.43 %	0.44 %
chacha20	Desencriptación	0.45 %	0.45 %	0.45 %	0.28 %	0.45 %	0.45 %	0.31 %	0.45 %	0.45 %
	Encriptación	0.36 %	0.43 %	0.45 %	0.40 %	0.45 %	0.43 %	0.44 %	0.44 %	0.45 %
	Suma de Verificación	0.29 %	0.30 %	0.42 %	0.24 %	0.25 %	0.27 %	0.27 %	0.25 %	0.31 %
des3	Desencriptación	0.45 %	0.44 %	0.45 %	0.35 %	0.45 %	0.44 %	0.22 %	0.45 %	0.45 %
	Encriptación	0.10 %	0.26 %	0.38 %	0.44 %	0.44 %	0.43 %	0.44 %	0.45 %	0.45 %
	Suma de Verificación	0.31 %	0.35 %	0.37 %	0.25 %	0.29 %	0.27 %	0.29 %	0.25 %	0.31 %

A10 - Porcentaje de uso de memoria RAM total utilizando algoritmo ECC y RIPEMD-160

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	0.45 %	0.44 %	0.45 %	0.19 %	0.45 %	0.45 %	0.20 %	0.45 %	0.45 %
	Encriptación	0.26 %	0.32 %	0.45 %	0.44 %	0.46 %	0.44 %	0.45 %	0.45 %	0.26 %
	Suma de Verificación	0.33 %	0.24 %	0.28 %	0.27 %	0.29 %	0.29 %	0.30 %	0.28 %	0.22 %
chacha20	Desencriptación	0.44 %	0.45 %	0.45 %	0.06 %	0.45 %	0.45 %	0.49 %	0.45 %	0.45 %
	Encriptación	0.45 %	0.37 %	0.42 %	0.44 %	0.48 %	0.43 %	0.45 %	0.45 %	0.38 %
	Suma de Verificación	0.25 %	0.32 %	0.34 %	0.28 %	0.26 %	0.24 %	0.24 %	0.25 %	0.23 %
des3	Desencriptación	0.45 %	0.45 %	0.44 %	0.17 %	0.44 %	0.45 %	0.33 %	0.45 %	0.45 %
	Encriptación	0.44 %	0.23 %	0.45 %	0.44 %	0.45 %	0.42 %	0.45 %	0.45 %	0.36 %
	Suma de Verificación	0.28 %	0.22 %	0.33 %	0.20 %	0.24 %	0.23 %	0.23 %	0.24 %	0.24 %

A11 - Porcentaje de uso de memoria RAM total utilizando algoritmo DH y RIPEMD-160

Porcentaje de Consumo de CPU

Algoritmo de Verificación SHA-512

En las Tablas A12 a A15 se puede observar el porcentaje de consumo de CPU en cada una de las pruebas realizadas en la RPI 3 con el algoritmo SHA-512 de suma de verificación.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	98.26 %	86.10 %	73.58 %	97.80 %	72.74 %	70.90 %	98.32 %	54.36 %	66.06 %
	Encriptación	63.98 %	42.86 %	30.50 %	78.06 %	53.68 %	26.16 %	75.96 %	64.92 %	26.18 %
	Suma de Verificación	93.00 %	84.02 %	99.76 %	75.16 %	99.06 %	99.80 %	92.80 %	99.60 %	85.36 %
chacha20	Desencriptación	97.80 %	99.52 %	55.86 %	97.48 %	94.52 %	62.88 %	97.90 %	49.16 %	57.66 %
	Encriptación	60.34 %	28.20 %	18.72 %	85.58 %	45.48 %	15.80 %	54.70 %	43.60 %	16.54 %
	Suma de Verificación	92.34 %	98.94 %	99.58 %	61.92 %	99.30 %	99.78 %	92.92 %	99.58 %	87.96 %
des3	Desencriptación	98.68 %	97.14 %	82.26 %	96.94 %	78.44 %	92.18 %	98.80 %	96.10 %	86.28 %
	Encriptación	97.12 %	59.54 %	57.34 %	79.22 %	84.38 %	56.54 %	55.28 %	81.74 %	52.78 %
	Suma de Verificación	87.54 %	55.64 %	99.78 %	57.20 %	99.18 %	80.00 %	92.88 %	99.18 %	99.74 %

A12 – Porcentaje de Consumo de CPU utilizando algoritmo ECDSA y SHA-512

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	98.20 %	91.06 %	75.70 %	98.08 %	93.38 %	76.76 %	98.20 %	83.46 %	61.82 %
	Encriptación	95.56 %	57.10 %	34.18 %	95.20 %	67.26 %	35.24 %	87.60 %	63.02 %	35.08 %
	Suma de Verificación	94.12 %	51.06 %	87.88 %	87.92 %	99.10 %	99.76 %	94.46 %	99.32 %	99.58 %
chacha20	Desencriptación	97.60 %	93.24 %	69.02 %	97.92 %	87.26 %	64.06 %	97.80 %	60.88 %	59.78 %
	Encriptación	89.72 %	39.26 %	20.46 %	88.42 %	44.10 %	22.46 %	68.58 %	35.48 %	22.42 %
	Suma de Verificación	94.10 %	99.42 %	99.76 %	76.02 %	99.58 %	99.74 %	94.28 %	99.58 %	86.64 %
des3	Desencriptación	97.84 %	96.72 %	80.44 %	98.68 %	92.74 %	90.20 %	98.38 %	94.60 %	84.54 %
	Encriptación	97.54 %	81.22 %	65.94 %	97.42 %	89.36 %	61.82 %	82.78 %	90.36 %	67.00 %
	Suma de Verificación	94.32 %	81.10 %	99.66 %	86.74 %	99.60 %	99.80 %	94.44 %	99.56 %	99.74 %

A13 - Porcentaje de Consumo de CPU utilizando algoritmo RSA y SHA-512

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	98.24 %	88.08 %	66.32 %	97.80 %	84.40 %	69.64 %	96.46 %	69.88 %	60.54 %
	Encriptación	93.56 %	43.36 %	31.28 %	92.52 %	61.30 %	26.60 %	40.92 %	69.86 %	29.24 %
	Suma de Verificación	92.02 %	99.58 %	87.84 %	69.20 %	99.52 %	99.78 %	92.34 %	99.60 %	99.70 %
chacha20	Desencriptación	97.88 %	88.66 %	59.30 %	97.22 %	72.92 %	52.06 %	95.82 %	76.34 %	69.16 %
	Encriptación	84.06 %	31.82 %	21.00 %	85.46 %	29.86 %	18.18 %	68.64 %	48.64 %	17.42 %
	Suma de Verificación	91.82 %	83.82 %	99.74 %	38.84 %	99.60 %	99.76 %	92.28 %	99.56 %	99.58 %
des3	Desencriptación	98.80 %	99.64 %	76.34 %	93.56 %	85.30 %	86.90 %	98.16 %	90.46 %	82.68 %
	Encriptación	97.06 %	65.88 %	60.90 %	95.22 %	82.02 %	59.72 %	41.92 %	77.82 %	59.00 %
	Suma de Verificación	92.48 %	67.30 %	99.76 %	55.36 %	99.58 %	99.78 %	92.34 %	99.56 %	86.74 %

A14 - Porcentaje de Consumo de CPU utilizando algoritmo ECC y SHA-512

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	97.80 %	93.50 %	69.36 %	96.84 %	75.64 %	73.86 %	95.72 %	82.70 %	59.52 %
	Encriptación	97.28 %	71.50 %	32.28 %	96.78 %	65.52 %	32.74 %	71.62 %	64.10 %	32.48 %
	Suma de Verificación	87.50 %	99.40 %	99.70 %	89.90 %	99.20 %	99.74 %	88.28 %	99.46 %	99.20 %
chacha20	Desencriptación	98.04 %	92.58 %	61.30 %	95.82 %	84.24 %	53.34 %	95.96 %	62.54 %	53.68 %
	Encriptación	91.28 %	38.34 %	19.02 %	83.52 %	47.06 %	19.16 %	30.30 %	41.40 %	19.40 %
	Suma de Verificación	89.86 %	99.44 %	99.72 %	88.78 %	99.08 %	99.60 %	88.82 %	99.52 %	99.46 %
des3	Desencriptación	98.80 %	97.94 %	79.68 %	97.46 %	75.56 %	78.76 %	98.68 %	84.26 %	87.92 %
	Encriptación	98.18 %	70.24 %	66.16 %	98.04 %	89.04 %	66.46 %	63.82 %	88.62 %	56.48 %
	Suma de Verificación	86.26 %	98.98 %	99.68 %	89.00 %	99.10 %	99.70 %	87.00 %	99.02 %	83.56 %

A15 - Porcentaje de Consumo de CPU utilizando algoritmo DH y SHA-512

Algoritmo de Verificación SHA1

El porcentaje de Consumo de CPU de las pruebas realizadas en la RPI 3 con el algoritmo SHA1 de suma de verificación se muestran en las Tablas A16 a A19.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	97.84 %	92.54 %	56.72 %	97.62 %	75.70 %	70.46 %	98.30 %	73.56 %	75.90 %
	Encriptación	62.34 %	39.80 %	30.58 %	96.42 %	58.42 %	28.80 %	42.84 %	66.16 %	25.22 %
	Suma de Verificación	92.52 %	81.74 %	99.58 %	70.24 %	99.52 %	99.68 %	92.34 %	99.30 %	84.32 %
chacha20	Desencriptación	97.96 %	99.48 %	57.28 %	97.52 %	60.96 %	62.90 %	97.96 %	58.92 %	61.16 %
	Encriptación	69.50 %	36.72 %	19.76 %	64.24 %	49.76 %	19.22 %	67.48 %	42.00 %	15.72 %
	Suma de Verificación	92.58 %	87.62 %	99.76 %	74.44 %	99.50 %	99.66 %	92.58 %	99.22 %	86.00 %
des3	Desencriptación	98.76 %	88.92 %	82.84 %	98.40 %	84.84 %	84.12 %	98.44 %	75.92 %	86.44 %
	Encriptación	97.92 %	50.36 %	65.84 %	86.78 %	88.14 %	60.80 %	70.62 %	80.12 %	60.14 %
	Suma de Verificación	92.36 %	81.84 %	99.56 %	92.48 %	99.48 %	99.72 %	92.22 %	99.56 %	54.62 %

A 16 - Porcentaje de Consumo de CPU utilizando algoritmo ECDSA y SHA1

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	98.26 %	89.12 %	63.64 %	97.14 %	85.86 %	72.50 %	98.26 %	73.44 %	76.44 %
	Encriptación	94.94 %	65.14 %	36.02 %	95.64 %	73.80 %	35.46 %	95.38 %	67.10 %	33.96 %
	Suma de Verificación	93.42 %	98.94 %	99.70 %	70.60 %	99.54 %	99.66 %	93.80 %	99.20 %	99.50 %
chacha20	Desencriptación	96.76 %	88.20 %	68.16 %	97.82 %	84.56 %	66.52 %	97.92 %	88.56 %	65.32 %
	Encriptación	90.34 %	47.22 %	21.72 %	88.70 %	38.58 %	22.80 %	88.50 %	36.08 %	23.90 %
	Suma de Verificación	94.06 %	99.36 %	86.30 %	62.58 %	99.48 %	99.72 %	93.94 %	99.52 %	99.52 %
des3	Desencriptación	98.70 %	96.50 %	84.36 %	96.68 %	96.60 %	86.38 %	98.90 %	90.40 %	84.54 %
	Encriptación	98.18 %	80.82 %	69.12 %	98.20 %	84.20 %	65.42 %	97.50 %	83.26 %	67.96 %
	Suma de Verificación	93.64 %	99.54 %	99.58 %	72.24 %	99.50 %	99.58 %	93.86 %	99.50 %	86.22 %

A17 - Porcentaje de Consumo de CPU utilizando algoritmo RSA y SHA1

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	98.14 %	94.60 %	65.68 %	98.06 %	76.02 %	73.44 %	97.86 %	75.64 %	75.36 %
	Encriptación	80.18 %	40.04 %	28.14 %	96.60 %	54.72 %	26.94 %	47.00 %	71.56 %	28.46 %
	Suma de Verificación	88.66 %	81.54 %	99.72 %	63.90 %	99.52 %	99.74 %	91.22 %	99.54 %	85.74 %
chacha20	Desencriptación	79.24 %	86.64 %	54.12 %	95.18 %	71.88 %	61.64 %	97.28 %	95.68 %	69.04 %
	Encriptación	84.82 %	32.08 %	17.26 %	68.58 %	45.42 %	16.32 %	69.96 %	40.44 %	16.56 %
	Suma de Verificación	91.52 %	99.30 %	95.32 %	90.74 %	99.52 %	99.76 %	91.88 %	99.50 %	99.70 %
des3	Desencriptación	98.24 %	98.10 %	89.70 %	98.80 %	73.94 %	82.40 %	98.68 %	82.88 %	79.46 %
	Encriptación	97.68 %	69.36 %	62.40 %	98.00 %	82.06 %	56.06 %	97.46 %	91.42 %	58.32 %
	Suma de Verificación	91.62 %	98.98 %	99.72 %	65.50 %	99.56 %	99.74 %	91.90 %	99.50 %	68.34 %

A18 - Porcentaje de Consumo de CPU utilizando algoritmo ECC y SHA1

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	98.26 %	94.70 %	64.98 %	96.14 %	71.24 %	72.96 %	98.22 %	81.00 %	71.06 %
	Encriptación	97.06 %	64.44 %	30.62 %	97.04 %	67.18 %	30.82 %	58.18 %	73.84 %	29.14 %
	Suma de Verificación	91.78 %	99.38 %	99.74 %	88.24 %	99.54 %	99.74 %	89.56 %	99.52 %	72.04 %
chacha20	Desencriptación	97.94 %	99.18 %	64.86 %	96.40 %	75.56 %	58.94 %	97.96 %	82.26 %	56.88 %
	Encriptación	90.36 %	44.98 %	20.46 %	89.58 %	46.64 %	20.94 %	53.66 %	47.72 %	19.96 %
	Suma de Verificación	90.46 %	98.66 %	99.72 %	71.96 %	99.56 %	99.76 %	90.16 %	99.58 %	99.60 %
des3	Desencriptación	98.28 %	99.52 %	81.56 %	98.20 %	78.40 %	82.72 %	98.36 %	88.96 %	84.84 %
	Encriptación	97.90 %	80.20 %	59.80 %	98.02 %	85.50 %	65.02 %	73.12 %	82.98 %	57.66 %
	Suma de Verificación	89.38 %	99.20 %	99.68 %	89.00 %	99.36 %	99.68 %	87.82 %	99.36 %	68.54 %

A19 - Porcentaje de Consumo de CPU utilizando algoritmo DH y SHA1

Algoritmo de Verificación RIPEMD-160

En las Tablas A20 a A22 se pueden observar los resultados obtenidos de las pruebas realizadas en la RPI 3 con el algoritmo RIPEMD-160 de suma de verificación. No se realizaron las pruebas con el algoritmo ECDSA porque no son compatibles.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	98.00 %	98.60 %	65.16 %	97.02 %	77.08 %	65.86 %	98.20 %	98.60 %	66.60 %
	Encriptación	96.34 %	64.16 %	35.30 %	96.82 %	66.00 %	36.02 %	77.60 %	71.30 %	33.88 %
	Suma de Verificación	94.44 %	83.88 %	99.80 %	62.44 %	99.20 %	99.54 %	94.58 %	99.64 %	99.54 %
chacha20	Desencriptación	98.02 %	98.36 %	50.30 %	98.04 %	66.52 %	55.48 %	93.26 %	80.84 %	58.08 %
	Encriptación	83.60 %	43.20 %	21.56 %	82.62 %	44.92 %	21.84 %	81.84 %	45.14 %	21.56 %
	Suma de Verificación	93.00 %	99.64 %	96.66 %	78.06 %	99.20 %	99.78 %	94.50 %	99.40 %	99.52 %
des3	Desencriptación	98.76 %	98.68 %	77.06 %	80.24 %	94.98 %	85.82 %	98.82 %	97.30 %	85.04 %
	Encriptación	96.24 %	82.66 %	66.98 %	97.98 %	85.44 %	63.98 %	79.18 %	85.24 %	65.00 %
	Suma de Verificación	94.36 %	83.18 %	99.70 %	86.78 %	99.28 %	99.74 %	94.68 %	99.62 %	99.70 %

A20 - Porcentaje de Consumo de CPU utilizando algoritmo RSA y RIPEMD-160

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	98.10 %	71.68 %	59.68 %	91.88 %	67.32 %	70.78 %	97.58 %	66.40 %	63.84 %
	Encriptación	93.30 %	51.36 %	28.30 %	81.06 %	68.10 %	27.20 %	77.50 %	64.50 %	29.80 %
	Suma de Verificación	91.34 %	99.48 %	99.78 %	63.54 %	99.54 %	99.64 %	93.16 %	99.64 %	99.76 %
chacha20	Desencriptación	97.92 %	89.04 %	66.32 %	97.78 %	65.14 %	65.74 %	97.26 %	84.58 %	62.78 %
	Encriptación	89.32 %	31.58 %	20.44 %	73.30 %	40.88 %	18.62 %	57.78 %	41.94 %	16.54 %
	Suma de Verificación	92.80 %	99.42 %	99.78 %	69.60 %	99.66 %	99.80 %	93.12 %	99.42 %	99.76 %
des3	Desencriptación	98.74 %	98.34 %	88.78 %	98.54 %	96.40 %	82.32 %	98.72 %	91.38 %	78.82 %
	Encriptación	97.46 %	73.26 %	66.46 %	94.16 %	80.82 %	61.74 %	50.26 %	81.62 %	63.00 %
	Suma de Verificación	91.92 %	48.62 %	99.68 %	38.22 %	99.64 %	99.78 %	93.10 %	99.64 %	87.00 %

A21 - Porcentaje de Consumo de CPU utilizando algoritmo ECC y RIPEMD-160

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	98.20 %	99.56 %	73.48 %	97.70 %	83.44 %	76.42 %	97.24 %	83.58 %	71.58 %
	Encriptación	96.56 %	58.96 %	31.62 %	96.68 %	66.54 %	33.06 %	84.86 %	67.38 %	30.94 %
	Suma de Verificación	92.42 %	99.68 %	99.86 %	93.46 %	99.78 %	99.86 %	90.50 %	99.70 %	89.00 %
chacha20	Desencriptación	97.92 %	88.76 %	63.38 %	96.50 %	78.28 %	63.52 %	97.66 %	73.72 %	56.30 %
	Encriptación	89.92 %	43.96 %	19.92 %	90.04 %	32.98 %	20.52 %	54.32 %	45.54 %	19.56 %
	Suma de Verificación	92.46 %	99.50 %	99.78 %	89.38 %	99.64 %	99.80 %	89.82 %	99.38 %	98.30 %
des3	Desencriptación	98.78 %	98.04 %	83.04 %	98.42 %	88.38 %	88.54 %	98.64 %	83.34 %	77.30 %
	Encriptación	98.14 %	76.18 %	67.00 %	97.86 %	86.12 %	65.12 %	37.74 %	85.82 %	67.24 %
	Suma de Verificación	92.46 %	67.80 %	99.74 %	90.76 %	99.66 %	99.70 %	91.74 %	99.64 %	85.40 %

A22 - Porcentaje de Consumo de CPU utilizando algoritmo DH y RIPEMD-160

Tiempo De Ejecución

Algoritmo de Verificación SHA-512

En las Tablas A23 a A26 se pueden observar los tiempos de ejecución de las pruebas realizadas en la RPI 3 con el algoritmo SHA-512 de suma de verificación.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	0.17	4.42	9.90	0.17	5.73	10.30	0.17	7.48	11.32
	Encriptación	1.42	7.26	18.20	1.17	5.58	21.28	0.99	4.65	20.88
	Suma de Verificación	0.04	1.21	1.40	0.42	0.70	1.41	0.04	0.71	2.17
chacha20	Desencriptación	0.15	2.68	9.75	0.15	2.83	8.39	0.16	6.80	9.36
	Encriptación	1.55	6.94	17.59	0.15	3.78	19.81	1.05	4.31	19.44
	Suma de Verificación	0.04	0.71	1.40	0.87	0.71	1.40	0.04	0.71	1.68
des3	Desencriptación	0.25	9.59	22.78	0.25	11.96	20.06	0.26	9.74	21.82
	Encriptación	0.23	14.31	28.99	0.83	9.92	29.41	0.50	10.42	31.83
	Suma de Verificación	0.04	2.36	1.40	2.58	0.70	2.04	0.04	0.71	1.40

A23- Tiempo de ejecución utilizando algoritmo ECDSA y SHA-512

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	0.17	4.23	9.60	0.17	4.00	9.35	0.17	4.52	11.91
	Encriptación	0.15	5.68	15.61	0.15	4.64	15.16	0.18	4.59	15.48
	Suma de Verificación	0.05	2.56	1.81	0.05	0.71	1.41	0.05	0.72	1.42
chacha20	Desencriptación	0.15	2.92	7.50	0.15	3.10	8.23	0.15	4.73	8.96
	Encriptación	0.14	4.74	15.66	0.14	4.21	14.15	0.67	4.98	14.42
	Suma de Verificación	0.05	0.73	1.42	0.07	0.71	1.43	0.05	0.72	1.94
des3	Desencriptación	0.26	9.64	23.10	0.26	10.11	20.56	0.26	9.88	22.14
	Encriptación	0.23	10.51	24.95	0.23	9.24	26.78	0.40	9.22	24.64
	Suma de Verificación	0.05	1.92	1.40	0.05	0.71	1.40	0.05	0.72	1.42

A24 - Tiempo de ejecución utilizando algoritmo RSA y SHA-512

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	0.17	4.27	11.09	0.17	4.64	10.40	0.18	6.10	12.26
	Encriptación	0.15	8.20	17.48	0.16	5.01	20.71	1.43	4.04	18.79
	Suma de Verificación	0.03	0.71	1.81	0.47	0.70	1.39	0.03	0.70	1.40
chacha20	Desencriptación	0.15	3.32	9.15	0.15	4.20	10.26	0.16	4.55	7.56
	Encriptación	0.15	7.20	15.27	0.16	6.06	17.82	0.34	3.48	18.89
	Suma de Verificación	0.03	1.16	1.39	1.07	0.70	1.40	0.03	0.70	1.40
des3	Desencriptación	0.25	9.33	24.32	0.27	11.23	21.45	0.26	10.58	22.53
	Encriptación	0.23	13.27	27.37	0.24	10.31	28.03	0.97	11.06	28.28
	Suma de Verificación	0.03	2.08	1.39	1.60	0.70	1.39	0.03	0.70	1.92

A25 - Tiempo de ejecución utilizando algoritmo ECC y SHA-512

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	0.17	4.04	10.82	0.18	5.72	9.99	0.17	5.88	12.62
	Encriptación	0.15	3.90	16.89	0.15	4.73	16.18	0.26	4.71	16.71
	Suma de Verificación	0.03	0.53	1.04	0.03	0.53	1.03	0.03	0.52	1.05
chacha20	Desencriptación	0.15	2.97	8.76	0.15	4.26	10.17	0.15	4.65	10.18
	Encriptación	0.13	4.76	16.88	0.15	3.60	16.23	1.19	4.48	16.21
	Suma de Verificación	0.03	0.52	1.04	0.03	0.53	1.04	0.03	0.53	1.04
des3	Desencriptación	0.25	9.53	23.45	0.25	12.58	23.75	0.26	12.03	21.42
	Encriptación	0.22	11.76	24.97	0.23	9.27	24.68	0.56	9.39	29.21
	Suma de Verificación	0.03	0.53	1.05	0.03	0.53	1.04	0.03	0.53	1.88

A26 - Tiempo de ejecución utilizando algoritmo DH y SHA-512

Algoritmo de Verificación SHA1

En las Tablas A27 a A30 se pueden observar los tiempos de ejecución de las pruebas realizadas en la RPI 3 con el algoritmo SHA-1 de suma de verificación.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	0.17	4.09	12.70	0.17	5.76	10.31	0.17	5.57	9.52
	Encriptación	1.67	8.29	17.96	0.15	5.22	18.87	1.71	4.41	21.36
	Suma de Verificación	0.04	1.56	1.18	0.05	0.60	1.20	0.04	0.61	1.98
chacha20	Desencriptación	0.15	2.67	9.48	0.15	4.82	8.44	0.15	5.10	9.09
	Encriptación	1.37	5.91	15.92	1.22	3.36	16.87	0.22	4.55	20.37
	Suma de Verificación	0.04	0.77	1.18	0.05	0.59	1.19	0.04	0.61	1.65
des3	Desencriptación	0.25	10.67	22.57	0.25	11.49	22.11	0.26	13.12	21.67
	Encriptación	0.23	16.49	25.03	0.29	9.40	27.26	0.52	10.60	28.05
	Suma de Verificación	0.04	1.46	1.18	0.04	0.60	1.20	0.04	0.60	3.38

A27 - Tiempo de ejecución utilizando algoritmo ECDSA y SHA1

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	0.17	4.23	11.59	0.17	4.47	10.13	0.17	5.18	9.48
	Encriptación	0.15	4.55	14.86	0.15	3.77	15.11	0.16	4.53	15.89
	Suma de Verificación	0.04	0.61	1.19	0.08	0.60	1.18	0.05	0.61	1.21
chacha20	Desencriptación	0.15	3.11	7.87	0.15	3.29	8.28	0.15	3.07	8.10
	Encriptación	0.14	3.63	14.41	0.14	4.54	13.69	0.14	5.20	13.26
	Suma de Verificación	0.04	0.61	1.66	0.90	0.61	1.19	0.05	0.61	1.19
des3	Desencriptación	0.25	9.65	22.15	0.26	9.62	21.55	0.26	10.39	22.15
	Encriptación	0.23	10.43	23.78	0.23	10.06	25.14	0.24	10.04	24.32
	Suma de Verificación	0.04	0.61	1.20	0.07	0.60	1.20	0.05	0.61	1.72

A28 - Tiempo de ejecución utilizando algoritmo RSA y SHA1

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	0.17	3.94	11.48	0.17	5.56	9.99	0.17	5.49	9.78
	Encriptación	0.34	8.05	20.21	0.15	5.73	20.11	1.13	3.94	19.51
	Suma de Verificación	0.03	1.33	1.18	0.81	0.59	1.20	0.03	0.59	1.74
chacha20	Desencriptación	0.35	3.59	10.18	0.15	4.59	8.69	0.15	2.79	7.54
	Encriptación	0.15	6.20	19.37	1.54	3.92	19.61	0.27	4.79	19.13
	Suma de Verificación	0.03	0.59	1.24	0.03	0.59	1.18	0.03	0.61	1.19
des3	Desencriptación	0.25	9.48	20.79	0.25	13.00	22.53	0.26	11.42	23.62
	Encriptación	0.22	12.70	26.78	0.23	10.27	29.70	0.24	9.10	28.67
	Suma de Verificación	0.03	0.60	1.17	1.17	0.59	1.17	0.03	0.59	2.82

A29 - Tiempo de ejecución utilizando algoritmo ECC y SHA1

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	0.17	3.90	11.34	0.17	5.74	10.07	0.17	5.29	10.33
	Encriptación	0.15	4.65	17.57	0.15	4.30	17.20	1.42	3.82	18.71
	Suma de Verificación	0.03	0.60	1.17	0.03	0.59	1.18	0.03	0.60	2.25
chacha20	Desencriptación	0.15	2.67	8.26	0.15	4.34	9.37	0.15	3.59	9.43
	Encriptación	0.14	3.78	15.53	0.14	3.68	14.63	1.30	3.57	15.84
	Suma de Verificación	0.03	0.60	1.17	0.44	0.59	1.18	0.03	0.59	1.21
des3	Desencriptación	0.25	9.38	22.96	0.25	12.24	22.56	0.26	10.68	22.19
	Encriptación	0.22	10.42	27.61	0.23	9.93	25.25	0.40	10.45	28.56
	Suma de Verificación	0.03	0.45	0.89	0.03	0.44	0.87	0.03	0.44	2.08

A30 - Tiempo de ejecución utilizando algoritmo DH y SHA1

Algoritmo de Verificación RIPEMD-160

En las Tablas A31, A32 y A33 se pueden observar los resultados obtenidos de las pruebas realizadas en la RPI 3 con el algoritmo RIPEMD-160 de suma de verificación.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	0.17	3.72	11.17	0.17	4.85	11.18	0.17	3.72	11.14
	Encriptación	0.16	4.73	15.17	0.15	4.65	14.80	0.41	4.03	16.18
	Suma de Verificación	0.05	1.41	1.62	0.08	0.82	1.63	0.05	0.83	1.65
chacha20	Desencriptación	0.15	2.71	10.46	0.15	4.21	10.13	0.17	3.47	9.36
	Encriptación	0.15	4.12	14.56	0.15	4.22	14.62	0.16	3.85	14.71
	Suma de Verificación	0.05	0.83	1.69	0.07	0.83	1.63	0.05	0.83	1.63
des3	Desencriptación	0.25	9.43	24.34	1.08	9.78	21.76	0.26	9.62	22.10
	Encriptación	0.23	10.48	24.60	0.23	9.80	25.69	1.24	9.84	25.47
	Suma de Verificación	0.05	1.56	1.63	0.06	0.82	1.63	0.05	0.83	1.65

A31 - Tiempo de ejecución utilizando algoritmo RSA y RIPEMD-160

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	0.17	5.65	12.39	0.19	5.68	10.54	0.18	6.17	11.60
	Encriptación	0.16	5.91	19.29	0.24	4.62	19.97	0.39	4.98	18.32
	Suma de Verificación	0.04	0.83	1.61	0.90	0.81	1.62	0.04	0.82	1.65
chacha20	Desencriptación	0.15	3.27	7.85	0.15	4.69	8.18	0.15	3.35	8.48
	Encriptación	0.14	6.96	15.64	0.99	4.33	17.56	0.59	4.72	19.29
	Suma de Verificación	0.04	0.81	1.61	1.05	0.81	1.61	0.04	0.82	1.62
des3	Desencriptación	0.25	9.45	20.91	0.25	9.60	22.63	0.26	10.35	23.62
	Encriptación	0.22	12.15	24.73	0.24	10.43	26.72	2.47	10.55	26.38
	Suma de Verificación	0.04	3.54	1.62	2.15	0.81	1.62	0.04	0.81	2.20

A32 - Tiempo de ejecución utilizando algoritmo ECC y RIPEMD-160

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	0.17	3.70	10.16	0.17	4.72	9.50	0.17	4.89	10.60
	Encriptación	0.15	5.78	17.18	0.15	4.58	15.88	0.20	4.43	17.51
	Suma de Verificación	0.04	1.15	2.27	0.04	1.14	2.28	0.05	1.14	2.83
chacha20	Desencriptación	0.15	3.30	8.32	0.15	3.70	8.34	0.16	3.86	9.65
	Encriptación	0.14	4.03	15.79	0.14	5.86	14.70	1.56	3.75	15.73
	Suma de Verificación	0.04	0.82	1.61	0.04	0.81	1.61	0.04	0.82	1.66
des3	Desencriptación	0.25	9.50	22.45	0.25	10.62	20.96	0.26	11.65	24.08
	Encriptación	0.23	10.97	24.58	0.23	9.77	25.25	1.54	9.90	24.76
	Suma de Verificación	0.04	2.14	1.62	0.04	0.81	1.62	0.04	0.82	2.36

A33 - Tiempo de ejecución utilizando algoritmo DH y RIPEMD-160

Nodo 3 – Raspberry PI 4

Porcentaje de Uso de Memoria

Algoritmo de Verificación SHA3-512

En las Tablas A34, A35, A36 y A37 se pueden observar los resultados obtenidos de las pruebas realizadas en la RPI 3 con el algoritmo SHA3-512 de suma de verificación.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	0.20 %	0.19 %	0.20 %	0.20 %	0.17 %	0.20 %	0.24 %	0.19 %	0.24 %
	Encriptación	0.17 %	0.18 %	0.19 %	0.19 %	0.19 %	0.20 %	0.24 %	0.19 %	0.24 %
	Suma de Verificación	0.10 %	0.10 %	0.10 %	0.10 %	0.10 %	0.10 %	0.22 %	0.11 %	0.22 %
chacha20	Desencriptación	0.19 %	0.20 %	0.19 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %
	Encriptación	0.20 %	0.20 %	0.19 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.19 %
	Suma de Verificación	0.11 %	0.11 %	0.10 %	0.10 %	0.10 %	0.10 %	0.10 %	0.12 %	0.10 %
des3	Desencriptación	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.24 %	0.20 %	0.24 %
	Encriptación	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.24 %	0.20 %	0.24 %
	Suma de Verificación	0.17 %	0.17 %	0.17 %	0.17 %	0.17 %	0.17 %	0.23 %	0.17 %	0.22 %

A34 - Porcentaje de uso de memoria RAM total utilizando algoritmo ECDSA y SHA3-512

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.19 %	0.20 %	0.20 %
	Encriptación	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %
	Suma de Verificación	0.17 %	0.17 %	0.17 %	0.17 %	0.17 %	0.17 %	0.10 %	0.14 %	0.10 %
chacha20	Desencriptación	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.23 %	0.20 %	0.23 %
	Encriptación	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.23 %	0.20 %	0.23 %
	Suma de Verificación	0.17 %	0.17 %	0.18 %	0.17 %	0.17 %	0.17 %	0.20 %	0.17 %	0.20 %
des3	Desencriptación	0.18 %	0.20 %	0.20 %	0.20 %	0.19 %	0.20 %	0.19 %	0.20 %	0.20 %
	Encriptación	0.20 %	0.20 %	0.20 %	0.19 %	0.18 %	0.19 %	0.20 %	0.19 %	0.19 %
	Suma de Verificación	0.10 %	0.10 %	0.10 %	0.10 %	0.10 %	0.10 %	0.10 %	0.10 %	0.10 %

A35 - Porcentaje de uso de memoria RAM total utilizando algoritmo RSA y SHA3-512

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	0.20 %	0.18 %	0.19 %	0.19 %	0.20 %	0.19 %	0.19 %	0.19 %	0.19 %
	Encriptación	0.20 %	0.18 %	0.23 %	0.19 %	0.20 %	0.20 %	0.19 %	0.20 %	0.19 %
	Suma de Verificación	0.10 %	0.09 %	0.11 %	0.10 %	0.09 %	0.10 %	0.09 %	0.10 %	0.10 %
chacha20	Desencriptación	0.18 %	0.18 %	0.19 %	0.19 %	0.20 %	0.19 %	0.23 %	0.18 %	0.23 %
	Encriptación	0.28 %	0.27 %	0.19 %	0.19 %	0.19 %	0.19 %	0.23 %	0.22 %	0.23 %
	Suma de Verificación	0.10 %	0.10 %	0.10 %	0.10 %	0.10 %	0.10 %	0.20 %	0.08 %	0.20 %
des3	Desencriptación	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.18 %	0.20 %	0.19 %
	Encriptación	0.19 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %
	Suma de Verificación	0.17 %	0.17 %	0.17 %	0.17 %	0.17 %	0.17 %	0.10 %	0.17 %	0.10 %

A36 - Porcentaje de uso de memoria RAM total utilizando algoritmo ECC y SHA3-512

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256cbc	Desencriptación	0.19 %	0.19 %	0.18 %	0.20 %	0.20 %	0.16 %	0.19 %	0.19 %	0.20 %
	Encriptación	0.19 %	0.20 %	0.20 %	0.20 %	0.19 %	0.20 %	0.20 %	0.20 %	0.19 %
	Suma de Verificación	0.08 %	0.11 %	0.10 %	0.10 %	0.10 %	0.10 %	0.12 %	0.10 %	0.09 %
chacha20	Desencriptación	0.20 %	0.20 %	0.19 %	0.20 %	0.18 %	0.20 %	0.20 %	0.19 %	0.20 %
	Encriptación	0.20 %	0.19 %	0.19 %	0.20 %	0.20 %	0.18 %	0.20 %	0.20 %	0.20 %
	Suma de Verificación	0.10 %	0.09 %	0.10 %	0.10 %	0.10 %	0.10 %	0.08 %	0.10 %	0.09 %
des3	Desencriptación	0.20 %	0.19 %	0.20 %	0.20 %	0.19 %	0.19 %	0.20 %	0.18 %	0.20 %
	Encriptación	0.20 %	0.20 %	0.19 %	0.18 %	0.18 %	0.20 %	0.20 %	0.20 %	0.20 %
	Suma de Verificación	0.10 %	0.13 %	0.11 %	0.10 %	0.10 %	0.10 %	0.09 %	0.10 %	0.09 %

A37 – Porcentaje de uso de memoria RAM total utilizando algoritmo DH y SHA3-512

Algoritmo de Verificación SHA1

Se presentan en las Tablas A38, A39, A40 y A41 los resultados obtenidos de consumo de memoria por cada una de las funciones teniendo en cuenta el algoritmo de verificación SHA-1.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	0.19 %	0.19 %	0.19 %	0.20 %	0.20 %	0.18 %	0.20 %	0.20 %	0.19 %
	Encriptación	0.20 %	0.21 %	0.20 %	0.19 %	0.20 %	0.19 %	0.20 %	0.20 %	0.20 %
	Suma de Verificación	0.10 %	0.10 %	0.11 %	0.10 %	0.10 %	0.10 %	0.10 %	0.10 %	0.10 %
chacha20	Desencriptación	0.19 %	0.19 %	0.20 %	0.18 %	0.20 %	0.19 %	0.19 %	0.19 %	0.20 %
	Encriptación	0.20 %	0.20 %	0.20 %	0.19 %	0.20 %	0.19 %	0.19 %	0.19 %	0.20 %
	Suma de Verificación	0.10 %	0.11 %	0.10 %	0.10 %	0.10 %	0.10 %	0.10 %	0.10 %	0.10 %
des3	Desencriptación	0.20 %	0.18 %	0.20 %	0.20 %	0.20 %	0.19 %	0.24 %	0.19 %	0.24 %
	Encriptación	0.19 %	0.19 %	0.20 %	0.20 %	0.20 %	0.17 %	0.24 %	0.19 %	0.24 %
	Suma de Verificación	0.10 %	0.10 %	0.10 %	0.10 %	0.10 %	0.10 %	0.22 %	0.10 %	0.22 %

A38 - Porcentaje de uso de memoria RAM total utilizando algoritmo ECDSA y SHA1

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.19 %	0.20 %	0.19 %
	Encriptación	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.18 %
	Suma de Verificación	0.18 %	0.17 %	0.17 %	0.17 %	0.17 %	0.17 %	0.10 %	0.14 %	0.10 %
chacha20	Desencriptación	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.23 %	0.22 %	0.23 %
	Encriptación	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.23 %	0.22 %	0.23 %
	Suma de Verificación	0.17 %	0.18 %	0.17 %	0.17 %	0.17 %	0.17 %	0.20 %	0.18 %	0.20 %
des3	Desencriptación	0.19 %	0.19 %	0.19 %	0.17 %	0.19 %	0.20 %	0.20 %	0.19 %	0.20 %
	Encriptación	0.20 %	0.19 %	0.20 %	0.19 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %
	Suma de Verificación	0.12 %	0.10 %	0.11 %	0.10 %	0.10 %	0.10 %	0.10 %	0.11 %	0.10 %

A39 - Porcentaje de uso de memoria RAM total utilizando algoritmo RSA y SHA1

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	0.16 %	0.18 %	0.19 %	0.19 %	0.20 %	0.19 %	0.20 %	0.18 %	0.20 %
	Encriptación	0.20 %	0.19 %	0.20 %	0.19 %	0.20 %	0.20 %	0.20 %	0.20 %	0.19 %
	Suma de Verificación	0.10 %	0.10 %	0.10 %	0.10 %	0.10 %	0.10 %	0.10 %	0.12 %	0.10 %
chacha20	Desencriptación	0.18 %	0.19 %	0.19 %	0.20 %	0.20 %	0.17 %	0.26 %	0.19 %	0.26 %
	Encriptación	0.24 %	0.32 %	0.20 %	0.19 %	0.19 %	0.19 %	0.26 %	0.22 %	0.27 %
	Suma de Verificación	0.10 %	0.10 %	0.10 %	0.10 %	0.10 %	0.10 %	0.23 %	0.10 %	0.23 %
des3	Desencriptación	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.17 %	0.20 %	0.20 %
	Encriptación	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %
	Suma de Verificación	0.17 %	0.17 %	0.17 %	0.17 %	0.17 %	0.17 %	0.10 %	0.17 %	0.10 %

A40 - Porcentaje de uso de memoria RAM total utilizando algoritmo ECC y SHA1

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	0.19 %	0.19 %	0.20 %	0.19 %	0.20 %	0.18 %	0.19 %	0.19 %	0.19 %
	Encriptación	0.20 %	0.19 %	0.20 %	0.19 %	0.20 %	0.18 %	0.20 %	0.18 %	0.18 %
	Suma de Verificación	0.10 %	0.10 %	0.10 %	0.10 %	0.10 %	0.10 %	0.10 %	0.10 %	0.10 %
chacha20	Desencriptación	0.20 %	0.17 %	0.20 %	0.20 %	0.20 %	0.20 %	0.19 %	0.20 %	0.20 %
	Encriptación	0.20 %	0.18 %	0.19 %	0.19 %	0.20 %	0.20 %	0.20 %	0.17 %	0.20 %
	Suma de Verificación	0.10 %	0.10 %	0.11 %	0.10 %	0.10 %	0.10 %	0.12 %	0.11 %	0.10 %
des3	Desencriptación	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.18 %	0.20 %	0.20 %
	Encriptación	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.16 %	0.20 %	0.19 %
	Suma de Verificación	0.16 %	0.16 %	0.16 %	0.16 %	0.16 %	0.16 %	0.11 %	0.16 %	0.07 %

A41 - Porcentaje de uso de memoria RAM total utilizando algoritmo DH y SHA1

Algoritmo de Verificación RIPEMD-160

En las tablas A42, A43, A44 se muestran los resultados registrados para las pruebas realizadas con el algoritmo RIPEMD-160. El mismo no es compatible con ECDSA, por esta razón no se incluye en el informe los resultados de este algoritmo asimétrico.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %
	Encriptación	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %
	Suma de Verificación	0.17 %	0.17 %	0.17 %	0.17 %	0.17 %	0.17 %	0.10 %	0.14 %	0.10 %
chacha20	Desencriptación	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.19 %	0.23 %	0.22 %	0.23 %
	Encriptación	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.23 %	0.22 %	0.23 %
	Suma de Verificación	0.17 %	0.17 %	0.17 %	0.17 %	0.17 %	0.17 %	0.20 %	0.18 %	0.20 %
des3	Desencriptación	0.20 %	0.20 %	0.20 %	0.19 %	0.20 %	0.19 %	0.19 %	0.20 %	0.20 %
	Encriptación	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.19 %	0.20 %	0.19 %
	Suma de Verificación	0.10 %	0.10 %	0.10 %	0.10 %	0.10 %	0.10 %	0.10 %	0.11 %	0.10 %

A42 - Porcentaje de uso de memoria RAM total utilizando algoritmo RSA y RIPEMD-160

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	0.20 %	0.20 %	0.20 %	0.19 %	0.19 %	0.19 %	0.20 %	0.20 %	0.19 %
	Encriptación	0.16 %	0.19 %	0.20 %	0.19 %	0.20 %	0.19 %	0.20 %	0.19 %	0.20 %
	Suma de Verificación	0.14 %	0.10 %	0.08 %	0.10 %	0.10 %	0.10 %	0.10 %	0.10 %	0.10 %
chacha20	Desencriptación	0.20 %	0.20 %	0.20 %	0.19 %	0.19 %	0.20 %	0.20 %	0.20 %	0.19 %
	Encriptación	0.24 %	0.27 %	0.20 %	0.19 %	0.20 %	0.20 %	0.20 %	0.18 %	0.19 %
	Suma de Verificación	0.08 %	0.12 %	0.12 %	0.10 %	0.10 %	0.10 %	0.08 %	0.10 %	0.10 %
des3	Desencriptación	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %
	Encriptación	0.20 %	0.20 %	0.19 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.18 %
	Suma de Verificación	0.17 %	0.18 %	0.17 %	0.17 %	0.17 %	0.17 %	0.10 %	0.17 %	0.10 %

A43 - Porcentaje de uso de memoria RAM total utilizando algoritmo ECC y RIPEMD-160

Algoritmo	Función	Imagen			Texto			video		
		1	50	100	1	50	100	1	50	100
aes-256cbc	Desencriptación	0.19 %	0.18 %	0.20 %	0.19 %	0.19 %	0.20 %	0.18 %	0.18 %	0.20 %
	Encriptación	0.20 %	0.19 %	0.18 %	0.20 %	0.20 %	0.18 %	0.20 %	0.18 %	0.20 %
	Suma de Verificación	0.10 %	0.09 %	0.08 %	0.09 %	0.10 %	0.10 %	0.10 %	0.09 %	0.10 %
chacha20	Desencriptación	0.18 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.19 %	0.19 %
	Encriptación	0.20 %	0.20 %	0.19 %	0.19 %	0.20 %	0.18 %	0.20 %	0.19 %	0.20 %
	Suma de Verificación	0.10 %	0.10 %	0.10 %	0.09 %	0.08 %	0.10 %	0.08 %	0.11 %	0.10 %
des3	Desencriptación	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.19 %	0.20 %	0.19 %
	Encriptación	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.20 %	0.19 %	0.19 %
	Suma de Verificación	0.16 %	0.16 %	0.16 %	0.16 %	0.16 %	0.16 %	0.10 %	0.16 %	0.10 %

A44 - Porcentaje de uso de memoria RAM total utilizando algoritmo DH y RIPEMD-160

Porcentaje de Uso de CPU

Algoritmo de Verificación SHA-512

Se muestran a continuación las tablas de la A45 a la A48 con los resultados obtenidos de las pruebas realizadas para evaluar el consumo de CPU utilizando el algoritmo de suma de verificación SHA-512.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	97.54 %	99.56 %	25.60 %	98.04 %	99.10 %	13.74 %	46.48 %	99.34 %	42.02 %
	Encriptación	65.76 %	12.46 %	14.42 %	80.78 %	49.90 %	17.02 %	97.36 %	20.20 %	30.58 %
	Suma de Verificación	81.26 %	81.96 %	65.38 %	92.78 %	99.60 %	90.10 %	93.34 %	81.76 %	99.78 %
chacha20	Desencriptación	96.58 %	99.66 %	23.38 %	93.84 %	98.78 %	10.94 %	77.95 %	99.60 %	29.84 %
	Encriptación	77.14 %	5.38 %	5.78 %	75.54 %	20.38 %	8.00 %	96.64 %	10.74 %	14.22 %
	Suma de Verificación	92.66 %	81.26 %	89.26 %	93.04 %	98.04 %	99.50 %	93.34 %	97.90 %	99.80 %
des3	Desencriptación	98.72 %	99.50 %	54.28 %	98.80 %	98.85 %	34.18 %	80.78 %	89.40 %	63.04 %
	Encriptación	98.30 %	26.90 %	32.74 %	98.75 %	66.63 %	29.44 %	98.34 %	38.38 %	48.26 %
	Suma de Verificación	92.92 %	81.16 %	84.54 %	93.25 %	97.45 %	99.80 %	93.68 %	80.68 %	99.50 %

A45 - Porcentaje de uso de CPU utilizando algoritmo ECSA y SHA-512

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	93.56 %	97.42 %	34.41 %	98.17 %	99.60 %	35.63 %	79.04 %	99.29 %	35.72 %
	Encriptación	97.64 %	68.18 %	30.38 %	92.25 %	43.65 %	27.95 %	84.86 %	69.22 %	35.83 %
	Suma de Verificación	95.13 %	86.17 %	83.82 %	95.29 %	99.72 %	97.97 %	90.10 %	85.52 %	99.83 %
chacha20	Desencriptación	97.76 %	99.70 %	23.30 %	97.86 %	78.36 %	35.58 %	96.80 %	99.52 %	32.10 %
	Encriptación	97.74 %	42.92 %	13.38 %	92.36 %	13.98 %	13.18 %	96.50 %	35.68 %	15.42 %
	Suma de Verificación	94.64 %	80.74 %	99.18 %	94.60 %	99.60 %	99.80 %	94.12 %	99.64 %	99.80 %
des3	Desencriptación	98.88 %	99.78 %	58.96 %	98.74 %	92.06 %	52.06 %	79.24 %	97.16 %	52.10 %
	Encriptación	98.64 %	62.80 %	46.14 %	97.90 %	52.16 %	44.60 %	98.76 %	76.73 %	54.10 %
	Suma de Verificación	94.02 %	99.26 %	68.74 %	94.28 %	84.64 %	99.80 %	94.40 %	90.11 %	99.80 %

A46 - Porcentaje de uso de CPU utilizando algoritmo RSA y SHA-512

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	97.06 %	96.12 %	32.82 %	98.12 %	99.40 %	15.24 %	46.48 %	99.36 %	42.02 %
	Encriptación	62.82 %	17.32 %	24.34 %	98.38 %	36.78 %	16.82 %	97.36 %	39.08 %	30.58 %
	Suma de Verificación	92.04 %	99.46 %	66.64 %	92.58 %	99.64 %	99.78 %	93.34 %	99.58 %	99.78 %
chacha20	Desencriptación	97.44 %	99.52 %	21.14 %	97.56 %	99.18 %	10.86 %	96.80 %	99.74 %	32.10 %
	Encriptación	77.36 %	11.64 %	10.84 %	97.46 %	15.28 %	5.86 %	96.50 %	9.52 %	15.42 %
	Suma de Verificación	87.64 %	81.82 %	84.62 %	92.40 %	99.62 %	99.80 %	94.12 %	98.76 %	99.80 %
des3	Desencriptación	98.55 %	99.66 %	53.96 %	96.64 %	99.22 %	35.18 %	82.98 %	98.04 %	69.76 %
	Encriptación	98.54 %	21.76 %	32.80 %	98.72 %	62.80 %	28.34 %	93.98 %	45.10 %	47.04 %
	Suma de Verificación	92.68 %	88.18 %	99.58 %	92.40 %	99.60 %	99.76 %	92.24 %	99.68 %	99.80 %

A47 - Porcentaje de uso de CPU utilizando algoritmo ECC y SHA-512

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	97.92 %	97.84 %	33.24 %	98.22 %	99.82 %	16.58 %	63.48 %	99.74 %	29.48 %
	Encriptación	97.50 %	9.58 %	18.42 %	98.40 %	59.08 %	15.46 %	97.30 %	48.50 %	31.68 %
	Suma de Verificación	91.78 %	99.38 %	99.06 %	92.40 %	99.66 %	99.80 %	74.46 %	99.44 %	99.80 %
chacha20	Desencriptación	97.52 %	99.62 %	31.54 %	94.58 %	99.14 %	15.68 %	63.10 %	99.76 %	29.38 %
	Encriptación	97.14 %	6.52 %	7.54 %	82.32 %	19.06 %	7.22 %	93.66 %	14.22 %	14.60 %
	Suma de Verificación	61.80 %	81.78 %	82.66 %	58.14 %	99.60 %	99.78 %	55.70 %	99.58 %	99.48 %
des3	Desencriptación	98.12 %	99.70 %	59.34 %	97.94 %	99.52 %	35.74 %	65.64 %	99.76 %	68.44 %
	Encriptación	81.18 %	28.22 %	29.64 %	98.48 %	58.66 %	29.64 %	95.38 %	48.46 %	51.02 %
	Suma de Verificación	50.46 %	99.62 %	84.62 %	78.14 %	99.58 %	99.80 %	91.52 %	99.58 %	99.68 %

A48 - Porcentaje de uso de CPU utilizando algoritmo DH y SHA-512

Algoritmo de Verificación SHA1

Los resultados de las ejecuciones realizadas en el nodo RPI 4 sobre la métrica de consumo de CPU utilizando algoritmo SHA1 se muestran en las tablas A49 a A52.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	98.28 %	99.02 %	28.52 %	98.06 %	99.48 %	15.66 %	64.66 %	91.52 %	40.12 %
	Encriptación	97.88 %	16.66 %	15.40 %	80.66 %	50.70 %	19.34 %	97.98 %	16.38 %	33.30 %
	Suma de Verificación	91.66 %	93.10 %	99.34 %	93.36 %	99.66 %	88.66 %	93.26 %	81.18 %	99.80 %
chacha20	Desencriptación	96.34 %	99.78 %	20.38 %	97.76 %	85.76 %	13.78 %	97.44 %	91.84 %	20.88 %
	Encriptación	78.26 %	4.40 %	5.66 %	60.92 %	36.16 %	7.60 %	95.28 %	6.92 %	13.78 %
	Suma de Verificación	92.78 %	99.60 %	85.52 %	93.28 %	99.60 %	99.80 %	92.40 %	81.44 %	99.52 %
des3	Desencriptación	98.54 %	99.26 %	50.90 %	98.08 %	95.68 %	32.14 %	51.70 %	99.58 %	55.92 %
	Encriptación	98.48 %	26.76 %	25.56 %	82.02 %	83.08 %	27.78 %	98.14 %	36.56 %	48.76 %
	Suma de Verificación	92.06 %	99.46 %	99.74 %	93.24 %	99.62 %	99.80 %	88.22 %	80.94 %	94.76 %

A49 - Porcentaje de uso de CPU utilizando algoritmo ECDSA y SHA1

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	98.10 %	98.98 %	33.52 %	98.26 %	99.82 %	36.20 %	77.00 %	98.24 %	34.76 %
	Encriptación	98.40 %	68.44 %	34.50 %	98.32 %	47.92 %	29.36 %	83.10 %	67.47 %	35.36 %
	Suma de Verificación	94.60 %	81.72 %	84.48 %	94.78 %	99.66 %	92.36 %	94.36 %	91.00 %	99.78 %
chacha20	Desencriptación	97.88 %	99.72 %	35.36 %	78.46 %	89.94 %	31.50 %	97.80 %	99.74 %	21.82 %
	Encriptación	97.28 %	46.34 %	12.82 %	97.42 %	15.76 %	13.60 %	97.20 %	39.30 %	15.46 %
	Suma de Verificación	94.30 %	99.36 %	64.16 %	94.68 %	99.62 %	99.80 %	94.74 %	99.23 %	99.50 %
des3	Desencriptación	98.70 %	99.72 %	59.10 %	98.86 %	92.64 %	56.46 %	80.18 %	98.16 %	56.28 %
	Encriptación	98.64 %	63.06 %	40.14 %	98.56 %	85.36 %	37.44 %	98.84 %	66.80 %	53.90 %
	Suma de Verificación	94.84 %	99.64 %	99.72 %	94.48 %	99.64 %	99.52 %	94.72 %	99.47 %	98.56 %

A50 - Porcentaje de uso de CPU utilizando algoritmo RSA y SHA1

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	96.73 %	99.52 %	26.80 %	97.70 %	91.26 %	13.36 %	80.88 %	99.02 %	40.08 %
	Encriptación	61.44 %	21.60 %	21.24 %	98.14 %	31.26 %	17.16 %	95.70 %	31.76 %	29.08 %
	Suma de Verificación	89.86 %	81.52 %	98.98 %	92.72 %	99.70 %	99.80 %	90.84 %	98.58 %	99.80 %
des3	Desencriptación	98.54 %	96.72 %	53.88 %	98.26 %	99.42 %	45.54 %	74.95 %	99.54 %	60.10 %
	Encriptación	88.84 %	22.58 %	44.84 %	98.96 %	79.54 %	30.66 %	98.50 %	50.10 %	45.16 %
	Suma de Verificación	92.88 %	80.98 %	99.72 %	92.04 %	99.70 %	99.80 %	92.96 %	99.62 %	99.80 %
chacha20	Desencriptación	97.44 %	99.54 %	26.80 %	94.76 %	83.60 %	11.58 %	41.64 %	92.26 %	22.56 %
	Encriptación	96.14 %	24.26 %	8.70 %	79.44 %	22.74 %	7.20 %	97.26 %	34.66 %	15.54 %
	Suma de Verificación	72.70 %	99.52 %	84.50 %	92.54 %	99.70 %	99.80 %	92.52 %	99.40 %	99.80 %

A51- Porcentaje de uso de CPU utilizando algoritmo ECC y SHA1

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	97.72 %	99.52 %	36.14 %	80.54 %	91.78 %	17.62 %	77.72 %	99.68 %	31.66 %
	Encriptación	98.06 %	15.00 %	16.32 %	96.48 %	48.84 %	20.72 %	94.02 %	23.86 %	34.86 %
	Suma de Verificación	74.04 %	99.60 %	99.28 %	87.64 %	84.06 %	84.58 %	55.78 %	99.66 %	99.80 %
chacha20	Desencriptación	95.24 %	93.36 %	27.26 %	97.72 %	99.76 %	11.42 %	42.76 %	99.70 %	35.96 %
	Encriptación	97.42 %	6.50 %	6.26 %	96.26 %	22.88 %	7.24 %	94.18 %	16.50 %	15.16 %
	Suma de Verificación	38.46 %	80.36 %	93.86 %	58.16 %	99.64 %	99.80 %	54.52 %	99.30 %	99.80 %
des3	Desencriptación	98.68 %	98.66 %	55.52 %	97.78 %	99.56 %	32.68 %	78.85 %	99.78 %	67.22 %
	Encriptación	98.08 %	29.02 %	26.16 %	98.72 %	63.18 %	29.34 %	97.16 %	40.80 %	48.20 %
	Suma de Verificación	39.10 %	79.94 %	99.50 %	73.46 %	99.62 %	99.80 %	73.70 %	99.62 %	99.80 %

A52 - Porcentaje de uso de CPU utilizando algoritmo DH y SHA1

Algoritmo de Verificación RIPEMD-160

En las tablas A53 a A56 se puede observar el consumo de CPU del algoritmo RIPEMD-160 en una RPI 3.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	97.94 %	99.66 %	38.08 %	98.02 %	99.84 %	29.46 %	96.84 %	99.56 %	35.84 %
	Encriptación	95.22 %	68.54 %	29.36 %	98.38 %	42.56 %	26.56 %	98.48 %	74.59 %	38.44 %
	Suma de Verificación	94.86 %	82.32 %	86.74 %	95.36 %	99.70 %	99.80 %	95.24 %	88.61 %	99.84 %
chacha20	Desencriptación	97.64 %	99.52 %	24.82 %	97.96 %	97.58 %	28.20 %	95.20 %	99.66 %	22.92 %
	Encriptación	97.52 %	30.16 %	11.64 %	97.16 %	16.18 %	12.62 %	93.80 %	42.90 %	14.04 %
	Suma de Verificación	95.08 %	82.94 %	68.98 %	95.00 %	99.72 %	99.66 %	95.34 %	91.32 %	99.84 %
des3	Desencriptación	95.32 %	99.42 %	56.46 %	98.66 %	98.78 %	54.96 %	98.62 %	92.48 %	57.70 %
	Encriptación	98.76 %	78.26 %	42.86 %	81.70 %	82.46 %	37.32 %	98.14 %	86.47 %	50.42 %
	Suma de Verificación	94.78 %	82.88 %	99.80 %	95.12 %	99.70 %	90.48 %	95.18 %	89.78 %	99.84 %

A53 - Porcentaje de uso de CPU utilizando algoritmo RSA y RIPEMD160

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	97.02 %	99.56 %	25.04 %	97.74 %	99.34 %	16.40 %	63.98 %	99.38 %	33.82 %
	Encriptación	75.93 %	13.22 %	21.96 %	80.20 %	58.34 %	16.84 %	98.28 %	30.80 %	29.54 %
	Suma de Verificación	92.98 %	65.70 %	75.46 %	93.40 %	99.74 %	99.68 %	91.44 %	99.42 %	99.88 %
chacha20	Desencriptación	96.90 %	94.52 %	19.28 %	96.06 %	83.16 %	10.36 %	40.50 %	99.26 %	26.36 %
	Encriptación	97.42 %	5.90 %	6.88 %	93.50 %	12.52 %	6.04 %	97.28 %	27.26 %	13.52 %
	Suma de Verificación	93.56 %	99.74 %	33.92 %	93.76 %	99.70 %	99.68 %	57.00 %	99.20 %	99.86 %
des3	Desencriptación	98.58 %	98.08 %	60.46 %	98.52 %	93.66 %	38.10 %	62.10 %	99.70 %	67.68 %
	Encriptación	48.66 %	27.34 %	42.22 %	81.96 %	61.62 %	30.44 %	96.24 %	54.04 %	50.20 %
	Suma de Verificación	93.68 %	99.66 %	96.00 %	93.58 %	99.72 %	99.88 %	93.12 %	99.14 %	99.88 %

A54 - Porcentaje de uso de CPU utilizando algoritmo ECC y RIPEMD160

Algoritmo	Función	Imagen			Texto			video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	98.04 %	99.64 %	43.96 %	97.96 %	99.82 %	16.46 %	76.72 %	99.86 %	35.56 %
	Encriptación	92.16 %	24.30 %	16.46 %	80.08 %	33.40 %	18.80 %	95.04 %	26.90 %	35.30 %
	Suma de Verificación	68.28 %	97.28 %	99.80 %	93.54 %	80.76 %	77.88 %	93.48 %	99.72 %	99.88 %
chacha20	Desencriptación	97.70 %	90.28 %	24.40 %	95.80 %	99.40 %	9.56 %	58.08 %	98.90 %	23.82 %
	Encriptación	96.28 %	4.24 %	5.84 %	96.36 %	42.46 %	7.90 %	88.56 %	15.96 %	16.22 %
	Suma de Verificación	44.14 %	99.72 %	95.18 %	75.32 %	93.44 %	80.34 %	67.54 %	56.22 %	99.88 %
des3	Desencriptación	98.30 %	99.34 %	49.88 %	97.86 %	99.36 %	35.92 %	96.10 %	92.06 %	62.48 %
	Encriptación	98.08 %	21.98 %	31.20 %	98.44 %	76.14 %	30.80 %	98.24 %	42.28 %	51.38 %
	Suma de Verificación	58.22 %	81.36 %	89.38 %	85.46 %	99.70 %	99.90 %	74.86 %	99.44 %	99.84 %

A55 - Porcentaje de uso de CPU utilizando algoritmo DH y RIPEMD160

Uso de Memoria Externa

Algoritmo de Verificación SHA3-512

Se presentan en las tablas A56 a A59 la cantidad de accesos memoria a memoria externa por cada una de las funciones teniendo en cuenta el algoritmo de verificación SHA3-512.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	35,922,083.60	531,880,534.80	1,043,765,770.20	35,972,194.20	529,749,985.00	1,041,697,387.80	36,478,369.40	533,574,251.00	1,046,475,315.00
	Encriptación	33,185,102.00	399,370,604.20	777,391,011.80	33,235,013.60	397,761,411.40	776,164,832.60	33,609,239.20	400,751,750.20	779,840,647.00
	Suma de Verificación	7,544,430.60	165,170,972.20	328,079,638.00	7,562,812.00	164,518,384.60	327,313,041.80	7,726,708.20	165,783,208.20	328,667,362.00
chacha20	Desencriptación	32,092,833.00	346,012,165.00	669,883,505.60	32,119,415.40	344,472,867.60	668,746,074.40	32,439,024.60	346,939,312.00	671,710,966.40
	Encriptación	30,357,402.00	262,107,898.20	501,389,899.60	30,389,865.80	261,275,544.80	500,472,909.00	30,621,942.80	263,010,749.60	502,531,967.80
	Suma de Verificación	7,544,324.00	165,189,910.40	328,013,558.60	7,565,949.80	164,491,610.20	327,377,435.80	7,724,766.60	165,752,557.00	328,713,739.20
des3	Desencriptación	125,628,775.60	5,408,488,300.00	10,862,313,441.60	126,314,880.40	5,384,757,419.80	10,841,580,417.60	131,644,051.80	5,427,078,535.20	10,887,939,015.40
	Encriptación	96,546,862.20	3,997,874,774.40	8,025,229,792.20	97,017,192.80	3,980,167,163.20	8,009,722,074.60	100,990,968.20	4,011,513,366.00	8,044,198,794.80
	Suma de Verificación	7,545,594.00	165,192,068.00	327,929,177.40	7,563,735.00	164,520,954.40	327,380,371.40	7,726,012.00	165,768,170.80	328,794,382.40

A56 - Cantidad de accesos a memoria externa utilizando algoritmo ECDSA y SHA3-512

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	35,919,879	532,032,083	1,043,972,337	35,977,306	529,821,218	1,041,792,816	36,481,958	533,624,028	1,046,363,797
	Encriptación	33,187,190	399,413,748	777,628,222	33,231,673	397,734,899	776,088,796	33,604,915	400,770,061	779,374,548
	Suma de Verificación	6,262,022	164,180,807	327,287,162	6,277,089	163,491,005	326,575,254	6,441,709	164,700,871	327,998,048
chacha20	Desencriptación	32,086,375	345,911,455	670,030,434	32,117,484	344,420,939	668,857,305	35,562,868	346,969,128	663,659,764
	Encriptación	30,355,368	262,140,510	501,432,334	30,383,197	261,108,209	500,250,339	30,620,147	262,982,474	502,577,357
	Suma de Verificación	6,262,695	164,210,959	327,202,033	6,278,057	163,515,902	326,713,489	9,059,067	164,792,323	330,243,905
des3	Desencriptación	125,633,727	5,408,602,016	10,862,188,981	126,258,831	5,384,609,679	10,841,367,820	131,642,314	5,427,047,135	10,888,312,558
	Encriptación	96,552,596	3,997,786,475	8,042,761,991	97,013,694	3,980,183,912	8,009,731,462	100,992,837	4,011,457,834	8,044,219,004
	Suma de Verificación	6,258,207	164,127,046	326,386,593	6,279,609	163,469,318	326,623,421	6,440,974	164,746,661	328,065,983

A57 - Cantidad de accesos a memoria externa utilizando algoritmo RSA y SHA3-512

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	36,303,609	552,063,965	1,084,434,560	36,369,996	549,801,256	1,082,350,953	36,897,453	553,880,982	1,086,968,751
	Encriptación	33,187,755	399,557,157	777,463,341	33,233,594	397,748,296	775,933,780	33,610,425	400,716,792	779,366,220
	Suma de Verificación	8,874,377	166,565,350	329,274,387	8,891,087	165,819,975	328,692,979	9,039,911	167,105,489	330,078,859
chacha20	Desencriptación	35,044,059	490,767,598	961,338,910	35,096,952	488,626,923	959,580,044	35,562,868	492,330,026	963,659,764
	Encriptación	30,354,489	262,161,555	501,233,308	30,385,703	261,069,109	500,632,130	30,620,147	262,843,545	502,577,357
	Suma de Verificación	8,863,713	166,592,573	329,201,338	8,899,651	165,855,754	328,656,861	9,059,067	167,067,922	330,243,905
des3	Desencriptación	101,485,165	4,238,549,592	8,509,288,333	101,982,232	4,219,968,985	8,492,536,001	106,188,300	4,252,982,398	8,529,299,406
	Encriptación	96,552,049	3,997,819,996	8,025,393,990	97,014,947	3,980,181,921	8,009,695,766	100,990,498	4,011,484,150	8,044,040,471
	Suma de Verificación	8,869,826	166,535,674	329,299,791	8,893,462	165,867,895	328,604,577	9,062,564	167,104,733	330,054,889

A58 - Cantidad de accesos a memoria externa utilizando algoritmo ECC y SHA3-512

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256cbc	Desencriptación	35,916,164	532,023,229	1,043,793,640	35,977,702	529,697,438	1,042,037,673	36,481,845	533,665,765	1,046,520,114
	Encriptación	33,189,025	399,517,840	777,598,036	33,244,323	397,829,504	776,023,282	33,605,662	400,722,195	779,404,049
	Suma de Verificación	6,262,116	164,158,942	327,201,385	6,279,308	163,485,142	326,606,092	6,440,180	164,733,403	327,985,097
chacha20	Desencriptación	32,082,341	346,095,129	669,644,224	32,127,221	344,498,928	668,711,707	32,447,983	346,926,727	671,464,994
	Encriptación	30,359,151	261,999,056	501,111,543	30,387,181	260,949,499	500,335,038	30,617,952	263,006,246	502,567,707
	Suma de Verificación	6,262,184	164,184,007	327,219,758	6,279,543	163,459,066	326,589,540	6,440,716	164,754,137	327,973,107
des3	Desencriptación	125,629,762	5,408,631,901	10,862,334,988	126,265,321	5,384,682,771	10,841,429,556	131,642,800	5,427,134,669	10,887,672,999
	Encriptación	96,549,827	3,997,787,612	8,025,107,770	97,014,730	3,980,177,223	8,009,676,953	100,988,817	4,011,455,989	8,044,287,915
	Suma de Verificación	6,263,817	164,224,780	327,167,756	6,281,063	163,527,899	326,580,169	6,441,859	164,711,443	328,047,064

A59 - Cantidad de accesos a memoria externa utilizando algoritmo DH y SHA3-512

Algoritmo de Verificación SHA-512

En las tablas A60 a A63 se muestra la cantidad de accesos a memoria externa de la RPI3 por cada una de las funciones cuando se utiliza el algoritmo de verificación SHA-512.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	35,917,117.60	531,868,324.60	1,044,080,068.80	35,977,109.60	529,536,900.20	1,041,946,540.60	36,481,579.40	533,733,065.40	1,046,405,852.00
	Encriptación	33,187,836.80	399,438,172.60	777,694,673.80	33,233,913.60	397,850,878.40	775,990,508.60	33,599,626.40	400,820,220.00	779,326,053.40
	Suma de Verificación	7,014,099.00	139,266,475.00	275,977,710.20	7,027,869.20	138,766,175.60	275,379,466.80	7,167,345.40	139,783,345.20	276,647,370.20
chacha20	Desencriptación	32,088,675.00	346,167,371.00	670,024,548.00	32,118,324.80	344,642,379.60	668,729,330.20	32,440,062.40	347,019,882.40	671,519,463.60
	Encriptación	30,354,839.40	262,136,784.40	501,448,891.40	30,382,116.40	261,111,970.40	500,511,499.40	30,620,130.80	262,868,641.20	502,611,998.00
	Suma de Verificación	7,013,667.00	139,310,796.20	275,792,366.20	7,028,076.80	138,704,899.00	275,409,756.60	7,168,185.80	139,841,793.20	276,492,835.40
des3	Desencriptación	125,633,460.20	5,408,513,284.00	10,862,253,404.20	126,256,065.00	5,384,671,285.00	10,841,387,688.75	131,639,856.80	5,427,044,406.60	10,888,202,028.20
	Encriptación	96,544,781.80	3,997,873,937.20	8,025,045,072.40	97,013,418.25	3,980,337,115.75	8,009,747,794.00	100,990,700.40	4,011,462,106.40	8,044,111,297.20
	Suma de Verificación	7,014,814.80	139,333,365.60	275,885,072.60	7,029,586.25	138,779,487.00	275,482,971.00	7,165,943.20	139,845,410.00	276,568,297.00

A60 - Cantidad de accesos a memoria externa utilizando algoritmo ECDSA y SHA-512

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	36,316,621	552,073,351	1,084,565,091	36,372,865	549,731,387	1,082,477,220	36,792,028	553,882,776	1,076,770,231
	Encriptación	33,190,141	399,495,656	777,491,338	33,232,921	397,795,372	775,957,070	33,605,062	400,737,437	779,334,151
	Suma de Verificación	7,736,832	110,848,140	217,309,081	7,738,566	110,418,672	216,898,467	7,187,325	106,981,304	217,284,250
chacha20	Desencriptación	35,042,728	490,821,607	961,146,054	35,098,551	488,658,118	959,442,152	35,563,544	492,376,496	963,577,773
	Encriptación	30,356,054	262,095,700	501,337,664	30,384,110	261,016,685	500,364,963	30,619,950	262,939,479	502,426,922
	Suma de Verificación	8,330,439	140,651,899	277,216,572	8,365,325	140,075,753	276,790,324	8,483,498	141,159,053	277,966,656
des3	Desencriptación	101,490,911	4,238,545,362	8,509,410,208	101,979,799	4,219,914,622	8,492,981,080	106,190,768	4,253,104,281	8,529,444,771
	Encriptación	96,550,072	3,997,889,080	8,025,260,772	97,020,119	3,980,239,570	8,009,790,698	100,990,708	4,011,519,398	8,043,933,852
	Suma de Verificación	8,342,744	140,740,727	277,250,949	8,349,769	140,056,773	276,539,844	8,485,940	141,118,800	277,934,204

A61 - Cantidad de accesos a memoria externa utilizando algoritmo RSA y SHA-512

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	35,921,640	531,911,829	1,043,910,170	35,976,185	529,631,282	1,041,904,975	36,481,579	533,575,163	1,046,405,852
	Encriptación	33,189,097	399,388,523	777,415,851	33,229,215	397,759,699	775,908,950	33,599,626	400,733,952	779,326,053
	Suma de Verificación	5,724,815	138,329,384	275,177,457	5,741,033	137,696,155	274,456,315	7,167,345	138,745,640	276,647,370
chacha20	Desencriptación	32,083,438	345,862,602	669,908,730	32,118,739	344,245,239	668,512,985	35,563,544	347,107,427	963,577,773
	Encriptación	30,360,995	262,099,746	501,438,370	30,383,840	261,144,307	500,506,511	30,619,950	262,860,340	502,426,922
	Suma de Verificación	5,729,602	138,350,097	275,175,821	5,741,988	137,741,192	274,519,891	8,483,498	138,808,017	277,966,656
des3	Desencriptación	125,630,326	5,408,630,300	10,862,377,841	126,261,008	5,384,757,162	10,841,304,457	131,639,923	5,426,940,703	10,887,907,975
	Encriptación	96,554,527	3,997,752,406	8,025,339,611	97,015,297	3,980,145,757	8,009,787,533	100,991,914	4,011,494,557	8,044,096,073
	Suma de Verificación	5,729,646	138,257,645	275,134,776	5,743,834	137,693,173	274,694,896	5,877,054	138,698,742	275,876,321

A62 - Cantidad de accesos a memoria externa utilizando algoritmo ECC y SHA-512

Algoritmo	Datos Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	35,916,435	531,898,745	1,043,959,292	35,978,656	529,640,383	1,041,980,335	36,483,636	533,652,492	1,046,271,622
	Encriptación	33,195,606	399,359,994	777,465,722	33,229,365	397,752,093	776,083,140	33,600,633	400,776,169	779,381,875
	Suma de Verificación	5,726,272	138,276,388	275,045,181	5,744,096	137,651,653	274,565,043	5,878,261	138,738,162	275,706,731
chacha20	Desencriptación	32,086,706	346,023,823	669,932,487	32,123,385	344,589,197	668,727,150	32,447,715	347,025,688	671,571,519
	Encriptación	30,357,023	262,083,442	501,410,366	30,387,239	261,057,217	500,266,279	30,624,035	262,984,030	502,644,947
	Suma de Verificación	5,727,333	138,242,761	275,081,952	5,740,693	137,679,251	274,728,154	5,876,293	138,707,147	275,844,706
des3	Desencriptación	125,631,608	5,408,639,795	10,862,225,076	126,256,613	5,384,687,466	10,841,320,475	131,643,420	5,426,960,086	10,887,728,236
	Encriptación	96,550,876	3,997,814,033	8,025,222,989	97,017,832	3,980,195,948	8,009,635,276	100,990,691	4,011,493,088	8,044,344,455
	Suma de Verificación	5,725,940	138,246,691	275,092,124	5,745,534	137,689,873	274,559,148	5,876,527	138,749,539	275,806,504

A63 - Cantidad de accesos a memoria externa utilizando algoritmo DH y SHA-512

Algoritmo de Verificación RIPEMD-160

Se observan en las tablas A64, A65 y A66 la cantidad de accesos a memoria externa que requiere cada una de las funciones utilizando el algoritmo de verificación RIPEMD-160. Este algoritmo no es compatible con ECDSA.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	36,305,459	552,044,674	1,084,707,121	36,374,485	549,800,947	1,082,334,574	36,893,569	553,897,685	1,086,912,892
	Encriptación	33,192,154	399,424,276	777,503,141	33,237,242	397,816,813	775,929,957	33,606,298	400,792,077	779,304,652
	Suma de Verificación	7,739,346	111,554,468	218,794,839	7,757,925	111,114,208	218,311,255	7,867,825	111,984,198	219,372,393
chacha20	Desencriptación	35,047,756	490,695,909	961,392,616	35,097,153	488,565,689	959,514,565	35,564,868	492,398,437	963,431,056
	Encriptación	30,359,978	262,173,111	501,377,846	30,380,994	261,109,771	500,310,876	30,617,823	262,954,034	502,571,099
	Suma de Verificación	7,747,224	111,616,942	218,853,764	7,754,663	111,120,074	218,373,742	7,856,598	111,951,933	219,327,597
des3	Desencriptación	101,483,461	4,238,490,598	8,509,343,018	102,006,157	4,219,873,118	8,492,861,777	106,193,633	4,253,064,637	8,529,046,715
	Encriptación	96,548,693	3,997,875,356	8,025,326,566	97,037,891	3,980,221,202	8,009,481,114	100,993,918	4,011,386,505	8,044,178,214
	Suma de Verificación	7,754,128	111,562,335	218,816,260	7,758,610	111,068,130	218,441,542	7,860,803	111,960,379	219,350,884

A64 - Cantidad de accesos a memoria externa utilizando algoritmo RSA y RIPEMD-160

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	35,915,280	531,949,595	1,043,719,456	35,973,684	529,628,906	1,041,946,358	36,483,545	533,601,939	1,046,364,006
	Encriptación	33,185,171	399,361,656	777,659,567	33,235,733	397,791,957	776,052,781	33,605,971	400,692,103	779,475,328
	Suma de Verificación	5,108,176	109,160,516	216,609,220	5,123,653	108,744,562	216,202,951	5,226,198	109,599,543	217,029,727
chacha20	Desencriptación	32,086,766	345,907,834	669,883,249	32,121,436	344,425,787	668,870,180	32,441,998	347,107,580	671,262,550
	Encriptación	30,359,294	262,114,667	501,613,641	30,382,575	261,101,815	500,507,425	30,617,694	263,059,135	502,544,501
	Suma de Verificación	5,114,715	109,304,146	216,589,523	5,122,791	108,763,850	216,137,250	5,230,614	109,527,958	217,139,023
des3	Desencriptación	125,630,803	5,408,626,651	10,862,497,243	126,260,461	5,384,757,618	10,841,272,998	131,641,697	5,427,077,401	10,888,068,012
	Encriptación	96,550,394	3,997,811,418	8,025,315,535	97,015,799	3,980,203,596	8,009,644,891	100,995,109	4,011,382,774	8,044,062,049
	Suma de Verificación	5,112,937	109,202,123	216,645,251	5,122,222	108,740,657	216,149,978	5,231,552	109,550,980	217,062,554

A65 - Cantidad de accesos a memoria externa utilizando algoritmo ECC y RIPEMD-160

Algoritmo	Función	Imagen			Texto			video		
		1	50	100	1	50	100	1	50	100
aes-256cbc	Desencriptación	35,924,983	531,900,124	1,043,827,544	35,978,513	529,725,360	1,041,916,889	36,477,563	533,506,442	1,046,341,883
	Encriptación	33,193,110	399,428,779	777,617,402	33,229,725	397,771,097	776,038,926	33,601,276	400,753,282	779,446,051
	Suma de Verificación	5,110,574	109,213,840	216,621,771	5,124,545	108,715,494	216,181,394	5,229,470	109,546,589	217,026,635
chacha20	Desencriptación	32,082,993	346,140,738	669,659,712	32,122,760	344,513,093	668,944,198	32,444,531	347,053,255	671,279,158
	Encriptación	30,356,509	262,169,882	501,465,194	30,381,975	261,193,273	500,432,836	30,621,912	262,922,502	502,670,080
	Suma de Verificación	5,111,097	109,249,187	216,723,214	5,125,730	108,756,086	216,232,360	5,229,796	109,563,041	217,053,095
des3	Desencriptación	125,631,221	5,408,679,935	10,862,195,483	126,258,212	5,384,702,070	10,841,230,640	131,644,485	5,427,099,337	10,888,090,932
	Encriptación	96,552,646	3,997,810,087	8,025,183,720	97,013,747	3,980,086,131	8,009,634,160	100,991,768	4,011,543,913	8,044,187,671
	Suma de Verificación	5,112,679	109,159,163	216,643,675	5,122,935	108,692,017	216,285,004	5,232,104	109,567,495	217,192,585

A66 - Cantidad de accesos a memoria externa utilizando algoritmo DH y RIPEMD-160

Tiempo de Ejecución

Algoritmo de Verificación SHA3-512

Se presentan en las tablas A67 a A70 los resultados obtenidos de tiempo de ejecución por cada una de las funciones teniendo en cuenta el algoritmo de verificación SHA3-512.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	0.08	1.97	12.82	0.08	1.85	22.71	0.36	1.88	10.55
	Encriptación	0.41	16.51	31.69	0.09	9.41	29.28	0.10	11.17	14.97
	Suma de Verificación	0.03	5.79	2.77	0.03	1.06	2.12	0.04	1.07	2.16
chacha20	Desencriptación	0.07	1.60	11.93	0.08	1.29	20.82	0.08	1.29	11.49
	Encriptación	0.33	17.84	34.00	2.19	8.61	24.53	0.06	12.98	11.56
	Suma de Verificación	0.03	2.30	3.16	0.03	1.06	2.12	0.04	1.73	2.15
des3	Desencriptación	0.14	5.32	19.14	0.14	5.40	30.56	0.25	6.79	16.99
	Encriptación	0.13	19.27	34.08	0.13	8.58	39.63	0.14	18.20	19.45
	Suma de Verificación	0.03	2.00	2.97	0.04	1.06	2.21	0.04	1.09	2.24

A67 - Tiempo de ejecución utilizando algoritmo ECSA y SHA3-512

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	0.28	1.87	14.17	0.08	1.87	10.58	0.22	1.87	9.25
	Encriptación	0.09	4.68	15.94	0.74	7.01	15.93	1.15	4.23	13.26
	Suma de Verificación	0.04	1.08	3.27	0.04	1.07	2.13	0.04	2.50	2.15
chacha20	Desencriptación	0.07	1.28	11.33	0.07	1.27	11.16	0.08	1.28	10.18
	Encriptación	0.06	2.75	12.84	0.06	4.98	12.93	0.07	3.91	11.88
	Suma de Verificación	0.04	2.88	2.84	0.04	1.14	2.13	0.61	1.08	2.15
des3	Desencriptación	0.14	5.31	20.30	1.36	5.37	20.66	0.14	5.44	16.33
	Encriptación	0.13	6.34	21.54	0.13	6.09	22.96	0.19	5.88	18.33
	Suma de Verificación	0.04	4.73	2.16	0.04	1.06	2.13	0.04	1.86	2.14

A68 - Tiempo de ejecución utilizando algoritmo RSA y SHA3-512

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	0.08	1.88	13.56	0.08	1.87	23.36	0.46	1.87	11.03
	Encriptación	0.09	17.97	21.88	0.86	3.61	26.57	0.10	8.29	16.07
	Suma de Verificación	0.08	2.48	2.13	0.03	1.06	2.12	0.16	3.17	2.14
chacha20	Desencriptación	0.07	1.31	12.71	0.07	1.37	23.74	0.08	1.29	10.18
	Encriptación	0.06	13.57	17.67	0.23	4.45	22.57	0.07	8.99	11.88
	Suma de Verificación	0.18	1.87	2.48	0.03	1.06	2.12	0.61	1.07	2.15
des3	Desencriptación	0.14	5.39	21.44	0.14	5.35	28.08	0.26	5.33	16.77
	Encriptación	0.65	20.67	22.55	0.19	8.22	33.52	0.13	11.80	20.87
	Suma de Verificación	0.03	1.06	2.69	0.04	1.06	2.14	0.03	1.07	2.15

A69 - Tiempo de ejecución utilizando algoritmo ECC y SHA3-512

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	0.08	2.63	10.50	0.08	1.86	24.84	0.39	1.97	11.94
	Encriptación	0.09	10.80	30.66	0.48	8.17	25.80	0.09	13.22	15.46
	Suma de Verificación	0.43	1.07	2.30	0.03	1.06	2.22	0.50	3.43	2.14
chacha20	Desencriptación	0.07	1.27	10.47	0.07	1.29	24.52	1.84	1.29	11.34
	Encriptación	0.06	14.97	30.59	0.06	5.69	20.07	0.06	15.46	12.14
	Suma de Verificación	1.14	1.62	2.49	0.04	1.21	2.40	0.19	1.07	2.14
des3	Desencriptación	0.14	5.31	19.69	0.14	5.29	32.96	0.39	5.34	16.39
	Encriptación	0.13	20.31	35.43	0.13	7.92	36.15	0.14	20.34	19.71
	Suma de Verificación	0.42	3.34	2.13	0.03	1.06	2.13	0.18	1.09	2.15

A70 - Tiempo de ejecución utilizando algoritmo DH y SHA3-512

Algoritmo de Verificación SHA1

En las tablas A71 a A74 se pueden observar los tiempos de ejecución que demandan las funciones utilizando el algoritmo de suma de verificación SHA1.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
des3	Desencriptación	0.14	5.34	21.38	0.14	5.55	33.58	0.50	5.35	19.52
	Encriptación	0.13	20.28	39.23	0.28	5.97	36.52	0.13	15.57	20.24
	Suma de Verificación	0.02	0.43	0.85	0.02	0.43	0.84	0.02	1.77	0.92
chacha20	Desencriptación	0.07	1.27	12.51	0.07	1.58	20.82	0.07	1.46	12.31
	Encriptación	0.25	20.64	31.13	1.32	6.75	21.91	0.06	17.98	12.94
	Suma de Verificación	0.02	0.43	1.26	0.02	0.42	0.84	0.02	1.48	0.85
aes-256-cbc	Desencriptación	0.08	1.89	13.30	0.08	1.86	25.64	0.28	2.14	9.68
	Encriptación	0.09	15.82	32.72	0.25	6.73	27.67	0.09	15.12	14.78
	Suma de Verificación	0.02	0.48	0.85	0.02	0.43	1.05	0.02	1.43	0.86

A71 - Tiempo de ejecución utilizando algoritmo ECDSA y SHA1

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	0.08	1.89	11.28	0.08	1.86	10.62	0.71	1.91	11.12
	Encriptación	0.09	3.57	14.09	0.09	5.92	17.54	0.15	4.43	14.00
	Suma de Verificación	0.03	1.23	1.32	0.03	0.44	0.95	0.03	0.69	0.86
chacha20	Desencriptación	0.07	1.28	7.66	1.42	1.51	8.30	0.07	1.29	12.08
	Encriptación	0.06	1.89	13.17	0.06	6.55	12.41	0.06	3.21	11.43
	Suma de Verificación	0.03	0.44	1.79	0.03	0.43	0.85	0.03	0.44	0.86
des3	Desencriptación	0.14	5.31	18.51	0.14	5.72	19.21	0.38	5.49	19.17
	Encriptación	0.13	8.84	24.87	0.13	5.74	26.33	0.13	8.20	18.19
	Suma de Verificación	0.03	0.43	0.85	0.03	0.43	0.87	0.03	0.44	0.88

A72 - Tiempo de ejecución utilizando algoritmo RSA y SHA1

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	0.09	1.89	14.32	0.08	2.13	28.93	0.19	1.90	9.50
	Encriptación	0.30	14.36	26.65	0.09	8.90	28.06	0.10	10.54	16.67
	Suma de Verificación	0.02	1.25	0.86	0.02	0.43	0.85	0.02	0.44	0.86
des3	Desencriptación	0.14	5.50	20.32	0.14	5.30	23.46	0.72	5.34	17.89
	Encriptación	0.15	25.46	22.19	0.13	6.54	32.02	0.13	10.38	21.98
	Suma de Verificación	0.02	1.73	0.86	0.02	0.42	0.85	0.02	0.43	0.86
chacha20	Desencriptación	0.07	1.28	9.84	0.07	1.80	23.29	1.05	1.44	11.63
	Encriptación	0.06	13.28	23.68	0.23	5.61	24.94	0.06	5.86	11.15
	Suma de Verificación	1.42	0.43	1.39	0.02	0.42	0.84	0.02	0.44	0.84

A73 - Tiempo de ejecución utilizando algoritmo RSA y SHA1

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	0.08	1.87	10.98	0.13	2.10	22.47	1.61	1.88	11.83
	Encriptación	0.09	17.50	30.13	0.10	8.25	25.27	0.10	14.17	14.23
	Suma de Verificación	0.78	0.43	0.85	0.02	0.84	1.43	1.88	0.43	0.85
chacha20	Desencriptación	0.07	1.40	10.30	0.07	1.27	23.70	1.85	1.29	7.74
	Encriptación	0.06	14.31	28.06	0.06	6.01	23.09	0.06	13.44	11.19
	Suma de Verificación	0.71	1.60	0.93	0.25	0.43	0.85	0.78	0.43	0.86
des3	Desencriptación	0.14	5.37	19.87	0.14	5.29	33.92	0.27	5.35	16.24
	Encriptación	0.13	19.07	38.14	0.13	8.06	34.17	0.14	16.04	21.02
	Suma de Verificación	0.63	1.73	0.85	0.23	0.42	0.85	0.38	0.43	0.85

A74 - Tiempo de ejecución utilizando algoritmo DH y SHA1

Algoritmo de Verificación RIPEMD-160

Las tablas A75, A76 y A77 muestran los tiempos de ejecución de las funciones en la RPI4 empleando el algoritmo de suma de verificación RIPEMD-160.

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	0.08	1.87	10.73	0.08	1.86	13.98	0.08	1.89	11.35
	Encriptación	0.10	3.56	16.72	0.09	6.75	18.21	0.10	3.69	12.74
	Suma de Verificación	0.03	1.21	1.59	0.03	0.57	1.13	0.03	0.79	1.12
chacha20	Desencriptación	0.07	1.28	10.63	0.07	1.30	9.80	0.07	1.29	11.58
	Encriptación	0.06	4.57	14.53	0.07	6.79	13.57	0.08	2.56	12.39
	Suma de Verificación	0.03	1.11	1.87	0.03	0.57	1.12	0.03	0.81	1.14
des3	Desencriptación	1.42	5.33	19.18	0.14	5.50	19.84	0.14	5.92	18.73
	Encriptación	0.13	6.30	23.16	0.28	6.26	26.37	0.13	5.75	19.56
	Suma de Verificación	0.03	1.14	1.12	0.03	0.57	1.31	0.03	1.28	1.14

A75 - Tiempo de ejecución utilizando algoritmo RSA y RIPEMD-160

Algoritmo	Función	Imagen			Texto			Video		
		1	50	100	1	50	100	1	50	100
aes-256-cbc	Desencriptación	0.08	1.88	15.32	0.08	1.86	23.31	0.31	1.89	11.22
	Encriptación	0.28	20.56	23.72	0.32	5.44	28.75	0.09	8.85	16.50
	Suma de Verificación	0.02	1.90	2.46	0.02	0.56	1.12	0.02	0.57	1.13
chacha20	Desencriptación	0.07	1.37	13.57	0.07	1.68	24.58	2.92	1.30	10.68
	Encriptación	0.06	14.78	26.32	0.07	8.77	28.89	0.06	7.98	12.74
	Suma de Verificación	0.02	0.56	5.51	0.02	0.56	1.13	1.16	0.57	1.12
des3	Desencriptación	0.14	5.41	18.41	0.14	5.74	28.97	1.49	5.34	16.06
	Encriptación	0.67	19.65	24.18	0.27	8.79	32.45	0.14	11.14	19.60
	Suma de Verificación	0.02	0.56	1.18	0.02	0.56	1.13	0.02	0.56	1.12

A76 - Tiempo de ejecución utilizando algoritmo ECC y RIPEMD-160

Algoritmo	Función	Imagen			Texto			video		
		1	50	100	1	50	100	1	50	100
aes-256cbc	Desencriptación	0.08	1.87	8.73	0.08	1.86	24.28	0.32	1.87	10.55
	Encriptación	0.10	11.26	29.27	0.25	9.28	26.11	0.10	11.84	13.76
	Suma de Verificación	0.26	0.58	1.13	0.02	0.85	1.80	0.02	0.57	1.12
chacha20	Desencriptación	0.07	1.53	11.04	0.07	1.28	27.23	0.54	1.30	10.92
	Encriptación	0.06	21.35	28.79	0.06	2.47	22.03	0.07	16.38	10.66
	Suma de Verificación	0.33	0.56	1.21	0.12	0.62	1.67	0.17	3.56	1.13
des3	Desencriptación	0.14	5.45	21.55	0.14	5.31	31.96	0.15	5.85	17.42
	Encriptación	0.13	22.55	32.25	0.13	6.59	31.80	0.13	14.79	19.18
	Suma de Verificación	0.28	1.85	1.36	0.03	0.56	1.12	0.14	0.57	1.16

A77 - Tiempo de ejecución utilizando algoritmo DH y RIPEMD-160