Universidad de los Andes
Caso de Estudio 3 – Canales Seguros – Infraestructura Computacional

Thomas Bonnett – 202225550
Camila Fajardo – 202220711

El presente documento describe la medición y análisis de los tiempos requeridos por el servidor para ejecutar tres operaciones críticas dentro de un protocolo de comunicación seguro: firma digital, cifrado de datos y verificación de integridad (HMAC). Para ello, se evaluaron dos escenarios principales: uno iterativo con un cliente realizando múltiples solicitudes secuenciales, y otro concurrente con múltiples clientes realizando solicitudes simultáneamente. A través de estas mediciones se busca comparar el comportamiento del sistema en diferentes condiciones de carga, entender el impacto de la concurrencia en el desempeño de las operaciones criptográficas y analizar las diferencias de tiempo entre cifrado simétrico y asimétrico. Finalmente, se presentan los resultados de los experimentos, las gráficas comparativas y un conjunto de observaciones sobre los patrones encontrados.

Estructura:

A continuación se presenta una descripción breve de las principales clases que componen la implementación del protocolo seguro. Cada clase tiene un rol específico dentro del sistema: algunas manejan la comunicación y sincronización entre cliente y servidor, mientras que otras implementan las funciones criptográficas necesarias para garantizar la confidencialidad, integridad y autenticidad de los mensajes intercambiados. Estas descripciones permiten entender de manera clara la estructura general del programa y la responsabilidad de cada componente dentro del flujo de ejecución.

Descripción de la clase Cliente2:

La clase Cliente2 implementa la lógica de un cliente que establece una conexión segura con un servidor mediante un protocolo de autenticación y comunicación cifrada. Utiliza criptografía asimétrica (RSA) para autenticación mutua y el protocolo de intercambio de claves Diffie-Hellman (DH) para derivar llaves de sesión simétricas (AES para cifrado de datos y HMAC-SHA256 para verificación de integridad). Una vez establecida la conexión segura, el cliente realiza un número configurable de consultas (envíos de solicitudes de servicio) al servidor, seleccionando aleatoriamente el servicio a solicitar. Al finalizar las consultas, envía un mensaje especial ("FIN") para cerrar la sesión de forma controlada. La

clase extiende Thread, permitiendo la ejecución de múltiples clientes en paralelo en modo concurrente.

Descripción de la clase ClienteConcurrente:

La clase ClienteConcurrente representa un cliente que establece una conexión segura con un servidor, ejecutándose en un hilo independiente para soportar múltiples conexiones concurrentes. Utiliza un protocolo de autenticación y comunicación segura que combina criptografía asimétrica (RSA) para autenticación inicial y criptografía simétrica (AES y HMAC) para el intercambio de datos protegido. Tras autenticarse y establecer llaves de sesión usando el protocolo Diffie-Hellman, el cliente envía una solicitud cifrada de un servicio seleccionado aleatoriamente y recibe una respuesta segura. Finalmente, envía un mensaje de cierre cifrado ("FIN") para terminar la comunicación. La clase es capaz de ser ejecutada muchas veces en paralelo, permitiendo pruebas de carga y concurrencia.

Descripción de la clase DelegadoServidor:

La clase DelegadoServidor representa el encargado de manejar individualmente la comunicación segura entre el servidor y un cliente específico. Cada vez que un cliente se conecta, se crea una instancia de esta clase, ejecutándose en su propio hilo para permitir múltiples conexiones concurrentes. El protocolo de comunicación implementado asegura autenticación, establecimiento seguro de llaves de sesión, intercambio de mensajes cifrados y autenticados, además de medición precisa de tiempos críticos de operación.

Descripción de la clase FuncionesCrypto:

La clase FuncionesCrypto proporciona un conjunto de funciones utilitarias para realizar operaciones criptográficas fundamentales utilizadas en protocolos de comunicación segura. Ofrece métodos para el cifrado y descifrado de datos usando RSA (asimétrico) y AES (simétrico), así como generación y verificación de códigos de autenticación de mensajes (HMAC) utilizando SHA-256.

Descripción de la clase ServidorPrincipal:

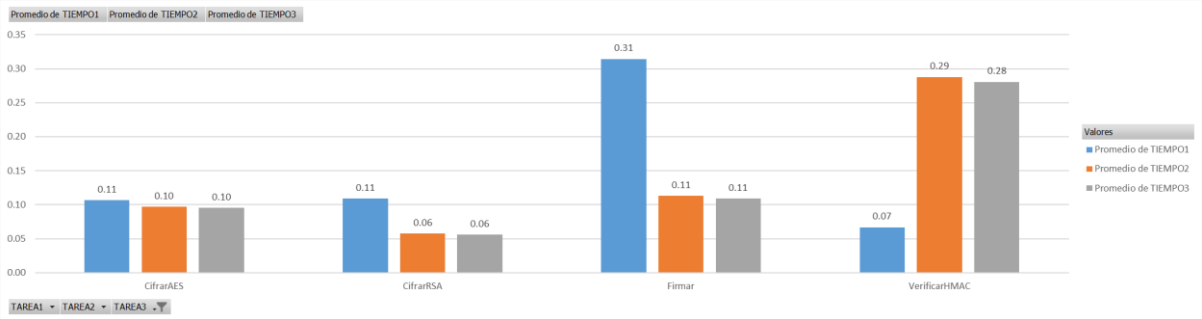
La clase ServidorPrincipal implementa el servidor principal encargado de aceptar conexiones de múltiples clientes y delegar la atención de cada cliente a un hilo

separado (DelegadoServidor). Además, gestiona la carga de las llaves públicas y privadas necesarias para la comunicación segura usando el algoritmo RSA.

Pruebas

Cada escenario se ejecutó 3 veces para tener una medida un poco más clara de los tiempos necesarios. Ahora bien, para la obtención de datos de las ejecuciones de cada escenario, se guardó la información de los tiempos de cada tarea (i) Firmar, (ii) cifrar la tabla y (iii) verificar la consulta en un archivo .csv por cada ejecución.

Escenario Iterativo



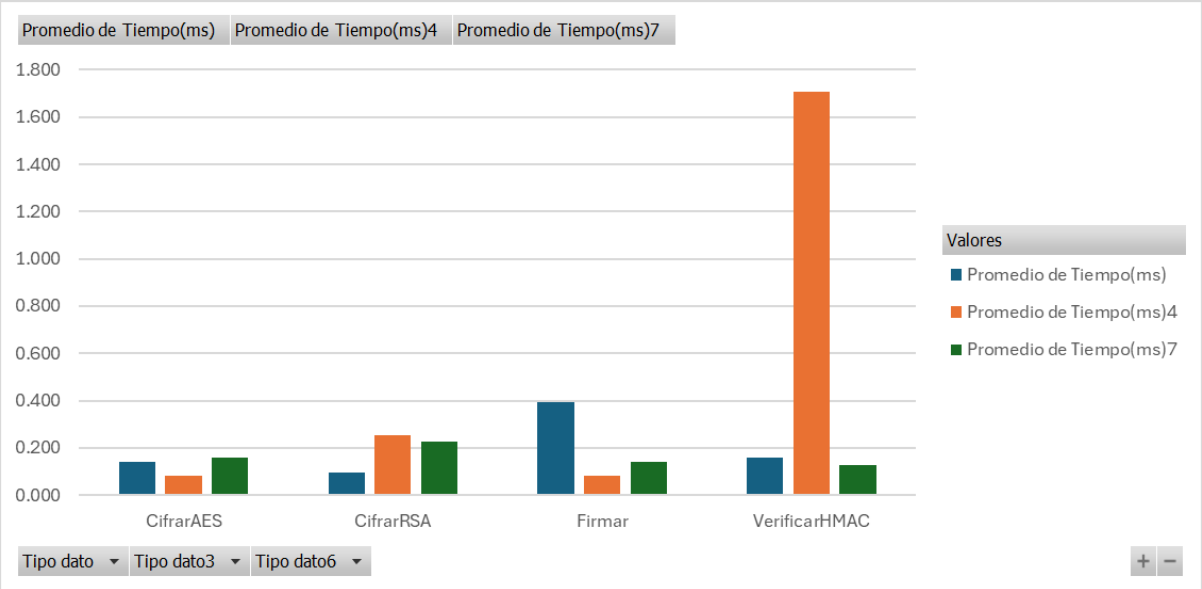
En el escenario iterativo se observa que el tiempo promedio para realizar el cifrado con AES es ligeramente mayor que el tiempo requerido para cifrar utilizando RSA. Específicamente, el cifrado simétrico con AES mostró promedios cercanos a 0.10–0.11 ms, mientras que el cifrado asimétrico con RSA estuvo alrededor de 0.06 ms. Esta situación, aunque inicialmente puede parecer contradictoria, se explica porque en el caso de mensajes pequeños, como los utilizados en este protocolo (solicitudes sencillas), la sobrecarga de inicializar el proceso de cifrado simétrico puede hacer que AES no sea tan eficiente como RSA en estas condiciones específicas.

Por otra parte, la operación de firma digital resultó ser la más costosa en términos de tiempo entre todas las tareas medidas. El tiempo promedio para firmar alcanzó valores de hasta 0.31 ms en una de las ejecuciones. Esto era esperado, ya que firmar implica el uso de operaciones de clave privada (RSA) que son inherentemente más pesadas y complejas que un simple cifrado o verificación de integridad.

Respecto a la verificación del HMAC, se registraron tiempos relativamente elevados, con promedios que alcanzaron 0.28–0.29 ms en dos de las mediciones. Este comportamiento se debe a que, en el escenario iterativo, el cliente realiza 32 consultas secuenciales y el servidor debe verificar el HMAC para cada una de ellas, acumulando así una carga de trabajo considerable a lo largo de toda la sesión.

Finalmente, se aprecia cierta variabilidad entre las diferentes ejecuciones (TIEMPO1, TIEMPO2 y TIEMPO3), especialmente en las tareas de firmar y verificar el HMAC. Esta variabilidad puede ser atribuida a factores propios del entorno de ejecución, como fluctuaciones en la carga del procesador o pequeñas interrupciones del sistema operativo, que son comunes en pruebas locales.

Escenario Concurrente con 4 delegados



En el escenario concurrente con 4 delegados se observa un comportamiento distinto en comparación con el escenario iterativo. Se nota que los tiempos de cifrado (tanto simétrico con AES como asimétrico con RSA) y de firma aumentaron ligeramente, aunque permanecieron dentro de rangos aceptables. El tiempo de cifrado con AES se mantuvo relativamente bajo (cercano a 0.1–0.2 ms), mientras que el cifrado con RSA presentó un ligero aumento en una de las mediciones, llegando hasta cerca de 0.2–0.25 ms en promedio.

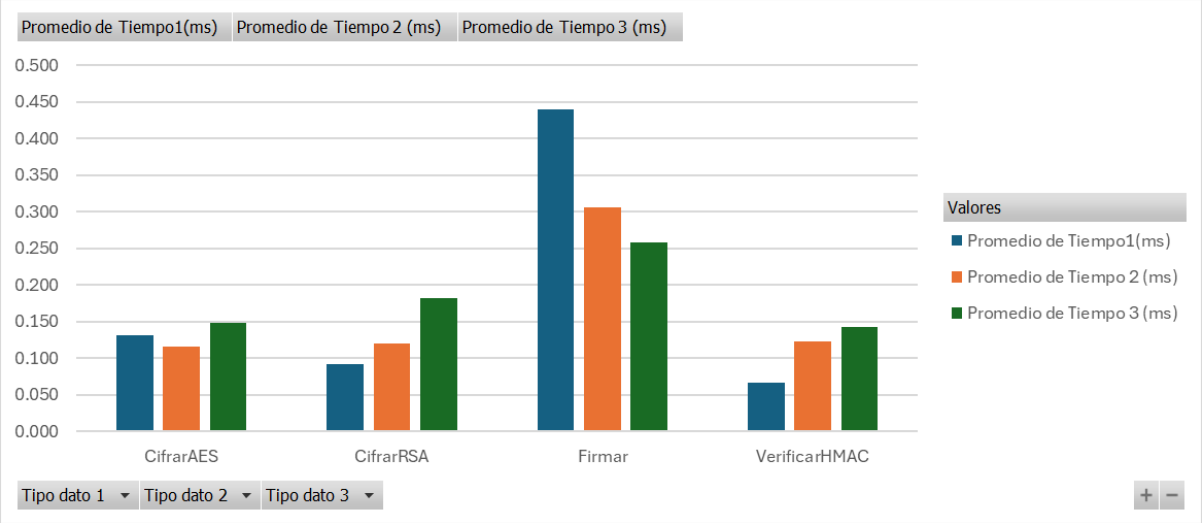
Un comportamiento interesante se da en la firma digital, donde en una de las ejecuciones el tiempo promedio para firmar se elevó hasta aproximadamente 0.4 ms, un incremento significativo respecto al escenario iterativo. Esto se puede atribuir a la sobrecarga generada por manejar múltiples conexiones simultáneas, que afecta la velocidad de operaciones intensivas en CPU como la firma digital RSA.

Sin embargo, el aspecto más llamativo de este escenario es el comportamiento de la verificación de HMAC. En una de las ejecuciones se registró un tiempo promedio extraordinariamente alto, cercano a 1.7 ms. Esto representa un salto abrupto respecto a las demás mediciones y sugiere que, bajo concurrencia, la verificación de múltiples solicitudes simultáneas puede saturar parcialmente los recursos del

servidor, generando retrasos acumulados en la verificación de integridad de los mensajes.

En general, aunque las operaciones de cifrado no se ven afectadas de forma drástica por la concurrencia, la verificación de integridad (HMAC) muestra una mayor sensibilidad al aumento de la carga, especialmente si las solicitudes de los clientes llegan en momentos muy cercanos o simultáneos, generando picos de latencia.

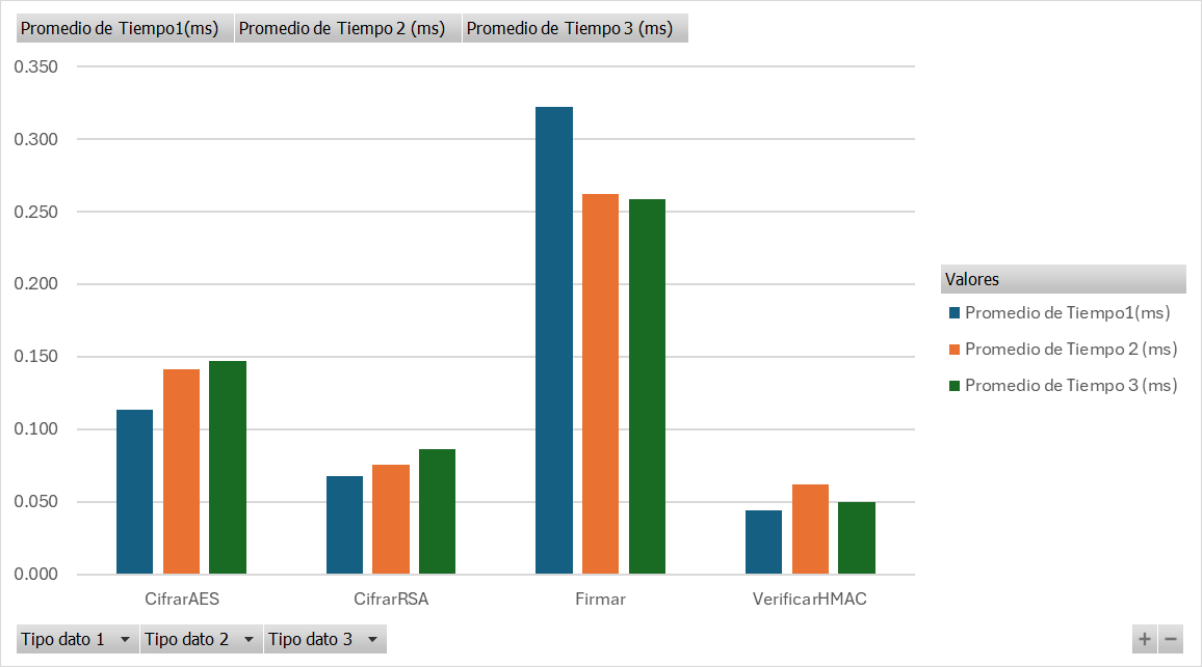
Escenario con 16 delegados



En el escenario de 16 delegados, los tiempos de cifrado AES y RSA se mantienen bajos y estables, alrededor de 0.1 a 0.15 ms. Aunque el cifrado RSA crece ligeramente respecto a escenarios anteriores, el impacto no es drástico. El tiempo

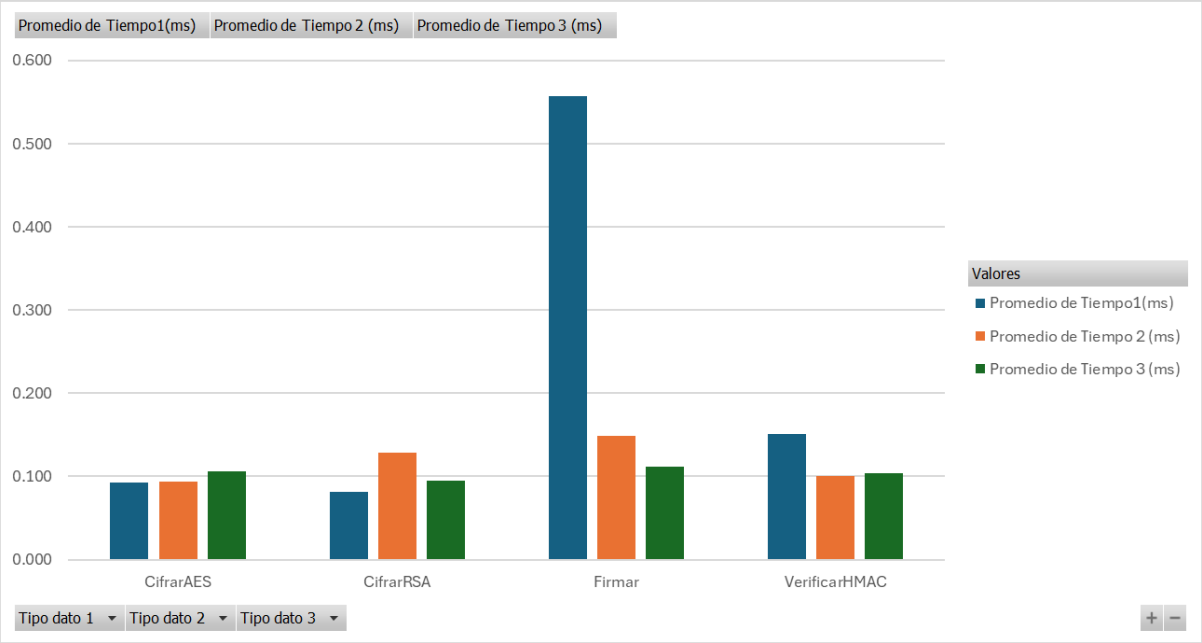
de firma, en cambio, muestra mayor variabilidad: en una medición alcanzó cerca de 0.45 ms, evidenciando que la operación de firma sigue siendo sensible a la carga concurrente. La verificación de HMAC también se incrementa un poco, pero de forma más controlada que en el escenario con 4 delegados. En general, se aprecia que el servidor maneja mejor el aumento de clientes, aunque las operaciones de firma continúan siendo las más costosas en tiempo bajo alta concurrencia.

Escenario con 32 delegados



En el escenario de 32 delegados, los tiempos de cifrado AES y RSA permanecen relativamente bajos y estables, con ligeros aumentos en comparación con escenarios anteriores. El tiempo de firma sigue siendo el más alto entre todas las operaciones, pero ahora se observa una ligera disminución respecto al escenario de 16 delegados, indicando posiblemente un mejor manejo de concurrencia o menores interferencias en algunas mediciones. La verificación HMAC se mantiene como la operación más rápida y estable. En general, el servidor parece adaptarse bien al incremento de clientes, manteniendo todos los tiempos controlados.

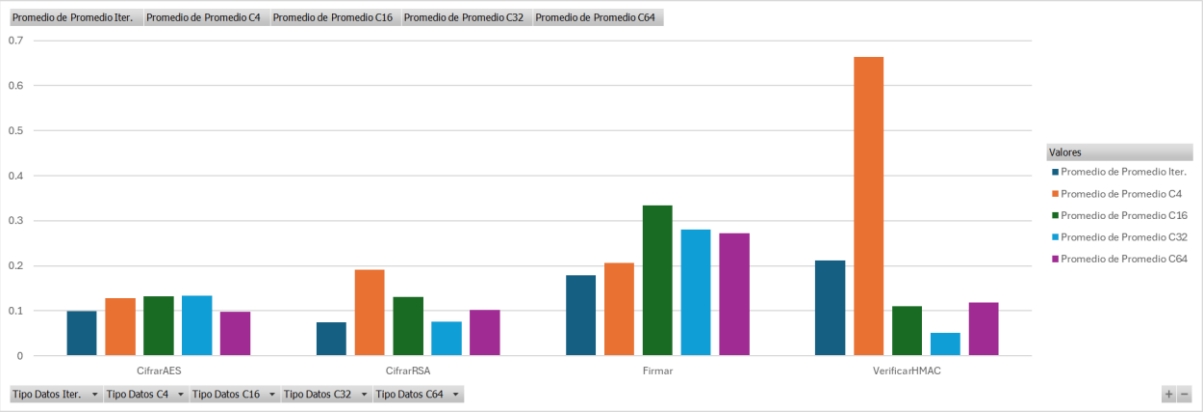
Escenario con 64 delegados



En el escenario de 64 delegados, se observa que los tiempos de cifrado AES y RSA se mantienen relativamente bajos y estables, muy similares entre sí y sin grandes variaciones respecto a escenarios con menos delegados. Sin embargo, el tiempo de firma presenta un aumento importante, alcanzando su valor más alto en comparación con los otros escenarios, lo cual sugiere que la operación de firma comienza a saturarse cuando el número de clientes concurrentes es muy alto.

Por otro lado, el tiempo de verificación HMAC también muestra un leve aumento, pero sigue siendo bastante manejable y considerablemente inferior al de las operaciones de firma. En general, aunque el servidor sigue respondiendo de manera aceptable para las tareas de cifrado y verificación, la firma se convierte en el principal cuello de botella cuando se incrementa la concurrencia.

Comparación de escenarios



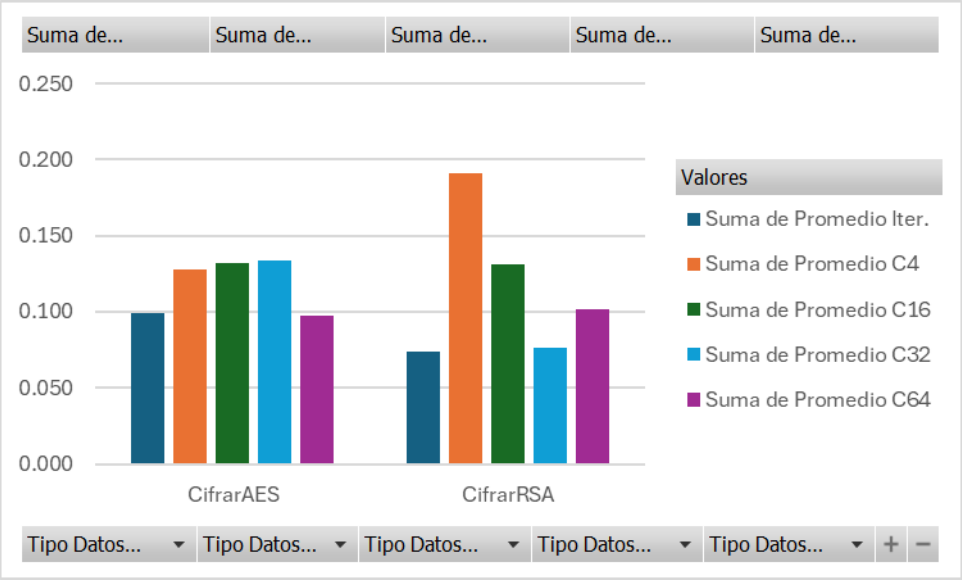
Al comparar todos los escenarios, se evidencia que las tareas de cifrado (AES y RSA) se mantienen bastante estables en todos los casos, con variaciones menores a medida que aumenta la concurrencia. El cifrado AES y RSA presentan tiempos similares en todos los escenarios, confirmando que estas operaciones no se ven fuertemente afectadas por la cantidad de clientes conectados.

En cuanto a la firma, se observa un incremento progresivo a medida que aumenta el número de delegados, siendo el escenario de 16 y 32 delegados donde el tiempo de firma alcanza su máximo, seguido de una ligera reducción en 64 delegados. Esto sugiere que inicialmente el sistema se ve más exigido con más clientes, pero luego podría estabilizarse gracias a la forma en que se maneja la concurrencia o a la eficiencia de las firmas en paralelo.

Por último, la verificación HMAC muestra un comportamiento particular: en el caso de 4 delegados, el tiempo de verificación es extremadamente alto comparado con los demás escenarios. Posteriormente, con 16, 32 y 64 delegados, los tiempos disminuyen notablemente, mostrando que a mayor concurrencia (más clientes trabajando), la operación de verificación HMAC tiende a estabilizarse y volverse más eficiente.

En general, los resultados indican que la firma digital es el proceso más sensible a la cantidad de clientes, mientras que el cifrado y la verificación mantienen un rendimiento aceptable en todos los escenarios.

Comparación RSA vs AES



En conclusión, al analizar las gráficas de comparación entre ambos cifrados y por la variación de los resultados obtenidos, podemos asumir que hubo casos en los que uno o más Threads tuvieron ejecuciones muy lentas o rápidas del cifrado RSA, ademas, se sabe que, en la ejecución de procesos en hilos, no se tiene una medida exacta de cuanto se demora un hilo en una tarea. Sin embargo, la ejecución si es consistente ya que en 3 de los 5 escenarios la ejecución del cifrado RSA fue más lenta que la ejecución del cifrado AES. Por otro lado, factores como la cantidad de procesos simultaneos no afectaron al procesador debido a su capacidad de cómputo (Ryzen 7 5800X 8-core). Sin embargo, en un sistema con menor capacidad de cómputo, es posible que la variación en los tiempos de ejecución sea más notoria, especialmente con un mayor número de hilos concurrentes, lo que podría afectar negativamente el rendimiento.

Estimación de procesos

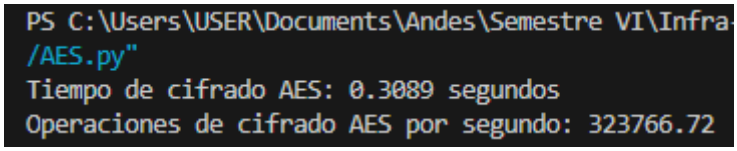
Para esto se deben recordar dos conceptos clave en cuanto al funcionamiento de cada algoritmo:

- AES es mucho más rápido debido a su naturaleza simétrica y las optimizaciones que permiten cifrar grandes cantidades de datos de manera eficiente.
- RSA, en cambio, es un algoritmo más costoso debido a la necesidad de realizar operaciones matemáticas complejas en grandes números (exponentes y módulos), lo que hace que el número de operaciones por segundo sea mucho menor.

Se utilizó un pequeño Script para cada tipo de cifrado, ambos se encuentran en la carpeta estimación.

El script de AES funciona indicándole un tamaño de 1KB por bloque y un numero de 100000 bloques utilizando una clave de 256 bits

Este nos arroja un valor de aproximadamente 323766 operaciones de cifrado por segundo:



Por su parte, en el script de RSA le indicamos que lo realice con 100000 y con un tamaño de bloque de máximo 256 obteniendo así el siguiente resultado.