Universidad de los Andes
Ingeniería de Sistemas y Computación
ISIS 2203 Infraestructura Computacional
Semestre 2024-20

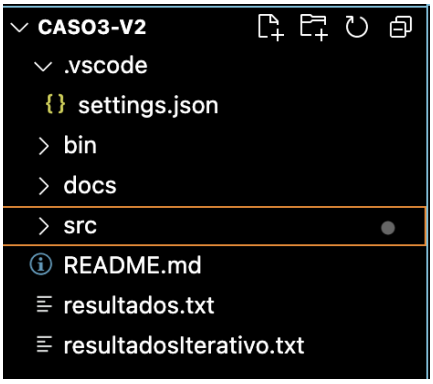
Caso 3

Angélica Ortiz - 202222480
María José Amorocho - 202220179

https://docs.google.com/spreadsheets/d/1JGPCRV4mHZUg4pZ6gEsYoBeRlOh8wcyLk-krlDe4KH0/edit?usp=sharing

1. Descripción de la organización de los archivos en el zip

Al abrir la carpeta se encontrarán las siguientes subcarpetas, las principales que vamos a utilizar para ejecutar el proyecto o ver la documentación de este son: docs, src y la carpeta principal donde se encuentra los resultados de los tiempos de cuando se utiliza iterativo y concurrente.



En la carpeta de src se encuentra la clase main que será la que se van a ejecutar. En esta carpeta se encuentran los siguientes archivos:

CifradoAES: Esta clase maneja el cifrado y descifrado de mensajes utilizando el algoritmo AES en modo CBC y claves. Contiene métodos para generar una clave AES a partir de un hash, crear un vector de inicialización (IV) aleatorio, cifrar mensajes y descifrar estos mensajes. Esto es para el cifrado simétrico.

CifradoHMAC: Esta clase permite generar y verificar códigos HMAC. Proporciona métodos para crear una clave HMAC a partir de un hash, para cifrar mensajes con HMAC y asegurar su integridad, y para comprobar si un mensaje coincide con el HMAC esperado. Esto se realiza para el cifrado asimétrico.

ClienteConDelegados: Esta clase es responsable de iniciar y gestionar múltiples instancias de clientes delegados de forma concurrente. Esto lo hace con respecto a los delegados que eligió el usuario y crea hilos independientes para cada cliente, permitiendo que se conecten y comuniquen simultáneamente con el servidor. Esto es para la parte concurrente.

ClienteDelegado: Esta clase representa un cliente delegado que se conecta al servidor y sigue un protocolo específico para la comunicación dependiendo del cifrado. Cada cliente delegado corre en su propio hilo y establece una conexión con el servidor mediante un socket,

intercambiando datos de entrada y salida para cumplir con el protocolo de comunicación definido. Esto es para la parte concurrente.

ClienteIterativo: Esta clase define un cliente que establece una conexión con el servidor y realiza 32 consultas de manera iterativa usando el mismo socket. Con un método para configurar el ID del cliente y del paquete, y una estructura para ejecutar las consultas secuencialmente. Este se utiliza para la parte iterativa.

Deposito: Esta clase simula un almacén o depósito que contiene información sobre el estado de diversos paquetes, almacenados en un HashMap.

DiffieHellman: Esta clase permite generar y verificar los parámetros necesarios para el intercambio de claves Diffie-Hellman, en este se utiliza openSSL,

Iterativo: Actúa como el controlador principal que inicializa y coordina el servidor y el cliente. Configura el tipo de cifrado y establece el camino SSL para el intercambio seguro de información.

ManejadorCliente: Es un hilo que maneja la comunicación con cada cliente de forma concurrente. Recibe la conexión, ejecuta el protocolo iterativo para procesar los datos del cliente, y cierra el socket al finalizar. Utiliza ProtocoloServidorIterativo para procesar las solicitudes del cliente.

Paquete: Representa un paquete con un identificador y un estado. Define los diferentes estados posibles. Facilita el seguimiento del estado del paquete en el sistema de entrega.

ProtocoloClienteIterativo: Establece un sistema de comunicación entre un cliente y un servidor. El cliente comienza generando un reto aleatorio cifrado, que el servidor debe descifrar para confirmar la conexión. Luego, se establece una clave compartida utilizando el protocolo Diffie-Hellman, con autenticidad verificada por una firma del servidor. Tras esto, el cliente genera una clave secreta compartida, que se utiliza para derivar claves de cifrado y autenticación.

ProtocoloServidorIterativo: En esta clase se implementa un protocolo de comunicación entre un servidor y un cliente utilizando el cifrado asimétrico y simétrico. El servidor inicia el proceso de autenticación respondiendo a un reto cifrado enviado por el cliente. Luego, establece una clave compartida con el cliente mediante el protocolo Diffie-Hellman, generando las llaves necesarias para cifrado y verificación. Después, el servidor recibe y verifica la autenticidad de los datos enviados por el cliente, calcula los hash correspondientes para validar la integridad. Si todo es correcto, consulta el estado del paquete en un depósito y envía la respuesta cifrada y autenticada al cliente.

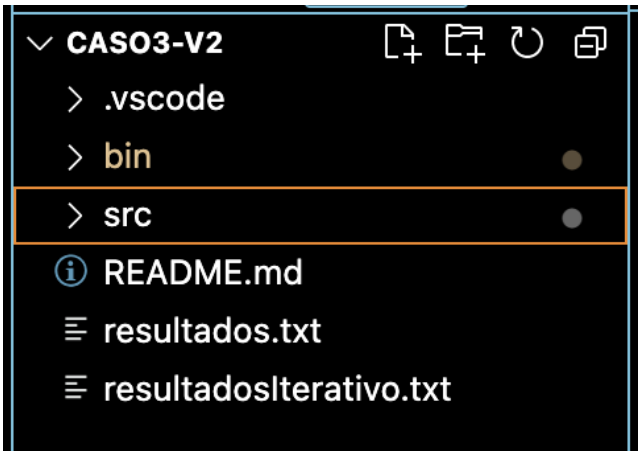
ServidorIterativo: La clase implementa un servidor que maneja las conexiones de clientes de forma secuencial. Por medio de un socket acepta conexiones y maneja consultas. El servidor acepta un cliente, procesa hasta 32 consultas por medio de su protocolo.

ServidorDelegados: Esta clase gestiona conexiones de clientes usando delegados, lo que permite manejar múltiples clientes de manera concurrente.Crea un nuevo thread manejador cliente para cada cliente conectado, permitiendo que varios clientes sean atendidos simultáneamente.

Tiempo: Por medio de esta clase se toman los tiempos de descifrar el reto, generar Gx, P, G y verificar la consulta.

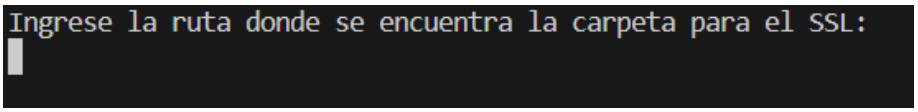
Despues de esto se tiene la carpeta donde se guardan la llave publica y privada generada en el primer paso.

En la carpeta principal se envc¿cuentran los resultados de tiempo del servidor concurrente (resultados.txt) y del iterativo (resultadosIterativo.txt).

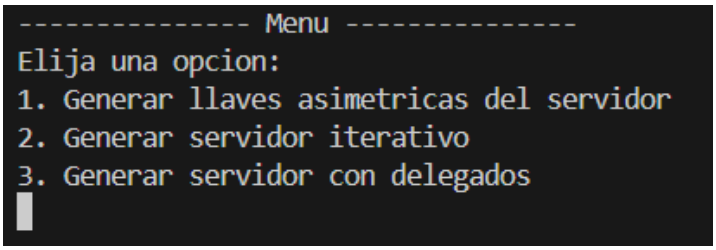


2. Instrucciones para correr servidor y cliente

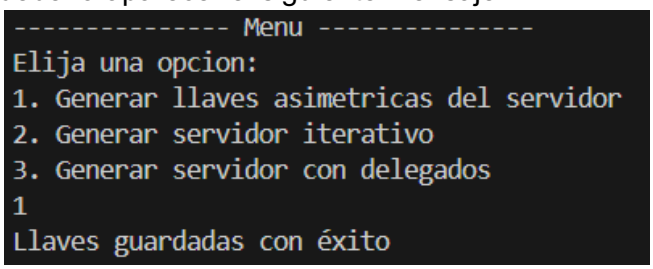
El código se corre desde la clase Main. Al ejecutarlo, lo primero que aparece es el siguiente mensaje



Aquí el usuario debe ingresar la ruta completa de la carpeta que contiene la aplicación de OpenSSL. Por ejemplo:
`C:\Users\majoa\Downloads\OpenSSL-1.1.1h_win32\OpenSSL-1.1.1h_win32`
Después de ingresar la ruta, se muestra el menú de la aplicación.



Lo primero que debería hacer es ejecutar la opción 1. Al seleccionar esta opción se generan las llaves simétricas del servidor. Estas llaves quedan guardadas en la ruta `src\Llaves`. En el archivo `LlavePrivadaServidor.txt` queda guardada la llave privada del servidor y en el archivo `LlavePublicaServidor.txt` queda guardada la llave privada de este. Si todo se generó correctamente debería aparecer el siguiente mensaje



Es importante recalcar que no se pueden ejecutar las opciones 2 y 3 del menú si primero no se ejecutó la opción 1. Si el usuario intenta ejecutar la opción 2 o 3 sin haber ejecutado la primera, el programa muestra el siguiente mensaje

```
--------------- Menu ---------------
Elija una opcion:
1. Generar llaves asimetricas del servidor
2. Generar servidor iterativo
3. Generar servidor con delegados
2
Primero debe generar las llaves
```

Una vez generadas las llaves, vuelve a aparecer el mismo menú y el usuario tiene la opción entre generar un servidor iterativo o un servidor con delegados.

Para iniciar el **servidor iterativo**, el usuario debe elegir la opción 2. A continuación se mostrarán las siguiente opciones

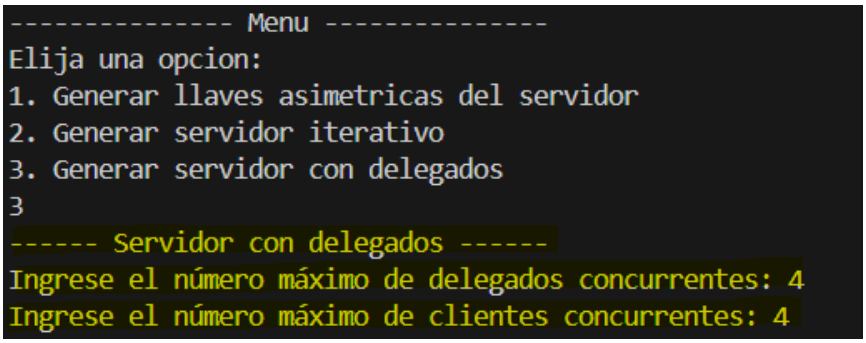
```
--------------- Menu ---------------
Elija una opcion:
1. Generar llaves asimetricas del servidor
2. Generar servidor iterativo
3. Generar servidor con delegados
2
--------------- Servidor iterativo ---------------
1. Cifrado del paquete con llave simetrica
2. Cifrado del paquete con llave asimetrica
```

El usuario debe **ingresar el número** de la opción que desea. Una vez realizada esta acción el programa creará el servidor y el cliente, que automáticamente realizará las 32 consultas de manera iterativa. Cada consulta se imprime de la siguiente manera:

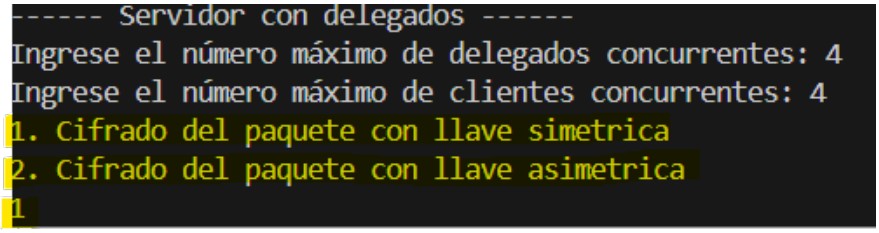
```
============ Ejecutando consulta #1 ============
C: SECINIT enviado
S: palabra de inicio recibida: SECINIT
C: R enviado
S: Descifrando R...
S: Envío Rta
C: Verificando Rta...
C: OK
S: Recibido: OK
S: Generando G, P y Gx...
S: Envío G, P y Gx
C: Verificando G, P y Gx...
C: Diffie-Hellman OK
S: Envío iv
C: Envío id cliente
C: Envío id paquete
S: Estado del paquete: RECOGIDO
C: Estado del paquete recibido. Estado: RECOGIDO
C: TERMINAR
```

S corresponde a un mensaje que envía el servidor y C a un mensaje que envía el cliente. Lo mismo se muestra con la opción de cifrado asimétrico del paquete.

Para el servidor concurrente, la dinámica es similar. Al presionar la opción 3 del menú se muestra lo siguiente:



Aquí el usuario debe colocar el número de delegados y de cliente que desee crear. En el ejemplo de la imagen, se crearon 4 clientes con 4 delegados. A continuación el usuario debe seleccionar si quiere cifrar el paquete con llave simétrica (opción 1) o con llave asimétrica (opción 2), como se muestra a continuación



En el ejemplo de la imágen, se usó el cifrado del paquete con la llave simétrica.
Una vez insertada la opción, el programa creará el número de clientes y delegados elegidos y comenzará a ejecutar el protocolo de comunicación por cada consulta.

Cabe resaltar que los tiempos no son impresos por consola en ningún caso. Estos quedan almacenados en dos archivos: `resultados.txt`, que almacena los tiempos de las consultas hechas en el caso concurrente y `resultadosIterativo.txt`, que almacena los tiempos de las consultas hechas en el caso iterativo.

3. Tablas de datos

Cifrado simetrico

Servidor iterativo

Cifrado simétrico				
# Consulta	Tiempo para descifrar el reto (ms)	Tiempo para generar G, P y Gx (ms)	Tiempo para verificar consulta (ms)	Tiempo para cifrar paquete (ms)
1	4.3371	2604.588	15.0045	0.2742
2	1.5609	1061.0985	3.6354	0.2538
3	0.6361	12678.7208	4.0303	0.3494

4	0.514	3273.8284	4.0985	0.3734
5	1.4677	2435.8733	2.3673	0.2994
6	0.5654	3627.2134	2.6744	0.2587
7	1.5336	5066.1053	2.8696	0.1996
8	1.3754	4853.3202	6.5445	0.6047
9	1.5844	1327.1199	3.3843	0.3159
10	1.3808	15408.7863	11.1146	1.4885
11	2.3582	696.9509	6.8391	1.8586
12	1.1215	10826.1362	2.1447	0.1548
13	1.6665	25121.1766	6.048	0.167
14	2.0949	513.9569	2.1587	0.2007
15	1.9971	6080.5431	1.9137	0.1976
16	0.5661	560.7165	1.5591	0.1068
17	0.9126	10563.5651	2.2405	0.1908
18	0.9359	4754.0869	1.6877	0.1283
19	1.3694	14103.8291	1.489	0.1347
20	1.2421	979.7334	1.7173	0.1572
21	0.5329	1303.855	1.9436	0.1498
22	1.2794	2106.0178	1.9842	0.1168
23	1.2099	3985.5967	1.5629	0.1104
24	1.3683	2683.801	2.09	0.1532
25	2.1208	4354.5913	1.7925	0.2028
26	1.2816	3310.5519	2.0923	0.2208
27	1.3151	976.0754	1.3791	0.0854
28	0.6963	7470.9872	1.4502	0.1097
29	1.9013	5989.4109	1.534	0.1251
30	0.7372	2950.2174	3.571	0.1705
31	1.2746	14016.1644	2.0851	0.1691
32	0.959	161.1042	1.5998	0.1568

Tiempo promedio para descifrar el reto (ms)	Tiempo promedio para generar G, P y Gx (ms)	Tiempo promedio para verificar consulta (ms)	Tiempo promedio para cifrar paquete (ms)
1.371753125	5495.178813	3.331434375	0.296390625

Servidor concurrente

Cifrado simétrico - 4 delegados				
# Consulta	Tiempo para descifrar el reto (ms)	Tiempo para generar G, P y Gx (ms)	Tiempo para verificar consulta (ms)	Tiempo para cifrar paquete (ms)
1	7.1341	346.54	18.1588	0.3167
2	7.3797	1284.4946	2.5536	0.02451
3	7.2543	1236.8588	2.5624	0.2803
4	7.4637	4075.4608	2.6496	0.2539

Cifrado simétrico - 8 delegados				
# Consulta	Tiempo para descifrar el reto (ms)	Tiempo para generar G, P y Gx (ms)	Tiempo para verificar consulta (ms)	Tiempo para cifrar paquete (ms)
1	6.4356	346.54	21.5834	0.3251
2	4.9064	1284.4946	17.9987	0.1933
3	6.7526	1236.8588	29.2717	0.3349
4	7.0434	4075.4608	21.0928	0.2243
5	7.699	3913.6156	23.7339	2.3019
6	9.6415	382.5876	36.7464	1.066
7	8.6089	6515.781	19.573	0.3404
8	5.2497	15921.0654	24.9856	0.3654

Cifrado simétrico - 32 delegados				
# Consulta	Tiempo para descifrar el reto (ms)	Tiempo para generar G, P y Gx (ms)	Tiempo para verificar consulta (ms)	Tiempo para cifrar paquete (ms)
1	7.0622	872.4704	19.3424	0.2701

2	1.0603	1609.9561	55.5905	0.1977
3	9.819	2251.7455	2.8838	0.1972
4	1.2653	3470.4181	3.3311	0.2002
5	0.8461	4625.8179	1.818	0.1858
6	0.9599	6526.2189	1.939	0.2054
7	13.1076	7542.4526	3.1512	0.3161
8	1.1676	7803.7836	3.2117	0.3902
9	10.3366	8393.1104	3.9647	0.397
10	3.9586	10929.6533	3.5287	0.3081
11	1.0445	12842.1754	4.869	2.3242
12	1.73	15950.9553	3.5336	0.3724
13	8.6636	17817.1949	4.6802	0.3427
14	7.3632	18927.0803	3.0254	0.2618
15	1.1199	20345.7306	2.8829	0.2512
16	5.387	20618.262	3.0945	0.2715
17	10.0033	33648.6347	2.7884	0.2433
18	12.0813	35163.969	2.5826	0.2275
19	1.2113	38618.2074	2.8605	0.2922
20	1.8909	38877.5437	2.4979	0.2325
21	3.9463	39625.291	2.0467	0.2057
22	6.5316	39815.4967	3.0949	0.2332
23	3.7429	40902.7346	2.3885	0.2217
24	1.4151	41537.9876	2.3193	0.1953
25	12.3823	44946.8279	2.1101	0.2038
26	1.452	45981.7359	2.4669	0.243
27	14.9025	46534.0085	2.0365	0.1759
28	0.9154	47806.7581	2.4712	0.209
29	0.9643	48287.0591	2.1075	0.1951
30	8.4527	53093.0464	2.0121	0.1722
31	6.2758	54764.5867	1.6393	0.1225
32	13.8829	55374.9346	2.1929	0.1745

Tiempo promedio para descifrar el reto (ms)	Tiempo promedio para generar G, P y Gx (ms)	Tiempo promedio para verificar consulta (ms)	Tiempo promedio para cifrar paquete (ms)
5.4669375	27047.05773	4.9519375	0.30746875

Cifrado asimétrico

Servidor iterativo

Cifrado asimétrico				
# Consulta	Tiempo para descifrar el reto (ms)	Tiempo para generar G, P y Gx	Tiempo para verificar consulta (ms)	Tiempo para cifrar paquete (ms)
1	2.9241	891.6319	0.2374	0.2374
2	1.777	3314.5202	2.4381	0.1511
3	0.16935	5115.2905	2.7339	0.2026
4	0.8578	5898.0532	3.8036	0.2858
5	2.2524	4956.3845	3.4045	0.2158
6	0.729	27307.2748	3.0235	0.2169
7	1.9298	5329.2128	2.1517	0.1815
8	1.6474	4695.9061	2.5831	0.1646
9	0.3679	14921.1046	2.8016	0.2334
10	1.3763	1858.8053	2.3942	0.1978
11	0.6816	836.1784	3.0048	0.2131
12	1.4421	5114.4179	1.9034	0.1231
13	1.0735	189.6079	2.3281	0.1479
14	0.7577	1401.8458	1.7068	0.1201
15	1.3185	7947.0251	2.2459	0.1862
16	0.7032	4952.0927	4.0804	0.2446
17	0.9541	894.4924	4.0639	0.1872
18	0.9163	7817.4432	2.4944	0.1993
19	0.9994	3912.9377	1.6518	0.1294
20	1.3451	1919.2619	1.89	0.1496
21	1.2057	2455.7071	1.6392	0.1284
22	0.7207	2073.2494	1.9346	0.1117

23	1.2387	10022.4674	2.526	0.191
24	1.2949	593.0785	1.6	0.0981
25	1.5013	5575.8335	1.5159	0.0877
26	1.2183	5559.2291	1.6045	0.1373
27	1.3379	1844.5992	2.431	0.213
28	2.0442	6425.5037	2.352	0.1817
29	0.8694	6042.1623	1.6911	0.1244
30	1.2873	307.091	1.5157	0.1154
31	1.1994	10079.4939	1.4584	0.1326
32	0.4166	697.4928	1.6111	0.1065

Tiempo promedio para descifrar el reto (ms)	Tiempo promedio para generar G, P y Gx (ms)	Tiempo promedio para verificar consulta (ms)	Tiempo promedio para cifrar paquete (ms)
1.204904688	5029.668588	2.27564375	0.169225

Servidor concurrente

Cifrado asimétrico - 4 delegados				
# Consulta	Tiempo para descifrar el reto (ms)	Tiempo para generar G, P y Gx	Tiempo para verificar consulta (ms)	Tiempo para cifrar paquete (ms)
1	4.851	6231.896579	14.4712	0.236
2	4.9608	11106.5369	4.2332	0.5864
3	4.7313	6421.068562	2.7696	0.1697
4	4.8571	10936.19869	2.1073	0.1604

Cifrado asimétrico - 8 delegados				
# Consulta	Tiempo para descifrar el reto (ms)	Tiempo para generar G, P y Gx	Tiempo para verificar consulta (ms)	Tiempo para cifrar paquete (ms)
1	7.5317	6231.896579	29.469	0.1775
2	7.7278	11106.5369	14.4762	0.387
3	8.2232	6421.068562	19.9519	0.2583

4	8.6407	10936.19869	26.2446	0.1976
5	8.2959	4141.832178	18.317	0.1506
6	9.5008	14822.20582	25.6084	1.3565
7	8.9806	14729.04907	24.4539	2.4755
8	8.7173	9706.899054	29.4974	0.5714

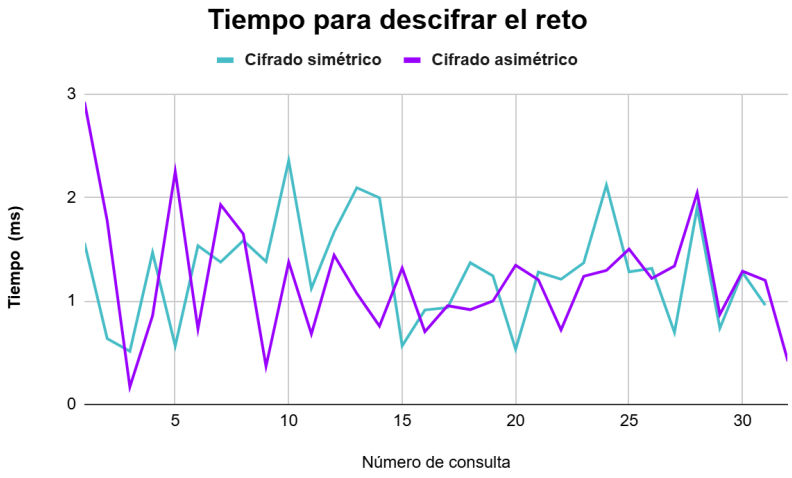
Cifrado asimétrico - 32 delegados				
# Consulta	Tiempo para descifrar el reto (ms)	Tiempo para generar G, P y Gx	Tiempo para verificar consulta (ms)	Tiempo para cifrar paquete (ms)
1	1.4546	1687.5095	12.3711	0.1996
2	1.4182	1830.0348	5.7981	0.2679
3	1.0385	2307.468	3.9106	0.3349
4	1.4854	2982.1829	2.8774	0.247
5	2.5425	4540.7512	2.916	0.9257
6	1.0206	4501.9909	126.4303	0.1874
7	1.4298	5639.7113	1.804	0.1432
8	1.4374	5726.7573	1.9453	0.1548
9	1.389	6961.5546	2.6533	0.1456
10	1.5387	7976.557	6.294	0.194
11	1.1465	10040.1304	4.875	0.3673
12	1.5481	11662.1481	3.9135	0.3083
13	10.5798	13033.8184	3.3755	0.2972
14	6.8536	15727.9985	3.9468	0.4483
15	8.3641	19875.658	4.1419	0.2521
16	1.073	21649.624	3.0988	0.2554
17	9.9743	25225.5384	3.4601	0.2344
18	1.1857	27737.6454	2.9973	0.2484
19	10.9007	32209.9895	2.7053	0.2612
20	8.6764	35756.0429	3.9478	0.2758
21	1.2947	36928.4782	2.5285	0.2496
22	1.0821	37085.2485	3.584	0.2427

23	9.3039	38536.6253	2.7235	0.2776
24	1.1157	39459.4041	2.762	0.2662
25	7.1888	45213.4572	2.9158	0.2948
26	1.0566	45352.0348	2.8196	0.2872
27	1.2848	45722.7792	2.4691	0.241
28	3.0959	46267.5829	2.9529	0.2856
29	1.0062	48329.677	2.5065	0.2234
30	9.8854	53057.1259	2.4735	0.2648
31	1.9556	54113.5495	2.5848	0.2306
32	1.2119	92163.8394	2.6558	0.2381

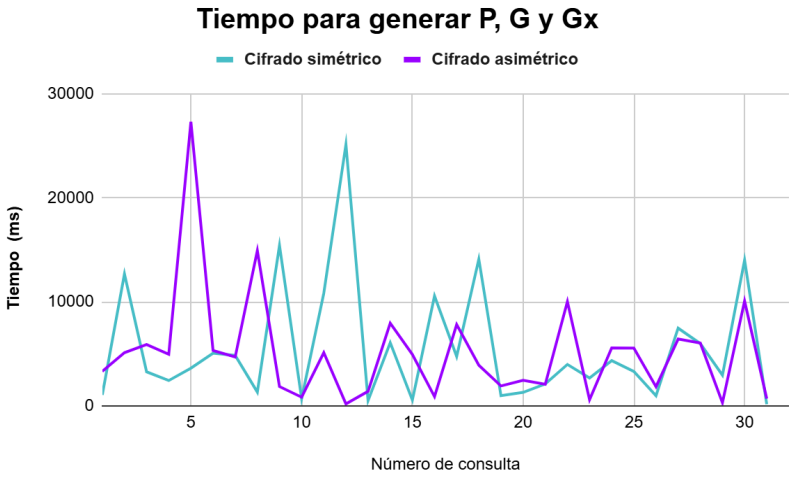
Tiempo promedio para descifrar el reto (ms)	Tiempo promedio para generar G, P y Gx (ms)	Tiempo promedio para verificar consulta (ms)	Tiempo promedio para cifrar paquete (ms)
3.579328125	26228.21603	7.388690625	0.276565625

4. Gráficas y análisis de resultados

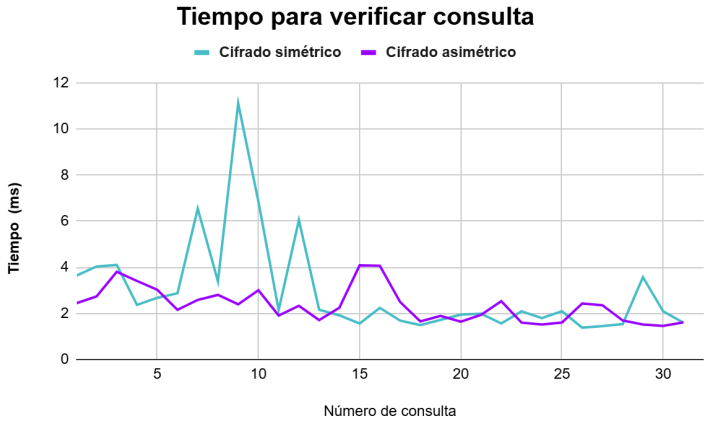
Cifrado simétrico vs cifrado asimétrico en servidor iterativo



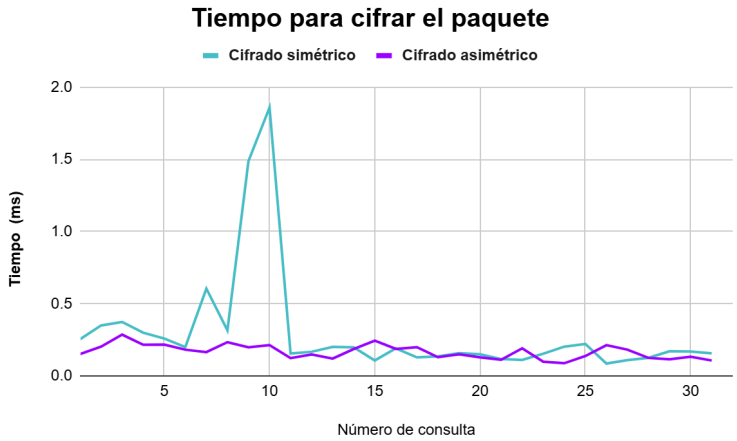
Al observar la gráfica, se puede evidenciar que el tiempo de descifrado con cifrado asimétrico es un poco más constante en comparación con el cifrado simétrico. No obstante, el cifrado asimétrico, presenta un pico muy pronunciado al inicio, indicando un tiempo alto en la primera consulta. Después de este inicio elevado, los tiempos del cifrado asimétrico se estabilizan, aunque siguen variabilidad, al igual que el estado simétrico.



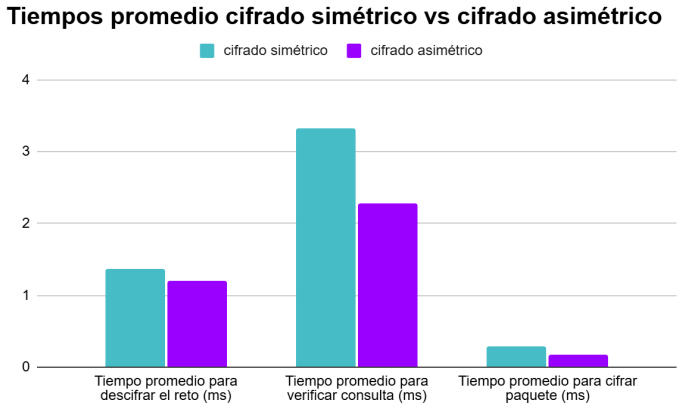
Al observar la gráfica, se puede evidenciar que el tiempo de ambos cifrados, tanto el cifrado simétrico como el asimétrico, presentan una alta variabilidad en los tiempos para generar los parámetros P, G, Gx.



Al observar la gráfica, se observa una comparación entre los tiempos de verificación utilizando cifrado simétrico y asimétrico. El cifrado asimétrico parece ser más rápido en la mayoría de las consultas, ya que las líneas de tiempo de este cifrado están generalmente más bajas que las del cifrado simétrico.



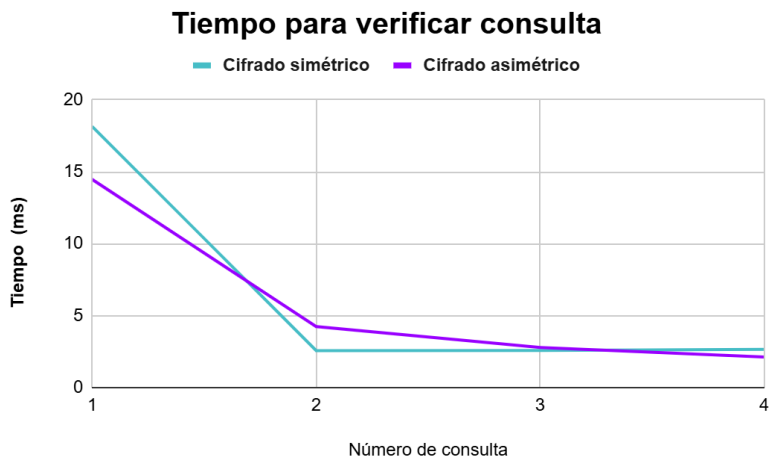
Esta gráfica indica que el cifrado asimétrico tiende a ser más eficiente al realizar el cifrado del paquete mostrando valores de tiempo menores en comparación con el cifrado simétrico.



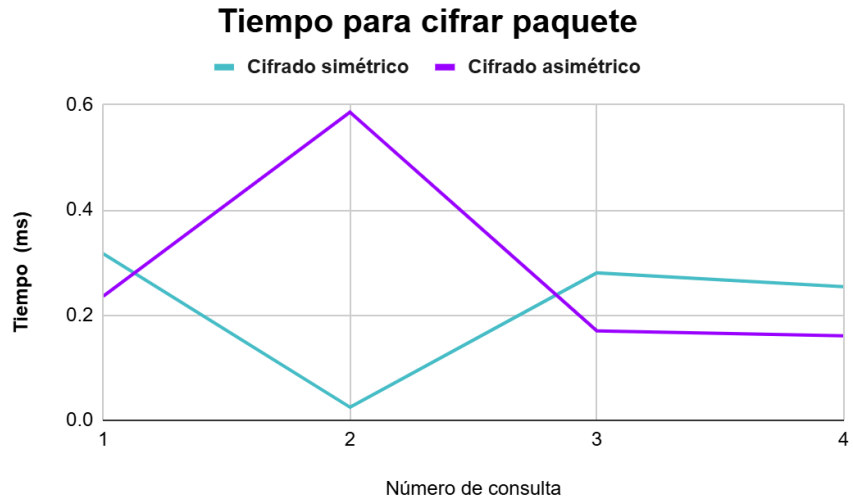
En esta grafica se evidencia que, el cifrado asimétrico tiene menores tiempos promedio en todas las operaciones, especialmente en la verificación de consultas frente al cifrado simetrico.

Cifrado simétrico vs cifrado asimétrico en servidor concurrente

Con 4 delegados

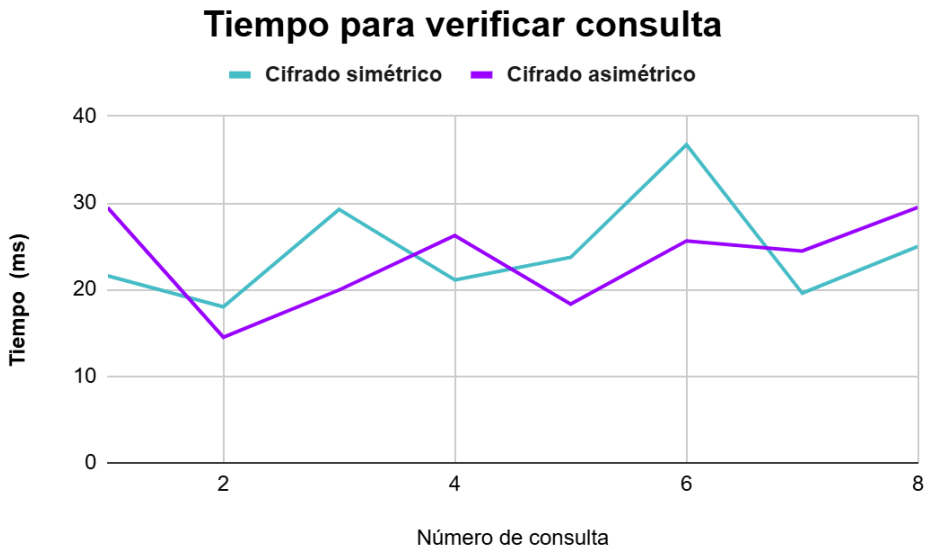


En esta gráfica se muestra una disminución significativa en los tiempos conforme aumenta el número de consultas, tanto para el cifrado simétrico como para el asimétrico. Al inicio, el cifrado simétrico presenta tiempos más altos, pero se estabiliza y reduce.

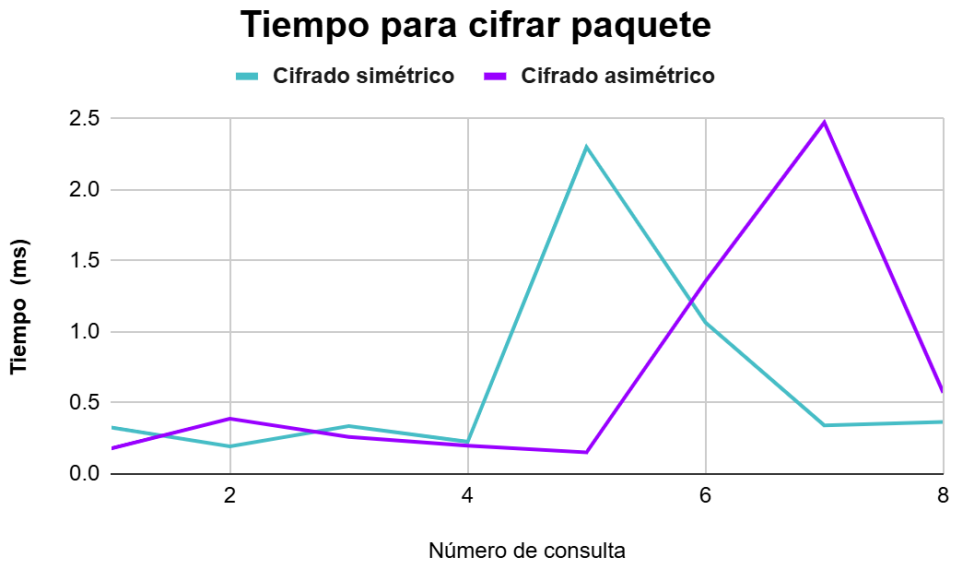


En la anterior gráfica, se observa que hay una mayor variabilidad en en los tiempos del cifrado asimétrico en comparación con el simétrico. No obstante, al final se estabilizan y son uniformes.

Con 8 delegados

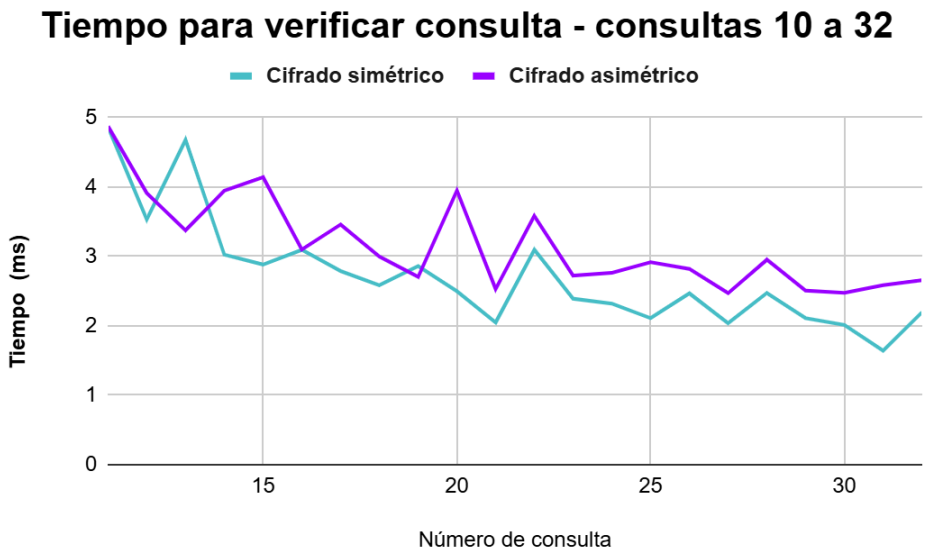


En esta gráfica se evidencia que hay viabilidad entre el cifrado simetrico y el asimetrico frente a la verificación de la consulta.en los tiempos del cifrado asimétrico en comparación con el simétrico. Sin embargo, al final presenta un mayor tiempo el asmétrico.

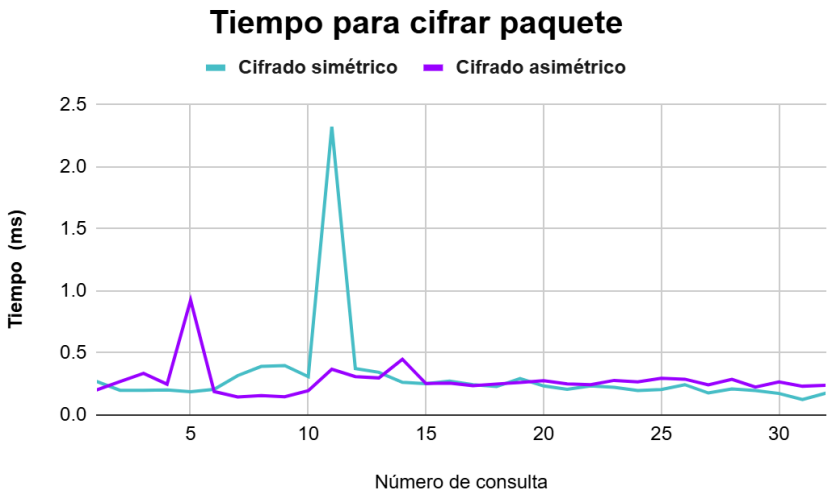


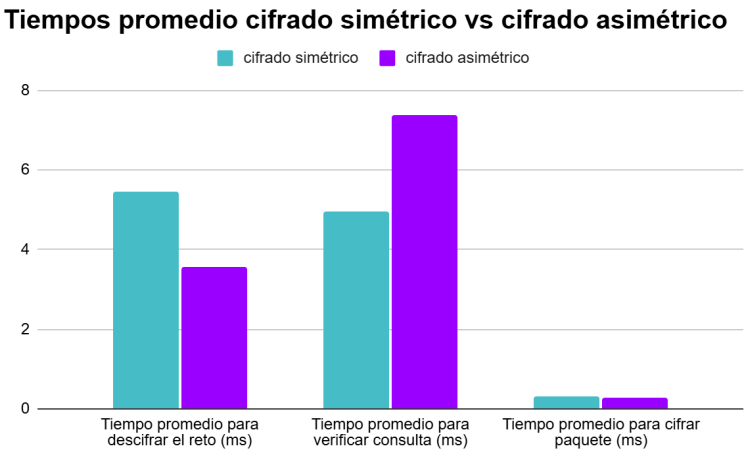
En esta gráfica, se puede evidenciar que para cifrar un paquete ambos cifrados presentan un pico, sin embargo el cifrado muestra un mayor tiempo desde 6 consultas.

Con 32 delegados



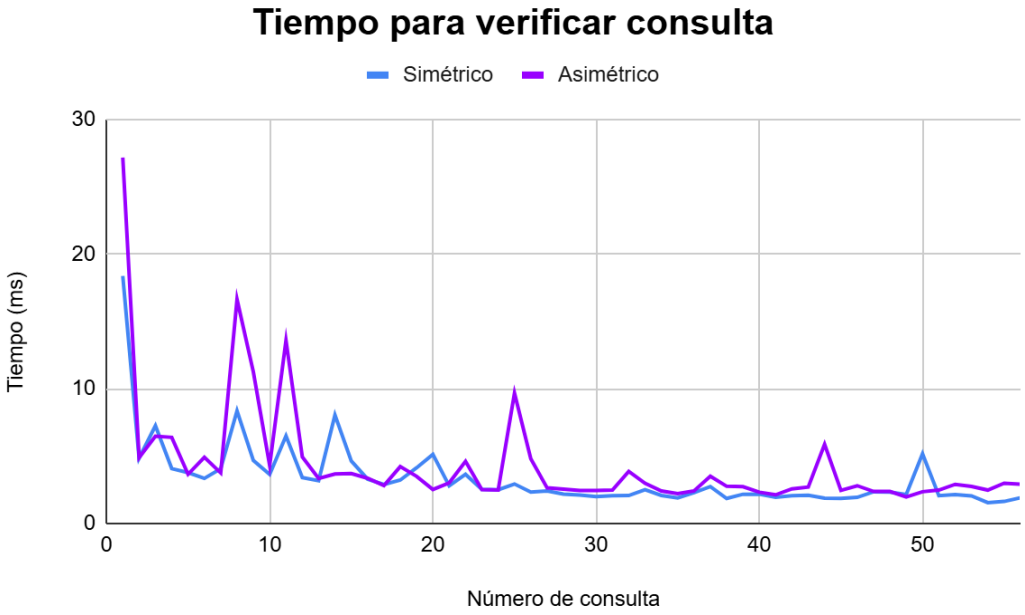
En esta gráfica, se evidencia un pico inicial significativo en el tiempo, especialmente para el cifrado asimétrico alrededor de las 5 consultas. Sin embargo, después de las primeras consultas, ambos tipos de cifrado tienden a estabilizarse, con el cifrado asimétrico generalmente mostrando tiempos mayores o similares al cifrado simétrico.



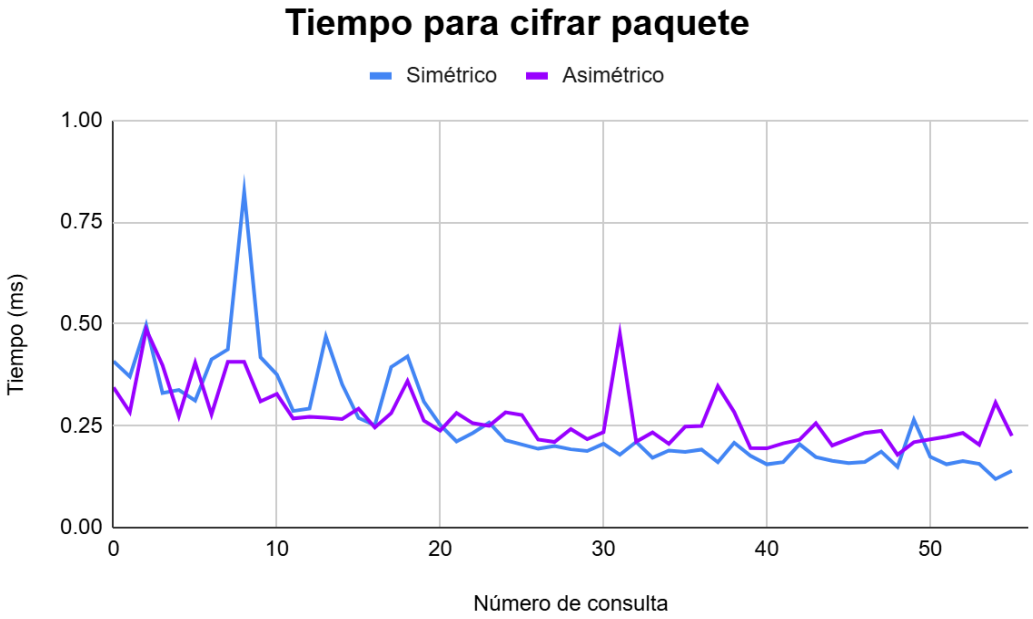


Caso adicional: mayor número de clientes concurrentes

Para este caso, se hizo la prueba con 56 clientes y 56 delegados. A continuación se muestran los resultados

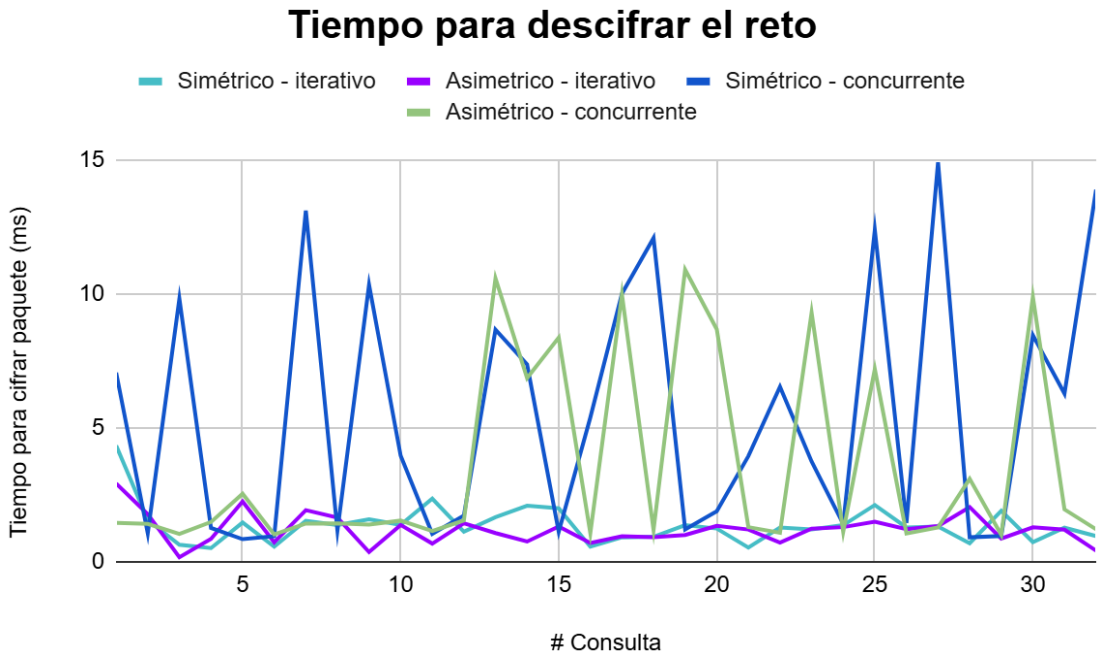


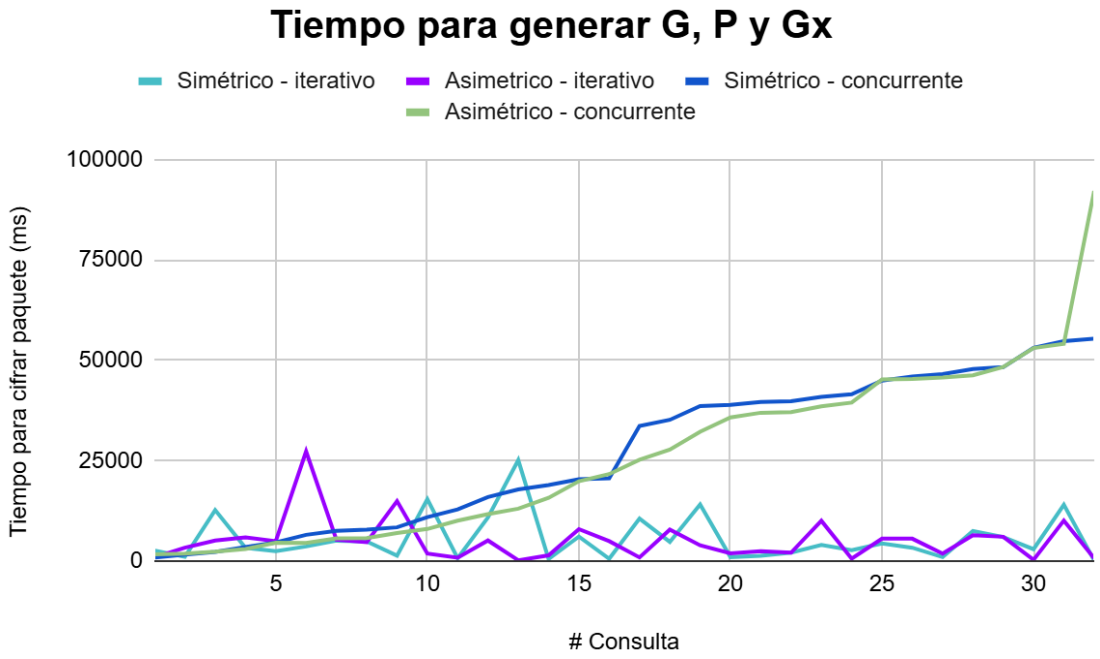
En esta gráfica se evidencia que para verificar una consulta con 56 delegados se presenta un mayor tiempo con el cifrado asimétrico. Asimismo, se puede observar que este cifrado presenta gran cantidad de picos.

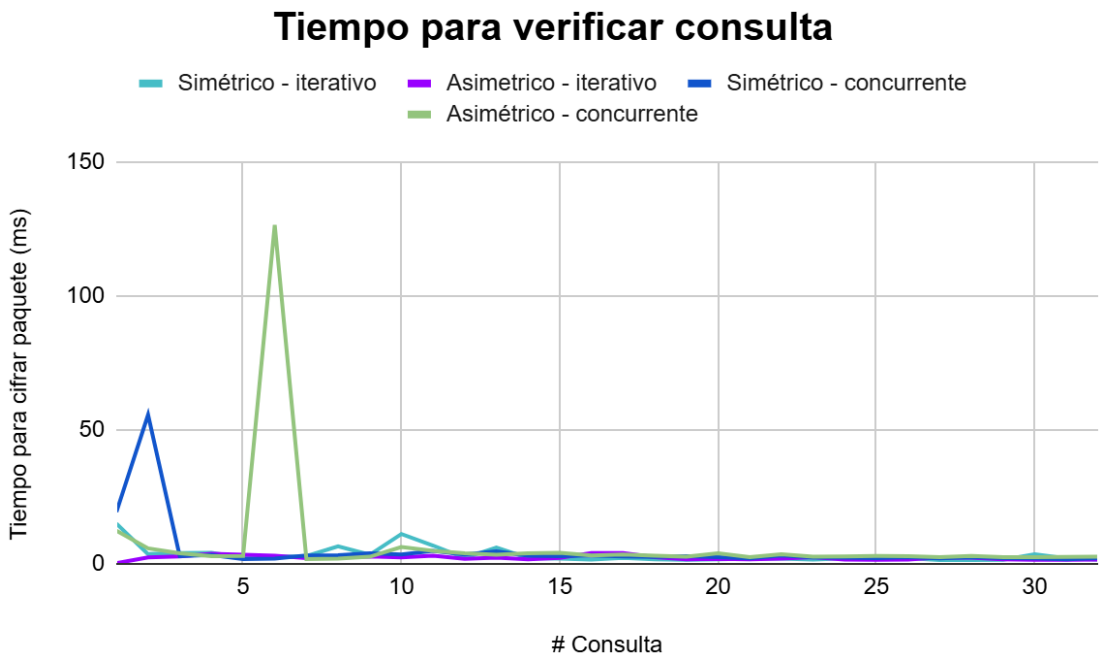


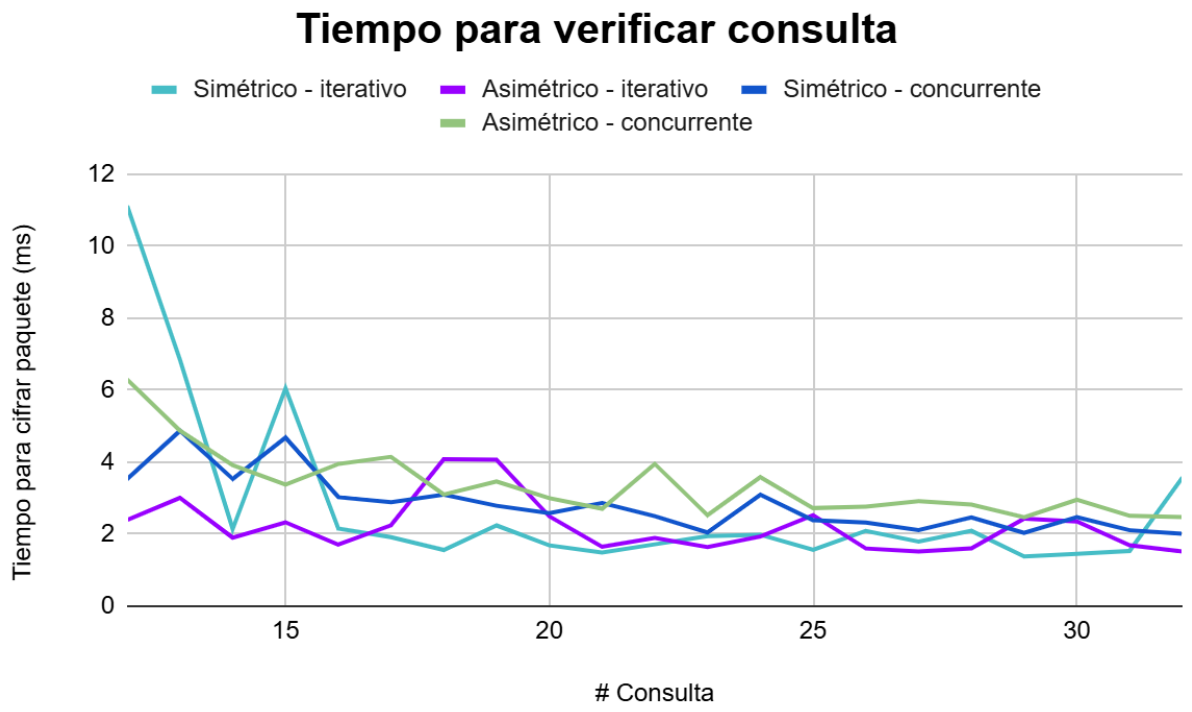
En esta grafica, se puede evidenciar que en la moyor parte de las consultas el cofrado asimétrico presenta un mayor numero. Sin embargo, en las primeras consultas se presentan unos picos del cifrado simetrico.

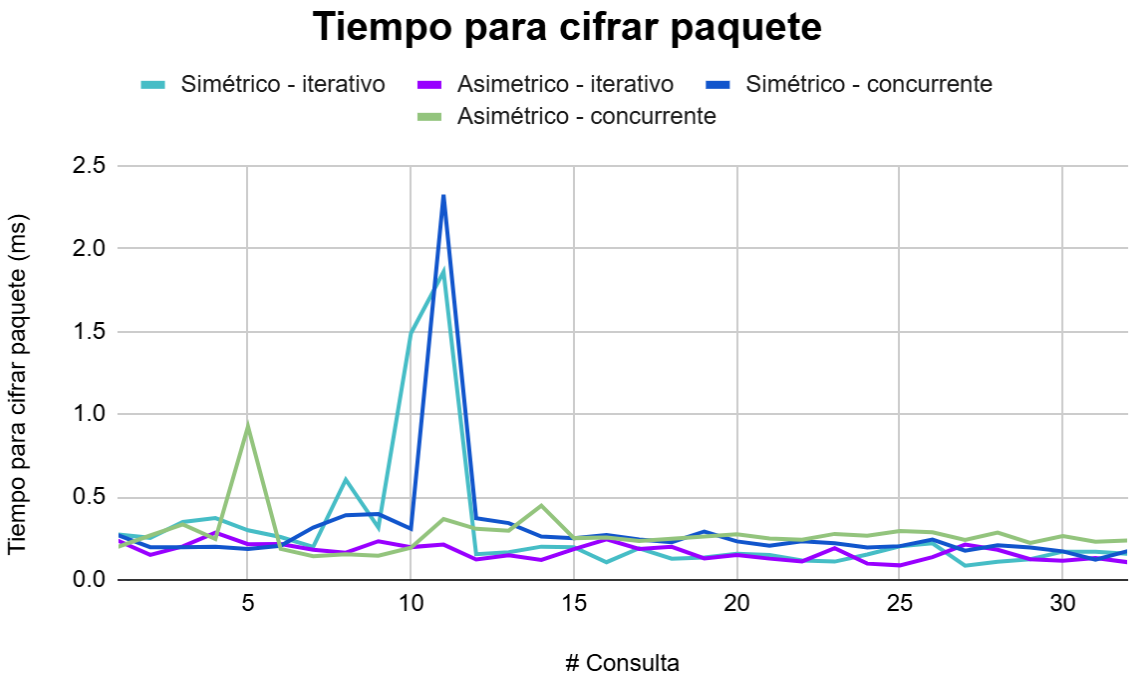
Servidor concurrente vs servidor iterativo

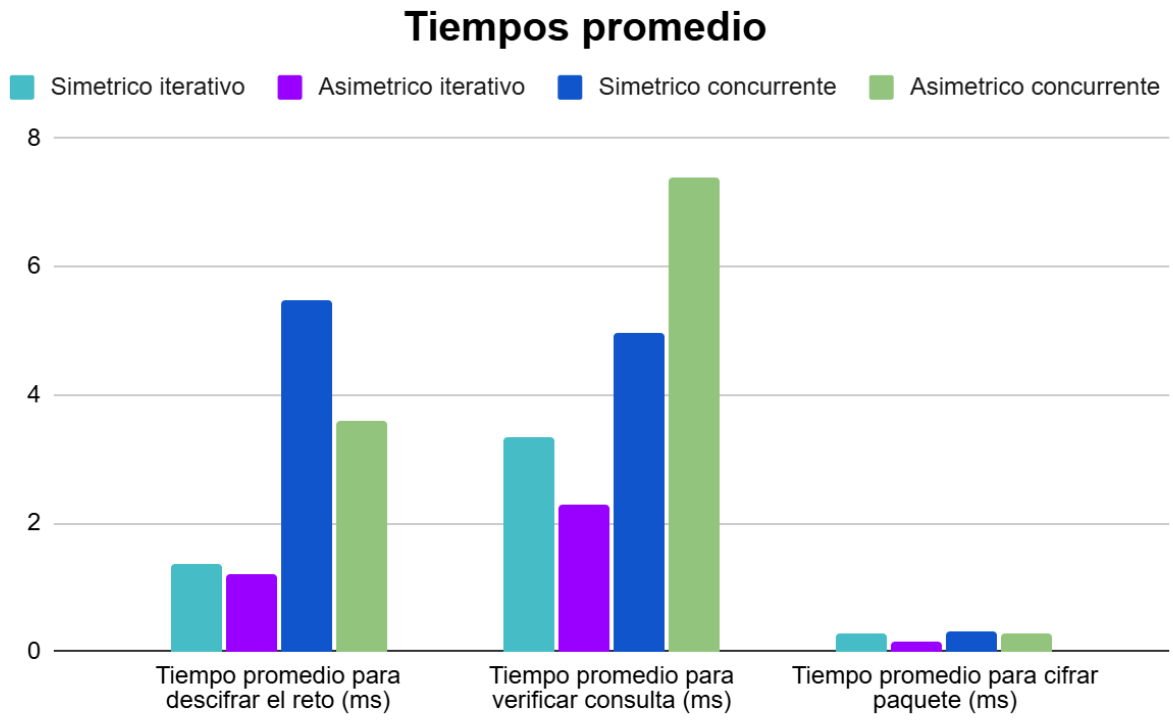





Tiempo para verificar consulta
Simétrico - iterativo
Asimetrico - iterativo
Simétrico - concurrente
Asimétrico - concurrente
Tiempo para cifrar paquete (ms)
150
100
50
0
5
10
15
20
25
30
Consulta


Tiempo para verificar consulta
Simétrico - iterativo
Asimétrico - iterativo
Simétrico - concurrente
Asimétrico - concurrente
Tiempo para cifrar paquete (ms)
12
10
8
6
4
2
0
15
20
25
30
Consulta





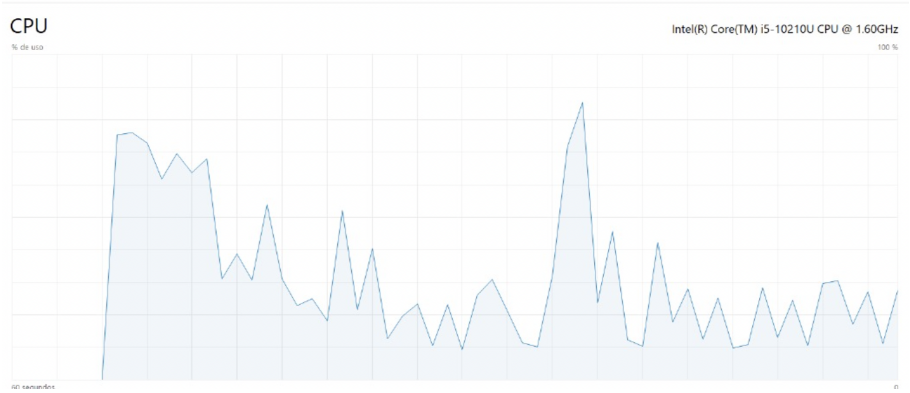
Las gráficas muestran que los métodos iterativos son generalmente más eficientes y rápidos en comparación con los concurrentes en las operaciones como descifrar retos, generar parámetros, verificar consultas y cifrar paquetes. En el caso del cifrado concurrente, tanto en simétrico como en asimétrico, los tiempos de respuesta suelen ser más altos y presentan mayor variabilidad a medida que aumenta el número de consultas, lo cual indica que el enfoque

concurrente introduce una mayor sobrecarga de procesamiento al manejar múltiples consultas en paralelo y asimismo se forman picos con algunas cargas.

5. Respuestas a las preguntas planteadas

Identificar la velocidad del procesador

La velocidad del procesador que utilizamos es de 1.60GHz, que es la frecuencia de reloj del procesador, como se evidencia en la siguiente imagen.



Estimación de cantidad de operaciones por segundo

Caso simétrico

Si se sabe que el algoritmo AES toma 18 ciclos por byte (Wikipedia, 2024) y que las pruebas fueron tomadas con un portátil con procesador Intel(R) Core(TM) i5 de 1.60GHz, entonces puede estimarse cuántos bytes se cifran por segundo. Esto es

$\frac{1.6 \times 10^9 ciclos}{1\ seg} \cdot \frac{1\ byte}{18\ ciclos} = 8.9 \times 10^7$ bytes cifrados por segundo.

Ahora bien, teniendo en cuenta que lo que se cifra es un número (el estado del paquete) representado como String, entonces se puede decir que el tamaño del mensaje es siempre de 1 byte. Debido a que el algoritmo maneja un relleno para alcanzar 16 bytes, entonces el mensaje a cifrar será de 16 bytes (1 de contenido del mensaje original y otros 15 de relleno). Así, si por mensaje hay 16 bytes a cifrar, entonces el número de mensajes que pueden cifrarse por segundo se puede calcular como:

$\frac{8.9 \times 10^7\ bytes}{1\ seg} \cdot \frac{1\ mensaje}{16\ bytes} = 5\,555\,555$ mensajes por segundo.

De esta forma, el procesador con el que se realizaron las pruebas, en teoría, podría cifrar aproximadamente 5 555 555 mensajes por segundo.

Además, con las pruebas hechas, también puede estimarse la cantidad de mensajes que pueden ser cifrados en 1 segundo. Para esto, se calcula un valor aproximado del promedio de todos los tiempos que se demora el equipo para realizar una operación de cifrado simétrico.

Este promedio corresponde a la suma del tiempo promedio que se demora en cifrar el estado del paquete con la llave simétrica en el caso iterativo y el tiempo promedio que se demora en cifrar el estado del paquete con la llave simétrica en el caso concurrente, dividido entre 2. De esta manera:

$$Promedio\ simétrico\ \approx \frac{0.296390625\ ms\ +\ 0.296390625\ ms}{2} \approx\ 0.3019296875\ ms$$

Ahora bien, sabiendo que el equipo es capaz de hacer un cifrado simétrico en aproximadamente 0.3019 ms entonces, el número de operaciones de cifrado que puede realizar en 1 segundo puede encontrarse mediante la siguiente expresión

$$\frac{1000\ ms}{1\ seg} \cdot \frac{1\ operación}{0.3019296875\ ms} = \frac{1000\ operaciones}{0.3019296875\ seg} \approx 3312.029394\ operaciones/seg$$

Así, se puede decir que el equipo puede ejecutar aproximadamente 3312 operaciones de cifrado simétrico en 1 segundo.

Claramente los resultados obtenidos en los dos métodos son bastante diferentes.

Caso asimétrico

Con las pruebas hechas, puede obtenerse un valor aproximado del promedio de todos los tiempos que se demora el equipo para realizar una operación de cifrado asimétrico.
Este promedio corresponde a la suma del tiempo promedio que se demora en cifrar el estado del paquete con la llave asimétrica en el caso iterativo y el tiempo promedio que se demora en cifrar el estado del paquete con la llave asimétrica en el caso concurrente, dividido entre 2. De esta manera:

$$Promedio\ simétrico\ \approx \frac{0.169225\ ms\ +\ 0.276565625\ ms}{2} \approx\ 0.2228953125\ ms$$

Ahora bien, sabiendo que, en un escenario práctico, el procesador es capaz de hacer un cifrado asimétrico en aproximadamente 0.2229 ms entonces, el número de operaciones de cifrado que puede realizar en 1 segundo puede encontrarse mediante la siguiente expresión

$$\frac{1000\ ms}{1\ seg} \cdot \frac{1\ operación}{0.2228953125\ ms} = \frac{1000\ operaciones}{0.2228953125\ seg} \approx 4486.411081\ operaciones/seg$$

Así, se puede decir que el equipo puede ejecutar aproximadamente 4486 operaciones de cifrado asimétrico en 1 segundo.