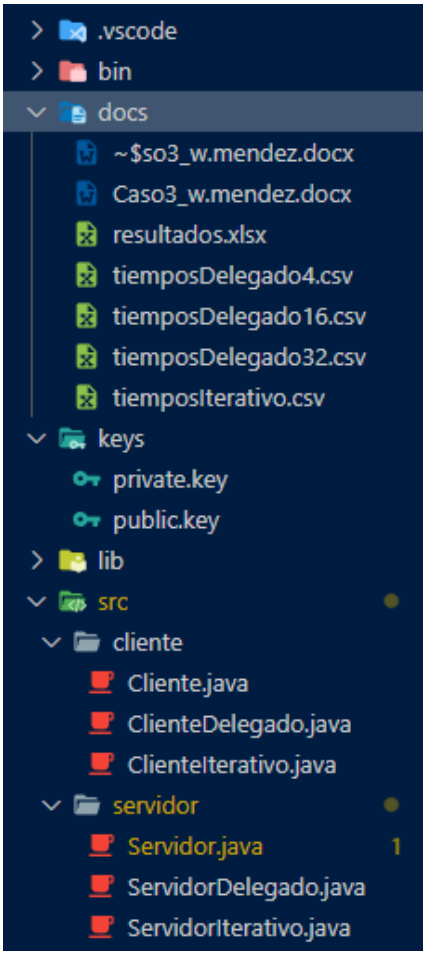
William Mendez – 202012662

Caso 3 Infraestructura Computacional

- Organización del repositorio :



El archivo de la entrega contiene un repositorio con el prototipo del caso, en el tenemos las carpetas:
1. Docs: Se encuentran los documentos con información de resultados y análisis del prototipo. Aquí se encuentra este documento, un archivo de Excel con la tabulación y las gráficas de los resultados para mayor visibilidad y 4 archivos CSV en los que se almacenan los tiempos de cifrado, en estos se muestran dos columnas para el tiempo de cifrar de forma asimétrica y de cifrar de forma simétrica.
2. Keys: Se almacenan dos archivos, private.key y public.key que almacenan la llave privada y pública del servidor, en la aplicación existe la opción de generar las llaves en cada ejecución sin embargo por simplicidad y como no era necesario este siempre usa la misma clave.
3. Src: Se almacenan dos paquetes con el código del proyecto, cada paquete define una de las clases principales, cliente y servidor. En cada paquete existen tres clases de la forma clase, claseDelegado y claseIterativo, las últimas 2 extienden de la primera para simplificar la ejecución, pues en la clase base se encuentran los métodos y atributos necesarios para procesar peticiones y cumplir el protocolo, y en las clases especializadas existe en cada una un método main para una ejecución más intuitiva.
4. Lib, bin y .vscode: Se almacenan archivos necesarios para la ejecución y edición del repositorio.

- Instrucciones para correr el prototipo:

  Como se menciona anteriormente, cada clase especializada contiene un método main que si se ejecuta la aplicación estará orientada a esa clase, por ejemplo, si se ejecuta el main de la clase ServidorDelegado se ejecutará la aplicación en modo servidor delegado, pedirá la cantidad de delegados por consola y abrirá el servidor para consultas de instancias cliente de la aplicación. Así, si se quiere ejecutar un servidor iterativo y realizar consultas como cliente iterativo hace falta abrir dos instancias del prototipo, ejecutar el main del servidor iterativo y el main del cliente iterativo en ese orden, pues si se ejecuta un cliente sin un servidor va a existir una falla.
  Cabe recalcar que las únicas entradas por consola que son necesarias para ejecutar los casos es indicar la cantidad de servidores o clientes delegados al ejecutar el main de ServidorDelegado o ClienteDelegado.

- Descripción del esquema para generación de las llaves:

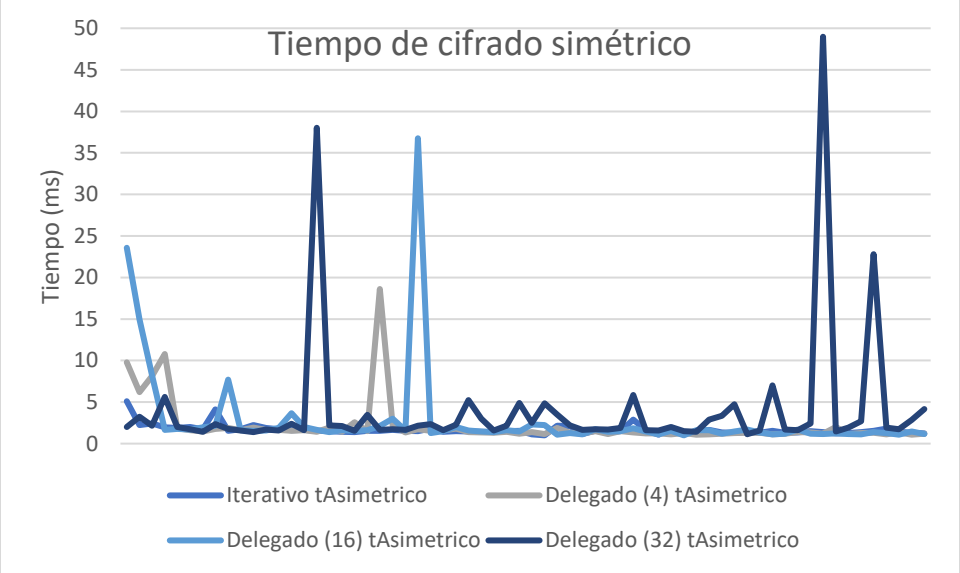
  Para generar las llaves se siguen los métodos planteados en el taller 8 de la clase, así que, para la generación, almacenado y carga de las llaves se usan la librería

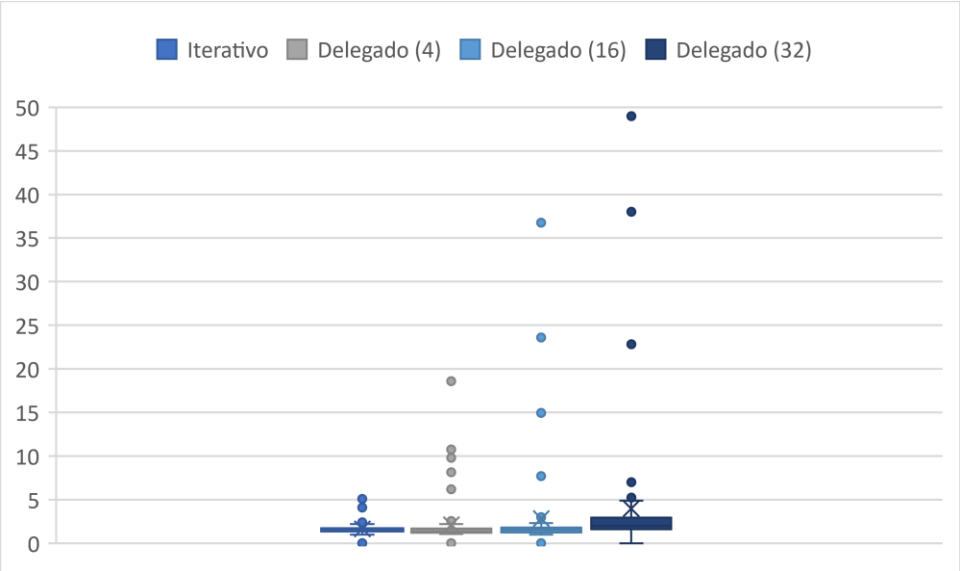
java.security haciendo uso de sus clases Key, PublicKey, PrivateKey, entre otras. Las llaves asimétricas, aunque no se generan con cada ejecución, es posible hacerlo al descomentar los llamados al método generateKey (que se basa en el taller 8) que están en los métodos main de las clases servidor. El almacenamiento y carga de llaves se realiza con la librería security y los métodos SaveKeyPair, LoadKeyPair y getPublicKey que están basados en esta página. Por razones de seguridad, aunque la llave privada y la pública están almacenadas en la misma carpeta, el cliente solo realiza lectura de la pública.

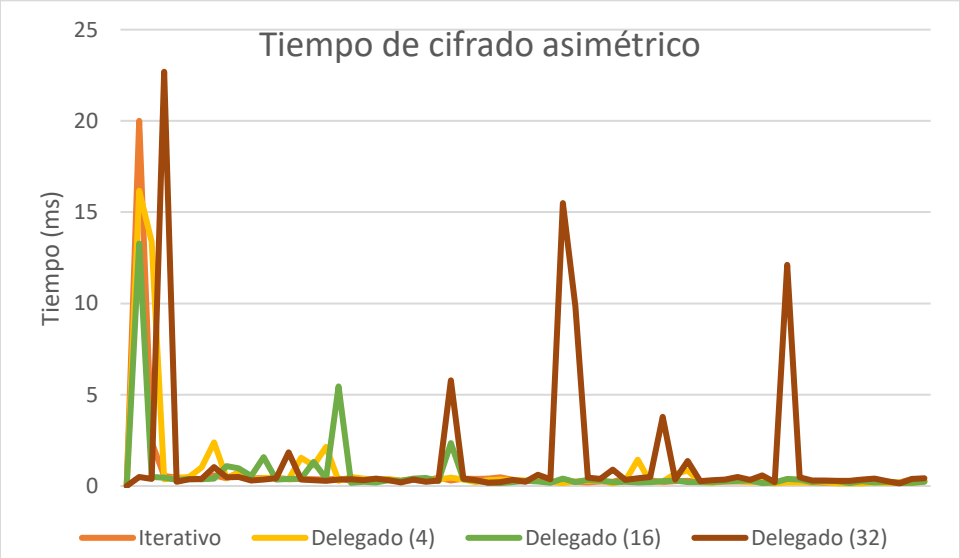
- Desempeño de la aplicación:
  1. Los resultados están en el archivo resultados.xlsx de la carpeta docs, sin embargo, más adelante se pueden ver las gráficas de los resultados.
  2. Nuevamente, la tabla está en el Excel y por su tamaño no se agrega en el documento, sin embargo, aquí están los promedios de tiempo obtenidos:

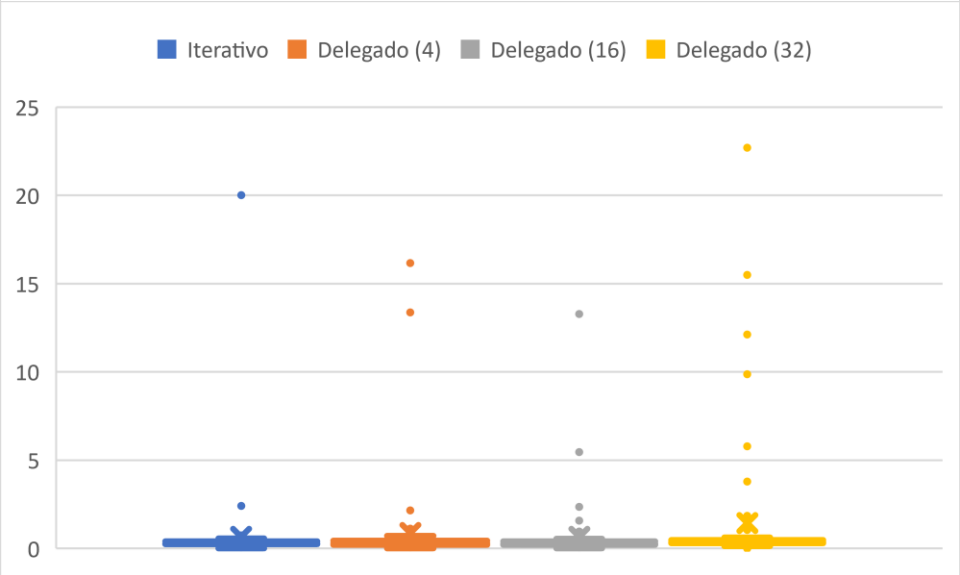
Iterativo tAsimetrico	Iterativo tSimetrico	Delegado (4) tAsimetrico	Delegado (4) tSimetrico	Delegado (16) tAsimetrico	Delegado (16) tSimetrico	Delegado (32) tAsimetrico	Delegado (32) tSimetrico
1,67	**0,66**	**2,17**	**0,88**	**2,87**	**0,66**	**4,05**	**1,47**

3.




Iterativo
Delegado (4)
Delegado (16)
Delegado (32)
50
45
40
35
30
25
20
15
10
5
0


Tiempo de cifrado asimétrico
Tiempo (ms)
25
20
15
10
5
0
Iterativo
Delegado (4)
Delegado (16)
Delegado (32)


Iterativo
Delegado (4)
Delegado (16)
Delegado (32)
25
20
15
10
5
0

4. Lo que se puede notar con los gráficos y la tabla es que en general, para este programa el cifrado simétrico es mucho más eficiente que el cifrado asimétrico. También, podemos notar que entre más servidores delegados se le agrega a la aplicación el rendimiento del cifrado asimétrico es peor, porque los procesos deben competir por el procesador para las operaciones pesadas de comunicación, cifrado y descifrado. Sin embargo, agregar más servidores delegados no afecta considerablemente el rendimiento del cifrado simétrico.
5. El procesador de mi máquina llega máximo a una velocidad de 2.9GHz
   En promedio, cifrar un reto con una llave asimétrica tarda aproximadamente 0,91 ms, esto significa por una regla de tres que en un segundo se pueden cifrar 1090 retos de esta manera.
   Por otro lado, cifrar con una llave asimétrica tarda casi 2,69 ms, por lo que en un segundo se pueden cifrar casi 372 retos.
   Estos cálculos se encuentran también consignados en el Excel debajo de la tabla de datos.

Referencias de la clase:

- Taller 5 – Servidores concurrentes – Java
- Taller 8 – Cifrado y control de integridad

Referencias externas:

- Manejo, almacenamiento y carga de llaves: https://snipplr.com/view/18368/saveload--private-and-public-key-tofrom-a-file
- Medición de tiempo de ejecución: https://javarevisited.blogspot.com/2012/04/how-to-measure-elapsed-execution-time.html#:~:text=There%20are%20two%20ways%20to,calls%20or%20events%20in%20Java.
- Conversión de unidades de tiempo: https://qph.fs.quoracdn.net/main-qimg-0e1f10feeea00b838f351064fd6460c1-lq
- Creación de HMAC y digest: https://www.baeldung.com/java-hmac
- Generación de un número para el reto: https://stackoverflow.com/questions/3709521/how-do-i-generate-a-random-n-digit-integer-in-java-using-the-biginteger-class