**Nombres: Diego Parraga González -202015540**
**Sebastián Cáceres Charari - 2020214172**
**Mariana Ruiz Giraldo - 202011140**
**Fecha: 07/11/2022**
**Carrera: Ingeniería De Sistemas**

# CASO 3 - MANEJO DE SEGURIDAD

**Universidad de los Andes Departamento de Ingeniería de Sistemas y Computación**
**ISIS 2203 Infraestructura Computacional**
**202220**

**Nombres: Diego Parraga González -202015540**
**Sebastián Cáceres Charari - 2020214172**
**Mariana Ruiz Giraldo - 202011140**
**Fecha: 07/11/2022**
**Carrera: Ingeniería De Sistemas**

# Análisis de la realización del caso

## 1. Contexto del Problema

Para poder comenzar con el proceso de diseño, primero fue necesario analizar el diagrama que muestra de manera general el modelo de programación MapReduce.

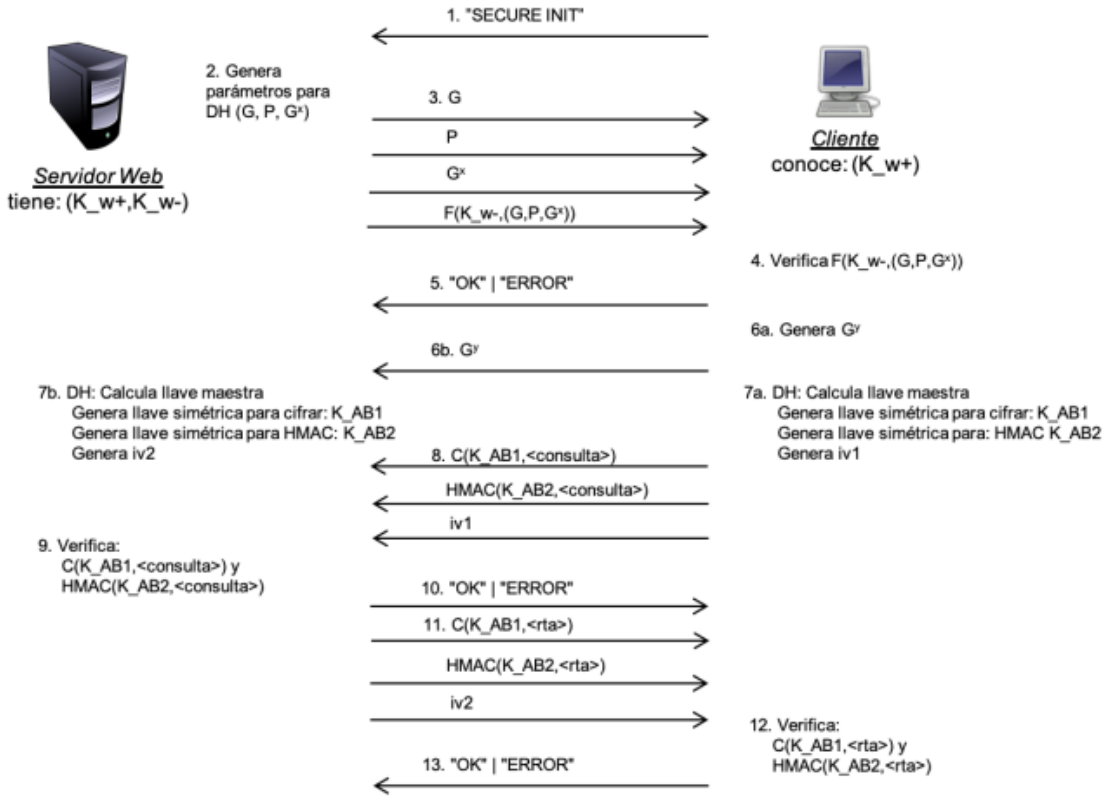


*Figura 1. Protocolo de comunicación entre el servidor y el cliente.*

En este modelo, los clientes pueden conectarse a una red predefinida. De esta forma poder comunicarse con el administrador de servidores. Se debe encriptar cualquier mensaje que viaje a través de la red. Manejando tanto cifrado simétrico como asimétrico según los casos que corresponda para asegurar la confidencialidad, autenticidad e integridad de la información.

El objetivo del caso es diseñar un mecanismo de comunicación para implementar la arquitectura descrita. Para este caso, los clientes serán threads en la misma máquina (Emulan la existencia de varios usuarios enviando solicitudes al servidor). La comunicación siempre pasa a través del puerto predefinido para la comunicación (4030). Por ejemplo, para comunicar a un primer cliente (A) con el servidor (B), A debe enviar un mensaje a través del socket de salida y main server desplegará un server thread (B) con el cual se estará comunicando con A para darle respuesta al mensaje recibido.

## 2. Diseño Detallado de la Solución

Para este proceso, identificamos los roles que podrían ser necesarios para el desarrollo de la solución de una manera general, es necesario entrar en detalle y dar más especificaciones sobre los roles, para descomponer correctamente y poder generar un sistema de clases que se relacionen correctamente. A continuación, se muestra un diagrama de clases que permite visualizar de una manera clara, las clases que fueron posible descomponer y como se van relacionando entre ellas.

**Nombres: Diego Parraga González -202015540**
**Sebastián Cáceres Charari - 2020214172**
**Mariana Ruiz Giraldo - 202011140**
**Fecha: 07/11/2022**
**Carrera: Ingeniería De Sistemas**

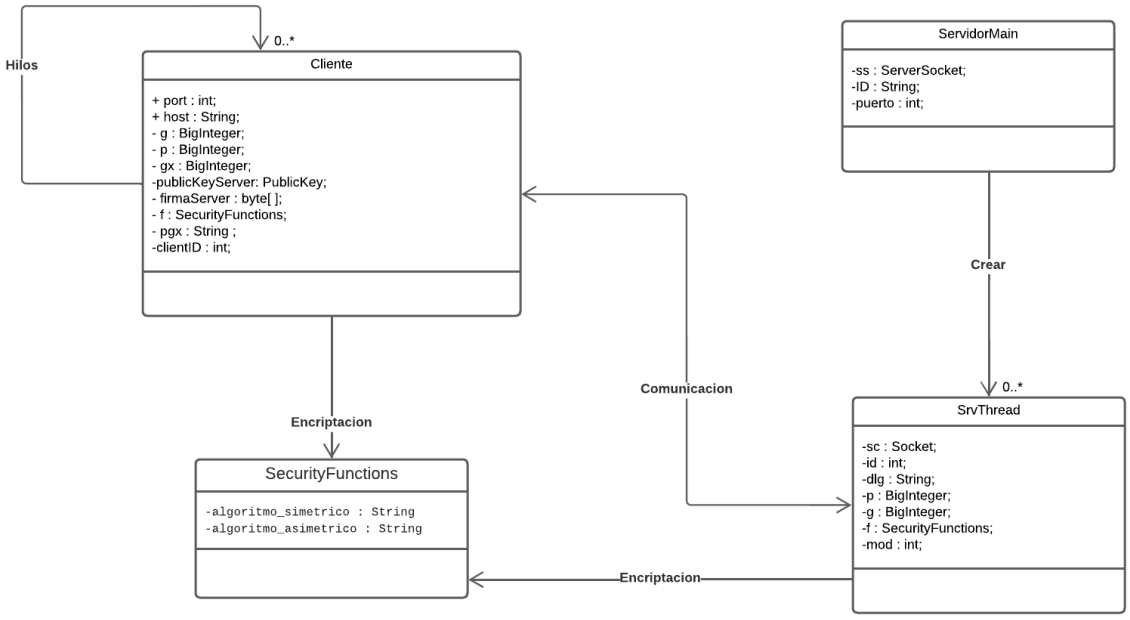


Figura 2. Diagrama de clases

En la figura 2, es posible visualizar 4 clases las cuales se describirán a continuación:

**Clase Main Server:** Es la que inicia el servicio de servidores, se debe inicializar antes que la clase clientes para que cuando lleguen las solicitudes ya se haya establecido el socked Server en ese puerto. Esta clase se encarga de inicializarlos y envía los clientes a unos respectivos asesores (SrvThreads), de forma se tiene una conversación exclusiva entre el cliente y su designado servidor. esto para que se pueda instaurar una llave simétrica privada solo esta comunicación en específico.

**Nombres: Diego Parraga González -202015540**
**Sebastián Cáceres Charari - 2020214172**
**Mariana Ruiz Giraldo - 202011140**
**Fecha: 07/11/2022**
**Carrera: Ingeniería De Sistemas**

**Clase Cliente:** La clase extiende de Thread y a su vez cuenta con su propio main independiente, este inicializa los hilos que funcionaran como clientes en la emulación. Para esto, hace un llamado recursivo a si mismo para crear e iniciar el numero de hilos que el usuario quiera crear. Esta clase se comunicará con un Servidor en el cual intercambiaran mensajes cifrados a través de las funciones de cifrado contenidas en SecurityFunctions. Adicionalmente, comprueba que los mensajes sean lo esperado antes de continuar para garantizar el correcto funcionamiento de la comunicación.

**Clase SrvSThread**: Es la clase que se comunica con la clase Cliente. Esta se encarga de tomar el mensaje del cliente y enviar una respuesta. Los mensajes envidos como recibidos deberán estar cifrados para garantizar los atributos de seguridad. Por lo cual el servidor usará los métodos de la clase SecurityFunctions para encriptar y desencriptar los mensajes como el cliente.

**Clase SecurityFunctions:** Esta clase se encarga de almacenar todos los métodos necesarios para la encriptar archivos, desencriptar, cargar las llaves privadas y públicas; y crear llaves simétricas.

**Nombres: Diego Parraga González -202015540**
**Sebastián Cáceres Charari - 2020214172**
**Mariana Ruiz Giraldo - 202011140**
**Fecha: 07/11/2022**
**Carrera: Ingeniería De Sistemas**

## 3. Funcionamiento de la solución

La clase ServidorMain se debe correr primero para crear el socket server que va a recibir las solicitudes. Seguido a esto, se debe correr la clase Cliente que será la encargada de enviar las solicitudes a el puerto que es una constante predefinida. Allí se preguntará cuantos Clientes desea atender y el servidor se encarga de generar Threads delegados para atender a los clientes. Al iniciar el servidor se cargan las llaves públicas y privadas que se encuentran adjuntas en la carpeta. Todos los clientes tienen acceso a la llave pública del servidor.

## 4. Preguntas:

### 4.1 Responda las siguientes preguntas:

#### 4.1.1 En el protocolo descrito el cliente conoce la llave pública del servidor (K_w+). ¿Cuál es la manera común de enviar estas llaves para comunicaciones con servidores web?

Las llaves públicas de los servidores web son conocidas por los clientes por medio de los certificados digitales. Un certificado digital es un archivo de datos que se aloja en el servidor web y que es firmado por una entidad certificadora, que garantiza la identidad del servidor. Cuando un cliente desea comunicarse con el servidor, este debe referenciar este archivo para obtener la llave pública y verificar la identidad del servidor. Este mecanismo es utilizado en HTTPS para asegurar la autenticación del servidor.

#### 4.1.2 El protocolo Diffie-Hellman garantiza “Forward Secrecy”, explique en qué consiste esta garantía

Forward Secrecy es una propiedad del protocolo Diffie-Hellman que garantiza que la llave utilizada para el intercambio de información en una sesión no se verá comprometida aún si la llave privada del servidor a largo plazo se ve comprometida en el futuro. Diffie-Hellman logra esto al no utilizar únicamente la llave privada para la

generación de las llaves de sesión. Además, las llaves de sesión se generan por medio de diferentes métodos.

### 4.2 Corra su programa en diferentes escenarios y mida los tiempos que el cliente demora para: Cifrar la consulta, genere el código de autenticación, verificación de la firma, calcular $G^y$.

El programa se corrió mínimo 3 veces para cada escenario y se hizo un promedio con todos los valores recopilados.

| Clientes | Tiempo (ns) | | | | |
|---|---|---|---|---|---|
| | Cifrar Consulta | Codigo MAC | Verificación de Firma | Calcular G^y | Total |
| 1 | 63422633 | 943600 | 15239267 | 5390300 | 84995800 |
| 4 | 463246600 | 5381567 | 23727750 | 20342167 | 512698083 |
| 16 | 9357985800 | 46607400 | 29248300 | 339118200 | 9772959700 |
| 32 | 18182754900 | 208483600 | 46601200 | 456674400 | 18894514100 |

Figura 3. Tiempos promediados de ejecución

En la figura 3 es posible observar los resultados de las ejecuciones en donde se promediaban los tiempos en aquellos casos en donde no era posible completar todo el proceso para completar el tiempo que habrían tardado estos procesos si se hubieran realizado. Es decir, en aquellos casos en los que no se cifraban las consultas o no se generaba el código de autenticación (MAC) porque la verificación de la firma no era correcta.

| Clientes | Tiempo (ns) | | | | |
|---|---|---|---|---|---|
| | Cifrar Consulta | Codigo MAC | Verificación de Firma | Calcular G^y | Total |
| 1 | 63422633 | 943600 | 15239267 | 5390300 | 84995800 |
| 4 | 463246600 | 3342017 | 23727750 | 14431917 | 504748283 |
| 16 | 1559664300 | 7767900 | 29248300 | 56519700 | 1653200200 |
| 32 | 956987100 | 16037200 | 46601200 | 38056200 | 1057681700 |

Figura 4. Tiempos de ejecución sin aproximaciones

En la figura 4 es posible observar los tiempos de ejecución de los procesos sin llevar a cabo aproximaciones para completar aquellos que no se pudieron llevar a cabo. Esto explica por qué en el caso de Cifrar Consulta, el tiempo de ejecución para 16 clientes es mayor que el de 32 clientes. En este caso en específico, en el caso de 16 clientes se llegaron a cifrar 11 consultas, mientras que en el caso de 32 se llegaron a cifrar 14 consultas, pero con un tiempo promedio por consulta menor.

### 4.3 Construya una gráfica con los datos de la tabla.

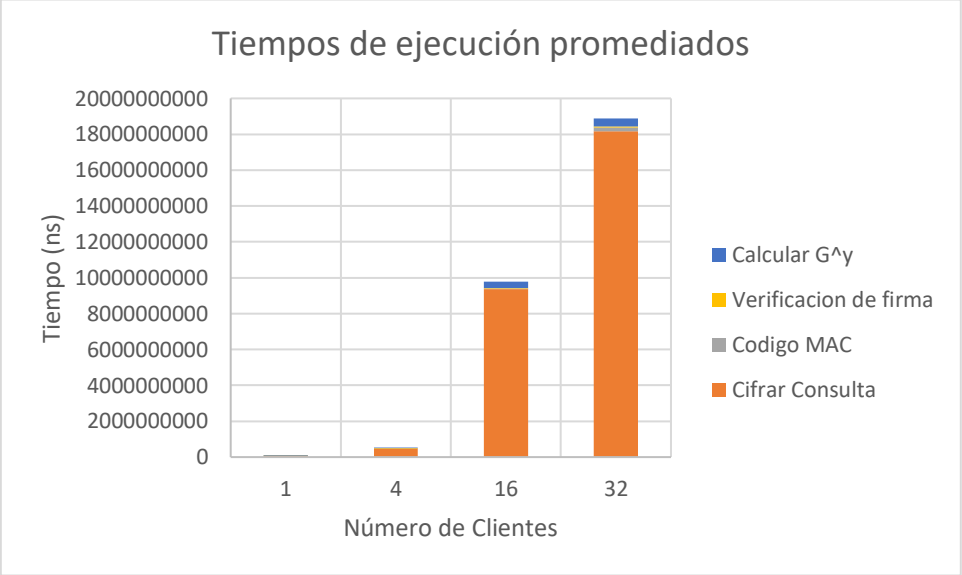


Figura 6. Grafica Tiempos promediados de ejecución

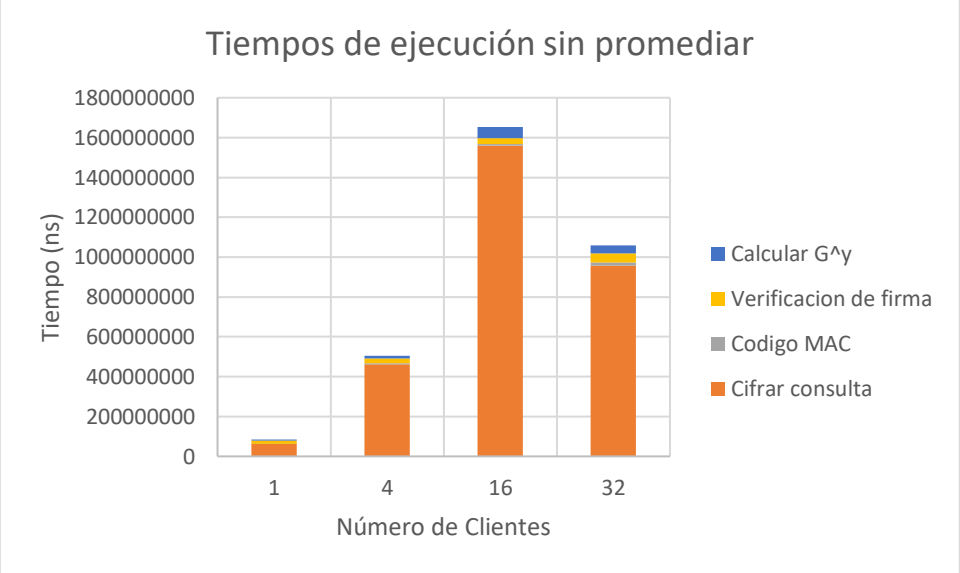


Figura 6. Grafica Tiempos promediados de ejecución

### 4.4 Escriba sus comentarios sobre los resultados.

En las graficas es posible identificar que el tiempo que toma cifrar una consulta es el que más contribuye al tiempo total de todo el proceso, lo cual tiene sentido porque es la función que realiza más operaciones matemáticas. Por otro lado, también es importante notar que el cambio entre los tiempos requeridos para 1 cliente no es muy diferente del de tiempo requerido para atender a 4 clientes, a pesar de ser 4 veces mayor en el caso de los tiempos de ejecución promediados. Sin embargo, esto mismo no ocurre si se compara la diferencia entre 4 y 16 clientes. Esta situación se puede explicar debido a los efectos de la concurrencia. Cuando se tomaba el tiempo antes de iniciar el proceso (sin importar si es Cifrar Consulta, Codigo MAC, Verificación de firma o Calcular G^y) y después se restaba al tiempo en el que se terminó el proceso, el tiempo calculado puede corresponder al tiempo que el procesador se demoró en realizar todo ese proceso más el tiempo en que el hilo estaba en espera. Por eso, asumimos que los tiempos que allí se presentan pueden ser mayores al tiempo real que lleva ejecutar un solo escenario a pesar de que la concurrencia permita que el tiempo total sea menor al tiempo si se atendiera a los clientes consecutivamente.

También es relevante mencionar que, a medida que el servidor iba atendiendo más clientes, los tiempos para cada proceso iban aumentando en cantidades que rondaban el 10% del dato anterior. La

explicación que se le da a este fenómeno radica en el efecto de la concurrencia sobre el procesador, pues se va añadiendo un Thread más por cada cliente.

Por último, cabe resaltar que los resultados presentados aquí pueden variar. Ya que en varias ejecuciones se dio la coincidencia de que menos de la mitad de los clientes pudieron completar sus pedidos satisfactoriamente, lo que afecta los datos en gran medida.

### 4.5 Identifique la velocidad de su procesador, y estime cuántas consultas puede cifrar su máquina, cuántos códigos de autenticación puede calcular y cuántas verificaciones de firma, por segundo. Escriba todos sus cálculos.

Las pruebas se realizaron en un Procesador Intel(R) Core(TM) i3-1005G1 CPU con una velocidad de reloj de 1.20GHz, lo que equivale a 1.2 x 10^9 instrucciones/segundo

En promedio, la máquina cifra 1 consulta en 0,33 s. Entonces, en un segundo se pueden cifrar:

$$\left(\frac{1\ consulta}{0{,}33\ \text{segundos}}\right) = 3\ consultas$$

En promedio, la máquina genera 341 códigos MAC en 0,0029 s. Entonces, en un segundo se pueden generar:

$$\left(\frac{1\ consulta}{0{,}0029\ \text{segundos}}\right) = 341\ codigos\ MAC$$

En promedio, la máquina hace 1 verificación en 0,0061 s. Entonces, en un segundo se pueden hacer:

$$\left(\frac{1\ consulta}{0{,}0061\ \text{segundos}}\right) = 163\ verificaciones$$

En promedio, la máquina puede calcular G^y en 0,0115 s. Entonces, en un segundo se pueden calcular:

$$\left(\frac{1\ consulta}{0{,}0115\ \text{segundos}}\right) = 87\ G\text{^}y$$

Somos conscientes de que en estos cálculos no se utilizó la velocidad de reloj de la máquina. Esto debido a que los valores obtenidos para la realización de cierta consulta ya tienen en cuenta la velocidad de reloj de la máquina y son específicos para la misma. Adicionalmente, queremos recalcar que estos valores se pueden ver afectados por otros procesos y/o aplicaciones que esté corriendo la máquina en el momento de hacer las pruebas, lo que influye en la cantidad de consultas que el procesador hizo por minuto.

Los cálculos y las gráficas presentadas en el informe pueden visualizarse con más detenimiento en el Excel que se incluye en el subdirectorio docs del zip.

## 5. Referencias

Java implementation of Diffie-Hellman algorithm between client and server. (2021, November 10). Recuperado el 5 de noviembre de 2022, de https://www.geeksforgeeks.org/java-implementation-of-diffie-hellman-algorithm-between-client-and-server/

What is an SSL certificate? (s/f). Cloudflare.com. Recuperado el 7 de Noviembre de 2022, de https://www.cloudflare.com/learning/ssl/what-is-an-ssl-certificate/

O'Brien, K. (2018, Enero 10). Diffie-Hellman and forward secrecy. Zwilnik.com. Recuperado el 7 de Noviembre de 2022, https://www.zwilnik.com/security-and-privacy/diffie-hellman-and-forward-secrecy/