Escuela Politécnica Superior

Diseño y desarrollo de un sistema de comunicación grupal seguro

Máster Universitario en Ingeniería Informática

Trabajo Fin de Máster

Autor:
Víctor García Ruiz

Tutor/es:
Rafael Ignacio Álvarez Sánchez

Universitat d'Alacant
Universidad de Alicante

Febrero 2017

Índice general

Índice de figuras

1 Introducción

A continuación, se realizará una introducción al proyecto propuesto, donde se explicarán las características principales del mismo. Además, se indicarán los motivos por los cuales se decidió desarrollar este proyecto y los objetivos que se pretenden alcanzar.

1.1 Proyecto propuesto

El proyecto propuesto se basa en el diseño y posterior desarrollo de un sistema de comunicación que permita realizar conversaciones grupales entre múltiples interlocutores en tiempo real, garantizando la seguridad del servicio y la privacidad de las conversaciones realizadas y demás información confidencial compartida entre los distintos clientes durante el uso del servicio propuesto.

Para el desarrollo de todas las partes de este sistema de comunicación propuesto el lenguaje de programación que se utilizará será Go. Destacando que es un lenguaje que puede ser compilado para todos los sistemas operativos de escritorio con mayor cuota de uso, que soporta potentes patrones de concurrencia y que cuenta con una extensa biblioteca estándar, de la cual muchos paquetes serán de utilidad para el desarrollo de este servicio.

Para tratar de garantizar la seguridad y la privacidad del servicio se implementarán numerosas medidas de protección. Entre esas medidas de protección se incluye el cifrado de las comunicaciones, utilizando para esta finalidad algoritmos criptográficos que sean lo suficientemente robustos y estén considerados actualmente como seguros para su uso en la práctica.

Se ha propuesto que las comunicaciones entre los distintos clientes del servicio se cifren punto a punto, no pudiendo descifrar el contenido confidencial enviado ninguno de los nodos intermediarios de la comunicación, solo pudiendo descifrarlo los receptores finales. Al tratarse de conversaciones grupales en las que pueden llegar a participar cantidades considerables de usuarios, es necesario utilizar un método eficiente para el cifrado punto a punto de los todos mensajes enviados a los participantes de la conversación, debido a que muchos de los métodos utilizados habitualmente para el cifrado de comunicaciones entre pares serían altamente ineficientes para comunicaciones grupales en tiempo real con un número de interlocutores medio o alto. Por lo tanto, tras estudiar el problema se opta por cifrar esas comunicaciones entre los usuarios del grupo mediante una clave de sesión común, que sería previamente compartida mediante el uso de criptografía asimétrica.

Además, al utilizar una topología de red centralizada para el servicio (debido a motivos de estabilidad, seguridad y complejidad), también se cifrará la comunicación entre cada cliente y el servidor centralizado del servicio. Esto se realizará para dificultar los intentos de vulneración de las comunicaciones por parte de atacantes externos y para ocultar algunos de los valiosos metadatos que no se cifrarían con la capa de cifrado punto a punto entre los clientes.

En cuanto a la decisión de qué usuarios tendrían acceso al servicio, esto sería tarea del usuario designado como administrador del grupo, el cual también podría participar en las conversaciones como cualquier otro usuario sin privilegios especiales.

Una vez que los usuarios hubiesen sido aceptados podrían participar en las conversaciones y reconectarse al servicio sin solicitar de nuevo la petición de acceso al actual administrador, siempre que se cumpliesen una serie de condiciones.

Para identificar a los usuarios que quisieran acceder al servicio de comunicación, reconectarse a este o a los que ya estén participando, se hará uso de pares de claves asimétricas y certificados digitales generados por los mismos, los cuales les permitirán aportar información que facilite su identificación y también les permitirán autenticar a los receptores de sus mensajes que ha sido él quien ha proporcionado dicha información.

Esas claves asimétricas y certificados digitales necesarios para identificar a cada usuario, podrían ser generados mediante una aplicación proporcionada, que sería desarrollada con la finalidad de generar esas claves y certificados de un modo sencillo, usable, guiado y a medida según los requisitos del servicio para estos.

Otra acción que podría realizar el administrador, aparte de la decisión de qué cliente puede acceder y cual no, es la de expulsar y revocar el acceso a los usuarios previamente aceptados en el servicio que él considerase oportunos, no permitiendo a los revocados continuar participando en la conversación.

El servicio propuesto además deberá cumplir la propiedad de forward secrecy en sus distintas capas de cifrado, tanto en la capa de comunicación entre los usuarios del grupo como en la capa establecida entre cada cliente y el servidor. Evitando de este modo que atacantes pudiesen llegar a tener acceso a información confidencial del pasado cifrada con claves ya desechadas.

Por lo tanto, en este proyecto propuesto se prestaría especial atención a la seguridad y a la privacidad del mismo, implementado además de los mencionados, otros mecanismos de seguridad que dificultasen la explotación de todo tipo de vectores de ataque posibles.

1.2 Justificación y objetivos

En el panorama actual cada vez se le da mayor importancia al grado de protección de la privacidad de los servicios, ya que los usuarios están más concienciados de la posibilidad de ser el objetivo de escuchas masivas o selectivas y de las consecuencias negativas que esto puede tener. En parte esto es debido a los programas destapados en los últimos años sobre vigilancia electrónica masiva llevadas a cabo por actores gubernamentales y también a la mayor repercusión de este tipo de vulneraciones de la privacidad de los usuarios en noticias de los distintos medios de comunicación.

Además de estar más concienciados con la privacidad de la información que comparten, también están un poco más concienciados con la seguridad de los sistemas y servicios de los que hacen uso habitualmente, debido a que cada vez la población tiene una mayor dependencia de las tecnologías de la información y a que los fabricantes y desarrolladores están más informados de los ataques que pueden llegar a sufrir si no aplican las medidas de seguridad suficientes, además de tener estos en cuenta que las vulneraciones graves pueden repercutir en daños difíciles de reparar en su imagen de marca.

Ante esta situación, actualmente algunos usuarios a la hora de elegir un nuevo servicio de comunicación prestan especial atención a las medidas de seguridad y privacidad que este ofrece, llegando en algunos casos incluso a migrar de servicio tras ya haberlo elegido si sienten que no es lo suficientemente seguro o si creen que se está haciendo un mal uso con sus datos, sobre todo aquellos usuarios que comparten información especialmente confidencial a través de ese tipo de servicios.

Por lo tanto, un servicio que fuese seguro por diseño y tuviese la seguridad realmente en cuenta desde el primer momento de su desarrollo y tras cada cambio importante en su ciclo de desarrollo, que implementase robustas medidas de protección, que no hiciese un mal uso con la información compartida por los usuarios y que además fuese eficiente, sería de interés para una gran parte de usuarios. Debido a la mayor concienciación de los usuarios y al gran auge de las aplicaciones y servicios de mensajería instantánea en los últimos años, donde la mayoría de los servicios de mensajería lanzados no implementan por defecto medidas robustas para proteger la seguridad y la privacidad de los usuarios, lo que lleva a que parte de los usuarios de esos servicios busquen otras alternativas más seguras.

El objetivo del sistema de comunicación propuesto es ofrecer una alternativa realmente segura a los servicios de comunicación actuales más populares, que en la mayoría de los casos no son lo suficiente seguros.

Una clara mejora respecto a muchos de esos servicios es el robusto cifrado punto a punto del que se hará uso, que estaría implementado mediante la doble capa de cifrado: la capa entre el cliente y el servidor y especialmente la capa de cifrado de los mensajes entre los interlocutores. En la capa de cifrado entre usuarios únicamente serían los interlocutores los que conocerían la clave que permitiría el descifrado de los mensajes recibidos, no siendo esa clave conocida por el servidor, ni siquiera tras su vulneración se vería comprometida la privacidad de las conversaciones pasadas o futuras, debido a que ha sido diseñado para no confiar excesivamente en el servidor del servicio por si fuese vulnerado, actuando en la mayoría de los casos simplemente como un túnel por el que se reenviarán los mensajes, donde los mensajes especialmente importantes además de ir cifrados irán firmados y serán verificados en su recepción para comprobar que no han sido alterados en tránsito.

Otra mejora que muchos de los servicios actuales no cumplen y que se pretende proteger mediante el uso de firmas digitales, es la robusta verificación de las identidades de los interlocutores, donde en muchos servicios actuales no se comprueba de manera segura que un usuario sea realmente quien dice ser y que no haya sido suplantado mediante vulnerabilidades del protocolo del servicio o de alguna de sus funcionalidades.

También se protegería la privacidad de las conversaciones realizadas en el pasado (propiedad de forward secrecy), por si se llegase a dar el caso de que se filtrase la clave de sesión con la que se cifran las comunicaciones. No pudiendo un atacante de este modo descifrar las conversaciones que fueron capturadas en el pasado, ya que se cifraron con claves efímeras que habrían dejado de existir tras haber sido sobrescritas.

Estas medidas de protección comentadas solo son algunas de las que se implementarán en el servicio de comunicación, habiendo muchas más, como por ejemplo: el añadido de marcas de tiempo a los mensajes para evitar ataques de reinyección de tráfico en el flujo de comunicación, el cifrado de los metadatos de las comunicaciones entre los clientes y el servidor, la posibilidad de poder almacenar cifrada la clave privada para evitar su uso en caso de sustracción sin conocer su contraseña, etc.

2 Estado del arte

En este apartado se repasará brevemente el estado del arte de los servicios de mensajería instantánea, indicando también las medidas de seguridad que suelen implementar y las que no, poniendo ejemplos de servicios famosos y comentando con mayor detalle las fortalezas y debilidades en cuanto a seguridad y a privacidad de dos de los más utilizados por los españoles.

2.1 Servicios de mensajería instantánea

La aparición del primer servicio de mensajería instantánea en línea se remonta a casi los orígenes de Internet, siendo este un servicio muy básico que permitía a un pequeño grupo de usuarios comunicarse en tiempo real mediante mensajes de texto. Desde entonces los servicios de mensajería no han parado de evolucionar ni de expandirse, aumentando considerablemente su popularidad en las posteriores décadas, debido a su utilidad para comunicarse y a la buena acogida por parte de los usuarios. Sobre todo, se han popularizado en la última década como consecuencia de la mayor penetración de Internet y al auge de los smartphones y del Internet móvil, llegando a convertirse en servicios de uso diario para gran parte de la sociedad de los países desarrollados. Remarcando que según estadísticas realizadas (1) el 57% de la población española utiliza servicios de mensajería instantánea a diario, siendo este el porcentaje de uso más alto de toda la Unión Europea, mientras que solo el 6% de la población española envía SMS a diario y un 62% de los españoles asegura no enviarlos nunca, pudiéndose comprobar a partir de estas estadísticas que se están convirtiendo en el claro sucesor de los mensajes SMS entre los españoles.

Como se ha comentado, en la actualidad este tipo de servicios están muy presentes en la sociedad de los países desarrollados, donde aumenta progresivamente su número de usuarios y son utilizados en todo tipo de ámbitos. Haciendo uso de estos tanto para conversaciones personales de baja importancia como para transmitir información considerada como confidencial entre miembros de grandes organizaciones.

Los servicios de mensajería instantánea evolucionan constantemente, dándoles a estos enfoques diferentes o añadiéndoles nuevas funcionalidades, con la finalidad de satisfacer las necesidades de los usuarios y hacerlas más atractivas para estos, debido en la mayoría de los casos a los intereses de las compañías que hay detrás para atraer a nuevos usuarios y lograr monetizar mejor el servicio. Algunas de esas funcionalidades que se han ido añadiendo respecto a los primeros servicios de mensajería instantánea y que han sido implementadas por una parte considerable de los servicios actuales son las siguientes: envío de archivos multimedia (imágenes, audios y vídeos), posibilidad de realizar llamadas de voz (incluso videollamadas), edición simplificada de imágenes y la posibilidad de compartir de un modo sencillo la geolocalización actual.

En el mercado existen una gran cantidad de servicios de mensajería instantánea, cada uno con sus peculiaridades y características propias. Aunque estos servicios, en ocasiones por cuestiones de marketing, se publiciten como seguros y que respetan la privacidad de los usuarios, esto no es siempre así, ya que no muchos realmente cumplen las medidas de seguridad y privacidad suficientes.

Debido a toda la información que se comparte a través de estos medios de comunicación, es importante proteger esa información compartida y evitar que sea obtenida por terceros con intenciones maliciosas o poco éticas.

Entre los servicios disponibles, casi la totalidad de los más famosos cifran el contenido de los mensajes en tránsito, pero algunos de ellos descuidan otros muchos detalles relacionados con la seguridad y privacidad, siendo habitual que en esos servicios no se protejan algunas de las siguientes características de seguridad y privacidad: cifrado de las conversaciones punto a punto, cifrado de los metadatos de las conversaciones, uso de claves efímeras, verificación de las identidades de los interlocutores o la posibilidad de auditar correctamente el código fuente de las aplicaciones cliente del servicio.

Algunas de las aplicaciones de mensajería más famosas que no cumplen la mayoría de las características de seguridad y privacidad comentadas anteriormente son las siguientes: Google Hangouts, Skype, SnapChat, y Yahoo Messenger. Como se puede apreciar son aplicaciones ampliamente conocidas y usadas en todo el mundo, pero aun así no cuentan con medidas de protección de la privacidad robustas.

Por otra parte, existen algunas aplicaciones de mensajería que cumplen todas o la mayoría de las características anteriores y son consideradas seguras por defecto, como por ejemplo: Signal, Silent Text y WhatsApp (tras la actualización de principios de abril del 2016).

Otras aplicaciones no cumplen la mayoría de las características comentadas por defecto, pero sí pueden llegar a cumplir la mayoría y permitir establecer conversaciones seguras y privadas si se indica específicamente para cada conversación (no siendo la opción por defecto), como por ejemplo: Telegram (los chats individuales secretos), Google Allo (los chats en modo incognito) y Facebook Messenger (los chats secretos).

Varios de esos servicios comentados como seguros por defecto o como opción adicional implementan variaciones del protocolo Signal (previamente conocido como protocolo TextSecure) del grupo Open Whisper Systems (2), en concreto lo implementa: WhatsApp, Google Allo, Facebook Messenger y por supuesto el servicio Signal (del mismo grupo). Se trata de un protocolo criptográfico diseñado en 2013 con la finalidad de crear servicios de mensajería instantánea que fuesen altamente seguros y protegiesen la privacidad de sus usuarios.

2.2 Servicios destacados y sus características

A continuación, se comentarán algunos de los servicios de mensajería instantánea más destacados del panorama actual, comentándose sobre estos sus principales características sobre seguridad y privacidad.

El servicio de mensajería instantánea más utilizado en todo el mundo y el más utilizado con un amplio margen de diferencia por los españoles es WhatsApp (3), que fue adquirido por la empresa Facebook a principios de 2014.

El 5 de abril del 2016 WhatsApp publicó una actualización de la aplicación que mejoraba su seguridad y privacidad considerablemente, implementando en el servicio el protocolo criptográfico Signal comentado anteriormente. Tras esa actualización el servicio pasó a ser más seguro, pero aun así WhatsApp tiene ciertas limitaciones no resueltas actualmente debido a problemas derivados del diseño previo o a intereses de Facebook para sacar un mayor provecho económico de la aplicación.

En ese nuevo protocolo implementado en WhatsApp se establece el cifrado punto a punto entre los usuarios, siendo las claves privadas generadas y almacenadas únicamente por los usuarios propietarios, también se establecen medidas de forward secrecy para proteger las conversaciones del pasado y de verificación de la identidad de los usuarios para evitar suplantaciones. Ese nuevo protocolo de cifrado punto a punto se utiliza para cifrar los mensajes, llamadas y demás archivos enviados, siempre que todas las partes tengan esa versión o una posterior, por lo tanto, con que únicamente un miembro del grupo no la tenga repercutirá en que todo el chat de grupo no utilizará ese nuevo protocolo.

Algunos de los puntos en contra que tiene en cuanto a seguridad y privacidad son los siguientes: el código no es público y no puede ser auditado para comprobar que no realice acciones sospechosas no documentadas oficialmente, almacena los archivos recibidos sin cifrar en el móvil y realiza copias de seguridad automáticamente de la base de datos que contiene las conversaciones (en la cual en ocasiones incluso no llega a eliminar correctamente los mensajes borrados) y además según algunos expertos WhatsApp no cifra muchos de los metadatos, donde dichos metadatos pueden ser utilizados por Facebook para cruzar información y obtener mejores perfiles de los usuarios para sus campañas de publicidad dirigida. (4)

Otro servicio de mensajería instantánea famoso y ampliamente conocido es Telegram, que fue lanzado en 2013 por dos de los fundadores de la famosa red social rusa VK, pero que actualmente ya no trabajan en esa compañía.

Telegram desde sus inicios tuvo en cuenta la seguridad y la privacidad de sus usuarios, dando lugar a un servicio que era más seguro que la mayoría de la competencia de ese momento. Esto provocó que se ganase la fama de ser la más segura, fama que todavía mantiene hasta el momento a pesar de que la competencia ha avanzado mucho en este aspecto y a que Telegram no ha implementado importantes medidas de seguridad últimamente, incluso ha desoído las peticiones de la comunidad que solicitan permitir chats secretos también para los grupos y utilizar el cifrado punto a punto como cifrado por defecto.

A continuación, se comentarán algunas de las características de seguridad y privacidad de Telegram, teniendo en cuenta que, aunque sea considerada por muchos como altamente segura, en realidad tiene múltiples limitaciones de seguridad y privacidad.

Todo en Telegram es cifrado mediante el protocolo propio MTProto (5), que utiliza AES de 256 bits, RSA de 2048 bits y el intercambio de claves seguras mediante Diffie-Hellman. Algunos expertos en criptografía han dudado de la seguridad del protocolo propio que han creado, debido a que desconfían de la robustez real del protocolo que han diseñado ellos mismos y a que consideran que no ha sido lo suficientemente probado como para asegurar que no sea vulnerable. Habiéndose incluso llegado a encontrar posteriormente debilidades en el protocolo (6), aunque de baja gravedad y de naturaleza más teórica.

Otras características de este servicio es que el cliente es de código abierto y está disponible para ser auditado en busca de debilidades o incluso puertas traseras, mientras que el servidor es de código cerrado y propietario. Además, los mensajes enviados por defecto se almacenan en la base de datos local sin cifrar, existiendo la posibilidad de que un atacante pueda sustraerla y leer las conversaciones no borradas sin la necesidad de conocer ninguna clave para ello.

Telegram por defecto cifra las conversaciones en tránsito, permitiendo conocer al servidor la información que se está compartiendo. Para evitar esto y añadir otras medidas de seguridad adicionales, Telegram ofrece chats secretos, donde la información irá cifrada punto a punto, para asegurar que los mensajes puedan ser leídos sólo por el destinatario. En esos chats secretos se cumple la propiedad de forward secrecy, ya que las claves utilizadas son renovadas periódicamente después de que hayan sido usadas más de 100 veces o de que hayan sido usadas durante más de una semana. En este tipo de chats también se verifica la identidad de los interlocutores, para evitar suplantaciones de identidad de los usuarios. Además, también ofrece la posibilidad en este tipo de chats de programar mensajes para que se destruyan automáticamente en ambos dispositivos que participan en el chat. Como se ha comentado anteriormente, actualmente los chats secretos no son la opción por defecto y solo están disponibles para conversaciones individuales y no para conversaciones grupales.

3 Planificación del proyecto

En este apartado del documento se mostrarán una serie de aspectos que han sido analizados sobre la planificación para desarrollar e implantar el proyecto propuesto. Indicando el modelo, entorno y tecnologías de desarrollo que serán utilizados durante todo el proceso, incluyendo también la descomposición del trabajo y la planificación temporal estimada para los distintos hitos, además de los análisis sobre los costes aproximados de implantación del servicio y los posibles modelos de negocio a establecer para monetizar el servicio sin perjudicar a los usuarios.

3.1 Metodología de desarrollo

Para la realización del proyecto se ha utilizado un modelo de desarrollo iterativo e incremental, donde el proyecto se divide en pequeñas etapas que dan como resultado prototipos que el usuario final pueda probar. Por lo que en cada una de esas pequeñas iteraciones se va mejorando y también ampliando las funcionalidades del servicio a desarrollar, priorizando en las funcionalidades esenciales para su correcto funcionamiento y en las que más beneficios aporten a los usuarios finales. (7)

Una de las razones por las cuales se decide utilizar este modelo de desarrollo es porque permite evaluar mejor el resultado durante su desarrollo mediante los prototipos resultantes de cada iteración. Facilitando de este modo la detección temprana de errores de diseño o de requerimientos.

Cada cierto periodo de tiempo mediante esos prototipos resultantes se muestra al tutor el progreso realizado, siendo ese periodo de tiempo de 1 o 2 semanas en las primeras fases del desarrollo. En esas reuniones, el tutor analiza los avances realizados en el nuevo prototipo mostrado, decidiendo a partir de este si está conforme con los cambios realizados y opinando sobre las posibles líneas futuras a seguir para etapas posteriores o los reenfoques que sería necesario realizar, en el caso de que no estuviese conforme con alguno de los aspectos del nuevo prototipo mostrado.

Siendo incluso utilizado un modelo similar durante la fase de diseño del proyecto, donde se van mostrando los cambios introducidos en el diseño en cada etapa con la finalidad de ir perfeccionándolo, aunque en esta fase no se entrega ningún prototipo funcional.

Siguiendo este modelo, se pretende evitar que falle la comunicación entre ambas partes y que no se terminen cumpliendo las expectativas previstas, pudiendo de otro modo sufrir considerables desviaciones respecto al resultado esperado. Además, permite asegurar que si hubiese problemas para llegar al plazo de entrega final se dispusiese de una versión totalmente funcional para realizar dicha entrega antes de la fecha límite inamovible.

3.2 Entorno y tecnologías de desarrollo

El lenguaje utilizado para desarrollar el sistema de comunicación grupal propuesto es Go, que se trata de un lenguaje de programación desarrollado e ideado por Google relativamente moderno, debido a que dicho lenguaje fue lanzando en noviembre del 2009 (8), teniendo en cuenta que la mayoría de los lenguajes de programación más utilizados en la actualidad son de la década de los 80 o de los 90.

Resumidamente algunas de las características que se pueden comentar de este lenguaje escogido son las siguientes: es compilado y eficiente en cuanto al proceso de compilación, con tipado estático (soportando inferencia implícita de tipos), cuenta con una sintaxis clara y concisa parecida a C, está disponible para la mayoría de sistemas operativos y arquitecturas actuales, es de código abierto y cuenta con una amplia documentación oficial, soporta potentes patrones de concurrencia y reflexión e incluye recolector de basura y una gran cantidad de paquetes útiles en su biblioteca estándar.

La biblioteca estándar de Go es especialmente útil en este caso, debido a que para este proyecto no es necesario utilizar ninguna otra biblioteca externa a dicha biblioteca estándar. Entre esos paquetes utilizados de la biblioteca estándar de Go, destacar el uso del paquete “crypto” (para todos los algoritmos, funciones, protocolos y estándares criptográficos utilizados) y del paquete “encoding” (para las distintas codificaciones utilizadas en el servicio, incluyendo Base64 y JSON).

En cuanto al entorno de desarrollo utilizado en este proyecto, se utiliza el ampliamente conocido IDE Eclipse en su versión Mars (lanzada en junio del 2015). Algunos de los motivos por los cuales se decidió utilizar este IDE son los siguientes: es un IDE muy completo y configurable, ofrece un amplio potencial mediante la inclusión de plugins adicionales, cuenta con una amplia comunidad, es de código abierto y multiplataforma, ofrece la posibilidad de trabajar con múltiples ventanas sobre el mismo proyecto simultáneamente (lo cual es útil cuando se desarrolla con múltiples monitores) y es un IDE que he utilizado en múltiples ocasiones en proyectos pasados con resultados satisfactorios.

Por defecto el IDE Eclipse no soporta el lenguaje de programación Go, por lo que para que soporte por completo dicho lenguaje se le ha añadido el plugin GoClipse (9). Ofreciendo mediante su inclusión las funcionalidades que Eclipse habitualmente proporciona para otros lenguajes de programación, como el coloreado de sintaxis, el reporte y resaltado de errores, el autocompletado de código y los accesos rápidos a las funciones de compilación, depuración, ejecución y formateo del código fuente.

El proyecto se divide en tres ejecutables distintos (server, client y certificateGenerator) y en cuatro archivos de código fuente. El ejecutable “server” es compilado a partir del código fuente mainServer.go, dependiendo también del código fuente usersList.go, mientras que los ejecutables “client” y “certificateGenerator” se compilan a partir de un único código fuente, llamados respectivamente mainClient.go y mainCertificateGenerator.go, sin depender estos de ningún otro código fuente propio para su compilación.

3.3 Estructura de desglose del trabajo

Para planificar con mayor precisión la hoja de ruta temporal del proyecto y poder monitorizar el progreso realizado durante su diseño y desarrollo, este se descompone en hitos (que normalmente serán los prototipos a mostrar) y elementos de distinto nivel.

En este caso al haber un único desarrollador, el nivel de detalle del desglose en elementos de cada fase será bajo, debido a que será bastante más fácil de predecir el coste temporal en este caso que en un proyecto en el cual se cuente con un equipo de desarrollo amplio donde existan muchas más variables a tener en cuenta.

A continuación, se muestran los distintos niveles de la estructura de desglose de trabajo (EDT) creada para el diseño y desarrollo del proyecto propuesto:

1. **Análisis previo**
   - 1.1. Recoger requerimientos iniciales
   - 1.2. Analizar requerimientos
   - 1.3. Definir alcance inicial
   - 1.4. Entrega del documento del análisis previo

2. **Diseño inicial del protocolo**
   - **2.1. 1ª versión del diseño del protocolo**
     - 2.1.1. Estudiar estado del arte
     - 2.1.2. Analizar medidas de seguridad a cumplir
     - 2.1.3. Diseñar protocolo
     - 2.1.4. Buscar debilidades en el protocolo diseñado
     - 2.1.5. Corregir debilidades encontradas
     - 2.1.6. Entrega del documento de la 1ª versión del protocolo inicial
   - **2.2. 2ª versión del diseño del protocolo**
     - 2.2.1. Analizar cambios a introducir
     - 2.2.2. Añadir cambios al diseño del protocolo
     - 2.2.3. Entrega del documento de la 2ª versión del protocolo inicial
   - **2.3. 3ª versión del diseño del protocolo**
     - 2.3.1. Analizar cambios a introducir
     - 2.3.2. Añadir cambios al diseño del protocolo
     - 2.3.3. Entrega del documento de la 3ª versión del protocolo inicial

3. **Desarrollo del sistema**
   - **3.1. 1ª versión del desarrollo**
     - 3.1.1. Analizar y escoger tecnologías a utilizar
     - 3.1.2. Estudiar tecnologías escogidas
     - 3.1.3. Preparar el entorno de desarrollo y pruebas
     - 3.1.4. Diseñar solución
     - 3.1.5. Desarrollar solución
     - 3.1.6. Evaluar solución
     - 3.1.7. Entrega de la 1ª versión del prototipo funcional
   - **3.2. 2ª versión del desarrollo**
     - 3.2.1. Analizar cambios a introducir
     - 3.2.2. Diseñar solución
     - 3.2.3. Desarrollar solución
     - 3.2.4. Evaluar solución
     - 3.2.5. Entrega de la 2ª versión del prototipo funcional
   - **3.3. 3ª versión del desarrollo**
     - 3.3.1. Analizar cambios a introducir
     - 3.3.2. Diseñar solución
     - 3.3.3. Desarrollar solución
     - 3.3.4. Evaluar solución
     - 3.3.5. Entrega de la 3ª versión del prototipo funcional
   - **3.4. 4ª versión del desarrollo**
     - 3.4.1. Analizar cambios a introducir
     - 3.4.2. Diseñar solución
     - 3.4.3. Desarrollar solución
     - 3.4.4. Evaluar solución
     - 3.4.5. Entrega de la 4ª versión del prototipo funcional
   - **3.5. 5ª versión del desarrollo**
     - 3.5.1. Analizar cambios a introducir
     - 3.5.2. Diseñar solución
     - 3.5.3. Desarrollar solución
     - 3.5.4. Evaluar solución
     - 3.5.5. Entrega de la 5ª versión del prototipo funcional
   - **3.6. 6ª versión del desarrollo**
     - 3.6.1. Analizar cambios a introducir
     - 3.6.2. Diseñar solución
     - 3.6.3. Desarrollar solución
     - 3.6.4. Evaluar solución
     - 3.6.5. Entrega de la 6ª versión del prototipo funcional

3.4 Hoja de ruta estimada

Una vez obtenida la estructura de desglose del trabajo se procede a realizar una hoja de ruta estimada (también llamado roadmap), que se trata de una planificación temporal donde se estima el esfuerzo necesario que se requerirá para realizar cada una de las actividades, siendo necesario indicar también las dependencias que existirán entre esas distintas actividades.

Una vez indicados los datos de entrada requeridos para generar la hoja de ruta, esta dará como resultado los plazos estimados en los cuales se finalizará cada actividad y se llegará a cada hito, pudiendo indicar también los solapes que pudiesen existir entre tareas que tuviesen restricciones.

Este tipo de planificaciones temporales pueden ser de gran utilidad para prever cuando se llegará a un hito determinado, para analizar si se producirán solapes entre recursos durante el desarrollo del proyecto, para monitorizar el avance realizado y tratar de asegurar que se llegará sin retrasos a las fechas límite establecidas y para comprobar como afectarían los reajustes a la planificación inicialmente estimada.

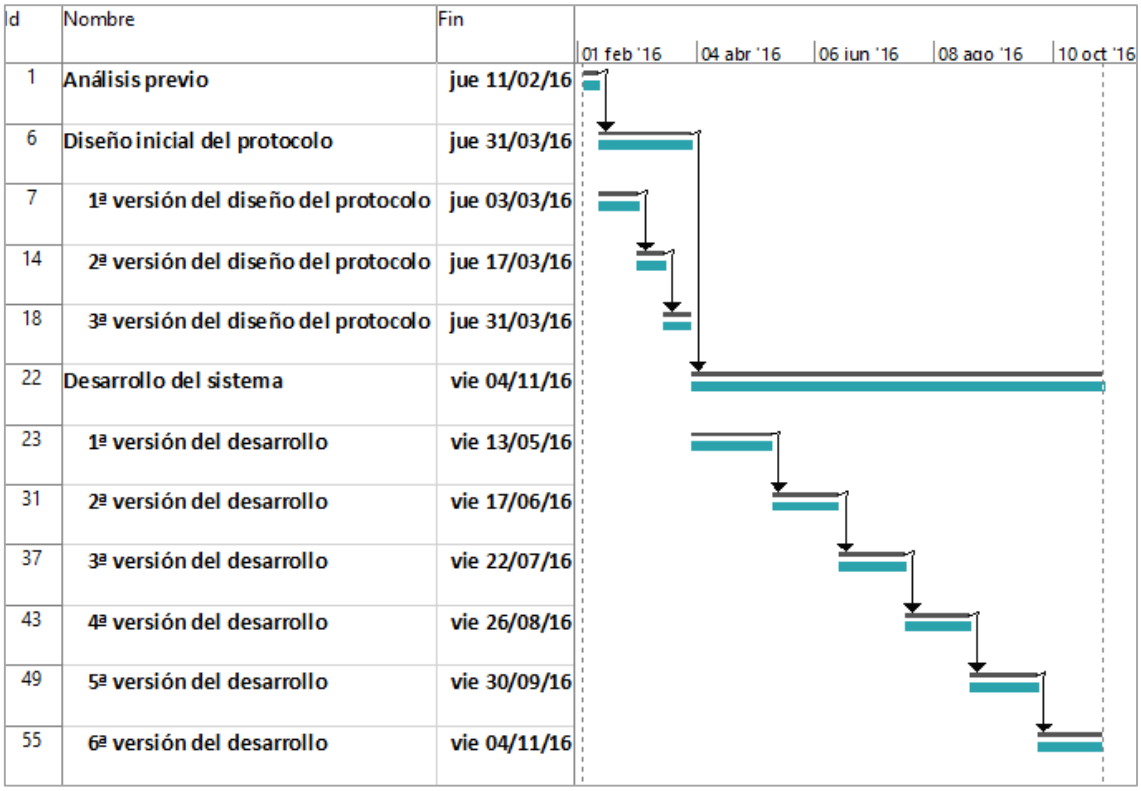

Figura 1: Hoja de ruta estimada para el diseño y desarrollo del proyecto

En la hoja de ruta estimada para este proyecto se puede apreciar el tiempo previsto que se dedicará a cada fase del proyecto, pudiendo llegar a tener el proyecto implementado con todas las funcionalidades para el 4 de noviembre del mismo año según la hoja de ruta generada, siempre que no se produzcan retrasos inesperados durante el transcurso del proyecto.

3.5 Posibles modelos de negocio

En este punto se comentarán algunos de los modelos de negocio que han sido analizados para tratar de monetizar en un futuro el sistema de comunicación desarrollado, cada uno de esos posibles modelos de negocio analizados trata de obtener beneficios de un modo distinto y no siempre se dirigen al mismo tipo de cliente.

A continuación, se comentarán algunos de esos posibles modelos de negocio con los cuales se quiere monetizar el proyecto desarrollado, no pretendiendo aplicar todos, solo uno o un par de ellos:

- Pago periódico por el alojamiento del servidor privado y por el servicio técnico oficial: en este modelo de negocio propuesto los ingresos vendrían por parte de las organizaciones o incluso de individuos que decidiesen disponer de un sistema de comunicación grupal seguro externalizado. Contando para ello con un servidor del servicio privado únicamente para ellos en instalaciones de confianza y con la supervisión y resolución de problemas por parte de un servicio técnico oficial con amplios conocimientos sobre el sistema. Ofreciendo al cliente mediante esta solución un servicio oficial de calidad, con la comodidad de no tener que preocuparse de administrarlo.

- Pago periódico por cada usuario para hacer uso del servicio: en este modelo los servidores estarían alojados en servidores de nuestra propiedad y los usuarios tendrían que abonar un pago periódico para hacer uso del sistema de comunicación. Pudiendo ofrecerse planes para grupos de usuarios a precios por usuario más reducidos, con la finalidad de que el servicio resultase más atractivo a las organizaciones. De este modo, el cliente pagaría únicamente por el uso que le diese y evitaría tener que montar y administrar el sistema de comunicación en su infraestructura.

- Pago por el servicio de verificación de cuentas oficiales: este modelo de negocio se basaría en diferenciar entre cuentas normales y cuentas oficiales de confianza especialmente verificadas, donde actuaríamos a su vez como una entidad certificadora que firmaría los certificados de esas cuentas que deseasen ser oficiales, mediante su previo pago y análisis para asegurar de que son quien dicen ser. Esto podría ser útil para organizaciones que utilizasen este servicio para comunicarse con sus clientes o empleados, debido a que de este modo los receptores podrían saber que se trata realmente de esa organización, dando mayor confianza y evitando posibles suplantaciones de identidad. Este pago sería realizado por cada certificado de identificación firmado emitido, contando esos certificados emitidos con un periodo de caducidad considerablemente prolongado.

- Pago por tener acceso a funcionalidades adicionales del servicio: en este caso los ingresos se obtendrían mediante la existencia de funcionalidades que aportasen valor adicional y les resultasen atractivas de adquirir a los usuarios finales. Dependiendo del tipo de funcionalidad o característica el pago sería periódico o único. Uno de los problemas de este modelo es que requiere que esas funcionalidades adicionales sean privadas y no de código abierto, para evitar que cualquiera pudiese compilar los códigos fuente por su cuenta y obtenerlas de manera gratuita, para ello se podría mantener el cliente de código abierto, pero dejar el servidor con código cerrado, siendo igualmente seguro, debido a que las conexiones son cifradas punto a punto y a que los clientes son quienes generan y almacenan las claves requeridas.

Posteriormente, una vez escogidos los modelos que se consideren más adecuados, se deberían realizar estudios más extensos y detallados del mercado sobre los modelos de negocio escogidos, para prever con mayor precisión la acogida que pudiese tener, para evitar errores cometidos por la competencia y para adaptarlo mejor a las necesidades deseadas y gustos de los clientes habituales de este tipo de servicios mediante esas modalidades.

3.6 Costes aproximados de implantación

Como se ha comentado, para poner en funcionamiento el servicio es necesario contar con la aplicación que actué como servidor ejecutándose y a la escucha de nuevas peticiones, debido a que el sistema de comunicación utiliza una topología de red centralizada. Por lo tanto, en este punto se analizarán los costes aproximados que podría tener alojar el servidor del sistema de comunicación propuesto, mediante el análisis de algunas de las opciones que se han considerado más adecuadas para su implantación.

Para alojar el servicio propuesto existen una infinidad de opciones posibles, debido a que no requiere de características poco usuales y a que es eficiente en cuanto al consumo de ancho de banda y de recursos de procesamiento, pero en este punto solo se analizarán algunas de las opciones ofrecidas por reconocidos proveedores de servicios en la nube que se han considerado más oportunas para este caso.

Se ha descartado la opción de alojar los servicios en instalaciones físicas propias desde un principio, debido a múltiples razones que se comentarán a continuación. Una de las más importantes es el elevado desembolso inicial que sería requerido para adquirir y montar la infraestructura necesaria, además de la necesidad de encontrar una ubicación adecuada para su montaje, la complejidad para realizar dicha tarea y el coste temporal requerido en el caso de querer montar una infraestructura que soporte una gran cantidad de usuarios en múltiples grupos simultáneamente. Esa infraestructura a montar sería aún más compleja y costosa si se quisiese

ofrecer el servicio con alta disponibilidad, teniendo que configurar y replicar gran parte de la infraestructura, además de ser altamente recomendable que la réplica se encontrase en una ubicación distinta. También comentar que con una infraestructura propia sería más complejo de administrar y difícil de prever los escalados ante aumentos considerables de la demanda, como por ejemplo si se produjese un gran aumento puntual de la carga tras una campaña de marketing sobre el servicio. Por lo tanto, se ha descartado la opción de realizar grandes desembolsos en una infraestructura física propia desde un principio, ya que el modelo de negocio podría no tener éxito y existiría el riesgo de no llegar a recuperar gran parte de la inversión realizada en dichas infraestructuras.

Por otra parte, pensando en reconocidos proveedores de servicios en la nube, uno de los pocos servicios ofrecidos en la nube como PaaS (plataforma como servicio) que tiene soporte para el lenguaje Go y que no se encuentra en fase experimental es Google App Engine, un servicio ofrecido por Google Cloud Platform. (10) A primera vista puede parecer una buena opción para alojar el servidor del servicio, el cual está desarrollado por completo en Go y el cual no requiere de paquetes externos a la biblioteca estándar para su funcionamiento, facilitando de este modo la implantación, además Google App Engine cuenta con plugins oficiales para el IDE Eclipse.

Pero tras analizar más detalladamente el servicio Google App Engine se puede comprobar que tiene múltiples puntos en contra que dificultarían considerablemente la implantación y harían necesario modificar el sistema propuesto, como que ese servicio PaaS está destinado principalmente a alojar aplicaciones web, por lo que está diseñado para procesar peticiones HTTP, teniendo amplias limitaciones a la hora de gestionar sockets, también tiene limitaciones para trabajar con múltiples hilos paralelamente, tampoco permite que para responder a una consulta se tarde un periodo superior a treinta segundos, además de que al ser un servicio PaaS no permite escribir ficheros en el sistema ni configurar la infraestructura sobre la que se ejecuta. Todas esas limitaciones comentadas afectarían al modo en el que se ha planteado el sistema propuesto, no pudiendo implantarlo en esta plataforma sin realizar grandes cambios, por lo que se decide descartar la opción de utilizar Google App Engine.

Otra opción válida para este caso de Google Cloud Platform es el servicio en la nube como IaaS (infraestructura como servicio) llamado Google Compute Engine o GCE, que ofrece máquinas virtuales en la nube con un alto rendimiento y escalabilidad. (10)

Por ejemplo, una infraestructura bajo demanda en Google Compute Engine válida para ofrecer servicio a una cantidad considerable de clientes iniciales (pudiendo escalarse fácilmente según creciese la demanda) estaría formada por 4 instancias de máquinas virtuales “n1-highcpu-4” con almacenamiento local mediante discos SSD de alto rendimiento, donde cada una de esas instancias virtuales contaría con 3,6 GiB de memoria RAM y 4 CPU virtuales (implementadas

mediante hyper-threading sobre Intel Xeon E5 de aproximadamente 2,4 GHz), utilizándose en esas instancias como sistema operativo una distribución GNU/Linux. Este tipo de instancia sería adecuada para el sistema de comunicación propuesto, debido a que este sistema propuesto no tiene un elevado consumo de memoria RAM, pero sí requiere de potencia de procesamiento en sus momentos de mayor carga.

Además, mediante la correcta configuración del sistema, esta infraestructura en la nube permitiría balancear la carga entre las distintas instancias y proporcionar un servicio con alta disponibilidad que realojase el servicio de comunicación ante problemas en su instancia actual.

Mediante esta infraestructura en la nube propuesta mediante GCE se podría dar servicio a un elevado número de usuarios en distintos grupos, permitiendo ofrecer este servicio a múltiples clientes simultáneamente tras su implantación. Por ejemplo, como una estimación de los usuarios que podría llegar a soportar cada una de las instancias, teniendo en cuenta que cuanto mayor sea el número de usuarios por grupo más recursos requerirá administrarlo y que se trata de una estimación inicial que puede mejorar en la práctica y tras futuras optimizaciones, soportaría aproximadamente: 33 grupos de 15 usuarios cada uno, 15 grupos de 30 usuarios, 6 grupos de 70 usuarios o 4 grupos de 100 usuarios.

El coste de esta infraestructura mediante GCE sería de aproximadamente 635€/mes (asumiendo que estuviese en funcionamiento 24x7), siendo una cifra fácil de amortizar en el caso de que el modelo de negocio propuesto tenga cierto éxito, debido a que las instancias propuestas podrían dar servicio a un gran número de clientes.

También podría ser válida para este caso la opción que ofrece el proveedor de servicios en la nube Amazon Web Services mediante su ampliamente conocido servicio IaaS Amazon EC2, que también se trata de entornos virtuales en la nube personalizables y altamente escalables. (11)

Una infraestructura similar a la propuesta para GCE en cuanto a rendimiento y características en EC2 sería la contratación de 4 instancias virtuales “c3.xlarge”, contando también con almacenamiento local mediante SSD, 4 CPU virtuales (implementadas mediante hyper-threading sobre el Intel Xeon E5-2680 a 2,8 GHz) y un sistema operativo GNU/Linux, pero estas instancias contarían con más memoria RAM, en concreto 7,5 GiB. Además, comentar que Amazon Web Services ofrece un kit de desarrollo para facilitar el proceso de desarrollo de aplicaciones en Go para infraestructuras EC2 (12).

El coste de esta otra infraestructura similar mediante EC2 sería de aproximadamente 665€/mes (asumiendo que estuviese en funcionamiento 24x7). Unos 30€/mes más que la infraestructura propuesta mediante GCE, pero teniendo en cuenta que incluye más memoria RAM y potencia de procesamiento por instancia.

4 Diseño y desarrollo del proyecto

En los siguientes puntos se explicará detalladamente el diseño y el protocolo definitivamente desarrollados para el servicio de mensajería instantánea propuesto, indicando también los diseños previos descartados y un resumen de las medidas de seguridad y privacidad implementadas.

4.1 Diseños previos descartados

Previamente al diseño definitivo del servicio que se terminó escogiendo y posteriormente implementando, se estudiaron otros diseños diferentes posibles. Estos diseños previos, tras ser analizados detenidamente, fueron descartados por diversos motivos, siendo algunos de esos motivos los siguientes: la baja eficiencia en cuanto al consumo de recursos de procesamiento, un aumento considerable del tamaño de gran parte de los mensajes y una menor estabilidad y control del servicio ante fallos del mismo o de la red.

A continuación, se comentarán las características de esos diseños previos realizados y se explicará en mayor profundidad los motivos por los cuales se terminaron descartando dichos diseños previos.

4.1.1 Topología de red peer-to-peer descentralizada

En un principio se estudió la posibilidad de implementar este servicio utilizando una topología de red peer-to-peer descentralizada, donde todos los nodos actuarían al mismo tiempo como cliente y servidor.

Con este diseño, cada nodo almacenaría en una caché propia al resto de clientes que hubiesen sido aceptados y sus correspondientes direcciones IP y puertos tras los cuales se encontraría su servicio escuchando; haciendo la función de una especie de tabla de enrutamiento que tendría que mantenerse actualizada conforme a los cambios que se fuesen produciendo, como el cambio de IP y puerto tras una reconexión de un usuario, la aceptación de nuevos clientes, la revocación de un usuario previamente aceptado, la designación de un nuevo administrador al grupo, etc.

De este modo, no existiría ningún servidor central en el servicio por el que tuviesen que pasar todas las comunicaciones realizadas, mejorando por tanto la privacidad del servicio. Debido a que si todo el tráfico pasase por un servidor central, este podría llegar a conocer ciertos metadatos valiosos sobre dichas comunicaciones.

Además, al tratarse de una topología de red descentralizada, sería menos vulnerable a ataques de tipo DoS (Denial of Service) que interrumpiesen el servicio, al no tener tanta dependencia de un servidor central que se encargase de gestionar funciones cruciales para el correcto funcionamiento del sistema.

Pero tras estudiar en mayor profundidad este diseño descentralizado, se pudo comprobar que aportaba más desventajas que ventajas al servicio de mensajería a implementar, por lo que se terminó descartando y optando por una topología de red centralizada con múltiples capas robustas de cifrado.

Entre esas desventajas se encuentra la mayor dificultad de asegurar en una topología descentralizada que se procesen correctamente las revocaciones de usuarios y los cambios de administrador, debido a que puede darse el caso de que no lleguen esos paquetes a ciertos nodos por errores de red o por interceptaciones de terceros con intenciones maliciosas. También sería más difícil determinar quién debería ser el siguiente administrador del grupo ante la desconexión repentina del actual administrador.

Otro problema de esta topología descentralizada, son las posibles pérdidas de integridad entre las distintas tablas caché de enrutamiento y configuraciones de los distintos nodos del servicio. Como por ejemplo en el caso de que uno de los nodos no recibiese el cambio de IP de uno de los usuarios tras su reconexión y siguiese enviando los paquetes a la dirección antigua, recibiendo ese nodo reconectado solo las respuestas del resto de usuarios; siendo estos problemas más fáciles de solucionar en una topología de red centralizada.

También sería más difícil establecer una dirección de encuentro que no variase ante la desconexión de sus nodos, como sería normalmente el caso de la dirección del servidor en una topología centralizada. Además, al actuar todos los nodos también como servidores, todos deberían ponerse a la escucha en la dirección y puerto indicados, lo cual puede ser un problema si se encuentra detrás de un firewall, sobre todo si no se pueden editar sus reglas.

En general, el diseño propuesto mediante peer-to-peer descentralizado protegía ligeramente más la privacidad de sus usuarios, pero era más complejo, menos estable y menos seguro respecto al diseño centralizado planteado.

4.1.2 Claves precompartidas entre pares

El primer diseño propuesto en cuanto al método de cifrado y descifrado de los mensajes enviados por los interlocutores del grupo, se basaba en la compartición de una clave simétrica diferente entre cada par de clientes antes de iniciar la conversación.

Para ello se generaría cada nueva clave con un generador de números pseudoaleatorios criptográficamente seguro y se compartiría punto a punto mediante el uso de criptografía de clave pública. De modo que esa clave solo sería conocida por ese par de usuarios, incluso aunque fuese compartida a través de un canal inseguro.

Para cifrar los mensajes una vez se hayan compartido las distintas claves entre los pares, se cifrará el contenido mediante un algoritmo de criptografía simétrica con una clave generada aleatoriamente para cada mensaje (a la cual llamaremos $K_{contenido}$).

Para que solo los usuarios del grupo admitidos y nadie más pueda descifrar los mensajes enviados, $K_{contenido}$ se cifrará con la clave precompartida entre el emisor y los distintos receptores; y estas claves cifradas para cada uno de los receptores se colocarán en la cabecera del mensaje, que posteriormente será reenviado a todo el grupo.

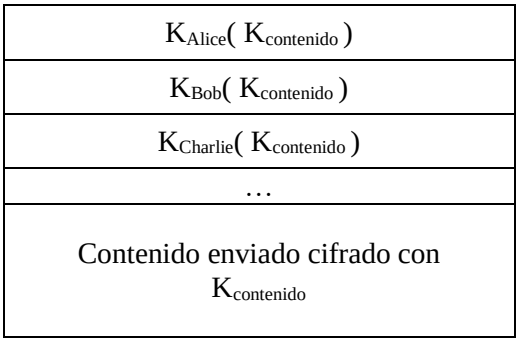

Figura 2: Estructura de un mensaje descifrable mediante múltiples claves precompartidas

De esta manera los usuarios receptores previamente aceptados tendrán la capacidad de descifrar el contenido del mensaje, sin que sea necesario cifrar el contenido con una clave diferente para cada uno de los distintos receptores del grupo, lo cual sería más costoso computacionalmente y haría las comunicaciones más pesadas. Por lo tanto en este diseño solo sería necesario cifrar el contenido con una sola clave debido a que podrían obtener $K_{contenido}$ descifrándola a partir de la cabecera con la clave cifrada para él.

Otra de las ventajas de este diseño es la facilidad con la que se pueden revocar usuarios, debido a que simplemente habría que no incluir la cabecera con $K_{contenido}$ cifrada para dicho usuario revocado.

Finalmente, este diseño se descartó por varios motivos, uno de los principales motivos por los cuales fue descartado era la necesidad de compartir un elevado número de claves precompartidas entre todo el grupo, debido a que esa cantidad crece considerablemente conforme crece el número de usuarios en el grupo, ya que cada miembro debe compartir una clave diferente por cada interlocutor. Por ejemplo, como se podrá calcular con la fórmula proporcionada posteriormente, para un grupo de 25 usuarios haría falta compartir unas 300 claves, lo cual resulta excesivo y muy poco eficiente.

A continuación, se muestra la fórmula con la que se pueden calcular la cantidad de claves diferentes que sería necesario compartir en este diseño planteado que finalmente se descartó (donde *m* es el número de miembros en el grupo):

$$\sum_{i=1}^{(m-1)} i = número\ de\ claves\ compartidas$$

Número de miembros	Claves compartidas totales
2	1
3	3
4	6
5	10
6	15
7	21
8	28
9	36
10	45
25	300
50	1225

Figura 3: Cálculo de claves precompartidas necesarias según número de miembros

Otro de los motivos por los cuales se descartó es que si existen muchos usuarios en el grupo aumenta notablemente el tamaño de cada mensaje, al tener que añadir la clave del contenido cifrada para cada receptor en las cabeceras, además del coste computacional de cifrar $K_{contenido}$ para cada uno de los receptores.

Por lo tanto, una vez analizado, se comprueba que este diseño solo es eficiente cuando el número de interlocutores es bajo, no siendo recomendable para grupos con un número medio o grande de interlocutores.

4.1.3 Claves públicas para cifrar la clave del contenido

Posteriormente se propuso otro diseño similar al anterior para el cifrado y descifrado de los mensajes, tratando de resolver el problema del gran número de claves que era necesario precompartir en grupos de tamaño medio o grande.

En este diseño, ya no se utilizaban claves simétricas precompartidas entre los distintos usuarios con la finalidad de cifrar $K_{contenido}$ para cada receptor del mensaje, sino que $K_{contenido}$ se cifraba mediante la clave pública de cada receptor y se seguían colocando dichas claves cifradas en la cabecera del mensaje.

De este modo, ya no sería necesario generar y precompartir claves entre cada par, sino que sería únicamente necesario compartir previamente su clave pública con el resto de usuarios, siendo para todos la misma y no siendo distinta entre pares de usuarios.

Además, en este diseño la revocación de usuarios sigue siendo igual de sencilla, pudiendo realizarla simplemente al no añadir $K_{contenido}$ cifrada para ese usuario con su clave pública en la cabecera de los mensajes.

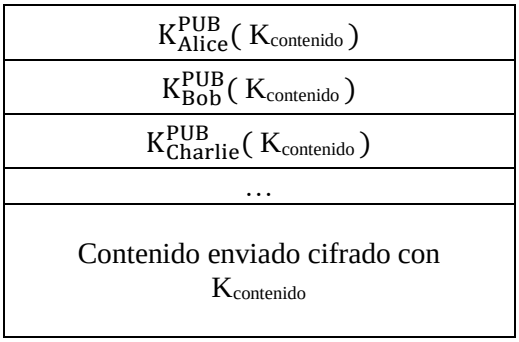

Figura 4: Estructura de un mensaje descifrable mediante múltiples claves privadas

Uno de los problemas de este diseño es que sigue aumentando considerablemente el tamaño de cada mensaje para grupos con un número medio o grande de interlocutores, al seguir introduciendo en la cabecera una $K_{contenido}$ cifrada diferente para cada uno de los distintos destinatarios.

Además, se hace un uso abusivo de la criptografía asimétrica al cifrar $K_{contenido}$ mediante este tipo de algoritmos para cada destinatario en cada mensaje, siendo estos algoritmos mucho más costosos computacionalmente que los algoritmos de criptografía simétrica habituales, haciéndolo en este sentido aún menos eficiente.

Por lo tanto, tras analizar este otro diseño, se comprueba que tampoco es eficiente para grupos con un número medio o grande de interlocutores, por lo que también es descartado y no será utilizado para la implementación definitiva.

4.2 Diseño definitivo implementado

En este punto se explicará con detalle el funcionamiento y las características del diseño que fue escogido definitivamente para la implementación del proyecto, tras haber descartado los diseños que no eran lo suficientemente óptimos para la correcta y eficiente resolución del problema propuesto.

4.2.1 Gestión de la clave de sesión común

En este diseño definitivamente escogido para desarrollar el servicio no se establece una clave precompartida diferente para cada par de usuarios ni se genera una clave aleatoria cada vez para cifrar el contenido del mensaje a enviar y se adjunta en la cabecera, sino que se hace uso de una clave de sesión común precompartida entre todos los interlocutores (a la cual en ocasiones se hará referencia como $K_{sesión}$).

Dicha clave común será utilizada para cifrar las conversaciones entre los distintos interlocutores del grupo, utilizándose tanto para cifrar el contenido de los mensajes de texto como para cifrar los archivos enviados en esa conversación. De manera que ese contenido cifrado solo podrá ser descifrado por los usuarios en los que se haya confiado previamente y se les haya proporcionado $K_{sesión}$. Tampoco siendo posible enviar mensajes legibles si no se conoce esa clave de sesión, al ser requerido que el contenido de todos los mensajes sea cifrado con esa clave común, protegiéndose de este modo ante ataques de inyección de paquetes en las comunicaciones por parte de atacantes externos con intenciones maliciosas.

De este modo el diseño definitivo cumple todos los requisitos propuestos de manera eficiente, al no aumentar considerablemente el tamaño de la cabecera de los mensajes, no necesitar compartir grandes cantidades de claves precompartidas entre los pares ni tampoco requerir un coste computacional alto para su ejecución, debido a que no se hace un uso abusivo de los algoritmos de criptografía asimétrica, ya que en el diseño definitivo todos los mensajes enviados se cifrarán con algoritmos de criptografía simétrica mediante una clave común previamente compartida entre los clientes, por lo que no será necesario añadir las claves de descifrado en la cabecera de los mensajes a enviar.

La clave $K_{sesión}$ que se utiliza para cifrar los mensajes tiene un tamaño de 256 bits y es generada de manera segura por el administrador del grupo, para ello se utiliza un generador de números pseudoaleatorios criptográficamente seguro (CSPRNG) que tenga la suficiente entropía como para usarse en protocolos criptográficos.

Para la implementación realizada como generador de números pseudoaleatorios se hace uso del paquete de Go “crypto/rand” (13) que incluye funciones ya implementadas para cumplir dicha finalidad, donde estas funciones a su vez hacen uso de utilidades para generar números pseudoaleatorios robustos del sistema operativo en el que se ejecutan. En concreto, en sistemas operativos de la familia Linux y Unix por defecto se hace uso de la piscina de entropía “/dev/urandom” (14), mientras que en sistemas operativos de la familia Windows se hace uso de la API “CryptGenRandom” (15).

Una vez generada $K_{sesión}$ esta es pasada por el administrador a los usuarios de confianza que hayan sido aceptados, enviándola cifrada de punto a punto mediante el uso de criptografía de clave pública para enviar esa clave a cada destinatario concreto, evitando que los usuarios ajenos puedan conocer su valor incluso si llegasen a interceptar esos paquetes de la comunicación. Además de enviarse $K_{sesión}$ siempre cifrada, también se adjunta siempre el resumen hash de la misma firmado por el administrador con su clave privada para evitar alteraciones y usos malintencionados.

Este diseño se asegura de que el servidor del servicio no tenga que conocer $K_{sesión}$ para su correcto funcionamiento ni pueda forzar el protocolo con intenciones maliciosas para conocerla. Evitando de este modo que si el servidor es vulnerado $K_{sesión}$ pudiese ser obtenida y alguien ajeno pudiese descifrar las conversaciones realizadas. Ya que si alguien ajeno obtuviese dicha clave secreta, el servicio quedaría comprometido, al poder descifrar las conversaciones enviadas y archivos mediante la interceptación de las comunicaciones.

Otra característica de este diseño es la posibilidad de revocar usuarios por decisión del administrador. Tras la revocación de usuarios, estos se expulsarán del grupo (no enviándoles más mensajes) y se cambiará $K_{sesión}$ para que no puedan descifrar los mensajes incluso si llegasen a tener acceso a estos. Esa característica de cambio de $K_{sesión}$ también será utilizada para cambiar dicha clave cada cierto tiempo, evitando de esta manera que si alguien llegase a descubrir la clave de ese momento pudiese descifrar conversaciones capturadas del pasado, al estar cifradas con claves diferentes e independientes entre ellas.

4.2.2 Envió de mensajes de texto cifrados

Una de las funcionalidades primordiales de este sistema de comunicación grupal es la del envío de mensajes de texto entre los distintos interlocutores del grupo, lo que permite a un usuario del grupo enviar una cadena de texto, la cual será retransmitida por el servidor al resto de usuarios que se encuentren conectados. Ese mensaje enviado estará cifrado por el usuario emisor y tendrá que ser descifrado por los usuarios receptores de dicho mensaje, no pudiendo ser descifrado en tránsito por el servidor ni por cualquier otro nodo intermediario que tuviese acceso a esa comunicación.

Como se ha comentado anteriormente, los mensajes de texto enviados a través del servicio serán cifrados con AES en modo CTR (counter) y con un tamaño de clave de 256 bits, debido a que se usará como clave $K_{sesión}$ que previamente habrá sido generada (con un tamaño de 256 bits) y compartida con todos los usuarios del grupo. En el modo de cifrado CTR se utiliza AES como si fuese un cifrador de flujo, generando un flujo de claves a partir del nonce de inicialización proporcionado, siendo el nonce un número generado aleatoriamente en cada nuevo uso. Dicho modo actualmente está considerado seguro, siempre y cuando que los nonces utilizados tengan la suficiente entropía y no se repitan.

Para ese cifrado del mensaje con AES en modo CTR se utilizará un nonce aleatorio de 128 bits que se adjuntará sin cifrar al principio del mensaje cifrado para que el receptor pueda obtenerlo y descifrar dicho mensaje cifrado. Ese nonce será generado con un generador de números pseudoaleatorios criptográficamente seguro, debido a que debe tener una alta entropía y nunca deberá repetirse.

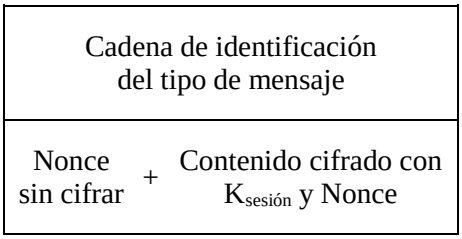

Figura 5: Estructura de un mensaje descifrable mediante clave de sesión

Como se puede apreciar en la estructura mostrada en la figura anterior, a todos los mensajes a enviar se les añade una cadena de identificación para que el receptor pueda saber el tipo de paquete que es y como deberá procesarlo. Por ejemplo, el identificador de un mensaje de texto es "MSG_TEXT", el de un mensaje con un archivo es "MSG_FILE" y el de un mensaje de control de denegación de acceso al grupo es "DENIED". En este caso, además, al ser un mensaje cifrado, al inicio del segundo campo se incluye el nonce generado aleatoriamente de 128 bits en claro y posteriormente la cadena de texto del mensaje escrito (junto a algunos metadatos) cifrada con $K_{sesión}$ y el nonce adjunto.

```
Mensaje de Carlos1   [20-Dec 18:15]: Buenos días, os tengo que comunicar algo.
Mensaje de Carla3    [20-Dec 18:15]: ¿Se trata de algo urgente?
Mensaje de Carlos1   [20-Dec 18:15]: No, por ahora...
Mensaje de Cristian2 [20-Dec 18:16]: Ok
Mensaje de Carla3    [20-Dec 18:17]: Ok
```

Figura 6: Conversación de ejemplo entre usuarios

En la anterior figura se muestra un ejemplo de una conversación utilizando el sistema de comunicación desarrollado, donde se puede ver el formato con el que se reciben las respuestas del resto de interlocutores del grupo. En dicho formato se muestra primero el nombre del usuario que envía dicho mensaje, siendo ese nombre el indicado durante la creación de su certificado, posteriormente aparece una marca de tiempo de cuando fue enviado y por último la cadena de texto escrita por el emisor.

4.2.3 Envío de archivos cifrados

Otra funcionalidad incluida en el sistema de comunicación desarrollado aparte del envío de mensajes de texto, es el envío de archivos entre los distintos interlocutores, pudiendo enviar ficheros sin restricciones de formatos ni de roles del usuario, pudiendo enviar y recibir archivos tanto el administrador como el resto de usuarios del grupo.

En este caso el cifrado y la estructura utilizada para el envío son muy similares a las del envío de mensajes de texto, utilizando también AES en modo CTR con la clave $K_{sesión}$ de 256 bits y un nonce aleatorio de 128 bits sin cifrar adjunto al principio, además de los metadatos del nombre del emisor y de la marca de tiempo al igual que en el mensaje de texto, en este caso también se adjunta en el mensaje el nombre del fichero que permitirá reconocerlo a los receptores y guardarlo

directamente con ese nombre si así lo desean. Todos esos metadatos incluidos irán cifrados con la clave de sesión, para evitar su modificación ante una intercepción sin conocer $K_{sesión}$ y para evitar también que puedan ser leídos por usuarios ajenos al grupo.

El usuario que desee compartir un archivo con el resto podrá utilizar la instrucción reconocida por la aplicación cliente que tiene la siguiente sintaxis: /enviar <ruta del archivo a enviar>. La aplicación cliente comprobará previamente a su envío que exista un archivo válido en dicha ruta y además si es válido también comprobará que dicho archivo no tenga un tamaño superior a 50 MiB. Esta limitación de tamaño se establece con la finalidad de evitar saturar demasiado la conexión del servidor por si este tuviese un ancho de banda reducido, teniendo en cuenta que el servidor ante la llegada de un archivo se encargaría de retransmitir ese archivo al resto de usuarios del grupo que estuviesen conectados.

```
/enviar
ERROR: Sintaxis incorrecta
Modo de empleo: /enviar "ruta del archivo a enviar"
/enviar archivo_inexistente.txt
mainClient.go:455: open archivo_inexistente.txt: no such file or directory
/enviar ../DirectorioPrueba/
ERROR: La ruta indicada pertenece a un directorio
/enviar ../Downloads/lubuntu-desktop-amd64.iso
ERROR: El archivo indicado supera el tamaño máximo permitido
/enviar output5k.dat
```

```
Se ha recibido un archivo con nombre "output5k.dat" que ha sido enviado por
Carlos1 [20-Dec 20:00]
¿Almacenar en el directorio actual el archivo recibido? (Y/N): Y
Ya existe un archivo con ese nombre en el directorio actual
Indique el nombre con el que quiere almacenar el archivo: nuevo5k.dat
Archivo "nuevo5k.dat" almacenado correctamente
```

Figura 7: Ejemplo de envío y recepción de un archivo

Como se puede apreciar el ejemplo de recepción de un fichero, al usuario receptor del mismo se le muestra el nombre del archivo recibido, además del nombre del emisor junto a la marca de tiempo de cuando fue enviado.

Los usuarios receptores podrán decidir si desean almacenar el fichero en el directorio actual o si por el contrario lo rechazan. Una vez recibido el fichero, dará la posibilidad de cambiarle el nombre, comprobando y no permitiendo su almacenamiento con un nombre ya existente en el directorio actual.

El archivo recibido en el caso de ser almacenado, por defecto se guardará con los permisos 644, es decir permisos de lectura y escritura para el usuario propietario de dicho sistema y solo lectura para el resto de usuarios.

4.2.4 Inclusión de marcas de tiempo

Una medida de protección simple pero eficaz que se les ha añadido a los mensajes de texto y a los mensajes de envío de archivos es la inclusión de marcas de tiempo, para dificultar los ataques de reinyección de tráfico en el flujo de comunicación (replay attack), donde los atacantes pretendan reintroducir mensajes cifrados capturados previamente, sin la necesidad de que el atacante conozca $K_{sesión}$ para realizar este tipo de ataque. Esa marca de tiempo añadida será mostrada a los receptores del mensaje, con el formato: [día-mes hh:mm].

Este ataque por ejemplo podría ser utilizado en el caso de que se tuviesen interceptadas las comunicaciones de uno o varios de los interlocutores, pero sin haber llegado a descubrir la clave de sesión, pudiendo en ese caso capturar el tráfico cifrado enviado por estos y posteriormente reintroducirlo en la comunicación para suplantar a uno de esos interlocutores, haciendo parecer que el usuario legítimo es el que realmente ha enviado ese mensaje en el caso de que no se aplicasen las medidas de protección.

Por ello, frente a este tipo de ataques, se han incluido marcas de tiempo, que indican en qué momento exacto fue enviado dicho fichero o mensaje. Esas marcas de tiempo siempre se cifran antes de ser enviadas y se descifran en su destino final, evitando que puedan ser alteradas por un atacante sin conocer $K_{sesión}$. Por lo tanto, en el caso de que un atacante reinyectase uno de esos paquetes con marcas de tiempo, el usuario podría comprobar que la hora de envío no corresponde con la actual y que sería la misma que la del mensaje original que se copió, en el caso de que el original le llegase a dicho usuario.

4.2.5 Generación de las claves asimétricas estáticas de identificación

En este sistema de comunicación desarrollado se hace uso de algoritmos de criptografía asimétrica y de certificados digitales para identificar y verificar a los distintos clientes. De este modo se podrá comprobar si dicho cliente es de confianza, además, se podrá verificar si la información crítica enviada por ese cliente es legítima y no ha sido alterada, evitando la suplantación de identidades dentro del grupo y la manipulación del protocolo criptográfico por parte de atacantes que tuviesen la posibilidad de interceptar el tráfico.

Cada cliente contará con un par de claves asimétricas distinto, ese par de claves no será obligado renovarlo cada cierto periodo tiempo, ya que solo servirá para identificar a dicho usuario y para firmar cierta información en la que sea necesario verificar la autenticidad e integridad. En ocasiones se hará referencia a dicho par de claves del usuario como: par de claves asimétricas estáticas (debido a que no cambiarán mientras se esté conectado) o con la notación $PubK_{estática}$ y $PrivK_{estática}$, según se trate de la clave pública o la privada de dicho par.

En ningún caso se utilizará ese par de claves estáticas para cifrar información importante. En este diseño implementado para el paso de claves a través de canales que puedan ser inseguros el usuario utilizará otro par de claves distinto a este, que será explicado en mayor detalle en el siguiente punto.

Con ese par de claves estáticas cada usuario generará y almacenará un certificado digital, que incluirá entre otros su $PubK_{estática}$, un conjunto de datos introducidos por el usuario que facilitarán su identificación y su firma, permitiendo comprobar que el certificado no ha sido manipulado durante su envío.

Como se ha comentado, uno de los usos que se le dará a ese par de claves estáticas y al certificado es el de identificar al cliente, para ello en el primer mensaje enviado al servidor se le adjuntará el certificado generado. El servidor, tras recibirlo, lo almacenará junto a otra información de ese usuario en su estructura en memoria que contiene información de los usuarios aceptados, donde dicha información será necesaria recuperarla posteriormente para que el administrador pueda realizar determinados procesos en el servicio. El administrador tras recibir el certificado, decidirá a partir de este si es de confianza y si le aceptará la petición de acceso al grupo.

Otro caso comentado, en el cual el cliente adjuntará en el mensaje enviado su certificado, es cuando ese cliente firme datos mediante su $PrivK_{estática}$ para otro cliente. Facilitando de este modo al cliente receptor la comprobación de la validez de la firma recibida y la obtención de sus datos de identificación, debido a que el certificado incluye, entre otros, su $PubK_{estática}$ y un conjunto de datos que lo identifican. Por ejemplo, esto se realizará cuando el administrador comparta $K_{sesión}$ firmada por él para un cliente en el cual confíe, teniendo que poder comprobar ese cliente receptor que la firma sea válida, para asegurarse de que no se trate de un atacante que esté intentando establecer sus claves al manipular la respuesta emitida por el administrador.

Una vez explicado qué son y qué uso se le dará, a continuación se explicará cómo generar el par de claves asimétricas estáticas y el certificado digital, siendo ambos necesarios para el uso de la aplicación cliente e incluso de la aplicación servidor.

Para ello se hará uso de la aplicación que ha sido desarrollada en Go, a la que se le ha llamado “CertificateGenerator". De este modo es bastante más sencillo y usable la generación de esos ficheros necesarios para los usuarios del sistema de comunicación. Además, reduce la probabilidad de errores por parte de los usuarios durante el proceso de generación del par de claves y del certificado, debido a que los genera a medida según los requisitos de la aplicación cliente y servidor. Evitando también que el usuario esté obligado a utilizar software de terceros menos usable y más complejo para esa finalidad. Teniendo el cliente que introducir en esta aplicación solo la información justa y necesaria, de un modo guiado durante el proceso e intuitivo.

Lo primero que hace la aplicación “CertificateGenerator" es generar el par de claves asimétricas. Como este sistema de comunicación utiliza el sistema criptográfico de claves asimétricas RSA, se generarán claves aleatorias RSA con un tamaño de clave de 2048 bits, siendo un algoritmo criptográfico y un tamaño de clave considerado actualmente como seguro.

El sistema criptográfico RSA (Rivest Shamir Adleman) es el más utilizado de su tipo, siendo un algoritmo que puede ser utilizado tanto para cifrar y descifrar como para firmar digitalmente. Este algoritmo basa su seguridad en el problema de factorización de grandes números enteros y se considera seguro debido a que actualmente no se conocen formas relativamente rápidas de descomponer un número grande en un producto de primos. (16)

Tras generar el par de claves RSA, la aplicación solicita una contraseña para almacenar ese par cifrado en disco, evitando con ese cifrado que un atacante pueda sustraer la clave privada del equipo y consiga utilizarla sin llegar a conocer la contraseña con la cual se cifró.

En la aplicación cliente y servidor antes de cargar $PrivK_{estática}$ se comprueba si está cifrada y en el caso de que no lo esté cargará la clave directamente. Pero para su creación mediante la aplicación desarrollada por defecto se solicita una clave no menor a 10 caracteres, para dificultar considerablemente los ataques de fuerza bruta que traten de adivinar la contraseña probando las distintas posibilidades.

Tras haber sido cifrado la clave RSA privada se almacenará en un fichero en formato PEM con el nombre por defecto de “Key.pem”, en el caso de que no exista un fichero con ese nombre en el directorio actual. PEM es un estándar que puede contener pares de claves asimétricas e incluso certificados completos, la información se incluye codificada en base64 y rodeada por cabeceras que identifican el tipo de contenido.

```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: AES-256-CBC,fce4dbab57363de0c30743fef11b6655

vZESlYRhXKPG02PzCFqP6kOiucyGh/+Fh6jLFWYPiUxCASnRhXAOtdmE7MWr6M1a
ADIiZW+74mVFllWVFTateHtWWYg6OUuai5gzsB3tYRJj33YMM9ArpHpS24zDTwkA
2B7QDAqMP6b0XAnK4Ur+yypFaP5KBTZSAXQfbUWf1I7aBMLIQ+t2gbZ9cSgRxzd/
z5FiUyhodb33ikw8BHIewa7NrQM46IacRmgk1pt3t4QjSK8Jqi5oKg4j4kJXcCsv
+MLixboCD/HuNTvnSoMOlMZqaL4xX0zPpHXFdmSWlcmSvEyn4kBsv2TyW5RQwGvo
...Más líneas codificadas en base64...
BWaQu5QvPtIj0lXmDhOtEXXZlSX2kBpNQ3HamWDky7OMghZoMbVZzWppd+DQFPDo
sehH2vJincRx5wj06jiJJIX0PgDvOfZiAxHsSGMbCsSdeXRj1WqL59TotPT80B51
nCAFzuIOV1QtOfXq+n1uI2jR86OJEAvcKYqKs1O4PCL1DrwOTUvCexYjUJSRmT+b
Lf5PENCXNp30kvNnXeaTnFAAIRvVjyu3Y4e6DU3YYtfPgTKaHvUYNmp0PSck35sX
wd1vz1DP/dtVI2+HAD1hCZXTe3KyQwWK7ynILJCI7of1PXPSj0SnjeXKBI0ZUHpQ
-----END RSA PRIVATE KEY-----
```

Figura 8: Ejemplo de clave RSA privada en formato PEM

En la figura anterior se puede ver un ejemplo de clave privada RSA generada y cifrada con la aplicación. También se puede comprobar que está cifrada y el tipo de cifrado por la cabecera previa a los datos codificados en base64. Para cifrar utiliza por defecto el algoritmo de cifrado simétrico AES en modo CBC (Cipher Block Chaining) con un tamaño de clave de 256 bits y un vector de inicialización de 128 bits, como función derivadora de claves se utiliza la función "deriveKey" (17) basada en el algoritmo de hash MD5, que en este caso la extiende hasta los 256 bits, debido a que es el tamaño de clave utilizado en esta situación para AES. En el modo CBC a cada bloque en plano antes de ser cifrado se le aplica el XOR del bloque cifrado previo, además para hacerlo único cada vez que se cifre se utilizará un vector de inicialización en el primer bloque.

Posteriormente se procede a generar el certificado del usuario, que contendrá entre otros $PubK_{estática}$ y datos que facilitarán su identificación. El estándar utilizado para la codificación de los certificados es X.509, un estándar utilizado en infraestructuras de clave pública ampliamente conocido, que da la posibilidad de codificar certificados de clave pública, permitiendo incluir campos con información identificativa del sujeto para el cual se emite y firmar el contenido para asegurar su autenticidad e integridad.

Por defecto para el certificado se utilizará como algoritmo de clave pública RSA, debido a que es el algoritmo utilizado en este servicio y con el cual se ha generado el par de claves RSA en el anterior paso. El tiempo de validez especificado por defecto es de 10 años tras el momento de su creación. Para el algoritmo de resúmenes hash por defecto se utilizará SHA512 (con un tamaño de salida hash de 512 bits), que será una de los algoritmos utilizados en la firma de este certificado.

SHA (Secure Hash Algorithm) es una familia de algoritmos criptográficos de resumen hash, a los cuales proporcionándoles una entrada de cualquier tamaño, devolverán como salida una cadena de longitud fija que identificará esa entrada y variará ante cualquier variación de la entrada por pequeña que sea, no pudiéndose obtener a partir de esa salida obtenida la entrada original.

```
victor@victor-K53SV:~$ ./bin/certificateGenerator
Clave RSA generada...
Introduzca una contraseña robusta para cifrar el archivo que contendrá la
clave RSA privada: P@ssW0rdPGC
Clave RSA cifrada y almacenada en "Key.pem"...

A continuación introduzca la información del usuario que será incluida en el
certificado
Nombre (CN): Víctor
Organización (O): UA
Unidad organizativa (OU): MUII
País (C): ES
Provincia/Estado (ST): C.Valenciana
Localidad (L): Alicante
Certificado X.509 generado...
Certificado almacenado en "Cert.pem"...
```

Figura 9: Ejemplo de generación de clave RSA y certificado X.509

Como se puede ver en el ejemplo de la figura anterior, la aplicación de generación de claves y certificados, tras generar y guardar la clave RSA, solicita al usuario información personal que será utilizada posteriormente para facilitar la identificación de dicho usuario cuando otro interlocutor reciba ese certificado.

Por último, el certificado recién generado será almacenado en disco sin cifrar mediante el estándar PEM con el nombre por defecto de "Cert.pem", en el caso de que no exista un fichero con ese nombre en el directorio. Lo guarda en un fichero en formato PEM, pero esta vez sin cifrar el contenido, debido a que el certificado no contendrá información crítica y a que el contenido de ese certificado debe poder ser interpretado por el resto de usuarios cuando sea enviado.

4.2.6 Generación y gestión de las claves asimétricas efímeras

En el sistema de comunicación desarrollado aparte de utilizarse claves asimétricas estáticas para identificar a los usuarios y para firmar, también se utilizan otras claves asimétricas distintas para cifrar información crítica para la seguridad y privacidad del sistema de comunicación.

Ese otro par de claves no se almacena nunca en disco y se genera aleatoriamente en ejecución cuando sea requerido, por lo que no serán estáticas y cambiarán cada periodo de tiempo y con cada ciertos eventos del servicio; por lo tanto, en ocasiones se hará referencia a estas claves como: par de claves asimétricas efímeras (debido a que son temporales) o con la notación $PubK_{efímera}$ y $PrivK_{efímera}$.

Ese par de claves efímeras se almacena en la memoria de ejecución volátil de la aplicación, donde se van sobrescribiendo las claves conforme se generan de nuevo cada vez. Por lo que las claves antiguas que ya no se utilizan quedarán sobrescritas en memoria y se perderán, no volviendo a tener acceso a la $PrivK_{efímera}$ antigua (que nunca es enviada a otros) para descifrar los mensajes que en un pasado fueron cifrados con la $PubK_{efímera}$ antigua, permitiendo cumplir con la propiedad de forward secrecy, que se explicará en mayor detalle en puntos posteriores. Por ejemplo, evitando de este modo que si el servidor es vulnerado y, tras su vulneración, va registrando las comunicaciones realizadas, este pueda llegar a descifrar las conversaciones registradas del pasado con una clave generada posteriormente, que es algo que podría suceder si se utilizasen las claves asimétricas estáticas para cifrar $K_{sesión}$.

La utilidad principal de las claves efímeras es compartir de manera segura las nuevas $K_{sesión}$ generadas a través de un canal que pueda ser inseguro, cifrando la clave de sesión con la $PubK_{efímera}$ del receptor, donde tras la renovación de las efímeras este perderá $PrivK_{efímera}$ y no se volverá a tener acceso a la clave privada necesaria para descifrar esa información secreta cifrada, dificultando que un atacante pudiese descifrar conversaciones del pasado.

Dicho par de claves efímeras de cada usuario se genera en dos situaciones distintas: una de ellas es cuando el cliente se conecta o reconecta al servidor y le tiene que pasar toda su información, mediante el mensaje con identificador "HELLO"; la otra situación es cuando se realiza un cambio de $K_{sesión}$ enviado por el administrador y se le envía posteriormente las nuevas efímeras generadas al servidor, a partir del mensaje con identificador "NEW_EPHEMERAL".

Cuando se generan nuevas claves asimétricas efímeras y se envía $PubK_{efímera}$ al servidor, este las almacena, sustituyendo las antiguas, si existiesen para ese usuario. Por lo tanto, el servidor va almacenando las últimas $PubK_{efímera}$ y la correspondiente firma de esas efímeras con $PrivK_{estática}$ del mismo usuario, para asegurar que dicha clave haya sido generada realmente por el usuario legítimo.

Esa información almacenada en el servidor sobre los usuarios aceptados donde se incluyen las $PubK_{efímera}$ y su firma, será enviada al administrador en conjunto cuando sea necesario realizar un cambio de clave de sesión, cifrándoles a cada usuario que considere de confianza la nueva $K_{sesión}$ generada con su $PubK_{efímera}$, previamente comprobando que su firma sea válida. Como se ha comentado anteriormente, tras recibir las nuevas claves los usuarios, generarán nuevas claves asimétricas efímeras y las enviarán al servidor para sustituir a las antiguas efímeras ya usadas.

Para ese par de claves efímeras aleatorias e independientes respecto al resto se utiliza el algoritmo RSA con un tamaño de clave de 2048 bits, al igual que con las claves asimétricas estáticas comentadas en el anterior punto. Para la firma se utiliza el algoritmo de resumen hash SHA512, sirviendo para que cuando el administrador reciba $PubK_{efímera}$ pueda comprobar que dicha clave pertenece al usuario esperado, para que un atacante no pueda intentar introducir una clave de la cual conozca su privada con la finalidad maliciosa de poder descifrar la información confidencial compartida.

4.2.7 Conexión con el servidor del servicio

El diseño realizado utiliza una topología de red centralizada, pasando por el servidor central todos los mensajes que sean enviados y recibidos por los clientes, actuando el servidor en determinados casos como un mero túnel que reenvía los mensajes de los clientes que se tienen que enviar punto a punto.

Aparte de la capa de cifrado con $K_{sesión}$ que se utiliza para cifrar las conversaciones entre los interlocutores del grupo ya comentada en anteriores puntos, también se utiliza una conexión cifrada par a par para la comunicación entre los distintos clientes y el servidor del sistema de comunicación.

Evitando de este modo que personas ajenas husmeen en dichas comunicaciones realizadas entre los clientes y el servidor, pudiendo estos llegar a conocer ciertos metadatos de importancia si no se cifrasen esas conexiones y tuviesen la capacidad de interceptarlas; aunque, de todas maneras, no serían capaces de conocer el contenido en plano de las conversaciones entre los distintos interlocutores, debido a que, además de esta capa de cifrado con el servidor, también existiría la otra capa de cifrado superior ya comentada que se encargaría de cifrar mediante $K_{sesión}$ los mensajes de texto y archivos enviados entre los interlocutores del grupo, no teniendo la capacidad el servidor de poder descifrar esa capa superior, pudiendo ser solo utilizada y descifrable por los clientes aceptados.

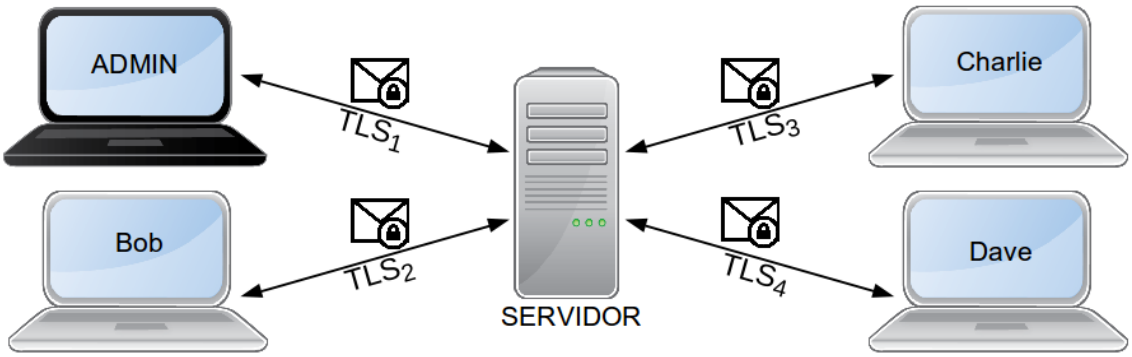

Figura 10: Diagrama de la capa de cifrado TLS con el servidor

Para dicha conexión cifrada con el servidor se utiliza el ampliamente conocido protocolo criptográfico TLS (Transport Layer Security) que es el sucesor de SSL; en concreto, se utilizará la versión 1.1 o superior para evitar versionas antiguas menos robustas y más propensas a ser vulneradas, además de permitir únicamente las suites de cifrado más robustas.

La suite de cifrado utilizada es "TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384" en este caso para el cifrado con el servidor. Como se puede identificar de esa cadena se utiliza AES en modo GCM (Galois Counter Mode) con un tamaño de clave de 256 bits para el cifrado de las comunicaciones tras haber establecido la clave simétrica, para los resúmenes hash se utiliza SHA384, como algoritmo de firmado se utilizará el algoritmo RSA y para el intercambio de la clave simétrica se hará uso de ECDHE que significa "Elliptic Curve Diffie-Hellman exchange with Ephemeral keys". Estando en la actualidad todos esos algoritmos con esos tamaños considerados como seguros.

Se utiliza el algoritmo de criptografía mediante curvas elípticas para el intercambio de la clave simétrica que se utilizará para cifrar esa capa de cifrado, debido a que, si está correctamente implementado, es considerado seguro y además tiene un mejor rendimiento en cuanto a costes de computación frente a otros algoritmos de criptografía asimétrica, como por ejemplo el algoritmo

criptográfico RSA, puesto que requiere claves de menor tamaño para ofrecer el mismo nivel de seguridad.

Además, la utilización de claves efímeras en ese proceso garantiza la propiedad de forward secrecy en esa capa de cifrado, protegiendo frente a posibles ataques de recolección de las comunicaciones por parte de un atacante externo para su posterior descifrado completo con una clave posterior a las utilizadas en ese momento para cifrar.

4.2.8 Estados del servidor y gestión de sus clientes

En el servidor, cuando los clientes se conectan son atendidos por hilos de ejecución diferentes, atendiendo a las peticiones recibidas de los clientes en cada uno de sus correspondientes hilos creados.

Otra tarea que realiza el servidor cuando se conecta satisfactoriamente un usuario es almacenar en una estructura (connMap) su IP y la relación de la conexión establecida con ese cliente, para tener todas las referencias a conexiones abiertas en un mismo lugar y, además, poder buscar de manera sencilla una conexión a partir de su IP. Cuando un usuario se desconecte o pierda la conexión con el servidor, este se borrará de dicha estructura, debido a que connMap deberá almacenar solo las conexiones activas.

En otra estructura también gestionada por el servidor (usersList) se almacena: el certificado X.509 proporcionado por el usuario (que contiene su $PubK_{estática}$), su fecha de acceso por primera vez, su dirección IP actual, la clave $PubK_{efímera}$ proporcionada y la firma electrónica de la misma mediante $PrivK_{estática}$. Permitiendo mediante la dirección IP del usuario relacionar ambas estructuras almacenadas en el servidor y obtener la conexión de ese usuario.

```
type User struct {
	Cert                  *x509.Certificate
	Address               string
	Registered            time.Time
	EphemeralPub          *rsa.PublicKey
	SignatureEphemeralPub []byte
}

var usersList list.List
```

Figura 11: Código de la estructura usersList del servidor

Esta información de ambas estructuras es de gran utilidad para el servidor a la hora de gestionar los usuarios, pudiendo consultar en cualquier momento su identidad y clave pública, también su clave efímera actual, en qué conexión se encuentra y si todavía sigue conectado, entre algunos de los usos posibles de esas estructuras de información. Esas dos estructuras mencionadas se almacenan en la memoria de ejecución, por lo que una vez finalizada la ejecución del servidor esa información no seguirá accesible y se perderá, no guardando ningún historial sobre estas.

En el caso de que se desconecten todos los clientes y el servidor siga activo, se reiniciará el estado de este, vaciando las estructuras con la información de los usuarios, volviendo a un estado idéntico al inicial; para ello borra todos los datos de los usuarios y reinicia todos los contadores del servicio. Esto se realizará debido a que siempre debe haber un usuario administrador que administre el grupo, haciendo que el usuario que se conecte tras el reinicio vuelva a ser considerado el primer usuario y, consecuentemente, el administrador.

Cuando los usuarios establecen la conexión con el servidor, estos siempre le entregan su certificado X.509, validándose siempre en el lado del servidor que el certificado sea válido y que no haya sido manipulado. Posteriormente, el servidor decidirá qué tipo de autenticación es requerida según el estado en el que se encuentra el cliente. Si es el primero en conectarse, será aceptado directamente, en el caso de que hubiese sido aceptado previamente podría tratar de reconectarse y por el contrario, si no es el primero ni ha sido aceptado en otras ocasiones, será necesario que solicite el acceso al administrador, siendo este el encargado de decidir si lo deniega o confía en él y le permite el acceso.

Para la designación del administrador del grupo, el servidor escogerá al usuario que esté conectado con mayor antigüedad, teniendo en cuenta que los usuarios de la estructura usersList se almacenan en orden de llegada, aunque también podría obtenerse la antigüedad mediante la fecha de registro por primera vez almacenada es esa misma estructura. Cada vez que el administrador comparta datos importantes es necesario que los firme y adjunte su certificado para que los receptores puedan comprobar que el administrador es legítimo, por si el servidor fuese vulnerado y escogiese un usuario con intenciones maliciosas.

4.2.9 Solicitud y paso de la clave de sesión común

En el caso de que el cliente que desee conectarse no sea el primero en entrar al grupo ni tenga la posibilidad de reconectarse (por haber sido aceptado en el grupo anteriormente), será necesario que ese cliente envíe una solicitud de $K_{sesión}$, que será reenviada al administrador para que decida si confía en él o no.

Esa solicitud de la clave de sesión contendrá su certificado, su $PubK_{efímera}$ recién generada y la firma electrónica de esta mediante su $PrivK_{estática}$. Tras ser reenviada esa petición con identificador "KEY_REQUEST" al administrador, este lo primero que hará es comprobar la autenticidad y validez de certificado recibido y de su $PubK_{efímera}$ a partir de sus firmas electrónicas para evitar intentos de engaño mediante su falseamiento. Si el certificado y clave efímera es válido, se le mostrará al administrador la información del cliente solicitante que viene incluida en dicho certificado, teniendo que decidir a partir de esa información mostrada si el cliente que realizo la solicitud es de confianza o no lo es.

```
KEY_REQUEST recibida:
-Nombre: Cristian2  -Correo electrónico: cristian2@fakemail.fake
-Localización: [ES][C.Valenciana][Alicante] -Organización y unidad: [UA][MUII]
-SHA512 del certificado:
ab078b90d107c71b79753dd1f4863f6173e1cfbde4eabcca0e092be3d0563c25
¿Confiar en Cristian2? (Y/N): y
```

Figura 12: Ejemplo de información mostrada al administrador sobre el solicitante

En el caso de que el administrador no confié en ese usuario y deniegue su solicitud, el solicitante será informado de su rechazo mediante un mensaje con identificador "DENIED" sin adjuntarle ninguna clave y el servidor cerrará su conexión, debido a que no habrá conseguido el permiso ni la clave necesaria para acceder al grupo.

En cambio, si el administrador confía en ese usuario y acepta su solicitud de paso de clave de sesión, se le enviará la respuesta a través del servidor con un mensaje con identificador "ACCEPTED". Esa respuesta enviada al cliente solicitante incluirá $K_{sesión}$ cifrada con la $PubK_{efímera}$ del solicitante, la firma del administrador del resumen hash de esa clave cifrada y el certificado del administrador, donde este último será verificado y servirá para mostrar información del emisor del mensaje cifrado y para obtener la $PubK_{estática}$ del administrador con la finalidad de poder comprobar su firma, siendo importante realizar dichas comprobaciones para garantizar que ha sido enviado por el administrador legítimo y no ha sido manipulado con intenciones maliciosas. Si en el momento de la solicitud el administrador todavía no tiene asignada ninguna $K_{sesión}$, la generará, debido a que puede darse el caso de que sea el primer y único usuario del servicio y todavía no la haya necesitado.

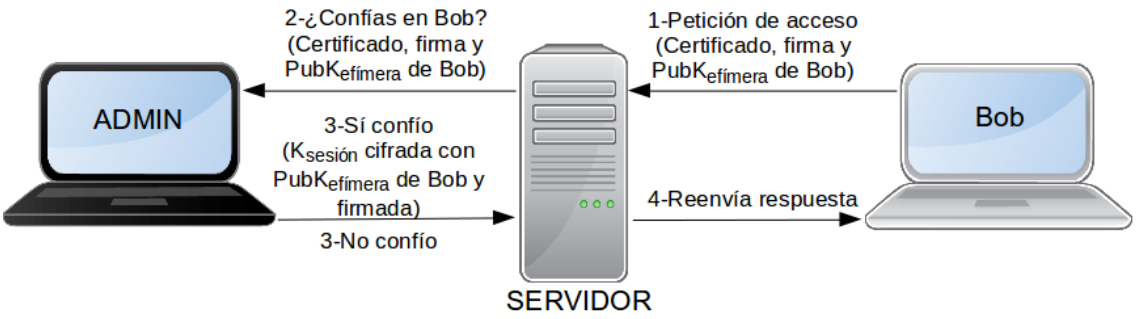

Figura 13: Diagrama de solicitud y paso de clave de sesión

Cuando el cliente solicitante reciba la respuesta del administrador, procederá a comprobar la validez del certificado y autenticidad de la firma adjuntada, en el caso de que sean válidas mostrará información del usuario que envió y cifró la clave de sesión común para asegurar que haya sido realizado por el administrador legítimo del grupo. Posteriormente descifrará mediante su $PrivK_{efímera}$ la $K_{sesión}$ que se incluye cifrada y almacenará su valor en su memoria volátil para cuando sea necesaria. Tras haber obtenido la nueva clave podrá descifrar las conversaciones de los interlocutores y también podrá participar en el grupo.

```
victor@victor-K53SV:~$  ../bin/client 127.0.0.1
La clave privada se encuentra cifrada, es necesario descifrarla para continuar
Introduzca la contraseña para descifrarla: P@ssW0rdPGC
Permiso de entrada CONCEDIDO
Clave aportada por el usuario:
-Nombre: Carlos1    -Correo electrónico: carlos1@fakemail.fake
-SHA512 del certificado:
c812b04d2c3fc6dd503a895d1efb408e32e3e91baf355dc1da88738fbfce19cf
Mensaje de Carlos1 [21-Dec 20:35]: Hola Cristian!!
```

Figura 14: Ejemplo de información mostrada al cliente tras ser aceptado

Como se ha comentado previamente, tanto el cliente administrador como el cliente solicitante, comprueban la validez de los certificados recibidos y muestran información del propietario, con la finalidad de poder verificar la identidad de la otra parte y asegurar que no haya sido manipulado. Esto es posible debido a que los certificados X.509 generados contienen información del propietario, su clave pública, un resumen hash del contenido firmado y otra información que puede ser valiosa.

Para las operaciones criptográficas mencionadas en esta implementación, siempre se usa (salvo algunos algoritmos para las conexiones cifradas con el servidor mediante TLS): para los certificados el estándar X.509, para el cifrado mediante algoritmos asimétricos se emplea RSA-OAEP (Optimal Asymmetric Encryption Padding), para el firmado RSA-PKCS#1v1.5 y en cuanto a las funciones de resumen hash se utiliza SHA512.

4.2.10 Reconexión de usuarios previamente aceptados

Los usuarios que hubiesen sido aceptados previamente y se hubiesen logrado conectar correctamente en un pasado, podrían en algunos casos concretos poder volver a reconectarse al sistema de comunicación sin tener que solicitar nuevamente la clave de sesión común al administrador del grupo.

Pero no siempre es posible reconectarse al sistema de comunicación, para que fuese posible se tendrían que cumplir una serie de requisitos obligatorios. El requisito principal es que ese usuario debe haber sido aceptado en un pasado en el grupo, estando todavía sus datos de identificación en las estructuras de usuarios del servidor. Tampoco puede haber sido revocado su acceso mientras se encontraba desconectado por decisión de alguno de los administradores del grupo. Otro requisito es que $K_{sesión}$ no debe haber cambiado desde que se desconectó la última vez del servicio.

Existen varios motivos por los cuales puede haber cambiado $K_{sesión}$ mientras el cliente estaba desconectado. Uno de los motivos posibles es que se haya producido una revocación de usuarios y otro es la posibilidad de que en el tiempo que haya estado desconectado se haya cambiado la clave debido a las medidas de forward secrecy implementadas en el servicio. En ambos casos los

usuarios desconectados son eliminados de las estructuras de información que almacena el servidor sobre los usuarios aceptados, debido a que un cambio de clave cuando un usuario no está conectado significa que no habrán podido recibir la nueva clave, por lo tanto, al no conocerla no podrán reconectarse.

Aparte de los requisitos comentados previamente, también es necesario comentar que el usuario que desee reconectarse debe recordar la clave de sesión común que estaba en uso antes de su desconexión. Dicha clave puede ser consultada mientras se esté conectado al servicio, mediante la instrucción reconocida por la aplicación cliente "/clave" (sin argumentos adicionales), tras introducir esa instrucción se mostrará $K_{sesión}$ actual codificada en base64, para evitar problemas con los caracteres especiales. En el caso de que el usuario que quiera reconectarse no la recuerde, deberá seleccionar la opción de que no la recuerda y volver a solicitar la clave al administrador del grupo.

El servidor reconoce que un cliente ha sido aceptado previamente en el momento de establecer la conexión, al comprobar que el certificado que proporciona ese cliente ya había sido aceptado por un administrador anteriormente.

Tras reconectarse el cliente satisfactoriamente y enviar los datos al servidor, este actualizará los datos del usuario en su estructura usersList, que ya ha sido comentada previamente en otros puntos. En dicha estructura del servidor se actualizará su dirección IP, por si ese usuario se reconectase desde un punto distinto o esa dirección fuese dinámica y hubiese cambiado, además de actualizar sus claves efímeras almacenadas con la nueva $PubK_{efímera}$ generada antes de la reconexión y su firma.

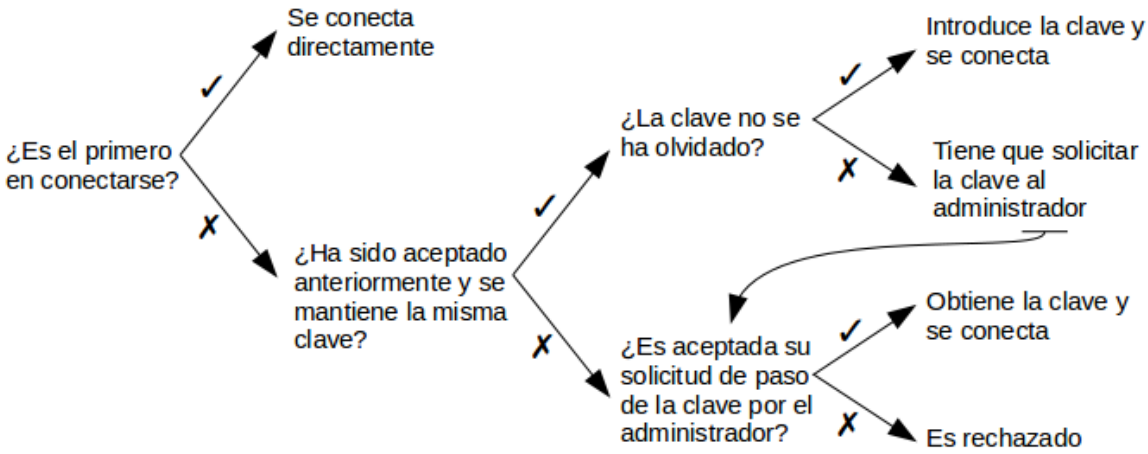

Figura 15: Diagrama de las distintas posibilidades de conexión

4.2.11 Revocación de usuarios previamente aceptados

Este servicio permite al administrador del grupo la acción de revocar a los usuarios que considere que deberían ser expulsados y no deberían poder participar o leer las conversaciones del resto de interlocutores.

Para realizar esa acción, el administrador deberá solicitarlo al servidor mediante la instrucción "/revocar" (sin argumentos), donde comprobará que realmente se trata del administrador del grupo y no está tratando de realizar esa acción otro usuario sin dicho privilegio. Además, el servidor comprobará que existan como mínimo 2 usuarios en el grupo (contando al administrador), ya que en ese caso no existiría la posibilidad de revocar a ningún usuario conectado.

Recordar que en este diseño finalmente implementado, la revocación de usuarios se realiza cerrando la conexión del revocado (no reenviándole más mensajes) y cambiando $K_{sesión}$ para que no pueda descifrar las conversaciones del resto de interlocutores por si incluso llegase a tener la capacidad de interceptarlas.

Si todo está correcto y cumple las condiciones comentadas anteriormente, el servidor le devolverá al administrador en un mensaje una estructura con la información imprescindible de todos los usuarios para esta operación, donde se incluye de cada uno: su certificado X.509, su $PubK_{efímera}$ actual y su firma. Permitiendo al administrador con esos datos comprobar su autenticidad y cifrarle a los no revocados la nueva $K_{sesión}$ mediante la $PubK_{efímera}$ proporcionada y pasársela a través de un canal posiblemente inseguro, al igual que se hace con las solicitudes de acceso aceptadas cuando un usuario trata de acceder al grupo.

Antes de enviar esa información de los usuarios obtenida de su estructura usersList, el servidor eliminará las entradas correspondientes a los usuarios que no estén conectados en el momento en el que se realiza esta acción, debido a que si no están conectados no les será posible a estos recibir la nueva $K_{sesión}$ y estarían obligados a volver a solicitarla realizando la conexión completa para la solicitud de $K_{sesión}$.

El administrador, tras recibir del servidor toda la información solicitada sobre los usuarios conectados, procederá a decidir qué usuarios revocará y cuáles no, pudiendo revocar múltiples usuarios a la vez. A los usuarios que no revoque tendrá que proporcionarles la nueva $K_{sesión}$ que haya generado aleatoriamente de nuevo; para esta finalidad, el administrador creará una estructura de datos donde añadirá para cada usuario no revocado: el certificado de ese usuario (para identificar esa posición a quién le corresponde), la nueva $K_{sesión}$ cifrada con la $PubK_{efímera}$ de dicho usuario y la firma por parte del administrador de la clave cifrada (para que pueda verificar que no ha sido manipulada). A los usuarios revocados, como no se les compartirá la nueva clave, les

añadirá solo su certificado, para que el servidor, tras recibir la respuesta del administrador a reenviar al resto de clientes no revocados, pueda revisar esa estructura y comprobar cuales ha revocado el administrador, cerrando las conexiones de los revocados y borrándolos de su usersList.

Tras realizarse el cambio de clave de sesión en el servicio, los clientes deberán generar nuevos pares de claves efímeras y enviarlas al servidor para que este contenga en sus estructuras las últimas $PubK_{efímera}$ generadas y pueda volver a proporcionarle al administrador la última versión de las efímeras de los usuarios en futuros cambios de clave.

Figura 16: Diagrama del proceso de revocación de usuarios

4.2.12 Renovación de claves periódicamente

En este servicio se ha implementado la propiedad de forward secrecy como medida de protección adicional para mejorar la privacidad y seguridad del mismo. La propiedad forward secrecy se basa en garantizar que en un sistema criptográfico, si se comprometen las claves utilizadas actualmente, no se comprometerán las anteriores claves usadas ni la correspondiente información que fuese cifrada con estas.

Evitando con esta protección adicional, que un atacante, en el caso de que consiga alguna de las claves secretas del servicio, pueda llegar a descifrar las conversaciones del pasado, al estar cifradas con claves anteriores que se van descartando y al no ser almacenadas en memoria no volátil, además estas claves temporales son generadas de manera totalmente independiente entre ellas, por lo que la obtención de un conjunto de estas no permitirá mediante su análisis obtener otras claves temporales utilizadas anteriormente o posteriormente.

De esta manera, con el objetivo de mejorar la seguridad y privacidad del servicio se asume que el sistema pueda llegar a tener alguna vulnerabilidad que permita a un atacante el filtrado parcial o

completo de alguna de esas claves temporales, existiendo también la posibilidad de que uno de los clientes de confianza sea vulnerado o engañado para proporcionar alguna de las claves utilizadas en ese momento.

En este servicio implementado, las dos capas de cifrado (capa inferior cliente-servidor y la capa superior entre todos los clientes aceptados) hacen uso de claves asimétricas temporales para compartir las claves simétricas con las que se cifrará finalmente la información enviada, pudiendo ambos clasificarse como criptografía híbrida, donde solo se cifra con las asimétricas para compartir la clave simétrica secreta, proporcionando la simétrica una velocidad de cifrado mucho mayor.

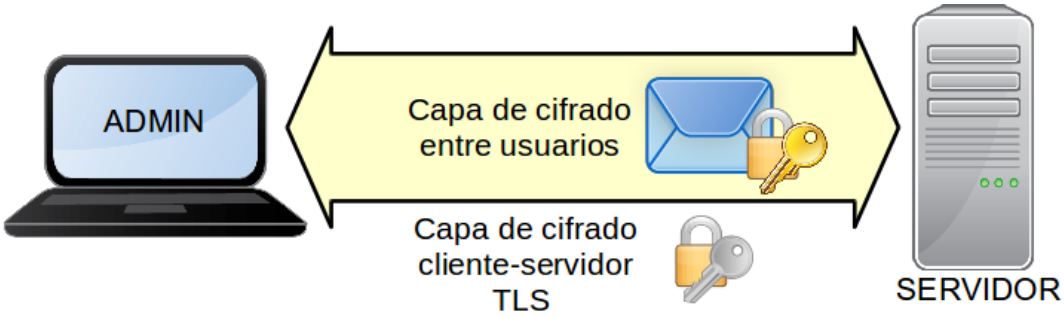

Figura 17: Diagrama de las capas de cifrado utilizadas en las comunicaciones

En la capa de cifrado inferior entre el cliente y el servidor (capa TLS), como se ha comentado anteriormente se utiliza una suite de cifrado que hace uso de ECDHE (en referencia a "Elliptic Curve Diffie-Hellman exchange with Ephemeral keys"), por lo que se utilizan claves nuevas en cada nueva conexión; asegurando, de este modo, la propiedad de forward secrecy en esta capa de cifrado.

En la capa de cifrado superior entre los distintos clientes aceptados, donde se utiliza $K_{sesión}$ para cifrar la información compartida entre los interlocutores y el par de asimétricas $PubK_{efímera}$ y $PrivK_{efímera}$ para compartir dicha clave de sesión a través de un canal posiblemente inseguro, son renovadas sus claves de forma periódica, asegurando de este modo la propiedad de forward secrecy para proteger las conversaciones de esta capa de cifrado.

El proceso de renovación de claves en la capa de cifrado superior, para cumplir la propiedad de forward secrecy, es muy similar al de revocación de usuarios comentado en el anterior punto, debido a que en ese proceso se cambia $K_{sesión}$ y también se cambian las claves asimétricas efímeras utilizadas para su cifrado; por lo que el proceso, donde el administrador solicita la estructura de información necesaria del resto de usuarios al servidor y este posteriormente devuelve otra estructura distinta con la información de los revocados y de los no revocados a los cuales se les pasa la nueva clave, es idéntico.

Una de las diferencias respecto al proceso de revocación es que esta funcionalidad no la invoca el administrador del grupo, sino que es llamada por un hilo de ejecución en el servidor donde se comprueba si el temporizador del tiempo transcurrido desde que se generó la anterior clave ha superado el valor especificado, siendo por defecto de 90 minutos y chequeándose si se ha superado dicho valor cada poco más de 10 minutos (no realizándose la renovación de claves cuando haya menos de 2 usuarios conectados). Se realiza en el servidor debido a que en determinadas situaciones los administradores podrían cambiar frecuentemente y no mantendrían el estado del temporizador entre ellos después de esos cambios que pudiesen suceder. Hay que tener en cuenta que el temporizador es reiniciado cada vez que la clave cambia, incluyendo, cuando en el proceso de revocación de usuarios la clave es cambiada y cuando el estado del servidor es reiniciado.

Resumidamente, se puede decir que en este sistema de comunicación se protege de manera robusta la seguridad de las claves antiguas y los datos cifrados con estas, al aplicar la propiedad de forward secrecy en sus dos capas de cifrado.

Por lo tanto, si un atacante externo quisiese descifrar las comunicaciones de los clientes con el servidor, este debería vulnerar la capa de cifrado TLS, no pudiendo de todas maneras descifrar las comunicaciones del pasado con la obtención de una clave generada posteriormente a esas comunicaciones.

En el caso de que el atacante quisiese descifrar las conversaciones entre los distintos interlocutores del grupo, tendría que descifrar la capa TLS establecida con el servidor más la capa superior de cifrado mediante $K_{sesión}$, la cual, en caso de obtención, tampoco permitiría el descifrado de conversaciones del pasado.

4.3 Resumen del proyecto desarrollado

En este punto se resumirá brevemente el sistema de comunicación desarrollado y se mostrarán ejemplos sencillos de su funcionamiento. Además, se enumerarán las medidas que han sido implementadas en dicho sistema con la finalidad de mejorar la seguridad y la privacidad del mismo.

4.3.1 Sistema de comunicación desarrollado

El sistema de comunicación propuesto ha sido desarrollado por completo mediante el lenguaje de programación Go, haciendo uso únicamente de su biblioteca estándar e implementándose conforme al diseño definitivo, propiedades y características explicadas anteriormente en este apartado.

Tras haber finalizado por completo la implementación del servicio de comunicación propuesto, este permite ejecutar el servidor del servicio y dejarlo a la escucha esperando a que se conecten los primeros clientes. Encargándose el servidor posteriormente del proceso de gestión de las conexiones con los distintos clientes, de redirigir los archivos y los mensajes enviados entre ellos (tanto los mensajes de texto como los de control) y de gestionar el estado del sistema de comunicación.

Para simplificar el proceso de creación de las claves asimétricas y certificados digitales requeridos y que los usuarios puedan identificarse y hacer uso del servicio, se ha desarrollado una aplicación para la generación de dichos archivos requeridos a medida (según los requisitos técnicos especificados), de una manera guiada para el usuario, permitiéndole fijar sus datos de identificación y la contraseña con la que se cifrará el archivo que contendrá su clave privada estática.

En el grupo de comunicación habrá usuarios sin privilegios especiales y un usuario administrador, siendo el administrador el encargado de gestionar la clave de sesión y compartirla entre el resto de usuarios en los cuales decida confiar. Aparte de decidir quién puede acceder al grupo, también puede revocar el acceso a usuarios del grupo que hayan sido previamente aceptados y cuyo permiso considere que deba ser revocado. Para el servicio diseñado siempre debe existir un administrador en cada grupo, en el caso de la desconexión del actual, otro usuario pasará a ocupar su lugar.

Los usuarios aceptados en el grupo (incluido el administrador) pueden comunicarse entre ellos mediante el envío de mensajes de texto o mediante el envío de archivos. Estos usuarios tienen la posibilidad de desconectarse y volver a conectarse al grupo, mientras no haya cambiado y recuerden la clave de la sesión, por lo tanto, teniendo que haberla solicitado de manera explícita previamente.

De manera resumida, este sería el funcionamiento del servicio de mensajería desarrollado, estando diseñado para ser seguro y privado (enumerándose de manera resumida en el siguiente punto las medidas implementadas para cumplir dichos requisitos).

A continuación, se mostrarán algunos ejemplos de ejecución sencillos y breves para que se aprecie mejor el funcionamiento del sistema de comunicación implementado (resaltando en morado la entrada de texto introducida por el usuario):

```
victor@victor-K53SV:~/server$ sudo ./bin/server
La clave privada se encuentra cifrada, es necesario descifrarla para continuar
Introduzca la contraseña para descifrarla: P@ssW0rdSeRVeR
Servidor a la espera de clientes...
Se ha conectado el 1er usuario 127.0.0.1:39553
Se ha concedido el acceso al usuario 127.0.0.1:39554
```

```
victor@victor-K53SV:~/client/C1$ ../bin/client 127.0.0.1
La clave privada se encuentra cifrada, es necesario descifrarla para continuar
Introduzca la contraseña para descifrarla: P@ssW0rdPGC
PRIMERO en entrar
KEY_REQUEST recibida:
-Nombre: Cristian2  -Correo electrónico: cristian2@fakemail.fake
-Localización: [ES][C.Valenciana][Alicante] -Organización y unidad: [UA][MUII]
-SHA512 del certificado:
ab078b90d107c71b79753dd1f4863f6173e1cfbde4eabcca0e092be3d0563c25
¿Confiar en Cristian2? (Y/N): Y
Bienvenido Cristian2.
Mensaje de Cristian2 [22-Dec 11:02]: Hola Carlos1.
```

```
victor@victor-K53SV:~/client/C2$ ../bin/client 127.0.0.1
La clave privada se encuentra cifrada, es necesario descifrarla para continuar
Introduzca la contraseña para descifrarla: P@ssW0rdPGC2
Permiso de entrada CONCEDIDO
Clave aportada por el usuario:
-Nombre: Carlos1    -Correo electrónico: carlos1@fakemail.fake
-SHA512 del certificado:
c812b04d2c3fc6dd503a895d1efb408e32e3e91baf355dc1da88738fbfce19cf
Mensaje de Carlos1 [22-Dec 11:02]: Bienvenido Cristian2.
Hola Carlos1.
```

Figura 18: Ejemplo de petición de acceso y posterior conversación

```
victor@victor-K53SV:~/client/C2$ ../bin/client 127.0.0.1
La clave privada se encuentra cifrada, es necesario descifrarla para continuar
Introduzca la contraseña para descifrarla: P@ssW0rdPGC2
Permiso de entrada CONCEDIDO
Clave aportada por el usuario:
-Nombre: Carlos1    -Correo electrónico: carlos1@fakemail.fake
-SHA512 del certificado:
c812b04d2c3fc6dd503a895d1efb408e32e3e91baf355dc1da88738fbfce19cf
/clave
La clave de la sesión actual es (Base64):
8kYTIr+Q8RIrfTtoU7v5QlzNPq6oTJiVMh2Oha2OtL0=
^C
victor@victor-K53SV:~/client/C2$ ../bin/client 127.0.0.1
La clave privada se encuentra cifrada, es necesario descifrarla para continuar
Introduzca la contraseña para descifrarla: P@ssW0rdPGC2
¿Recuerda la clave de sesión? (Y/N): Y
Introduce la clave de sesión (Base64):
8kYTIr+Q8RIrfTtoU7v5QlzNPq6oTJiVMh2Oha2OtL0=
Se ha RECONECTADO satisfactoriamente
```

Figura 19: Ejemplo de reconexión de un usuario previamente aceptado

4.3.2 Expectativas de seguridad y privacidad

El servicio de mensajería desarrollado está protegido frente a múltiples tipos de ataques que pretendan vulnerar la seguridad o privacidad del mismo. Uno de los aspectos que más se protegen es la seguridad de la clave de sesión y la privacidad de las conversaciones realizadas, cifrando para ello dichas conversaciones, además de cifrar también las conexiones entre el servidor y los clientes.

Aparte del cifrado de las comunicaciones, se verifica la identidad de las partes que intervienen y la información de mayor importancia es protegida, para prevenir suplantaciones, manipulaciones del contenido y demás usos maliciosos.

Otros de los aspectos que se han tenido en cuenta al diseñar el servicio son la posibilidad de renovar las claves de cifrado que se utilizan para cifrar las conversaciones cada cierto tiempo, de almacenar cifradas localmente las claves privadas y de revocar usuarios del grupo de manera segura.

A continuación, se enumerarán resumidamente algunas de las medidas implementadas en el proyecto para cumplir esos y otros aspectos de seguridad y privacidad:

- La clave de sesión que compartirán los interlocutores se pasa siempre cifrada punto a punto, cifrándola mediante criptografía asimétrica con la clave pública efímera del destinatario.
- La clave de sesión además de ir siempre cifrada punto a punto, también va siempre firmada por el administrador y se comparte únicamente bajo el consentimiento explícito del mismo.
- En la fase de compartición de la clave se muestra información verificable de ambas partes (del administrador y del solicitante de la clave) para asegurar que no haya sido suplantado.
- El servicio ha sido diseñado para que el servidor no necesite ni pueda conocer la clave de sesión compartida entre los usuarios, estando diseñado para que no pueda llegar a conocer la clave incluso ante la alteración del protocolo por parte del servidor para intentar forzar el paso de la misma.
- Los mensajes y archivos enviados entre los interlocutores van en todo momento cifrados con la clave de sesión precompartida, además se les añade una marca de tiempo con la fecha y hora con la finalidad de dificultar la realización de ataques de reinyección. Dicha marca de tiempo se añade en la parte cifrada para que no pueda ser alterada sin el conocimiento de la clave.

- Los mensajes recibidos no son almacenados y la clave de sesión es almacenada temporalmente en el espacio de memoria que le corresponde al proceso, dificultando su obtención en el caso de que el cliente sea vulnerado.

- Se utilizan dos capas de cifrado, una capa TLS para cifrar las comunicaciones entre el servidor y los clientes y otra capa superior en la cual se hace de uso de la clave de sesión precompartida entre los clientes para cifrar las conversaciones entre interlocutores.

- Solo se utilizan algoritmos, funciones, protocolos y estándares criptográficos considerados actualmente como seguros. Por ejemplo, en el caso de TLS solo se permiten las suites de cifrado más robustas y las versiones de TLS más modernas (versión 1.1 o superior), debido a que las versiones más modernas son menos proclives a ser afectadas por vulnerabilidades.

- Para la generación de números aleatorios se utilizan siempre generadores de números pseudoaleatorios criptográficamente seguros, que tengan la suficiente entropía como para utilizarse en las funciones criptográficas en las cuales sea necesario.

- Siempre que se recibe un certificado X.509 o un contenido firmado tiene que ser validado para asegurar que no ha sido manipulado.

- Tras la revocación de un usuario este será expulsado, cambiándose la clave de sesión y las claves asimétricas efímeras para que ese usuario expulsado no pueda descifrar las conversaciones futuras si llegase a tener acceso a estas.

- Cada cierto periodo de tiempo se renueva la clave de sesión con la que se cifran las conversaciones y también las claves asimétricas efímeras, por si un atacante por cualquier motivo llegase a conocerlas, no pudiese descifrar las conversaciones del pasado que hubiesen sido cifradas con otras claves anteriores. En la capa de cifrado TLS con una finalidad similar, se utilizan claves efímeras nuevas en cada conexión establecida.

- Los clientes nunca utilizarán sus claves asimétricas estáticas para cifrar información confidencial, estas solo serán utilizadas para firmar. Para el cifrado de las claves de sesión precompartidas punto a punto se hará uso de las claves asimétricas efímeras, que son generadas de manera aleatoria por cada usuario, se almacenan en memoria volátil y se van sobrescribiendo tras cada renovación.

- Las claves asimétricas estáticas utilizadas para firmar, podrán ser almacenadas de manera segura mediante su cifrado con una contraseña establecida por el propietario. Evitando de este modo que, en caso de ser sustraída, pueda ser utilizada para suplantar al propietario sin conocer dicha contraseña con la que se cifró.

5 Conclusiones

A continuación, se incluye una valoración de los resultados del proyecto realizado y un breve análisis sobre las posibles líneas futuras que se podrían seguir para mejorar la solución desarrollada actualmente.

5.1 Valoración de los resultados

En este proyecto llevado a cabo se ha logrado diseñar e implementar desde cero un sistema de comunicación cumpliendo con los requisitos propuestos inicialmente, dando como resultado un servicio de comunicación grupal funcional que incluye las características básicas de un servicio de mensajería en cuanto a funcionalidad y que ha sido diseñado y desarrollado para ser seguro y privado, teniendo en cuenta esas propiedades durante todo el ciclo de vida del proyecto para asegurar que se cumpliesen.

El sistema de comunicación diseñado y desarrollado implementa robustas medidas para proteger la seguridad y la privacidad de sus usuarios, de las conversaciones realizadas y del propio sistema, con la finalidad de evitar que puedan ser vulneradas por parte de un atacante.

Además de ser seguro, también ha sido diseñado para que sea eficiente en cuanto al consumo de recursos de procesamiento y de memoria y en cuanto al tamaño extra de las comunicaciones, permitiendo además que escale con mayor facilidad y pueda llegar a dar servicio a más usuarios por servidor. En gran parte esto es debido al diseño utilizado, basado en el cifrado de las conversaciones con una clave simétrica de sesión común a todos los interlocutores.

La utilización del sistema de comunicación implementado no requiere de elevados conocimientos técnicos por parte de sus usuarios ni del uso de aplicaciones de terceros para su utilización, debido a que trata de ser intuitiva, de no mostrar detalles técnicos innecesariamente y, además, incluye una aplicación para la generación a medida de las claves y certificados requeridos.

Por lo tanto, este sistema de comunicación puede ser de especial interés para aquellos individuos u organizaciones que estén especialmente concienciados con la seguridad y privacidad de sus servicios de mensajería o que traten con datos altamente confidenciales y no quieran confiar en las soluciones actuales del mercado, debido a que muchas de ellas no las consideren lo suficientemente seguras, no saben con certeza si cederán datos o metadatos a terceros o no pueden auditarlas correctamente debido a que su código no está disponible.

Ante el auge de este tipo de servicios y el aumento de la concienciación sobre la seguridad y la privacidad, a ese tipo de individuos u organizaciones comentados les puede ser de utilidad un servicio de mensajería que protegiese realmente la seguridad y la privacidad de sus usuarios.

5.2 Líneas futuras

Entre las posibles líneas futuras que, si bien quedan fuera del alcance de este trabajo, podrían resultar de interés para el proyecto, cabe destacar las siguientes:

- Diseñar e implementar interfaces gráficas para la aplicación cliente: que sean sencillas, intuitivas, faciliten el uso de la aplicación a los usuarios y la haga más atractiva; que muestren toda la información necesaria y no limiten las funcionalidades disponibles para los clientes. Para esta finalidad se ha pensado en el desarrollo de una interfaz web, que permitiría su uso en las distintas plataformas sin la necesidad de realizar modificaciones para portarla.

- Añadir la funcionalidad de realizar videoconferencias, permitiendo la posibilidad de realizar videoconferencias cifradas entre los distintos interlocutores del grupo, además también se podría añadir la posibilidad de compartir el escritorio propio con el resto de interlocutores. Habría que estudiar el ancho de banda que consumiría esta funcionalidad y analizar que calidades de vídeo y audio habría que ofrecer y si sería recomendable ajustar esas calidades automáticamente según demanda.

- Portar la aplicación de mensajería a las plataformas móviles Android e iOS: el sistema ha sido desarrollado en Go, siendo actualmente este un lenguaje que no soporta oficialmente la creación de aplicaciones nativas para las plataformas móviles Android e iOS. Existen soluciones experimentales (con numerosas limitaciones) que podrían servir para portar una aplicación al completo en Go o partes de la aplicación, mediante bibliotecas desarrolladas en Go enlazadas con otros lenguajes, como por ejemplo Java para Android o Objective-C para iOS. (18) (19) Habría que estudiar con mayor detenimiento que opción escoger, si intentar portarla en Go a móviles o portarla en otro lenguaje, en esta segunda opción pudiendo escoger una plataforma de desarrollo para móviles que permitiese compilar para ambas plataformas móviles.

- Estudiar y desarrollar en mayor profundidad el modelo de negocio: en apartados anteriores se han estudiado superficialmente diversas opciones con las que se podría monetizar el sistema de comunicación. Posteriormente se deberían realizar estudios más extensos y detallados sobre esos modelos de negocio y escoger definitivamente los que se llevarían a cabo, para después diseñarlos e implantarlos.

6 Bibliografía y referencias

1. **TNS Opinion & Social.** E-Communications and the Digital Single Market. http://ec.europa.eu/COMMFrontOffice/publicopinion/index.cfm/Survey/getSurveyDetail/instruments/SPECIAL/surveyKy/2062.

2. **Marlinspike, Moxie.** Open Whisper Systems - Advanced cryptographic ratcheting. https://whispersystems.org/blog/advanced-ratcheting/.

3. **WhatsApp.** WhatsApp Security. https://www.whatsapp.com/security/.

4. **CCN-CERT.** IA-21/16 - Riesgos de uso de WhatsApp. https://www.ccn-cert.cni.es/informes/informes-ccn-cert-publicos/1746-ccn-cert-ia-21-16-riesgos-de-uso-de-whatsapp/file.html.

5. **Telegram.** MTProto Mobile Protocol. https://core.telegram.org/mtproto.

6. **Jakobsen, Jacob y Orlandi, Claudio.** Aarhus University - On the CCA (in)security of MTProto. https://eprint.iacr.org/2015/1177.pdf.

7. **ProyectosAgiles.org.** Desarrollo iterativo e incremental. https://proyectosagiles.org/desarrollo-iterativo-incremental/.

8. **Griesemer, Robert, y otros.** Open Source Blog - Hey! Ho! Let's Go! https://opensource.googleblog.com/2009/11/hey-ho-lets-go.html.

9. **GoClipse.** Goclipse - GitHub Documentation. https://goclipse.github.io/.

10. **Google.** Google Cloud Platform - Products. https://cloud.google.com/products/.

11. **Amazon Web Services.** AWS - EC2 details. https://aws.amazon.com/es/ec2/details/.

12. **Amazon Web Services.** AWS - AWS SDK for Go. https://docs.aws.amazon.com/sdk-for-go/v1/developer-guide/welcome.html.

13. **Go team.** Go Project - Crypto/Rand Package. https://golang.org/pkg/crypto/rand/.

14. **Lacharme, Patrick, y otros.** The Linux Pseudorandom Number Generator Revisited. https://eprint.iacr.org/2012/251.pdf.

15. **Microsoft.** Windows Dev Center - CryptGenRandom function. https://msdn.microsoft.com/en-us/library/windows/desktop/aa379942(v=vs.85).aspx.

16. **Lucena López, Manuel José.** Criptografía y Seguridad en Computadores. 4-0.11.0. 2015.

17. **Go team.** Go Project - PEM_Decrypt/deriveKey function. https://golang.org/src/crypto/x509/pem_decrypt.go#L79.

18. **Dogan, Jaana Burcu.** GitHub - Go mobile subrepository. https://github.com/golang/go/wiki/Mobile.

19. **Kim, Hana.** GopherCon Talks - Go on Mobile. https://talks.golang.org/2015/gophercon-go-on-mobile.slide.

20. **Open Whisper Systems.** Signal App. https://whispersystems.org.