Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE GRADO

Envío seguro de información a través de aplicaciones de mensajería móvil

Grado en Ingeniería Mecánica

ALUMNO: Pablo González Álvarez

DIRECTOR: Norberto Fernández García

CURSO ACADÉMICO: 2019-2020

Universidade Vigo

Centro Universitario de la Defensa
en la Escuela Naval Militar

TRABAJO FIN DE GRADO

Envío seguro de información a través de aplicaciones de mensajería móvil

Grado en Ingeniería Mecánica

Intensificación en Tecnología Naval

Cuerpo General

Universidade Vigo

RESUMEN

En la actualidad, la necesidad de comunicación entre miembros de unidades militares o entre cuerpos y fuerzas de seguridad es innegable. Para llevar a cabo estas comunicaciones, en el ámbito de las Fuerzas Armadas Españolas existen numerosos sistemas de mensajería. Estos sistemas se consideran seguros, pero en muchos casos pueden no resultar cómodos y rápidos o no estar disponibles para los usuarios puesto que, por ejemplo, en algunos casos requieren terminales específicos para su uso. Por otro lado, existen multitud de aplicaciones de mensajería comerciales tales como Whatsapp o Telegram, de una gran popularidad, facilidad de uso y accesibilidad, pero sobre las que nos pueden surgir dudas acerca de las condiciones de seguridad que ofrecen.

Teniendo en cuenta lo anterior, en el presente trabajo se pretende desarrollar una aplicación de envío y recepción de mensajes que aúne seguridad y accesibilidad. El principal objetivo no es otro que el de conseguir enviar información a través de aplicaciones de mensajería instantánea comerciales, que aportan comodidad y rapidez, añadiendo sobre ellas una capa extra de seguridad, incluyendo aspectos tales como cifrado de la información, y mecanismos para proteger su integridad. El resultado final, al que denominaremos MILChat (Military Chat o en español Chat Militar), no pretende ser un medio de comunicación de información crítica, pero sí podría servir para intercambiar de manera segura información relacionada con el hacer diario en el ámbito militar y policial.

PALABRAS CLAVE

Aplicación, mensajería instantánea (IM), Android, seguridad, cifrado

AGRADECIMIENTOS

En primer lugar, agradecer a mis compañeros de promoción el haberse prestado voluntarios para probar la aplicación y ejercer un juicio crítico, así como en especial a mis compañeros de camareta por solventarnos dudas mutuamente a medida que salen de la cabeza en nuestros respectivos proyectos.

Al director del proyecto, Don Norberto Fernández García, por apoyar la idea que surgió de mi corta experiencia en la Armada y darle forma hasta diseñar un trabajo alcanzable, así como ayudarme a conseguir los objetivos marcados y por ser comprensivo con la difícil situación vivida por esta promoción en las últimas semanas.

Al CC Don Diego Mejías Mendoza, por orientarme en el ámbito de las comunicaciones y ayudarme a recabar información y documentación sobre el proyecto y siempre prestarse a resolver cualquier duda que pudiese surgir. Además, gracias por ser el inspirador de la idea fundamental del trabajo.

Por supuesto, agradecer a mi familia, sobre todo a mis padres, Pablo e Isabel, el apoyo incondicional a lo largo de toda mi vida y en especial en el período de estancia en la Escuela Naval Militar.

A mi hermana, Alba, y a mi abuela, Maribel, por prestarme toda su energía y aconsejarme siempre que lo he necesitado.

A mis amigos, por siempre estar ahí y recordarme quién soy y de dónde vengo.

A todos, muchísimas gracias.

CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1 INTRODUCCIÓN Y OBJETIVOS

En este capítulo se introducen los motivos que impulsaron el desarrollo de este proyecto, así como la explicación de algunas decisiones tomadas previamente al desarrollo del mismo. Para finalizar, se exponen los objetivos a alcanzar y la organización de la presente memoria.

1.1 Introducción y motivación

En un contexto militar o de fuerzas y cuerpos de seguridad, no es un mal excepcional, ni mucho menos poco frecuente, el uso de aplicaciones comerciales de mensajería instantánea en dispositivos móviles de forma inadecuada.

A menudo tenemos la necesidad de compartir información de un determinado nivel de seguridad de forma cómoda, rápida, sencilla y accesible y, para ello, nos valemos de sistemas de mensajería al que hoy en día la gran mayoría de personas tienen acceso. El problema es que esta práctica no es segura. Los terminales móviles individuales y privados de cada usuario pueden estar comprometidos y, por supuesto, no están acreditados como sistema de comunicaciones en el ámbito militar.

Está claro que en este ámbito encontramos numerosos sistemas de comunicaciónes para llevar a cabo dicha práctica, pero no siempre son, haciendo alusión a la descripción anterior, cómodos, rápidos, sencillos o accesibles. Suele ser tedioso y largo el envío de cualquier tipo de información, ya sea texto, archivos o, a mayores, contenido multimedia y, por ello al final, estos sistemas suelen caer prácticamente en desuso.

Además, dejando a un lado que las aplicaciones determinadas usadas por cada individuo no están acreditadas como sistema de comunicaciones militar, estas no son ni mucho menos confiables y seguras. Algunas, como veremos más en detalle, usan cifrado desde hace relativamente poco tiempo y otras ni si quiera lo usan, de modo que nuestras comunicaciones pueden ser fácilmente interceptadas.

Asimismo, Android [1] es a día de hoy un sistema operativo usado por millones de usuarios en todo el mundo. No resulta utópico pensar que, dado el uso habitual de esta plataforma por todo tipo de personas, sería muy útil y accesible tener algún tipo de aplicación que nos permitiese comunicarnos de manera confiable en este entorno. Y ahondando más en el sistema informático, sería beneficioso para los usuarios que se tratase de una interfaz similar o parecida a la habitual en aplicaciones de mensajería como Whatsapp o Telegram.

Cabe destacar que en este proyecto no se pretende crear una alternativa que sustituya totalmente a las redes o sistemas de comunicaciones que ya se encuentren implementados en un determinado

ejército. Simplemente, se trata de complementar a éstas y servir como una herramienta eficaz y rápida para mensajes o conversaciones de menor entidad o importancia. Es decir, se pretende conseguir un intercambio de mensajes de cierta confidencialidad o sensibilidad de un modo intuitivo y ágil sin pérdidas de tiempo en sistemas más tediosos ni distracciones de otros quehaceres del propio trabajo.

Por ello, el proyecto se centra en abordar las limitaciones de seguridad que poseen los sistemas de mensajería instantánea de uso común, dejando a un lado algunas vulnerabilidades que derivan del hecho de que los dispositivos móviles individuales de cada usuario puedan verse comprometidos. Este tipo de deficiencias no han sido explícitamente abordadas, pero se tienen en cuenta en el desarrollo del trabajo y se contemplan en el apartado 5 de esta misma memoria.

Por todo lo anterior, se ha decidido proceder en este proyecto de la forma explicada en el apartado 1.3.

1.2 Elección del sistema operativo

Existen varios sistemas operativos (SO) en los que es posible desarrollar la aplicación que tiene por objetivo este proyecto. Tras un análisis de los principales SO, se ha decidido implementar nuestra *app* (aplicación) en *Android*.

Apple, por su parte, no permite desarrollar aplicaciones y publicarlas en su *App Store* [2] (tienda de la empresa Apple donde se pueden descargar las aplicaciones de dicho OS) de forma gratuita. La tienda de aplicaciones de Android, Google Play Store [3], tampoco es gratuita para desarrolladores, aunque, en este último sistema operativo, sí que se pueden instalar aplicaciones que no se encuentren publicadas en dicha tienda a diferencia de en iOS (Iphone Operative System).

Asimismo, se han tenido en cuenta otros SO como Windows Phone, pero el dominio de los dos mencionados anteriormente sobre los demás es aplastante, tal y como muestran estudios recientes [4] sobre qué SO es el más influyente. Como se puede ver en la Figura 1-1, Android es el SO dominante, por ejemplo con un 61 % en Europa y un 84 % en Latinoamérica, sobre un 39 % y un 16 % ostentado por iOS respectivamente. También es cierto que, en América del Norte, con un 48 % frente a un 52 % y en China, con un 42 % frente a un 58 %, de Android frente a iOS respectivamente, el sistema operativo de Google se ve superado, pero por un margen muy escaso.

Por lo que se ha mencionado anteriormente, el presente proyecto se centrará en la plataforma Android, descartando programar en el sistema operativo iOS, simplificando así el desarrollo del mismo, que solo se centrará en un sistema operativo.

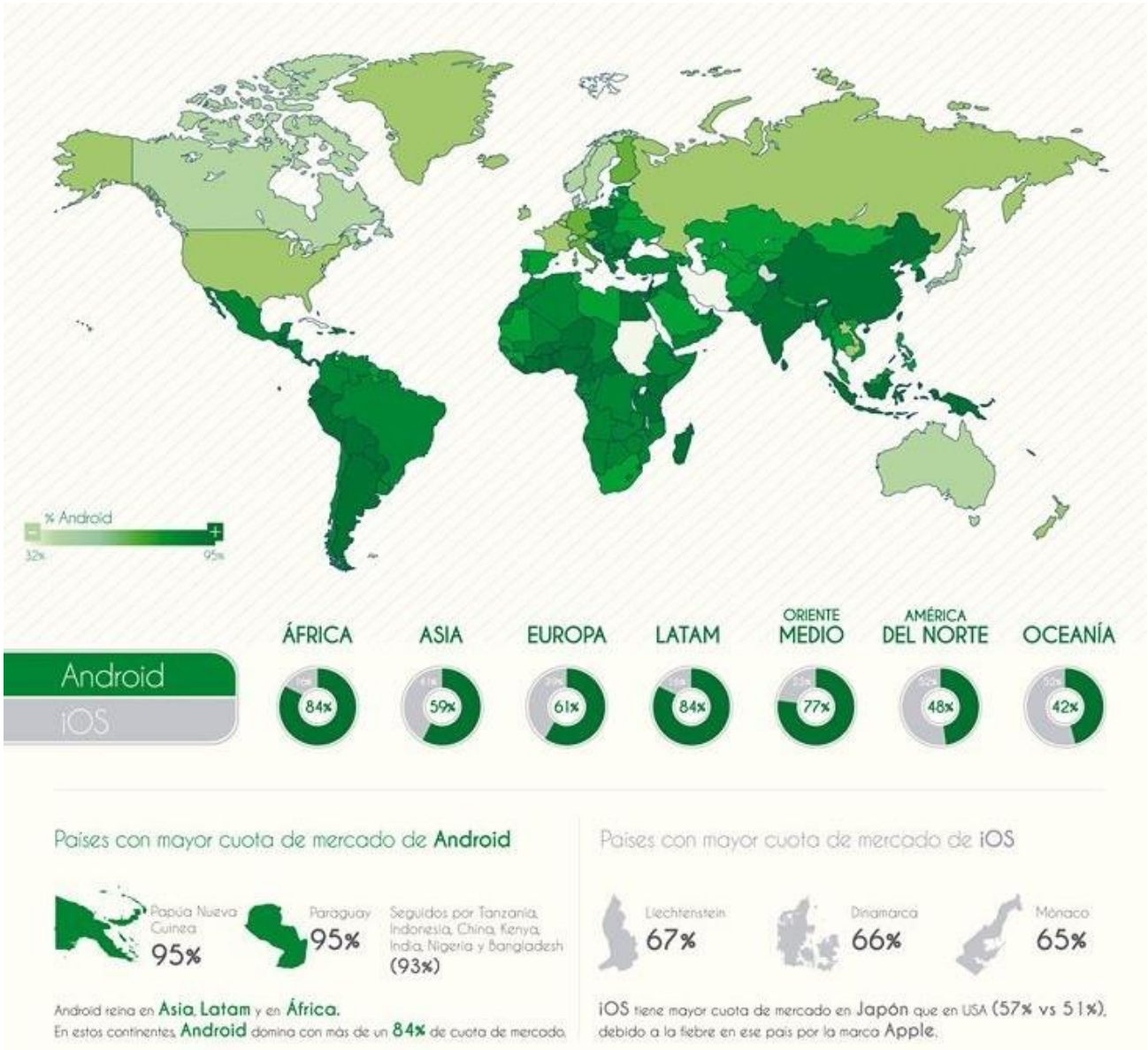

Figura 1-1 Comparación Android vs iOS [4]

1.3 Objetivo del proyecto

El objetivo del presente proyecto es el de conseguir enviar información a través de un sistema de mensajería móvil de forma segura. Para ello, elaboraremos una aplicación de edición de texto, a la que llamaremos en adelante MILChat, para el sistema operativo Android en la que se podrá redactar un determinado mensaje. Tras ello, mediante mecanismos de encriptación, se codificará el mensaje anteriormente redactado para ser enviado, de manera segura, a través de una aplicación de mensajería móvil comercial. El destinatario del mensaje lo recibirá en su dispositivo móvil particular a través de dicha aplicación de mensajería comercial y podrá descifrarlo a través de la misma aplicación de edición de texto MILChat.

También se pretende hacer un breve repaso de las aplicaciones de mensajería instantánea que se encuentran actualmente en el mercado. Tanto de las más usadas y conocidas como de las que están más enfocadas a ofrecer un entorno seguro. De esta forma, se intentará crear al lector cierto criterio

para elegir qué aplicación usará finalmente para el envío de información a través de la aplicación que desarrolla el presente trabajo.

1.4 Organización y estructura de la memoria

El trabajo que nos ocupa se organiza y estructura como se explica a continuación:

- **Capítulo 1.** Introducción y objetivos. Se exponen los motivos de la realización del proyecto, la explicación de algunas decisiones previas y los objetivos a cumplir.

- **Capítulo 2.** Estado del arte. Se da una breve introducción histórica sobre los orígenes de la mensajería instantánea y sus características principales, además de hacer una reseña sobre las actuales aplicaciones principales de mensajería instantánea y las herramientas usadas en el proyecto.

- **Capítulo 3.** Desarrollo del TFG. Se explican los requerimientos del trabajo y se dan unas nociones básicas sobre *Android*, necesarias para la realización del proyecto, además de terminar explicando el proceso de implementación de la aplicación.

- **Capítulo 4.** Resultados y pruebas. Se detalla cómo se ha llevado a cabo la instalación de la aplicación y se hace un resumen del funcionamiento de esta, finalizando con las pruebas que se le han hecho para comprobar su correcto funcionamiento.

- **Capítulo 5.** Conclusiones y líneas futuras. Por último, se hace un resumen de las conclusiones obtenidas al acabar el trabajo y se dan algunas indicaciones sobre posibles líneas futuras a seguir tras este proyecto.

2 ESTADO DEL ARTE

En este capítulo se dará una pequeña reseña histórica acerca de la evolución de la mensajería instantánea y sus características principales. También se hará un repaso de las aplicaciones de mensajería instantánea comerciales más usadas, conocidas en inglés como *instant messaging*, así como las que están más enfocadas a requisitos de seguridad. Por último, se hará un breve repaso de las herramientas utilizadas para la realización de nuestro proyecto.

2.1 Contexto histórico y funcionamiento

2.1.1 Orígenes históricos

La mensajería instantánea, conocida en inglés como *instant messaging* cuyo acrónimo es IM es una forma de comunicarnos en tiempo real entre uno o varios usuarios a través de texto, el cual es enviado a través de una red como Internet. Su principal diferencia con el correo electrónico es que este no ocurre en tiempo real, sino que está pensado para la comunicación asíncrona (las partes en comunicación no tienen que estar activas a la vez), mientras que las *apps* de mensajería instantánea están pensadas para comunicación síncrona, en las que las dos partes sí que están usualmente activas a la vez (conversando).

En la década de 1970, hicieron su primera aparición los primeros sistemas de mensajería intantánea, implementados en el sistema PLATO (Programmed Logic Automated Teaching Operations, en español, Lógica Programada para Operaciones de Enseñanza Automatizadas) [5], cuya terminal se puede ver en la Figura 2-1. Más tarde, en las décadas de los 80 y 90 fueron lanzados en sistemas Linux, primeramente, y luego en otros sistemas operativos, para que fueran usados por todo el mundo a través de Internet. El primero en ser utilizado ampliamente fue ICQ (*I seek you*, en castellano, Te busco) [6]. A partir de aquí, hicieron aparición numerosos sistemas distintos con protocolos propios o compartidos. Cabe destacar algunos muy extendidos en la primera década del siglo XXI como las distintas variantes de Windows Messenger, desarrollado por Microsoft [7], como podemos observar en la Figura 2-2.

Con la aparición de los móviles y, particularmente, con la llegada de los móviles inteligentes o *smartphones*, los sistemas de mensajería fueron incluidos en dispositivos, llevando a los IM en computadoras a la extinción gradual y permitiendo que las personas por medio de su dispositivo móvil pudieran comunicarse en formato de texto con cualquier parte del mundo.

Figura 2-1 Terminal PLATO [5]

Figura 2-2 Interfaz Windows Live Messenger [7]

En la actualidad, no solo es posible la compartición de texto, sino que además se puede compartir contenido multimedia como fotografías, vídeos o audios, además de tener la posibilidad de gestionar llamadas o videollamadas a través de estas aplicaciones mediante el uso de Internet. Además, son muchas las empresas que usan este tipo de herramientas asiduamente en sus obligaciones diarias, así como también se ven potenciadas en el ámbito educativo.

2.1.2 Funcionamiento básico

La mensajería instantánea "es un servicio de comunicación en tiempo real entre dispositivos como computadoras, tabletas, celulares, etc" [8]. La IM se basa en el uso de programas conocidos como clientes de IM que se instalan en un dispositivo. Tanto receptor como emisor deben tener instalada la aplicación para poder enviarse mensajes entre sí.

La comunicación, de un modo resumido y simple [9], se sucede en varios pasos, como se puede ver en la Figura 2-3:

1. Se produce un inicio de sesión en un cliente de IM. De esta forma conseguimos autenticación y fiabilidad de quién nos envía un mensaje ya que nuestro usuario será único.
2. El cliente de IM se conecta a un servidor usando Internet y un determinado protocolo de comunicación como Skype, XMPP, ICQ, MSNP [10], etc.
3. El servidor o cliente verifica nuestra identidad y crea un registro de nuestra conexión y nuestros contactos.
4. El cliente de IM comprueba los usuarios en línea (usuarios también con sesión iniciada y disponibles para comunicación al mismo tiempo) y le da esa información al usuario.
5. Se selecciona el contacto al que se quiere enviar un mensaje, se escribe y se manda dado que el servidor conoce la dirección IP y el puerto de destino.
6. La comunicación se sucede de forma bilateral el tiempo que sea necesario siguiendo el mismo proceso.

7. Una vez se cierra el cliente, esta información se comunica a los demás usuarios que nos tengan en su lista y nuestro servidor destruye el registro creado cuando nos conectamos.

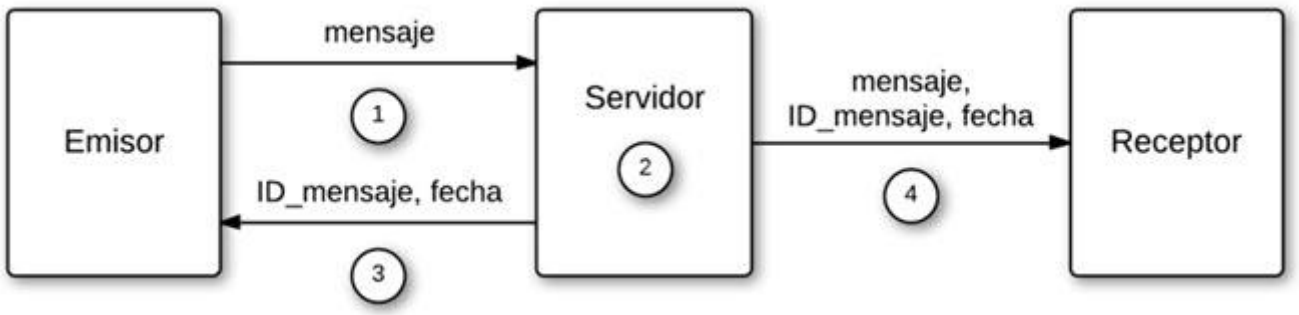

Figura 2-3 Esquema funcionamiento IM [9]

2.2 Parámetros de seguridad

Como se ha tratado en la asignatura "Fundamentos de redes de ordenadores" [11], existen una serie de parámetros de seguridad que debe cumplir cualquier red en un ciberentorno. Estos son totalmente extrapolables a un sistema de mensajería instantánea y, además, en un IM de un ámbito militar son claramente prioritarios.

Se definen como propiedades de seguridad las siguientes:

- **Confidencialidad:** solo los usuarios que pretenden comunicarse entre sí deben tener acceso a la información.
- **Integridad:** el contenido de la comunicación no debe verse alterado por terceras partes ajenas a los comunicantes.
- **Autenticidad:** tanto el emisor como el receptor deben poder confirmar la identidad de la otra parte.
- **No repudio:** una vez enviada la información, el emisor no debe poder negar que él originó el mensaje.
- **Disponibilidad:** las partes que desean comunicarse deben poder disponer de los recursos necesarios para hacerlo.

Cabe destacar que garantizar el cumplimiento de todos estos parámetros no es sencillo y, a menudo, existen problemas o fallos que no garantizan una seguridad total. En cualquier caso, si queremos obtener un sistema de mensajería seguro, debemos optimizar al máximo todos estos parámetros y conseguir que estos huecos o errores de seguridad sean mínimos.

Además, según diversas fuentes expertas en análisis de aplicaciones de mensajería instantánea [12], debemos contemplar algunas propiedades adicionales para determinar si podemos considerar una aplicación segura.

- **Cifrado extremo a extremo:** Puede ser el parámetro más importante ya que implica que los mensajes de chat se codifiquen de forma que solo remitente y destinatario o destinatarios puedan disponer de las claves necesarias para desencriptar el mensaje.

- **Código fuente abierto:** Consiste en dejar accesible el código fuente de la aplicación de forma que expertos puedan evaluar la aplicación y realizar auditorías, además de que también cualquier persona pueda contribuir a mejorar el código de dicha IM. Aunque se puede pensar que por culpa de la ingeniería inversa [13], esto pueda resultar contraproducente, no resulta así dado que es una gran ventaja que se pueda analizar el código de la aplicación para facilitar la detección y corrección de agujeros de seguridad.

- **Recopilación de datos:** Si bien es algo habitual en aplicaciones sin cifrado extremo a extremo, también se puede dar en las que lo tienen. Muchas aplicaciones recopilan metadatos, que son una huella digital electrónica que dejamos inevitablemente al mandar un mensaje (hora del mensaje, datos de contactos, dirección IP, etc.).

2.3 Guía de seguridad de las TIC

El Centro Criptológico Nacional (CCN), elabora y redacta una serie de publicaciones, en su mayoría sin clasificar, que introducen doctrina acerca del empleo de las tecnologías de la información y la comunicación (TIC). En nuestro caso, hemos consultado varias publicaciones que resultan interesantes para el tema que nos ocupa. Estas publicaciones no pueden ser adjuntadas al trabajo por motivos legales, pero pueden ser consultadas en la web del CCN [14], dado que son "sin clasificar".

Uno de los documentos consultados es la CCN-STIC-101 [15], que trata sobre los procedimientos de seguridad de las TIC. En ella se define el procedimiento de acreditación que habría de seguir cualquier sistema de comunicación que pretenda manejar información clasificada. Si el alcance de este trabajo de fin de grado fuese mayor y se pretendiese enviar información con un determinado nivel de seguridad, habría que contemplar las medidas recogidas en este documento, pero no es el caso dado que simplemente se pretende realizar el envío de información menor a través de un entorno seguro y más reservado que el que proporcionan las aplicaciones de mensajería instantánea comercial.

Otra publicación revisada para la realización del trabajo es CCN-STIC-407 [16], que aporta contenido teórico sobre los sistemas de telefonía móvil y establece recomendaciones de seguridad en su utilización. Esta guía resultaría muy útil en el caso de abordar el problema de que un dispositivo móvil pueda verse comprometido. Sin embargo, en nuestro trabajo esta vulnerabilidad se obvia en favor de obtener una aplicación que resulte rápida, cómoda y sencilla, aunque, de cualquier forma, se recomiendo la lectura de este documento para así hacer un uso de los dispositivos móviles adecuado en un ámbito militar.

La que nos afecta de una forma más directa es la CCN-STIC-440 [17], que habla sobre la seguridad de las redes TIC en el ámbito específico de la mensajería instantánea. Aunque es cierto que la publicación data de 2007 y que el avance de la tecnología año a año es innegable, aún así se pueden sacar ciertas conclusiones y medidas que debemos tener en cuenta al realizar una aplicación con el objetivo que se encomienda. En esta publicación se habla de numerosas cosas que pueden ser de

interés, pero por economía del presente trabajo decidimos centrarnos en los aspectos fundamentales en los que este documento puede asesorarnos y servirnos de guía para nuestra aplicación.

En ella, se evalúan los parámetros fundamentales de seguridad que debe cumplir una red de mensajería instantánea, como son la autenticidad, la integridad, la confidencialidad o la disponibilidad, que ya se explicaron en el apartado 2.2, anteriormente expuesto. Para completar estas variables, la publicación añade un concepto más: la privacidad.

En el documento, se define la privacidad como "el derecho a mantener el secreto sobre la información en su transferencia o almacenamiento" [17], lo cual nos recuerda bastante al parámetro de confidencialidad definido anteriormente en este mismo trabajo. Como la característica es la misma pero el nombre que se le atribuye es distinto según la fuente consultada, esto nos lleva a ver qué entiende dicha publicación por confidencialidad pues ésta ya no puede poseer la misma definición que la que hemos establecido en el trabajo.

Efectivamente, la confidencialidad queda definida como "la protección de información frente a posibles accesos no autorizados" [18]. Se explica en el documento que la confidencialidad afecta principalmente a dos aspectos fundamentales:

- **El intercambio no autorizado de información:** para evitar que comunicaciones indebidas o no permitidas se lleven a cabo, un sistema debe impedir las conexiones con máquinas no controladas y, además, evitar el uso de posibles vulnerabilidades por un atacante para acceder a las máquinas implicadas.

- **El acceso a la información producida por el sistema IM:** debe llevarse un registro de los accesos a la información que se hacen desde un determinado sistema de IM y que éstos solo estén disponibles para personas autorizadas a tal fin.

2.4 Sistemas de mensajería comerciales

2.4.1 Datos de los principales sistemas

Tras la total consolidación del correo electrónico tanto en computadoras personales como, posteriormente, en dispositivos móviles, llegaron las aplicaciones de mensajería instantánea. Estas aplicaciones permiten un chat en tiempo real entre una o varias personas en cualquier parte del mundo, simplemente sería necesaria una conexión proporcionada por los ISP *(Internet service provider).* Hoy en día es muy improbable no poder contactar con casi cualquier persona de esta forma y, además, poder compartir al instante contenido multimedia, archivos, ubicaciones, chat de voz y vídeo, etc. Además, es el sustituto de las llamadas telefónicas, dado que encima estas tenían mayor coste y no ofrecían la posibilidad de comunicación privada cuando se está con otras personas.

Existen infinidad de aplicaciones de mensajería instantánea en general, y en el mundo del *smartphone* en particular. La mayoría comparten las mismas funcionalidades, así como el mismo tipo de interfaz y no se separan demasiado en el ámbito de la seguridad. A continuación, en la Tabla 2-1, se detallan algunas aplicaciones seleccionadas bien por su popularidad y no por ser las mejores en todos los sentidos.

Sistemas de mensajería	Nº de usuarios (millones) / Año	Empresa	Utilidades	Necesidad tarjeta SIM
Whastapp [19]	1.500 / 2018	Facebook	Mensajería, archivos, multimedia, videoconferencia.	Sí
Telegram [20]	200 / 2018	Telegram FZ-LLZ	Mensajería, archivos, multimedia, conferencia.	Sí (solo la primera vez)
Facebook Messenger [21]	1300 / 2019	Facebook	Mensajería, multimedia, videoconferencia.	No
Skype [22]	300 / 2016	Microsoft	Mensajería, archivos, multimedia, videoconferencia.	No
WeChat [23]	1000 / 2018	Tencent	Mensajería, multimedia, videoconferencia.	Sí

Tabla 2-1 Características principales de los IM (elaboración propia)

2.4.2 Whatsapp

No cabe duda de señalar a la pionera en el mundo de la mensajería instantánea como la principal aplicación de transmisión de mensajes. No solo es la que posee mayor cantidad de usuarios a nivel mundial, sino que además su influencia se extiende por todos los rincones del mundo. Fue fundada en 2009 por Jan Koum, un antiguo ingeniero de la empresa Yahoo y en 2014 fue comprada por la empresa Facebook. En la actualidad posee unos 1.500 millones de usuarios como hemos mencionado anteriormente.

Esta aplicación utiliza el protocolo XMPP (Extensible Messaging and Presence Protocol), que no es más que un protocolo extensible y abierto, basado en XML (Extensible Markup Language, en español, lenguaje de marcado extensible) que fue concebido para mensajería instantánea. El usuario solo necesita registrarse la primera vez con su teléfono para acceder a su propia cuenta y los números teléfonicos de otros para poder comunicarse con estos. A diferencia de otras, basa su funcionamiento en un servidor como intermediario, es decir, el servidor guarda un mensaje que un terminal haya originado hasta que el receptor se conecte y pueda obtenerlo. Tras esto, el servidor borra el mensaje, con lo que no actúa como una nube.

Uno de los grandes defectos de Whatsapp es su vulnerabilidad en temas de seguridad. Si bien es cierto que hasta hace aproximadamente [24] tres años no usaba cifrado para las comunicaciones, actualmente sí que emplea encriptación extremo a extremo [25], es decir, ni siquiera el servidor puede descifrar los mensajes que le llegan, esto solo es plausible para emisor y receptor haciendo uso de técnicas de criptografía de clave pública, como se aclara en la Figura 2-4. El problema de todo esto reside en que existen numerosos casos en los que esta aplicación ha tenido incidentes de seguridad y filtraciones [26]. Por lo que, resulta claro que no es extremadamente complicado para los piratas informáticos, conocidos como *hackers*, acceder a contenido teóricamente seguro.

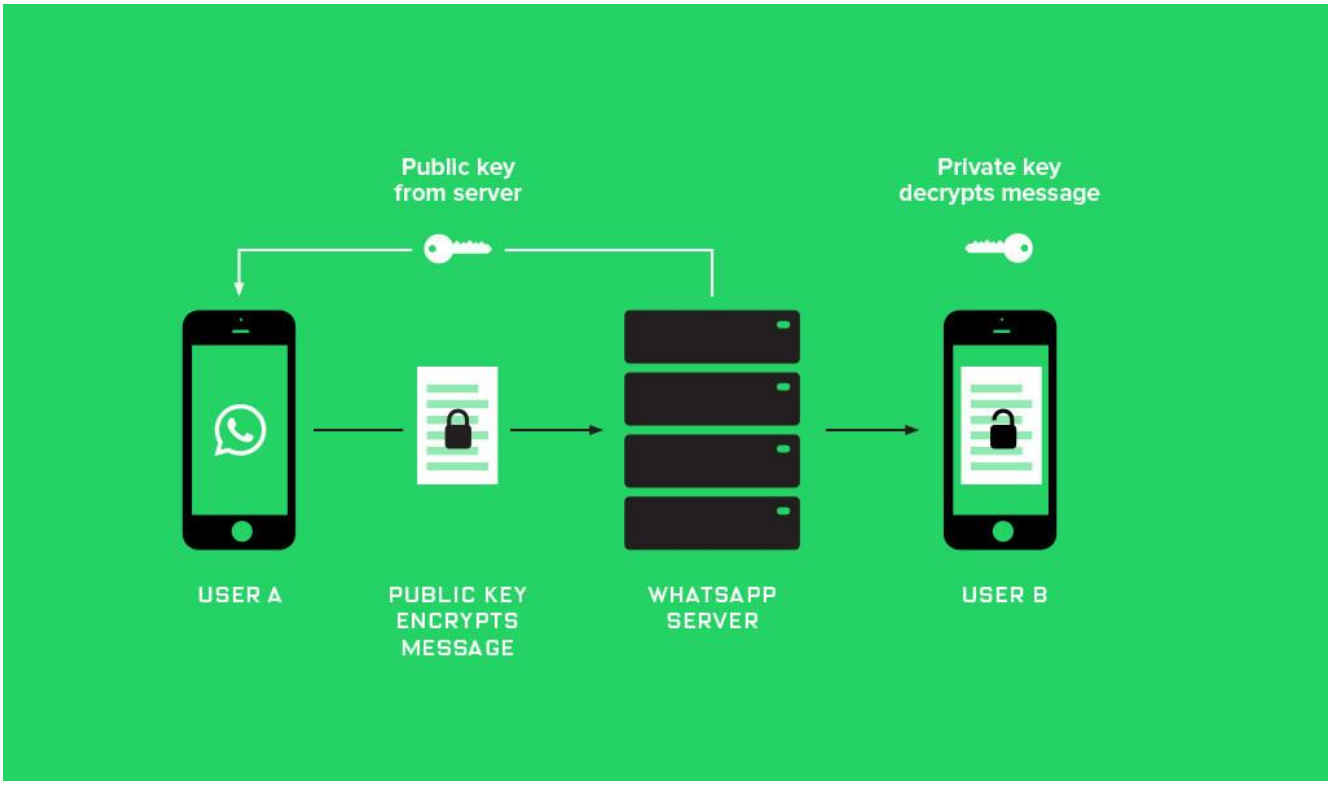

Figura 2-4 Cifrado extremo a extremo de Whatsapp [22]

2.4.3 Telegram

No siendo de las primeras en aparecer en el mercado (2013), Telegram ha sabido hacerse un hueco y competir con los principales IM. Por supuesto, tiene cobertura en los principales sistemas operativos como pueden ser iOS, Android, Windows, etc.

Se basa en un protocolo desarrollado por ellos mismos, el MTProto Protocol [27], que es abierto para permitir el desarrollo de nuevos clientes. Esta aplicación basa su funcionamiento en la nube, es decir, almacena todos los mensajes en un servidor que permite la reanudación de una comunicación desde cualquier terminal. Como es habitual, permite el envío de diferente contenido multimedia como pueden ser archivos, fotos, vídeos, etc. Además, también es posible efectuar llamadas, pero no videollamadas.

Cabe destacar dos cualidades principales para nuestro enfoque particular. Una de ellas es la posibilidad de efectuar chats secretos [28], como se puede ver en la Figura 2-5, que para simplificar, son conversaciones que se autodestruyen y no forman parte de la nube de datos (ver Figura 2-5). La otra cualidad es acerca de las importantes medidas de seguridad que nos ofrece este cliente de IM. Al tener código abierto, cualquier experto en la materia puede comprobar sus medidas y técnicas de seguridad. Para las conexiones servidor-cliente y cliente-cliente se realiza un cifrado AES256 [29] y otro RSA2048 [30], en los que no entraremos en detalle por no ser objetivo del presente trabajo, y para la compartición de claves se usa el esquema de Diffie-Hellman [31], que no es más que un protocolo empleado principalmente para el intercambio de claves simétricas.

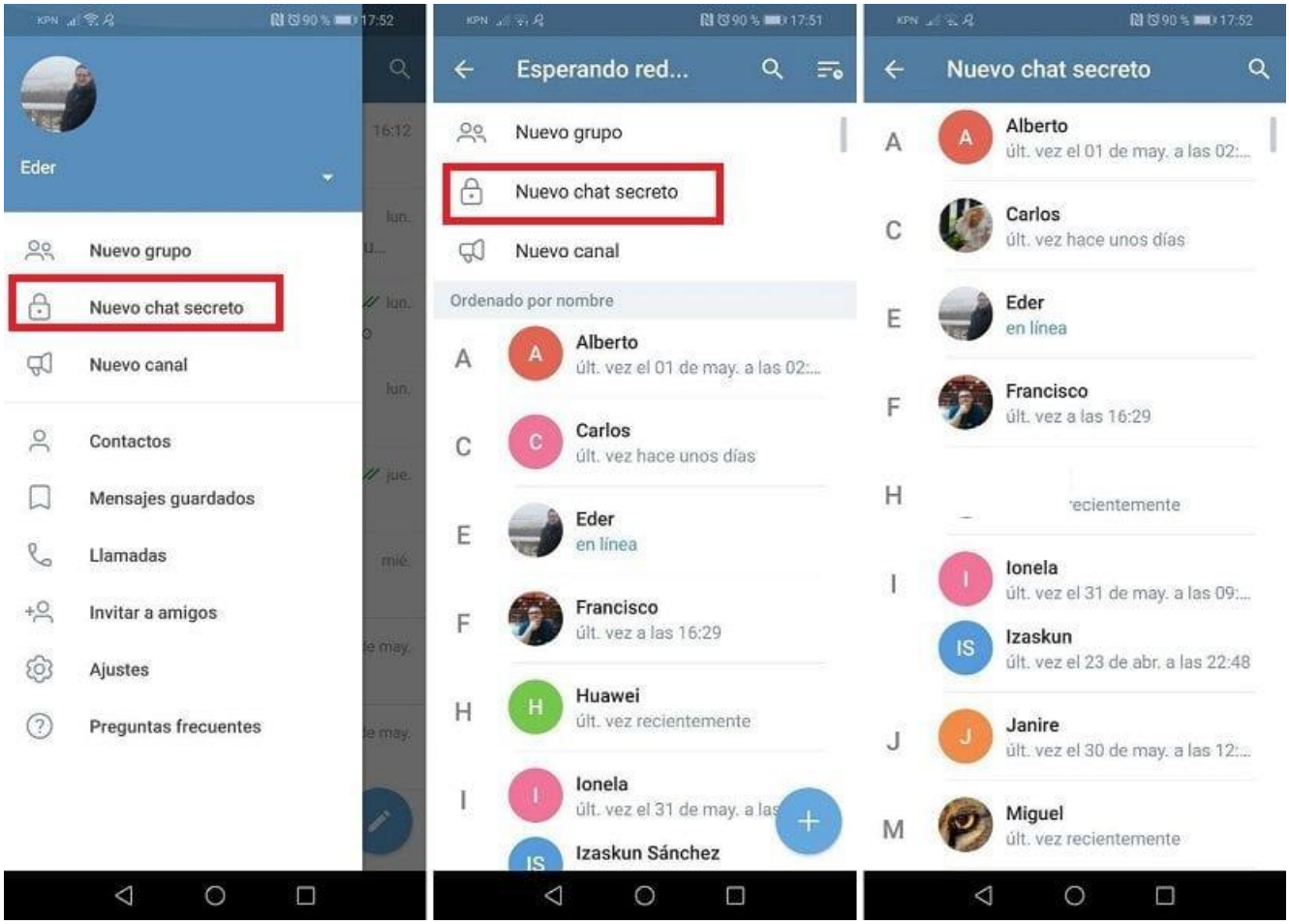

Figura 2-5 Creación chat secreto Telegram [25]

2.4.4 Facebook Messenger

Del mismo modo en que Facebook fue creciendo y haciéndose cada vez más popular, en esta última década, Facebook Messenger ha ido haciéndose un gigantesco hueco en el mercado de la mensajería. En la Figura 2-6 podemos ver su inferfaz principal [32].

Figura 2-6 Interfaz Facebook Messenger [29]

Al igual que su rival Telegram usa un protocolo abierto, el MQTT (Message Queue Telemetry Transport), el cual está optimizado para su utilización en dispositivos móviles. El gran defecto de este código es que no cifra los mensajes de extremo a extremo, sino entre cliente y servidor. De todas formas, si deseamos añadir seguridad podemos conseguirla a través del modo "chat secreto". En este modo conseguimos un cifrado extremo a extremo que aporta cierta seguridad. También al igual que la aplicación expuesta en el punto anterior, almacena las conversaciones en la nube y para que estas sean borradas basta con hacerlo desde la propia aplicación a nivel usuario, aunque obviamente en el modo "chat secreto" no son almacenadas.

2.4.5 Skype

Esta aplicación de IM debe su popularidad, sobre todo, al servicio de VoIP (Voz sobre IP). La mayoría de sus usuarios, en origen, comenzaron a usar esta aplicación para conferencias y videoconferencias, aunque en la actualidad, posee un correcto sistema de mensajería instantánea con múltiples posibilidades. La interfaz típica de Skype [33] la podemos ver en la Figura 2-7.

En la actualidad, usa un código privado desarrollado por la misma empresa, el cual está basado en el modelo *peer to peer* [34]. En términos de seguridad, usa un modelo de cifrado cliente-servidor por medio del algoritmo AES256 para todas sus comunicaciones. Cabe destacar que no almacena las conversaciones en el servidor, aunque sí que es conocido que todas las conversaciones son monitorizadas por la Agencia de Seguridad Nacional estadounidense (NSA) [35].

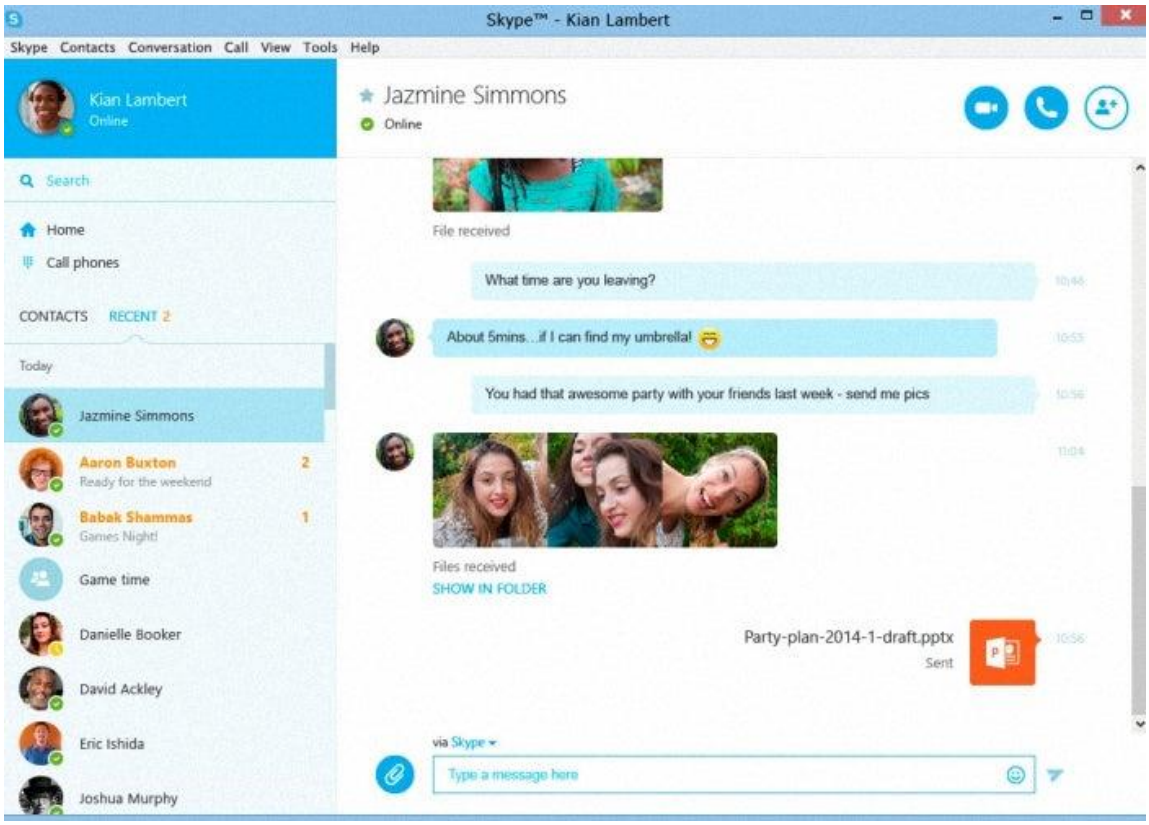

Figura 2-7 Interfaz Skype [30]

2.4.6 WeChat

Aunque es menos conocida en España, a nivel mundial esta aplicación de mensajería instantánea está bastante extendida, sobre todo en China. Además de ofrecer servicios de mensajería, conferencia y vídeoconferencia, también posee, entre otros, un servicio de red social que dispara la popularidad de esta IM.

En el ámbito de la seguridad, no implica novedades frente a otras competidoras del mercado. Usa un cifrado extremo a extremo y utiliza un modelo de cliente-servidor para el almacenamiento de mensajes. La gran controversia de esta aplicación se debe a dos principales causas. Por un lado, el servicio de geolocalización [36], el cual permite encontrar la ubicación exacta de otros usuarios, como podemos ver en la Figura 2-8. Este ha sido el origen de varios crímenes [37], lo que ha causado bastantes reacciones negativas en la población china.

Por otro lado, varias fuentes distintas [38] indican que la empresa propietaria de la aplicación, Tencent, comparte datos de sus usuarios con el gobierno chino y que, además este, censura los contenidos que se muestran en la red social de la IM.

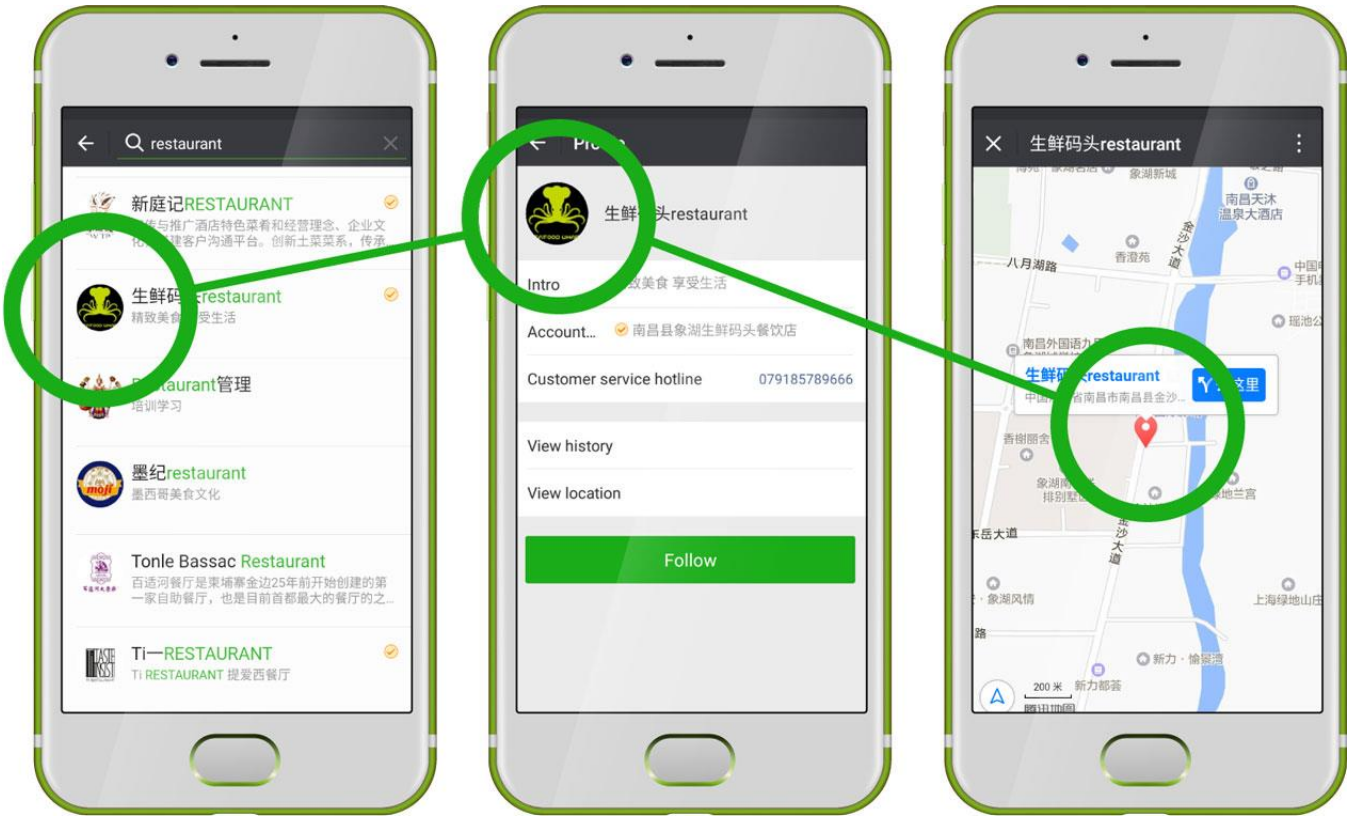

Figura 2-8 Servicio de geolocalización de WeChat [33]

2.5 Sistemas de mensajería instantánea enfocados a la seguridad

2.5.1 Introducción

Tras lo expuesto, se observa una clara tendencia mundial al uso de los sistemas de mensajería instantánea para todo tipo de asuntos diarios, formales e informales. El mundo militar no es una excepción. La comodidad y sencillez de estas aplicaciones impulsan a los usuarios a querer usarlas

también para su trabajo. Además, su accesibilidad permite que todos los rangos de edades tengan cabida en la utilización de estos sistemas.

Es por ello que ya se están desarrollando e incluso se han implementado algunas aplicaciones en distintos ejércitos o cuerpos policiales a nivel estatal e internacional. Los requisitos de este tipo de clientes en términos de seguridad son, obviamente, mayores que cualquier otro tipo de usuario.

A continuación, se exponen algunos de los ejemplos más conocidos en este ámbito.

2.5.2 Stashcat

Stashcat [39] es una aplicación de mensajería instantánea alemana que, además combina utilidades de almacenamiento en la nube. Esta aplicación está principalmente enfocada a la comunicación entre autoridades del ámbito de la seguridad y cuerpos de emergencias por su tecnología de alta seguridad y confidencialidad.

En 2017 [40] fue anunciado que Stashcat sería utilizada por la policía de Baja Sajonia como un sistema de mensajería seguro exclusivo para ese grupo cerrado de usuarios. Posteriormente, en 2018 [40], el Ministerio del Interior de Hesse anunció que la policía usaría HePolChat como sistema seguro de envío de texto, imágenes, vídeo y sonido. Este último sistema no es más que la versión privada de Stashcat para la policía de ese estado.

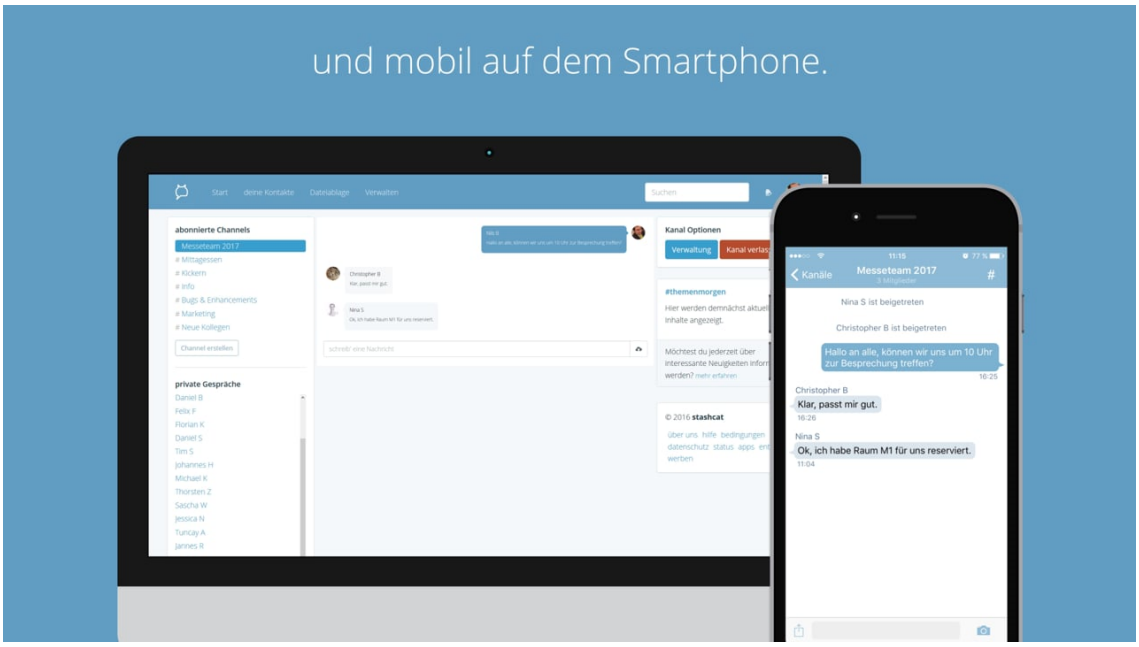

Figura 2-9 Interfaz Stashcat [36]

Esta IM, cuyo interfaz podemos ver en la Figura 2-9, se basa en intercambio de información a través de Internet. Posee un código privado de la empresa además de cifrado extremo a extremo doble (RSA y AES). Los mensajes no tienen límite de caracteres y la longitud del texto no afecta a la velocidad de transmisión, por lo que la comunicación se efectúa de forma rápida. El alojamiento de los datos se lleva a cabo localmente, en servidores propios del condado en cuestión o a través de otros servidores si se desea, con lo cual la información se almacena en un lugar relativamente seguro y

privado. Además, como punto fuerte, la aplicación permite la compartición de la geolocalización, con sus múltiples posibilidades en el ámbito policial y militar.

2.5.3 WICKR

Wickr [41] es una aplicación de mensajería instantánea americana multiplataforma (Android, iOS, Windows, etc.) que cubre un amplio sector empresarial e individual, aunque su principal enfoque sea el ámbito de la seguridad. Esta IM creada hace unos ocho años, fue elegida en 2018 [42] como la aplicación líder en criterios de encriptación y manejo de claves, entre otras muchas propiedades que la hacen muy segura.

Su funcionamiento es análogo al de otras aplicaciones de mensajería, permitiendo el envío de texto, multimedia y archivos, pero con algunas capas de seguridad extra con respecto a otras competidoras. De la misma forma que otras competidoras, usa la técnica de cifrado *end to end* (extremo a extremo), es decir, cifra de manera local en el dispositivo originador con una clave determinada y lo descifra de forma local también en el dispositivo de destino con la misma llave. Por tanto, la clave de cifrado es única para cada comunicación [43], como se puede ver en la Figura 2-10, lo cual hace a la aplicación robusta frente a amenazas y, además, al igual que ocurría en Whatsapp, la empresa no puede acceder a los mensajes que se intercambian entre usuarios. Posteriormente, pasado el tiempo de vida, la información caduca y se realiza una limpieza, tanto en el dispositivo como en los servidores de la IM, para no dejar rastro de las comnicaciones, ni en caché ni en RAM, ni en cualquier otro lugar donde pueda quedar constancia de dichos mensajes.

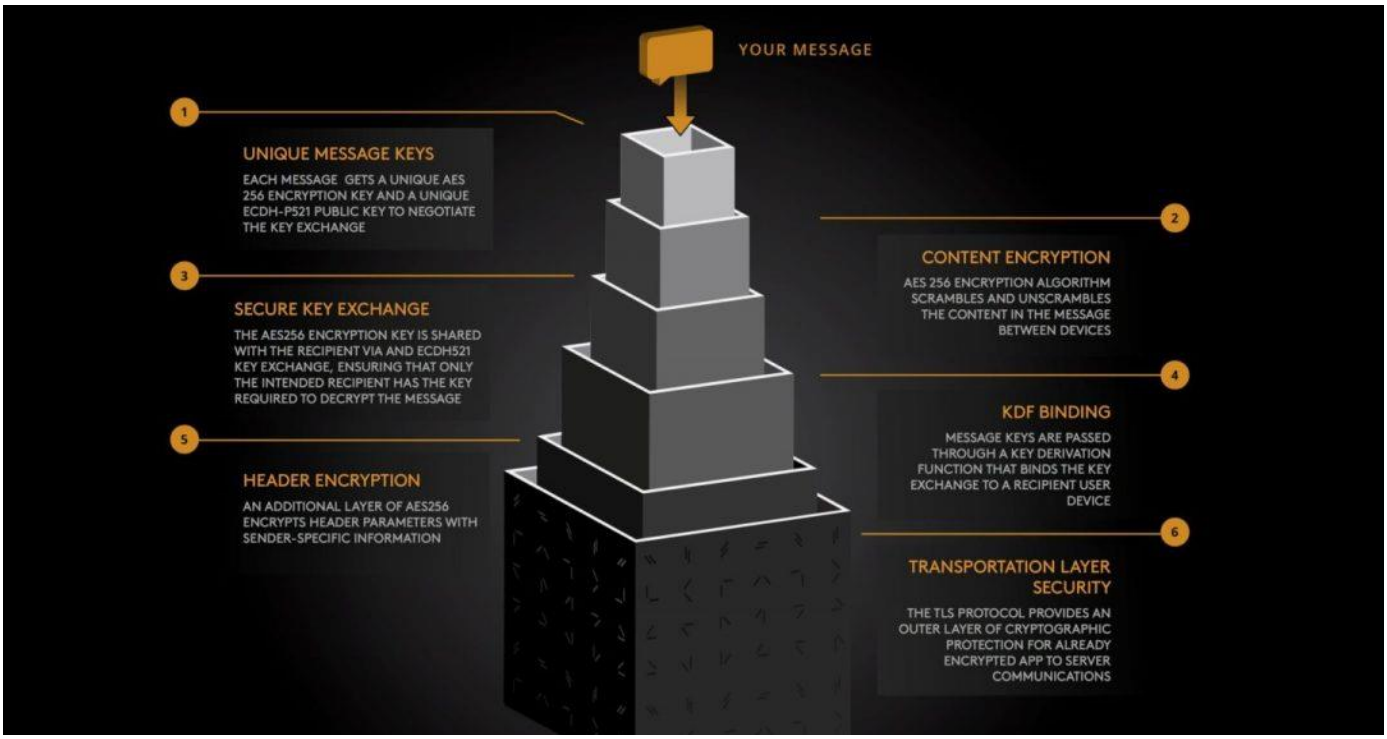

Figura 2-10 Esquema funcionamiento Wickr [40]

Para distinguirse de otras IM, la aplicación, a mayores, requiere un inicio de sesión con clave y usuario, los cuales no son almacenados *on-line* sino que residen en el dispositivo localmente y también borra los metadatos de las comunicaciones salientes para no dejar otro tipo de información como fecha, autor, equipo de creación, etc.

2.5.4 Signal

Signal [44], nacida en 2014, no es más que otro ejemplo de aplicación de mensajería instantánea enfocada principalmente a la seguridad, y es que este tipo de aplicaciones vienen ganando seguidores poco a poco de entre los usuarios de IM más comerciales, pero menos optimizadas en el ámbito de la defensa frente a amenazas informáticas.

Como sus competidoras directas, esta aplicación permite el envío de contenido multimedia, texto y archivos, así como conferencia y videoconferencia. Una peculiaridad de ésta plataforma, cuya interfaz podemos ver en la Figura 2-11, es que si no dispone de acceso, por cualquier motivo, a la red de datos, puede realizar el envío del mensaje a través del servicio SMS/MMS [45] [46]. En cuanto al campo de la seguridad, esta IM posee un protocolo abierto [47], lo que permite a cualquier interesado examinar el código y contribuir con los desarrolladores a la mejora de este, además de un cifrado extremo a extremo, de forma que la empresa no tiene acceso a las comunicaciones entre dos o más usuarios. Si se desea, se pueden almacenar los mensajes en una base de datos cifrada de forma local en el dispositivo o, si se prefiere, se pueden escribir mensajes temporales que desaparecerán del dispositivo emisor y receptor al cabo de cierto tiempo. También ofrece la posibilidad de bloquear los mensajes con una contraseña de forma que solo se pueda acceder a ellos tras la introducción de esta, así como la imposición de límites de caracteres en cada mensaje.

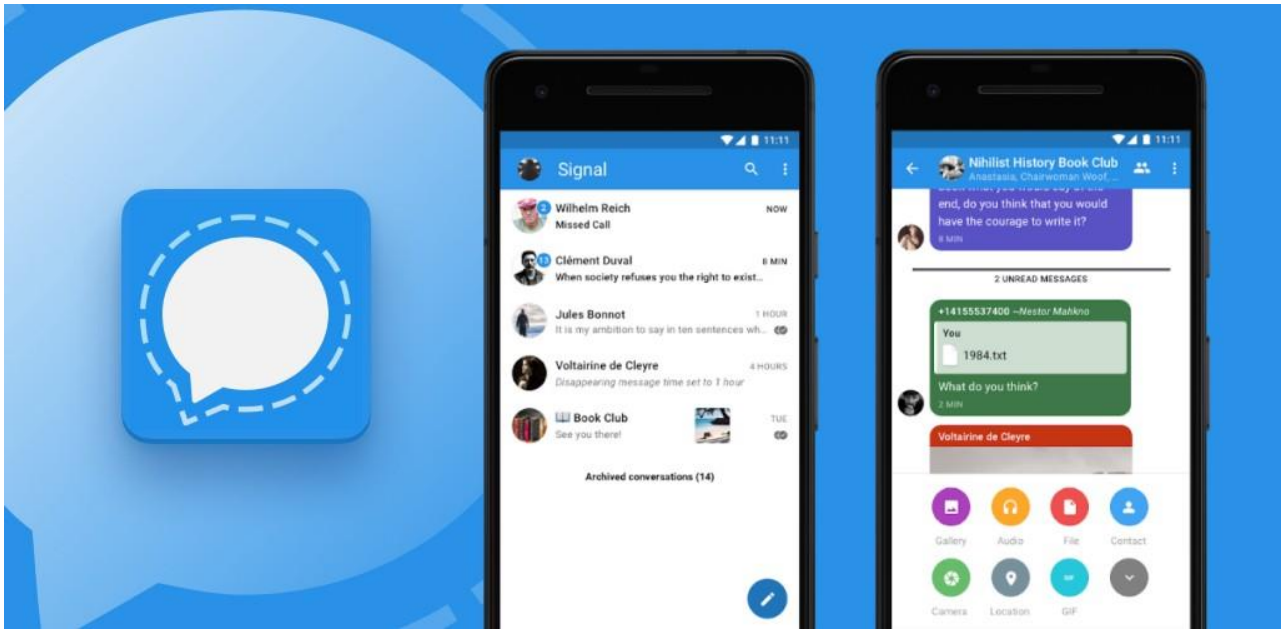

Figura 2-11 Interfaz Signal [41]

2.5.5 Aplicación UME

En el contexto nacional, la Unida Militar de Emergencias (UME) [48] no ha querido quedarse atrás frente a otras unidades análogas del marco internacional. Esta unidad se ha dado cuenta de la importancia de un sistema de mensajería seguro propio para la rapidez y fluidez de las comunicaciones en operaciones. Para ello, ha sacado a concurso una propuesta de licitación para que una empresa privada se encargue de diseñar e instalar un “servicio de mensajería instantánea seguro” [49].

Los requisitos que la UME ha establecido para este servicio son similares a los que ofrecen las aplicaciones descritas en apartados anteriores. A saber:

- Se desea un sistema que permita a sus militares "comunicarse de manera rápida y eficiente, con tiempos de respuesta mínimos y disponibilidad absoluta" [50], es decir, se desea instantaneidad en la respuesta y el envío y accesibilidad trescientos sesenta y cinco días al año, veinticuatro horas al día.

- Capacidad de envío de texto, multimedia (imágenes y vídeos) y archivos.

- Crear grupos o salas de chat donde se pueda incluir y excluir a los usuarios que se tenga a bien.

- Difusión de mensajes de alerta a múltiples dispositivos a la vez y que se pueda verificar su entrega y lectura.

- Conocimiento de envío, recepción y lectura correctos respectivamente.

- La seguridad es primordial. "La información transmitida por el sistema debe estar cifrada" [50] y además, deberá cumplir con el esquema nacional de seguridad, regulado en las Guías STIC. En este caso concreto, deberá atender primordialmente a a la "Guía CCN-STIC-440 Seguridad en la mensajería instantánea".

- Posteriormente al diseño e instalación de la aplicación, la Unidad Militar de Emergencias asumirá el control completo del sistema: altas, bajas, modificaciones, borrado de ficheros, etc.

- La información transmitida será almacenada en servidores del Ministerio de Defensa y en ningún caso será accesible por la empresa desarrolladora o cualquier otro tercero.

Así, la UME finaliza su pliego de condiciones acerca de la aplicación de mensajería instantánea segura que desea implementar para facilitar y simplificar el sistema de mando y comunicaciones.

2.6 Resumen de características de aplicaciones IM

Con el fin de dar al usuario un cierto criterio basado en información comprobada a la hora de seleccionar la aplicación final de la que se valdrá para realizar el envío de la información, se expone la siguiente tabla comparativa con la síntesis de aplicaciones que se ha realizado en el apartado anterior.

La Tabla 2-2 se ha realizado atendiendo a los criterios de seguridad expuestos en las distintas fuentes en las que nos hemos basado para realizar el trabajo. En esta se indican si se poseen o no las distintas propiedades de seguridad recogidas en "Parámetros de seguridad".

Parámetros	Whatsapp	Telegram	Facebook Messenger	Skype	WeChat	Stashcat	Wickr	Signal	UME
Confidencialidad	Sí	No	No	No	Sí	Sí	Sí	Sí	No
Integridad	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí
Autenticidad	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí
No repudio	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí
Disponibilidad	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí
Cifrado ext.-ext.	Sí	No	No	No	Sí	Sí	Sí	Sí	No
Código fuente abierto	Sí	Sí	Sí	No	No	No	No	Sí	No
Recopilación de datos	Sí	Sí	Sí	Sí	Sí	Sí	No	No	No
Privacidad (confidencialidad Guía STIC 440)	No	No	No	No	No	No	No	No	Sí

Tabla 2-2 Comparativa aplicaciones IM (elaboración propia)

2.7 Herramientas utilizadas

2.7.1 Java

Java [51] es un lenguaje de programación orientado a objetos que permite que las aplicaciones, una vez compiladas, puedan ejecutarse en una máquina virtual sin verse afectadas por la arquitectura propia de la computadora [52]. El objetivo era implementar una máquina virtual y un lenguaje con una estructura y sintaxis similar a *C++*, aunque finalmente, se optó por escribir un lenguaje de programación desde cero, aunque guardaba ciertas similitudes en su sintaxis con C/C++, para que este no estuviese ligado a ningún tipo de computadora en concreto.

Este lenguaje fue creado en 1991 por la empresa Sun Microsystems, de la mano de varios ingenieros que pretendían desarrollar una tecnología capaz de programar la siguiente generación de dispositivos inteligentes, a través de un lenguaje de programación sencillo de aprender y utilizar. Inicialmente, al lenguaje se le denominó como Oak, por un roble que había fuera de la oficina del director de la empresa, pero tuvieron problemas con otras marcas comerciales que poseían ya ese nombre. Un año más tarde de su lanzamiento, se comenzó a comercializar con productos que empleaban esta tecnología. Esto no fue fructífero al principio, aunque gracias al desarrollo simultáneo de la web y con su tremenda revolución en el concepto de Internet, comenzó a adquirir popularidad gracias a la posibilidad que permitía de crear Applets, que eran pequeños programas que se incrustaban en páginas web y eran ejecutados por el navegador. Finalmente, en 1995 se funda la empresa Java Soft por SunWorld [53] para dedicarse exclusivamente al desarrollo de de productos basados en este lenguaje informático.

El lenguaje incorpora una serie de características fundamentales que lo hacen muy loable [54]:

- **Sintaxis:** muy parecida a *C* y *C++*, pero con una gestión de la memoria dinámica automática.
- **Interpretado:** un programa escrito en este lenguaje se compila a código *bytecode*, el cual es interpretado por la máquina virtual Java.
- **Multiplataforma:** una vez compilado el programa, el fichero resultante se ejecuta en la máquina virtual siendo independiente de la plataforma.
- **Seguro:** la máquina virtual de Java se encarga de controlar que el programa compilado no ejecute operaciones no autorizadas sobre los recursos del sistema.

El lenguaje consta de las siguientes capas fundamentales [55], mostradas en la Figura 2-12:

- ***Device hardware*:** se trata del soporte físico, *hardware*, sobre el cual se ejecuta el lenguaje.
- ***OS* y *drivers*:** el SO es el software principal de un sistema informático, el cual gestiona los recursos de *hardware* y provee servicios a las aplicaciones. Asimismo, los *drivers* o controladores de dispositivo, son los programas informáticos que permiten que el sistema operativo interactúe con los distintos periféricos.
- ***Porting layer for platform integrator*:** esta capa se encarga de compatibilizar la instalación de Java en las distintas combinaciones posibles de *hardware*, OS y *drivers*.
- ***Java VM*:** La JVM se comunica con el hardware a través del OS y los *drivers*, abstrayendo al usuario de los mismos. Esto lo hace por medio de la interpretación del *bytecode*.
- ***Java profile*:** es una herramienta integrada en el paquete de productos de *Oracle Java* la cual permite monitorizar la ejecución del código *bytecode* en el nivel de la máquina virtual.
- ***Java API*:** está compuesta por las distintas librerías originales empleadas normalmente en las aplicaciones. Una API (*Application Programming Interface*) es una colección de componentes que dota a los programadores de medios para desarrollar aplicaciones Java. Las distintas API permiten realizar todo tipo de operaciones como puede ser procesar un XML por medio de la API JAXP (Java API for XML Processing).
- ***Partner app*/API:** son las librerías y aplicaciones desarrolladas por terceros que se ejecutan en Java.

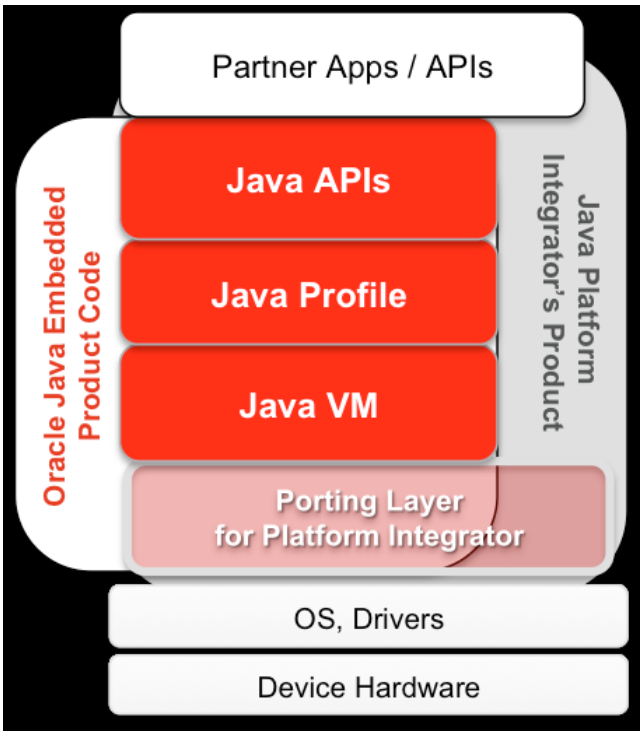

Figura 2-12 Plataforma Java [52]

2.7.2 Android

2.7.2.1 Introducción e historia

Android, palabra que casi toda la población mundial ha oído actualmente, es sin duda el sistema operativo móvil más usado a día de hoy, lo que lo hace objetivo de las agencias de inteligencia internacionales. Desarrollado por Google y basado en Kernel de Linux entre otro software de código abierto, fue diseñado *smartphones* y tabletas y, posteriormente, para relojes inteligentes, automóviles y televisores. Su código fuente principal es como hemos mencionado, abierto y trabaja bajo la licencia Apache. Dicho sistema operativo cuenta ya con once versiones principales y dieciocho contando las revisiones, llamando a cada una con un nombre. Aunque actualmente Android Q ó 10 no ha sido implementado en numerosos terminales, el 19 de febrero de 2020 se anunció la versión 11 [56].

En 2005 Google compra Android Inc. Por aquel entonces no se sabía demasiado de sus funciones más allá de que desarrollaban software para teléfonos móviles, lo que dio pie a rumores sobre que Google iba a entrar en el mercado de los dispositivos móviles. También en ese año se crea el logo y mascota del sistema operativo, Andy, el androide verde conocemos, así como una primera versión prototipo de la plataforma para dipositivos móviles, basada en Linux. En 2007, tras la creación de la Open Handset Alliance [57], un consorcio de setenta y ocho compañías de telecomunicaciones dedicadas al desarrollo de estándares aniertos para dispositivos móviles, se solicita la patente y se anuncia la primera versión del sistema operativo: Android 1.0. pero retrasándose el primer terminal con el SO hasta el año siguiente. Ya en 2011 alcanza unas cuotas de mercado del 50,9 %, duplicando al segundo en la lista, iOS [58]. Durante los siguientes años, numerosos desarrolladores de diversa índole, siguen creando aplicaciones para el sistema operativo, lo que extiende exponencialmente la funcionalidad de este. En 2018, ya existían más de dos millones de aplicaciones disponibles para Android, sin tener en cuenta las que existen en otras tiendas no oficiales.

Es cierto que este sistema operativo en ocasiones le sirve como herramienta de comercio al gobierno de los Estados Unidos de América pues la dependencia de numerosas compañías de teléfonos

móviles, vehículos, sonido y un largo etcétera es total. Son muchos los dispositivos electrónicos que para funcionar necesitan de este sistema operativo móvil y, como ya pasó con la compañía Huawei, por determinados intereses o circunstancias, Google puede negarse a dar cobertura de actualizaciones a algún cliente, con las desastrosas consecuencias que eso trae para dicha empresa [59].

2.7.2.2 Arquitectura Android

Los componentes principales del sistema operativo [60] son los que se pueden ver en la Figura 2-13:

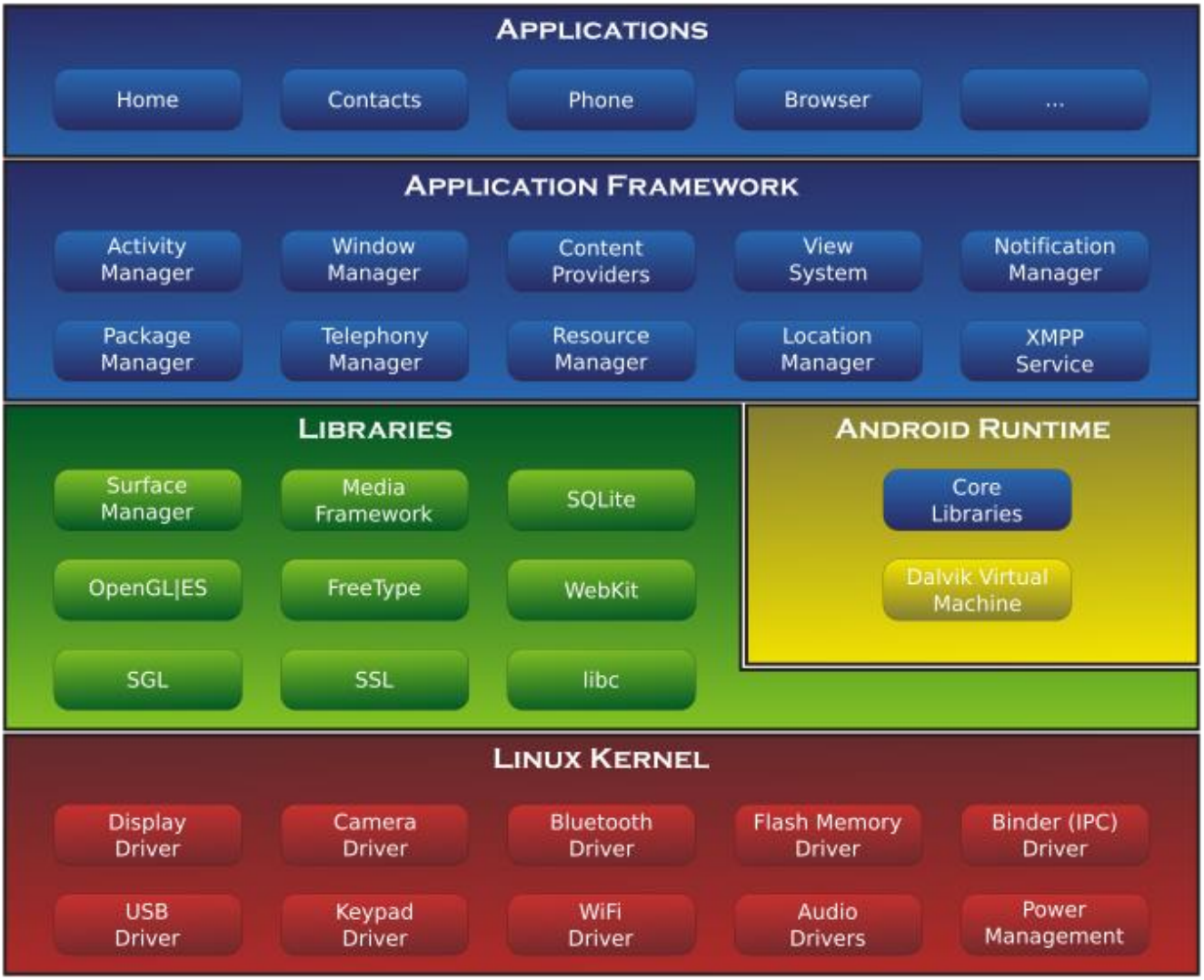

Figura 2-13 Arquitectura Android [60]

- **Aplicaciones:** existen diversas aplicaciones como correo electrónico, servicio de mensajes cortos, calendario y mapas entre otras que están escritas en lenguaje Java [61] y son propias del sistema operativo o pueden instalarse en él proviniendo de otros desarrolladores.

- **Marco de trabajo de las aplicaciones:** es un espacio en el que cada aplicación puede publicar sus capacidades y cualquier otra puede hacer uso de estas siguiendo unas reglas de seguridad.

- **Bibliotecas:** son un conjunto de librerías de lenguaje C y C++ usadas por varios componentes del sistema.

- **Android Runtime:** es un grupo de bibliotecas base que proporcionan la mayor parte de las funciones disponibles en las librerías del lenguaje Java.

- **Núcleo Linux:** Android depende del sistema operativo Linux para los servicios base del sistema como seguridad, gestión de memoria, gestión de procesos, pila de red y modelo de controladores.

2.7.2.3 Seguridad

Según un estudio de la empresa *Symantec* [62], actualmente renombrada como NortonLifeLock [63], Android es, en comparación con iOS, mucho más seguro pero bastante más atacado. En el estudio se reflejan solo trece vulnerabilidades graves en el sistema operativo que nos atañe mientras que para iOS se reflejan trescientas ochenta y siete. No obstante, ambos sistemas operativos se esfuerzan en ser cada vez más seguros, pero como ya sabemos, los ataques también son cada vez más diversos y sofisticados, con lo que la seguridad seguirá siendo previsiblemente un aspecto que requerirá atención en el futuro.

Como ya hemos comentado antes, se cree que Android es usado por las agencias de inteligencia como la NSA (Agencia de Seguridad Nacional estadounidense) y el GCHQ (Cuartel General de Comunicaciones del Gobierno británico). Ambas tendrían acceso a todo tipo de datos como SMS, geolocalización, correos, notas o mensajes de los usuarios que usan dispositivos que se valen del sistema operativo Android [64] [65].

2.7.3 Android Studio

Android Studio [66], cuya interfaz se puede ver en la Figura 2-14, es el IDE *(Integrated Development Environment)* oficial para la implementación de aplicaciones Android. En este entorno se proporcionan todas las herramientas necesarias para el desarrollo de aplicaciones en cualquier tipo de dispostivo Android se pueda ejecutar (móviles, tabletas, vehículos, relojes inteligentes, etc.).

En este entorno se ofrecen, entre otras, las siguientes funcionalidades: desarrollo de código en editor inteligente, capaz de corregir y sugerir al programador a medida que éste escribe; plantillas con código predeterminado; control de versiones integrado para disponer de cualquiera de las herramientas existentes a este fin en el sistema operativo; y un emulador de cualquier dispositivo que ejecute Android para probar la aplicación desarrollada. Cabe destacar que se puede probar en un emulador externo al IDE o en un dispositivo real, aunque las funcionalidades que ofrece el emulador de Android Studio son suficientes para casi cualquier prueba.

Como factor clave para la elección de este IDE para desarrollar nuestra aplicación entre otros muchos (IntelliJ IDEA, Xamarin o Titanium Appcelerator SDK*)*, tenemos su gran popularidad. Es el IDE usado por excelencia por la mayoría de programadores Android [67] y, por tanto, cuenta con numerosos recursos en la web para ayudar a crear nuestro propio código, así como numerosos tutoriales o vídeos de autoayuda *on-line.*

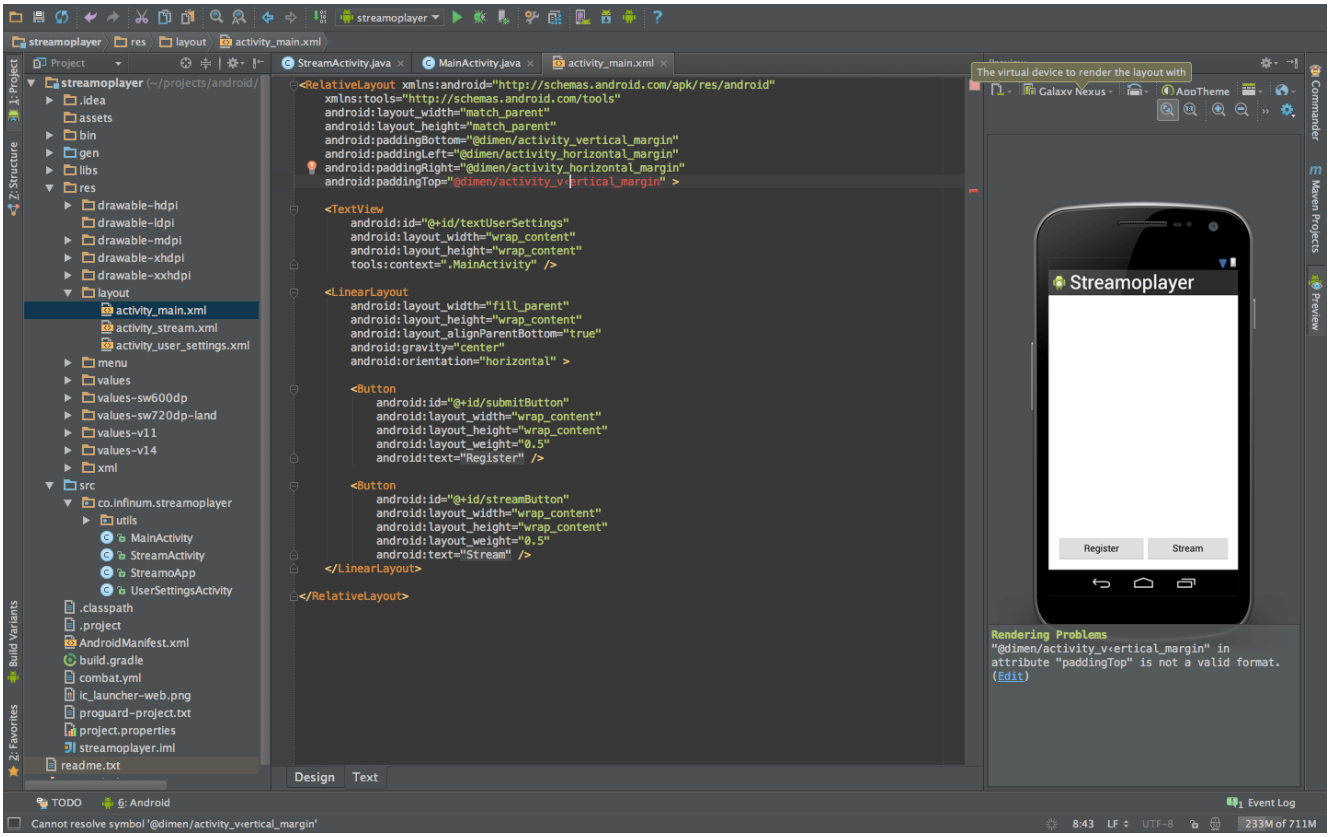

Figura 2-14 Interfaz IDE Android Studio [63]

3 DESARROLLO DEL TFG

En este capítulo se pretende explicar los requerimientos que ha de cumplir el proyecto, así como las nociones más básicas de desarrollo en Android. Para finalizar, se detallará cómo se ha realizado la aplicación, se explicarán las partes más significativas del código de la misma y el porqué de las decisiones que se han ido tomando acerca de ésta, así como sus funciones principales.

3.1 Demanda del proyecto

Se desea realizar una aplicación que permita el envío de información a través de cualquier plataforma de mensajería instantánea móvil ya existente. La elección de la aplicación usada debe ser libre para el usuario, simplemente debemos asegurarnos de la seguridad de esta: cifrado, integridad, etc.

Para ello la aplicación debe:

- Permitir al usuario editar texto para que pueda escribir el mensaje que se desee enviar previamente a la emisión.
- Encriptar el mensaje con una clave a elección y almacenarlo en un archivo de bytes, además de permitir la elección de la aplicación a través de la cual será enviado.
- Tras la recepción, abrir la aplicación al clicar en el archivo recibido y desencriptar éste solo con conocimiento de la clave necesaria.

3.2 Conceptos básicos de Android

Previo al desarrollo de una aplicación en esta plataforma, debemos conocer una serie de conceptos básicos. A continuación, se trata de exponer esos conceptos y componentes básicos, la estructura habitual de un proyecto en Android y una breve introducción sobre las versiones que existen del sistema operativo.

3.2.1 Componentes básicos

Los principales componentes que tiene un proyecto de Android son los siguientes:

- ***Layout* (diseño):** supone toda la matriz visual de la interfaz del usuario.

- ***Activity* (actividad):** es la pantalla o cada pantalla (según aplicación), que se muestra al usuario, conformándose así la interfaz.

- ***Fragment* (fragmento):** es una parte de la interfaz que puede ser reutilizada, añadida y/o eliminada de la aplicación sin interferir en el resto de elementos de la pertinente actividad. Esto facilita la implementación de varias configuraciones de pantalla sin necesidad de reescribir el código.

- ***Intent* (intención):** es un propósito de realizar una acción. Se suele emplear para comunicarse e intercambiar información entre diversas actividades y/o servicios.

- ***Content provider* (proveedor de contenidos):** se usa también para compartir datos entre aplicaciones, así como para administrar el acceso a un conjunto de datos estructurado.

- ***Service* (servicio):** es un proceso, parecido a una actividad, que es ejecutado en segundo plano, sin necesidad de que este interactúe con el usuario.

3.2.2 Estructura típica de una aplicación en Android

Existen numerosos archivos en un proyecto de Android. Estos están organizados según una estructura de directorios, que puede ser visualizada de dos formas distintas: vista Android y vista Project. La última es la representación real de la estructura de los archivos del proyecto, que es más tediosa e incómoda. La primera es una estructura distinta que facilita la navegación, organizando los archivos en módulos y tipos de archivos.

En la vista Android (ver Figura 3-1), dentro de cada módulo, los archivos se estructuran en los siguientes grupos:

- ***manifests:*** aquí se recoge la información esencial de configuración de la aplicación en el archivo *AndroidManifest*. Entre otras cosas, se especifican aquí:
  - La versión de la aplicación (API).
  - Las actividades (*activities*) que van a emplearse.
  - Todo tipo de permisos requeridos por la aplicación. Por ejemplo, para acceder a otras aplicaciones o para acceder a la memoria.

- ***java:*** recoge los diversos archivos de código fuente en lenguaje Java.

- ***res:*** contiene los recursos de la aplicación, formados en su mayoría por ficheros XML. A su vez este está formado por varios directorios:
  - *drawable*: en este directorio se almacenan las imágenes. Cabe reseñar que los nombres de estos archivos deben estar compuestos únicamente por números y las letras del abecedario inglés en minúsculas.
  - *layout*: en este directorio se almacenan los ficheros cuya función es describir el aspecto de la interfaz de usuario de las distintas actividades.
  - *menu*: en este directorio se encuentran los ficheros XML con los menús correspondientes a las distintas actividades.

- *mipmap*: este directorio funciona de la misma manera que el directorio drawable, con la única salvedad de que en éste únicamente se emplazan las imágenes que no se usan dentro de la aplicación, como es el caso del icono de la aplicación.
- *values*: en este directorio se almacenan diversos recursos XML que contienen valores simples como son: *strings* (cadenas de caracteres), *arrays* (vectores), enteros, estilos y colores. Al ser recursos XML podrían ser almacenados juntos en un único documento XML, sin embargo, para facilitar su uso se separan en distintos ficheros acorde con su contenido.

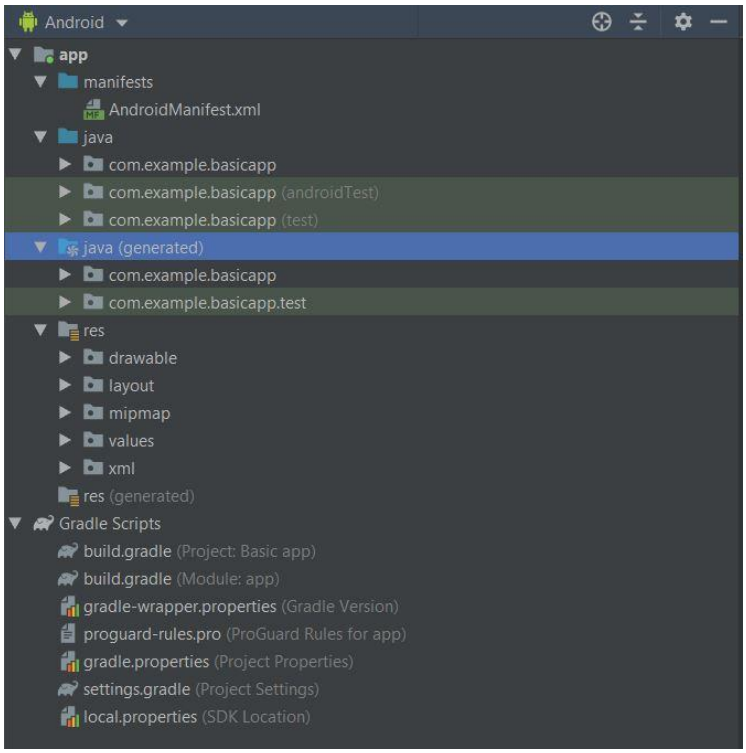

Figura 3-2 Vista Android (elaboración propia)

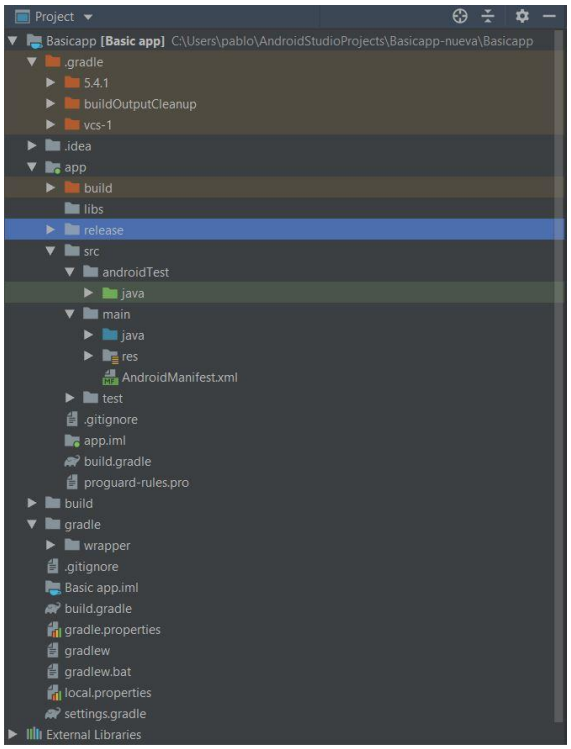

Figura 3-1 Vista Project (elaboración propia)

Para la vista más tediosa pero más parecida a la estructura original, la vista *Project* (ver Figura 3-2), cabe mencionar algunos archivos y ficheros:

- ***build*:** es la carpeta dónde se guardan los archivos originados tras el proceso de compilación.
  - *src*: es la carpeta contenedora del directorio *main*, donde podemos hallar los distintos archivos de código y recursos mencionados en la vista anterior (*manifests*, *java y res*).

- ***build.gradle*:** esta fichero contiene las configuraciones principales de compilación:
  - *minSdkVersion*: versión de Android, la misma inclusive, a partir de la que se puede ejecutar la aplicación.
  - *targetSdkVersion*: versión de Android para la que la aplicación ha sido desarrollada y probada.

3.2.3 Niveles de API

Al comienzo de la creación de un nuevo proyecto de Android se elige una versión del sistema operativo. Los niveles de *API* en los que la aplicación podrá ser ejecutada vienen determinados por la versión escogida.

En cada uno de los niveles se incluyen ciertas mejoras o avances de clases y métodos, además de mantener todos los anteriores. Por ello, un buen conocimiento de los requisitos de nuestra *app* es necesarios para determinar la elección de la versión, en la que también hay que tener en cuenta que las versiones más actuales son compatibles con los dispositivos de un menor número de usuarios. En cualquier caso, el sistema operativo permite que cierta *app* sea compatible con varios niveles de API, a través del valor previamente configurado en *build.gradle*, comparando este con el sistema en el que se desea instalar.

A continuación, en la Tabla 3-1, se recogen todas las versiones de Android existentes desde su creación hasta la actualidad.

Nivel de API	**Versión de *Android***	**Nombre**	**Fecha de lanzamiento**
1	1.0	Apple Pie	23 de septiembre de 2008
2	1.1	Banana Bread	9 de febrero de 2009
3	1.5	Cupcake	25 de abril de 2009
4	1.6	Donut	15 de septiembre de 2009
5-7	2.0-2.1	Eclair	26 de octubre de 2009
8	2.2-2.2.3	Froyo	20 de mayo de 2010
9-10	2.3-2.3.7	Gingerbread	6 de diciembre de 2010
11-13	3.0-3.2.6	Honeycomb	22 de febrero de 2011
14-15	4.0-4.0.5	Ice Cream Sandwich	18 de octubre de 2011
16-18	4.1-4.3.1	Jelly Bean	9 de julio de 2012
19-20	4.4-4.4.4	KitKat	31 de octubre de 2013
21-22	5.0-5.1.1	Lollipop	12 de noviembre de 2014
23	6.0-6.0.1	Marshmallow	5 de octubre de 2015
24-25	7.0-7.1.2	Nougat	15 de junio de 2016
26-27	8.0-8.1	Oreo	21 de agosto de 2017
28	9.0	Pie	6 de agosto de 2017
29	10.0	Android 10	3 de septiembre de 2019
30	11	Android 11	mayo de 2020

Tabla 3-1 Niveles de API (elaboración propia)

3.3 Ficheros principales de una aplicación

Existen ciertos archivos en un proyecto que son dignos de explicación:

- *AndroidManifest*
- *Build.gradle*
- Ficheros de recursos
- *Activities*

3.3.1 Android Manifest

Este archivo, como se dijo anteriormente, contiene datos básicos de la aplicación como su nombre de paquete base, la actividad que se ejecuta al lanzarla, referencias a servicios, *intents*, permisos que se usan, etc. A continuación, por orden de aparición en el código, se recogen las principales entradas del fichero:

- ***Package*:** se trata del nombre que recibe el paquete base del código fuente de nuestra aplicación que puede ser usado comercialmente a posteriori. En nuestro ejemplo, que se puede ver en la Figura 3-3, se le nombra como *"com.example.basicapp"*.

- ***Uses-permission*:** se trata de los permisos necesarios para habilitar determinadas funciones que pueden modificar la experiencia de usuario. De serie, el código viene sin ningún permiso reflejado, por lo que no se puede emplear ciertas funciones internas del dispositivo móvil en el que se ejecuta la aplicación. En nuestro ejemplo, como podemos ver en la Figura 3-3, solicitamos permiso para leer y escribir en la memoria interna del dispositivo.

- ***Application*:** en este apartado se pueden configurar una serie de parámetros básicos como el nombre, el tema empleado, los mecanismos de *intent*, el *provider* necesario para la gestión de contenidos, etc. Además de todo esto, en este apartado se declara la *activitie* que se ejecuta al arrancar la aplicación.

3.3.2 Build.gradle

En este archivo, se recogen, entre otros, los parámetros principales necesarios que son fundamentales para poder compilar el código. A continuación, por orden de aparición en el código, se explican los más importantes:

- ***compileSdkVersion*:** como se explicó anteriormente, indica la versión de API empleada para compilar. En nuestro caso, se ha usado la versión más actualizada existente en el momento de iniciar el proyecto, la 29, como se puede ver en la Figura 3-4, evitando así el uso de API obsoletas al beneficiarnos de las características de compilación del más actualizado.

- ***minSdkVersion*:** como también se explicó anteriormente, se usa para establecer la versión a partir de la cual se podrá ejecutar nuestra aplicación, incluyendo la numerada. En nuestro caso, nuestra *app* no podría ser si quiera instalada en versiones anteriores a la 19 (ver Figura 3-4).

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.MILChat.MILChat">
    <uses-permission android:name="android.permission.READ_INTERNAL_STORAGE"></uses
        permission>
    <uses-permission android:name="android.permission.WRITE_INTERNAL_STORAGE"></uses-
        permission>
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name="com.MILChat.MILChat.MainActivity"
            android:screenOrientation="portrait">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
            <intent-filter>
                <action android:name="android.intent.action.SEND"/>
                <category android:name="android.intent.category.DEFAULT"/>
                <data android:mimeType="application/octet-stream"/>
            </intent-filter>
            <intent-filter>
                <action android:name="android.intent.action.VIEW"/>
                <category android:name="android.intent.category.DEFAULT"/>
                <data android:mimeType="application/octet-stream" android:host="*"
                 android:scheme="content"/>
            </intent-filter>
        </activity>
        <provider android:name="androidx.core.content.FileProvider"
            android:grantUriPermissions="true"
            android:exported="false"
            android:authorities="es.enm.basicapp">
            <meta-data android:name="android.support.FILE_PROVIDER_PATHS"
                android:resource="@xml/filepaths"/>
        </provider>
    </application>
</manifest>
```

Figura 3-3 *AndroidManifest* (elaboración propia)

- ***targetSdkVersion*:** Este parámetro solo sirve para indicar la versión en la que la aplicación ha sido desarrollada y probada, por lo que, si la plataforma de uso final fuese superior a la indicada en este parámetro, se habilitarían una serie de funciones de compatibilidad para asegurar el correcto funcionamiento de la *app*. Por defecto, si no se indica este parámetro, el valor sería igual al del *minSdkVersion*.

- ***versionCode*:** este parámetro indica el número de versión de la aplicación, el cual debe ser modificado cada vez que hagamos una modificación en el proyecto y volvamos a publicar esta.

```
apply plugin: 'com.android.application'

android {
    //VERSIÓN DE API EMPLEADA PARA COMPILAR
    compileSdkVersion 29
    buildToolsVersion "29.0.2"
    defaultConfig {
        applicationId "com.MILChat.MILChat"
        //VERSIÓN A PARTIR DE LA CUÁL SE PUEDE EJECUTAR LA APP
        minSdkVersion 19
        //VERSIÓN EN LA QUE LA APP HA SIDO PROBADA Y COMPILADA
        targetSdkVersion 29
        //NÚMERO VERSIÓN DE LA APLICACIÓN
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'),
            'proguard-rules.pro'
        }
    }
}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'androidx.appcompat:appcompat:1.1.0'
    implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test.ext:junit:1.1.1'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.2.0'
    implementation 'com.google.android.material:material:1.1.0'
}
```

Figura 3-4 *Build.gradle* (elaboración propia)

3.3.3 Ficheros de recursos

Para terminar de diseñar nuestro proyecto, existen numerosos recursos que sirven para funciones muy distintas. A continuación, se recogen algunos de los más importantes:

- ***Drawable*:** este tipo de recursos son cualquier tipo de imágenes usados en la aplicación, ya sean de extensión *jpg*, *png o svg*.

- ***Values*:** este tipo de recurso es una carpeta contenedora de más ficheros donde se recogen determinados valores usados a lo largo del código como colores, cadenas de caracteres o estilos.

- ***Layout*:** son, básicamente, ficheros que determinan el aspecto visual de la *app*. Dentro del diseño se pueden incluir controles y elementos de interfaz de usuario de distinto tipo como *textView, button, toast, editText,* etc.

3.4 Mecanismos de *intent*

Un *intent* es un mecanismo que sirve, básicamente, para invocar componentes, los cuales, en Android, son las *activities* (actividades que pueden ejecutarse en segundo o primer plano y muestran una pantalla determinada al usuario). Es decir, nos sirven para llamar a aplicaciones externas.

3.4.1 Composición de un intent

Un *intent* es un objeto de mensajería que se usa para solicitar una acción de otro componente de una *app*. Se compone de varios elementos, que en general incluyen el identificador de la acción del *intent* y una serie de datos a pasar a la aplicación destinataria de éste. Los datos se representan mediante pares con la forma "clave/valor", donde la clave se trata normalmente de un identificador del dato y el valor puede ser cualquiera de los tipos fundamentales (boolean, int, double, etc.). Estos se añaden al *intent* mediante el método "putExtra". Así, por ejemplo, en la Figura 3-5 podemos ver que se añade al *intent* los datos de una URI (*Uniform Resource Identifier*, en español Identificador de Recursos Uniforme), que permite identificar un fichero para poder compartirlo entre la aplicación que origina el *intent* y la que lo recibe. Para poder aclarar el tipo de datos que contiene el fichero se utiliza el método "setType", que especifica dicho tipo mediante un código MIME (*Multipurpose Internet Mail Extensions*, en español, Extensiones multipropósito de correo de Internet).

Para poder saber cual es la aplicación destinataria del *intent*, tal como se verá en la sección 3.4.2, tenemos que distinguir dos posibilidades: que se haga de manera implícita o explícita. En cualquier caso, la aplicación destinataria tiene que haberle indicado al sistema Android con anterioridad que está preparada para recibir *intents*, utilizándose para ello el AndroidManifest. Esto se puede ver en la Figura 3-3, donde se puede comprobar que la aplicación MILChat se registra para poder recibir *intents* de dos tipos: "SEND" y "VIEW".

```
Intent shareIntent = new Intent();
shareIntent.setAction(Intent.ACTION_SEND);
shareIntent.putExtra(Intent.EXTRA_STREAM, uri);
shareIntent.setType("application/octet-stream");
shareIntent.setPackage("com.whatsapp");
startActivity(Intent.createChooser(shareIntent,
getResources().getText(R.string.bt_send_String)));
```

Figura 3-5 Ejemplo de *intent* implícito (elaboración propia)

3.4.2 Tipos de intent

Hay dos tipos principales de *intent*: los implícitos y los explícitos.

- ***Intent* explícito:** este tipo siempre especifica qué aplicación lo gestionará, bien incluyendo el nombre del paquete de la aplicación de destino o el nombre de la clase del componente, es decir, solo sería válido para comunicarse con una aplicación determinada, sin dejar al usuario la elección de esta.

- ***Intent* implícito:** intuitivamente al leer la definición del *intent* implícito, este tipo deja libre la elección de la aplicación de gestión al usuario, es decir, no nombran el componente específico, sino que declaran una acción general para realizar, lo cual permite que un componente de otra aplicación que se haya registrado para recibir el tipo de *intent* enviado.

En nuestro caso, hemos gestionado la aplicación con los dos tipos de mecanismos de *intent*: explícito (ver Figura 3-5) que hace alusión a una IM en concreto (Whatsapp en este caso) e implícito (ver Figura 3-6) que deja a elección del usuario la aplicación por la que se desea mandar el mensaje encriptado.

```
Intent shareIntent = new Intent();
shareIntent.setAction(Intent.ACTION_SEND);
shareIntent.putExtra(Intent.EXTRA_STREAM, uri);
shareIntent.setType("application/octet-stream");
startActivity(Intent.createChooser(shareIntent,
getResources().getText(R.string.bt_send_String)));
```

Figura 3-6 Ejemplo de *intent* explícito (elaboración propia)

3.5 Mecanismos de seguridad

3.5.1 Cifrado

Para tratar de dar seguridad a la información que se desea enviar, un método elegido es la encriptación del mensaje. A continuación, se exponen los principales métodos de encriptación.

3.5.1.1 AES (*Advanced Encryption Standard* [68])

El funcionamiento de este método de encriptación se basa en manipular los datos utilizando una serie de operaciones binarias complejas. De esta forma, los datos quedarían reordenados según una clave de distinta longitud dependiendo del tipo de método (128, 192 o 256 bits). Solo con esta clave se podrían revertir las operaciones realizadas para desencriptar y obtener el mensaje inicial.

Así pues, AES se puede considerar un modelo de cifrado simétrico en el que, como podemos ver en la Figura 3-7, el emisor encripta el mensaje con una clave secreta, la cual es compartida entre remitente y destinatario, y cuando el receptor recibe el mensaje solo tiene que desencriptarlo con la clave privada que ambos comparten. En la aplicación a desarrollar, se hará uso de AES como mecanismo de protección de la confidencialidad.

3.5.1.2 RSA (*Rivest, Shamir y Adleman* [68])

Se trata de un sistema de cifrado de clave pública, es decir, todos los usuarios poseen una clave privada, la cual solo es conocida por ellos mismos y una clave pública compartida a través de una infraestructura de distribución de claves a la que todos tienen acceso.

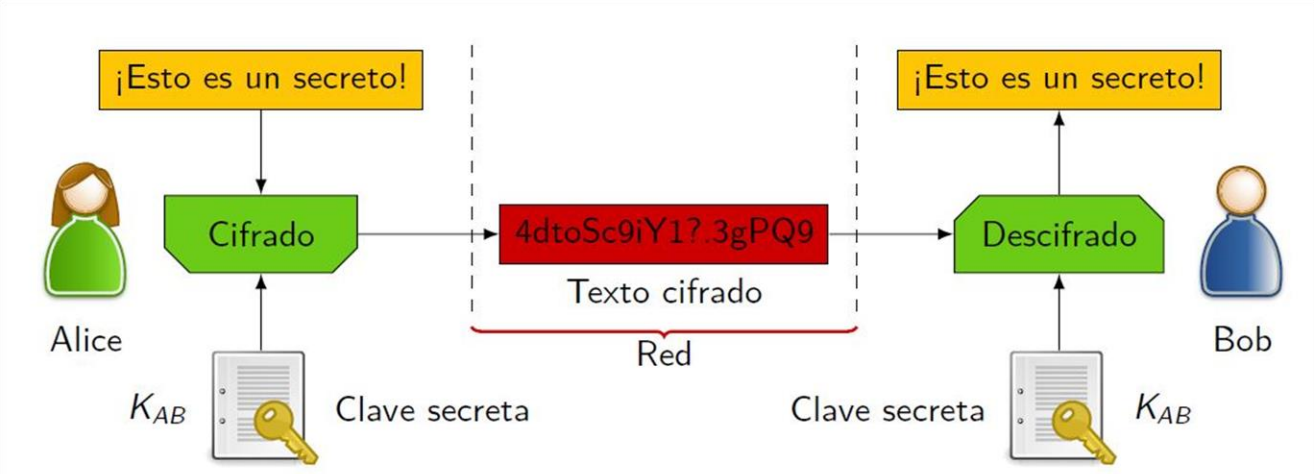

Figura 3-7 Ejemplo de cifrado de clave privada [11]

En la Figura 3-8 se puede observar cómo funcionaría un esquema de cifrado de clave pública como el de RSA. Básicamente, el remitente cifra un mensaje con la clave pública del usuario al que quiere mandarle la información y, de esta forma, es solo el usuario de destino el que puede descifrar el mensaje encriptado puesto que solo él posee la clave privada necesaria para descifrarlo.

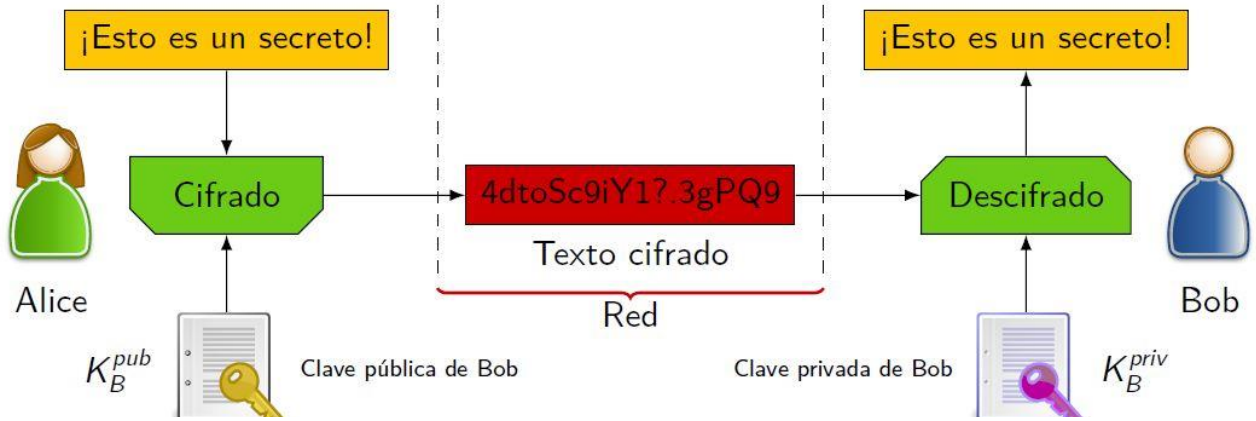

Figura 3-8 Ejemplo de cifrado de clave pública [11]

3.5.2 Integridad SHA (Secure Hash Algorithm)

Como se explicó en el apartado 2.2, la integridad no es más que la certeza o seguridad de que la información enviada no se ha visto alterada en ningún momento por una tercera persona ajena al emisor y receptor del mensaje. Para ello, usaremos un esquema de *hashing* [69] (ver Figura 3-9) conocido como SHA, que no es más que una construcción criptográfica no reversible que reduce el mensaje a un resumen o huella de mucho menor tamaño (256, 512 bits o similar). Este mecanismo es usado para la comprobación de la integridad y autenticidad de la información.

En nuestro caso añadiremos a los mensajes cifrados una huella SHA256, como se puede ver en la Figura 3-10, que se verificará en recepción para comprobar que el mensaje no ha sido alterado. Existen distintas variantes y evoluciones del algoritmo como SHA256, SHA384 o SHA512, aunque es cierto que la más usada es la primera.

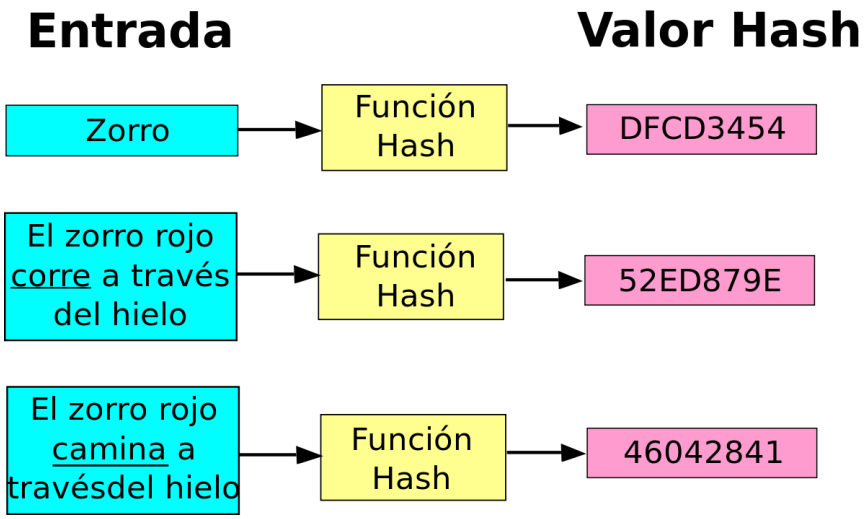

Figura 3-9 Esquema de hashing [66]

```
public static byte[] SHA256 (byte datos[]) {

    byte[] digest = null;
    try{
        MessageDigest md = MessageDigest.getInstance("SHA-256");
        digest = md.digest(datos);
    } catch (NoSuchAlgorithmException e) {
        e.printStackTrace();
    }
```

Figura 3-10 Función que genera SHA (elaboración propia)

3.5.3 Prevención de un ataque por repetición

Se trata de una vulnerabilidad más a la que puede estar expuesta una aplicación de mensajería instantánea en este caso. Es un tipo de ataque en el cual el atacante captura la información que viaja por la red, por ejemplo, un comando de autenticación que se envía a un sistema informático, para, posteriormente, enviarla de nuevo a su destinatario, sin que este note que ha sido capturada (ver Figura 3-11). Si el sistema informático o aplicación es vulnerable a este tipo de ataques, el sistema ejecutará el comando, como si fuera legítimo, enviando la respuesta al atacante que puede así obtener acceso al sistema [70].

Para protegerse de este tipo de ataques el sistema informático puede tomar medidas como usar un control de identificación de comandos o de sellado de tiempos (*timestamp*), junto con el cifrado y la firma de los comandos con el fin de evitar que sean reutilizados. En nuestro caso, en la aplicación usaremos un control de identificación de mensajes de sellado de tiempos, a través de la fecha y hora de envío de cada mensaje, que se añadirá como parte de la información citada que se envía al receptor.

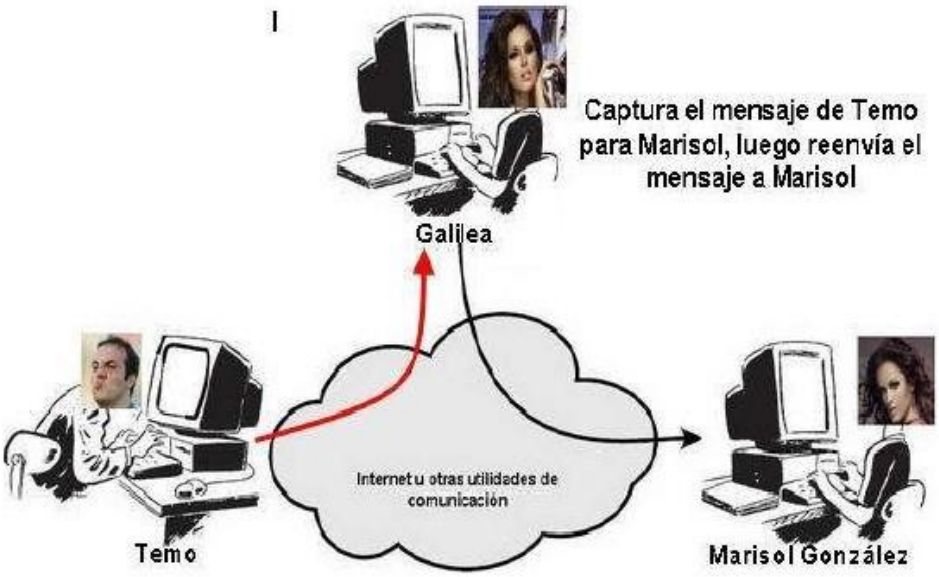

Figura 3-11 Esquema de ataque por repetición [70]

3.6 Desarrollo de la aplicación

En este apartado se pretende explicar en detalle cómo se ha desarrollado la aplicación y cómo funciona. La *app* es muy sencilla de usar. Como se puede ver en la Figura 3-12, se origina un mensaje en MILChat que luego se envía a un contacto a través de la aplicación de IM deseada. A continuación, el receptor abre el mensaje que le ha llegado por la IM comercial y lo visualiza con MILChat. Evidentemente, sin estar en posesión de la aplicación desarrollada, MILChat, no sería posible acceder al contenido del mensaje.

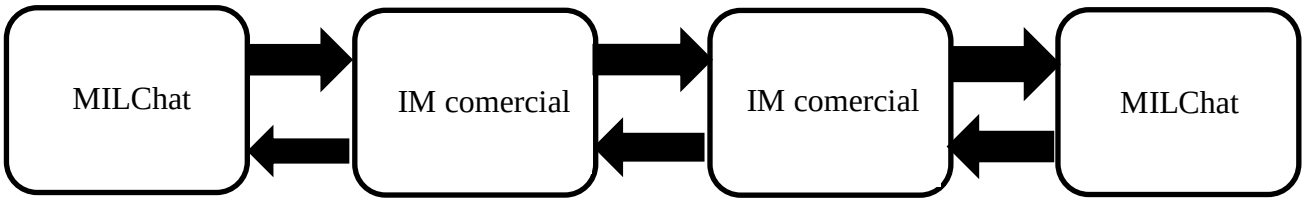

Figura 3-12 Esquema de funcionamiento MILChat (elaboración propia)

Como mencionamos anteriormente, se desea completar el envío de información de un modo "seguro" a través de otra aplicación de mensajería instantánea comercial ya existente. Para ello, siguiendo el funcionamiento lógico de ésta, la hemos dividido en sus partes más importantes:

- ***MainActivity*:** pantalla y actividad principal donde el usuario puede redactar un mensaje y una contraseña para posteriormente presionar un botón de "encriptar y enviar". Además, en esta pantalla se visionará el mensaje recibido, tras la introducción de la contraseña correcta, se tendrá la posibilidad de borrar el contenido de la pantalla con un botón y se podrá seleccionar la aplicación comercial para el envío del mensaje.

- ***SendEncrypt*:** acción que, mediante una contraseña que el usuario introduzca, encripte el mensaje deseado, lo convierta en un archivo *.bin* (archivo de *bytes*) y realice el envío de este a través de la aplicación IM comercial.

- ***Decrypt*:** de forma análoga a la acción anterior, esta actividad deberá, previa introducción de la contraseña correcta, descifrar el mensaje recibido a través de la aplicación (IM) cualquiera por la que la información llegue.

- ***GenerateKey*:** en esta acción generaremos la contraseña cifrada necesaria para encriptar y desencriptar nuestro mensaje.

A continuación, desarrollaremos en detalle cada una de estas.

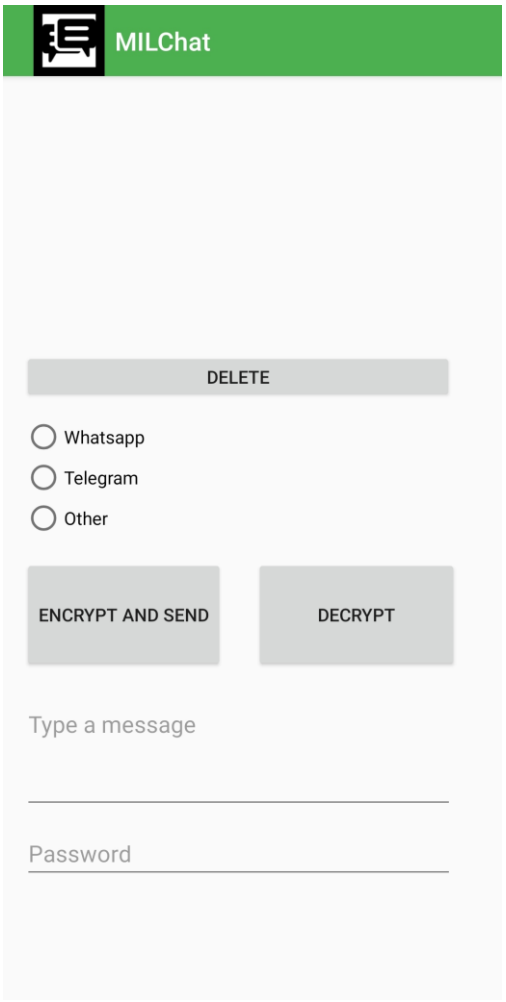

Figura 3-13 Menú principal (elab. prop.)

3.6.1 MainActivity

Como se ha descrito anteriormente, nuestra aplicación posee un menú principal, como se puede ver en la Figura 3-13, formado por varios elementos. Una caja de visionado de texto (*TextView o* TV) en la parte superior de la pantalla, debajo del nombre de ésta, para ver el mensaje recibido, dos cajas de edición de texto (*EditText o* ET), una para el mensaje y otra para la contraseña, tres botones (*Button o* BT), uno para enviar y encriptar, otro para desencriptar y otro para borrar todo justo debajo del TV, y un *RadioGroup* en el que existen tres botones *RadioButton*, que sirven para seleccionar la aplicación medio a través de la cual se quiere enviar el texto.

Esta actividad, la principal, al contener todos los accesos a todas las demás actividades, es la más larga. Trataremos de explicar lo que hace la parte del código más reseñable:

- ***Bt_delete*:** para empezar, tenemos el funcionamiento del botón "*delete"* (ver Figura 3-14). El empleo de este es muy sencillo, al pulsar el botón simplemente se borran todos los campos que se hayan podido rellenar (la caja mensaje, la caja de edición del mensaje y la de la clave), estableciendo cadena vacía en cada uno de estos.

```
btdelete.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        tv_message.setText("");
        et_message.setText("");
        et_password.setText("");
    }
});
```

Figura 3-14 Botón "*delete*" (elaboración propia)

- ***Bt_decrypt*:** este es el último de los botones de la función principal. En este simplemente se descifra el mensaje recibido a través de la aplicación de IM comercial (ver Figura 3-15). Para ello, el receptor debe introducir la clave correcta en el ET "password". Finalmente, al pulsar el botón se envía el mensaje cifrado y la clave a la función "desencriptar" y, tras eso, se visualiza el mensaje desencriptado por pantalla.

```
btdecrypt.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        try {
            if (tv_message != null) {
                datosencriptadosString = et_message.getText().toString();
                textoSalida = desencriptar(datosencriptadosString,
                et_password.getText().toString());
                tv_message.setText(textoSalida);
            }
        } catch (Exception e){
            e.printStackTrace();
        }
    }
});
```

Figura 3-15 Botoón "*decrypt*" (elaboración propia)

- ***Bt_encrypt*:** este es el funcionamiento de otro de los botones de la actividad principal (ver Figura 3-16). En este caso, si los dos ET están rellenos, se obtiene el mensaje introducido por el usuario y la fecha y hora de redacción de éste, a través de la función "*getDate*". Más tarde, se unen en una cadena y se mandan a la función "encriptar", que será explicada más adelante, la cadena mensaje más fecha-hora y "contraseña" y se publica el mensaje encriptado en la caja de visualización, lo que resulta útil a la hora de depurar. Posteriormente, se convierte el mensaje ya cifrado a un vector de *bytes* y se hace uso de la función *hash* para calcular la huella del mensaje y el sello, que será explicada más tarde. Después, se escribe en un archivo binario el *hash* de comprobación de la integridad, el sello temporal y el mensaje cifrado en vector de *bytes*. Por último, según la opción marcada en el *RadioGroup*, se llama a uno de los tres *intent* que permiten seleccionar de entre tres opciones posibles la aplicación de mensajería instantánea que se utilizará para transportar el mensaje cifrado. Dos de las opciones están asociadas a las aplicaciones más populares del mercado: Whatsapp y Telegram, mientras que la tercera nos permitiría elegir entre otras intaladas en el dispositivo.

```
btencrypt.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if ((et_message.getText().toString().trim().length()==0) ||
            (et_password.getText().toString().trim().length()==0)){
            tv_message.setText("El campo 'message' y/o el campo 'password' están
             vacíos. Por favor, rellénelo.");
        } else {
            try {
                // Obtengo el mensaje introducido por el usuario
                String msj = et_message.getText().toString();
                //Obtengo el sello temporal y lo concateno con el mensaje
                //Date d = new Date(System.currentTimeMillis());
                String d = getDate();
                msj = "[" + d + "] " + msj;
                //Cifro sello temporal y mensaje, para terminar convirtiéndolo a bytes
                msjEncryptString = encriptar(msj, et_password.getText().toString());
                tv_message.setText(msjEncryptString);
                byte[] msjEncryptBytes = msjEncryptString.getBytes();
                // Obtengo el hash del sello temporal y el mensaje
                byte[] hash = SHA256(msjEncryptBytes);
                //Guardo el resultado en un fichero en formato binario
                File documentsPath = new File(context.getFilesDir(), "");
                File file = new File(documentsPath, "Mensaje.bin");
                Log.i("Test", file.getAbsolutePath());
                FileOutputStream out = new FileOutputStream(file);
                out.write(hash);
                out.write(msjEncryptBytes);
                out.flush();
                out.close();

                Uri uri = FileProvider.getUriForFile(context, "es.enm.basicapp", file);
                Log.i("Test", uri.toString());

                if (r1.isChecked()==true){
                    Intent shareIntent = new Intent();
                    shareIntent.setAction(Intent.ACTION_SEND);
                    shareIntent.putExtra(Intent.EXTRA_STREAM, uri);
                    shareIntent.setType("application/octet-stream");
                    shareIntent.setPackage("com.whatsapp");
                    startActivity(Intent.createChooser(shareIntent,
                    getResources().getText(R.string.bt_send_String)));
                }

                if (r2.isChecked()==true){
                    Intent shareIntent = new Intent();
                    shareIntent.setAction(Intent.ACTION_SEND);
                    shareIntent.putExtra(Intent.EXTRA_STREAM, uri);
                    shareIntent.setType("application/octet-stream");
                    shareIntent.setPackage("org.telegram.messenger");
                    startActivity(Intent.createChooser(shareIntent,
                    getResources().getText(R.string.bt_send_String)));
                }

                if (r3.isChecked()==true) {
                    Intent shareIntent = new Intent();
                    shareIntent.setAction(Intent.ACTION_SEND);
                    shareIntent.putExtra(Intent.EXTRA_STREAM, uri);
                    shareIntent.setType("application/octet-stream");
                    startActivity(Intent.createChooser(shareIntent,
                    getResources().getText(R.string.bt_send_String)));
                }
            } catch (Exception e) {
                e.printStackTrace();
```

Figura 3-16 Botón "*encrypt*" (elaboración propia)

- **Recepción de un mensaje:** a continuación, tenemos el funcionamiento de la recepción de un mensaje. Básicamente, esta función se ejecuta cuando al enviar una comunicación seleccionamos como aplicación medio a nuestra propia *app*, lo que resulta útil a la hora de depurar, o cuando pulsamos sobre un mensaje recibido en otra aplicación de mensajería (ver Figura 3-17). Tras diversas operaciones, se consigue extraer el *hash*, que ha sido originado en una función de menor importancia llamada "SHA256". En ella, simplemente se obtiene el *hash* recibido y se calcula de nuevo otro con el mensaje que acaba de llegar para finalmente compararlos y verificar así que no ha sido alterado en el camino. Después, se envían el sello temporal y el mensaje original cifrado a la caja de edición de mensaje para continuar con la función de descifrado, explicada más adelante.

```
// Entra aquí cuando...
    // - Ponemos la propia aplicación (en lugar de WhatsApp) como destino del intent de
        compartir
    // - Usamos WhatsApp pero seleccionamos el mensaje y hacemos un compartir

    //Declaración URI
    Uri uri = null;
    String strMsj, strHash, strHashMsj, strTimeMsj;
    String strTime = null;
    if ( Intent.ACTION_SEND.equals(action) && type != null) {
        if ("application/octet-stream".equals(type)) {
            uri = intent.getParcelableExtra(Intent.EXTRA_STREAM);
        }
    } //Entra aquí cuando hacemos clic directamente sobre los datos en WhatsApp
    else if ( Intent.ACTION_VIEW.equals(action) && type != null) {
        if ("application/octet-stream".equals(type)) {
            uri = getIntent() != null ? getIntent().getData() : null;
        }
    }
    if (uri != null) {
        Log.i("Test", uri.toString());
        try {
            byte[] data = getBytes(context, uri);
            // Extraer hash de los datos (primeros 32 bytes)
            byte[] hashMsg = Arrays.copyOfRange(data, 0, 32);
            // Convertir a texto en base64 el hash que viene con el mensaje
            strHashMsj = Base64.encodeToString(hashMsg, Base64.DEFAULT);
            // Extraer los datos (resto de bytes que contienen el sello temporal y el
               mensaje)
            byte[] timeMsjEncryptBytes = Arrays.copyOfRange(data, 32, data.length);
            // Recalcular el hash para ver si coincide con el recibido
            byte[] hash = SHA256(timeMsjEncryptBytes);
            // Calcular el texto en base64 del hash recalculado
            strHash = Base64.encodeToString(hash, Base64.DEFAULT);
            // Comparar el texto base64 de los hash para ver si son iguales
            if (!strHash.equals(strHashMsj)) {
                strMsj = "ERROR: Hash no coincide";
            } else {
                //Imprimir en la interfaz el sello temporal con el mensaje
                strMsj = new String(timeMsjEncryptBytes, StandardCharsets.UTF_8);
                et_message.setText(strMsj);
            }
        } catch (Exception e) {
            strMsj = e.getLocalizedMessage();
            strHash = "ERROR: Excepción";
        }
    }
}
```

Figura 3-17 Código de recepción de un mensaje (elaboración propia)

3.6.2 SendEncrypt

Este método corresponde al botón "*encrypt and send*", es decir, cuando el usuario pulse en ese botón, se llevarán a cabo una serie de acciones que posibilitarán el funcionamiento deseado. Como ya hemos explicado anteriormente, el "*bt_encrypt*", da lugar a un algoritmo secuencial que hace uso de varios métodos entre los que se encuentra la función de enviar y cifrar.

Como se puede ver en el código de la Figura 3-18, esta función recibe dos parámetros: los datos que el usuario desea transmitir y la contraseña que ha sido elegida por el transmisor para el mensaje. Posteriormente, se hace una llamada al método que genera la huella de la contraseña, que será explicado en apartados posteriores. Tras eso, se elige el método de cifrado, en nuestro caso "AES", y se encripta con la clave introducida, posteriormente a su paso por la función "*GenerateKey*", la cual nos facilita la huella o resumen SHA-256 que usaremos para cifrar el mensaje. Por último, los datos ya cifrados y almacenados en un vector de *bytes* se pasan a una cadena y se devuelven a la *MainActivity*.

```
private String encriptar(String datos, String password) throws Exception{
    SecretKeySpec secretKey = generateKey(password);
    Cipher cipher = Cipher.getInstance("AES");
    cipher.init(Cipher.ENCRYPT_MODE, secretKey);
    byte[] datosEncriptadosBytes = cipher.doFinal(datos.getBytes());
    String datosEncriptadosString = Base64.encodeToString(datosEncriptadosBytes,
    Base64.DEFAULT);
    return datosEncriptadosString;
}
```

Figura 3-18 Actividad de cifrado (elaboración propia)

3.6.3 Decrypt

Nos encontramos en este apartado ante el método opuesto y complementario al explicado en el punto anterior. Esta función está asocidada a la pulsación del botón "*decrypt*", mediante la cual el receptor del mensaje será capaz de ver el contenido de este una vez descifrado.

Para ejecutar esta función, el mensaje a descifrar lo tendremos en nuestra caja de edición de mensaje expuesto, con lo que solo nos faltaría para proceder a la desencriptación, la contraseña original con la que el emisor cifró la información, al estar utilizando una técnica de cifrado simétrico.

El funcionamiento del código es análogo al de la función de cifrado, pero a la inversa. El método recibe una cadena de datos a descifrar y una contraseña introducida por el usuario. En primer lugar, se llama a la función que nos genera la huella de la contraseña, necesaria para proceder al descifrado del mensaje. Tras esto, se determina el método de descifrado, de nuevo AES, y se selecciona el modo "desencriptación". Luego se obtiene el mensaje descifrado como un vector de *bytes*, por lo que lo convertimos en cadena de caracteres y se devuelve a la *MainActivity*, como se puede observar en la Figura 3-19.

```
private String desencriptar(String datos, String password) throws Exception{
    SecretKeySpec secretKey = generateKey(password);
    Cipher cipher = Cipher.getInstance("AES");
    cipher.init(Cipher.DECRYPT_MODE, secretKey);
    byte[] datosDescodificados = Base64.decode(datos, Base64.DEFAULT);
    byte[] datosDesencriptadosBytes = cipher.doFinal(datosDescodificados);
    String datosDesencriptadosString = new String(datosDesencriptadosBytes);
    return datosDesencriptadosString;
}
```

Figura 3-19 Actividad de descifrado (elaboración propia)

3.6.4 GenerateKey

Como se ha explicado en las dos funciones anteriores, este método es necesario para poder obtener una huella o *hash* de la clave introducida. Ésta se supone compartida por emisor y receptor, pues estamos ante un mecanismo de cifrado simétrico en el que la clave es la misma para ambas partes y debe transmitirse por otro medio antes de realizar la comunicación. El medio utilizado en la práctica para el intercambio de claves queda fuera del alcance del presente proyecto.

Hablando del código, el funcionamiento es el siguiente. La función espera que se le pase la contraseña que previamente ha introducido el usuario. Lo primero que hacemos es generar la instancia de SHA-256 que es necesaria para calcular la huella de la clave. Luego, pasamos la cadena de datos *password* a un vector de *bytes* siguiendo el estándar UTF-8 [71], que no es más que una codificación de caracteres. Finalmente, se calcula la huella o resumen SHA-256 de la contraseña y ésta es la que se utilizará a la hora de especificar la clave de cifrado o descifrado de AES.

```
private SecretKeySpec generateKey(String password) throws Exception{
    //GENERACIÓN SHA-256
    MessageDigest sha = MessageDigest.getInstance("SHA-256");
    //PASO A BYTE DE LA CADENA
    byte[] key = password.getBytes("UTF-8");
    key = sha.digest(key);
    //CÁLCULO DE LA HUELLA SHA DE LA CONTRASEÑA
    SecretKeySpec secretKey = new SecretKeySpec(key, "AES");
    return secretKey;
}
```

Figura 3-20 Cálculo de huella de la clave (elaboración propia)

Con la consecución de estas funciones o métodos, conseguimos que la aplicación cumpla con los objetivos deseados: edición de un mensaje que se envíe de forma cifrada a través de una aplicación externa y que se pueda recibir y descifrar mendiante el uso de una clave previamente compartida.

4 RESULTADOS Y PRUEBAS

En este capítulo se pretende explicar cómo se ha llevado a cabo la instalación y prueba de la aplicación MILChat en dispositivos móviles y un resumen del funcionamiento para acabar con una búsqueda de errores y depuración del código hasta llegar a una versión suficientemente sólida y robusta de la aplicación.

4.1 Instalación de la aplicación

El primer paso para usar la aplicación y comprobar su correcto funcionamiento es instalarla en algún dispositivo móvil. Para ello, debemos generar el arhivo APK (*Android Application Package*, en español Paquete de Aplicación Android) ya que no se tiene licencia de desarrollador de Google Play Store.

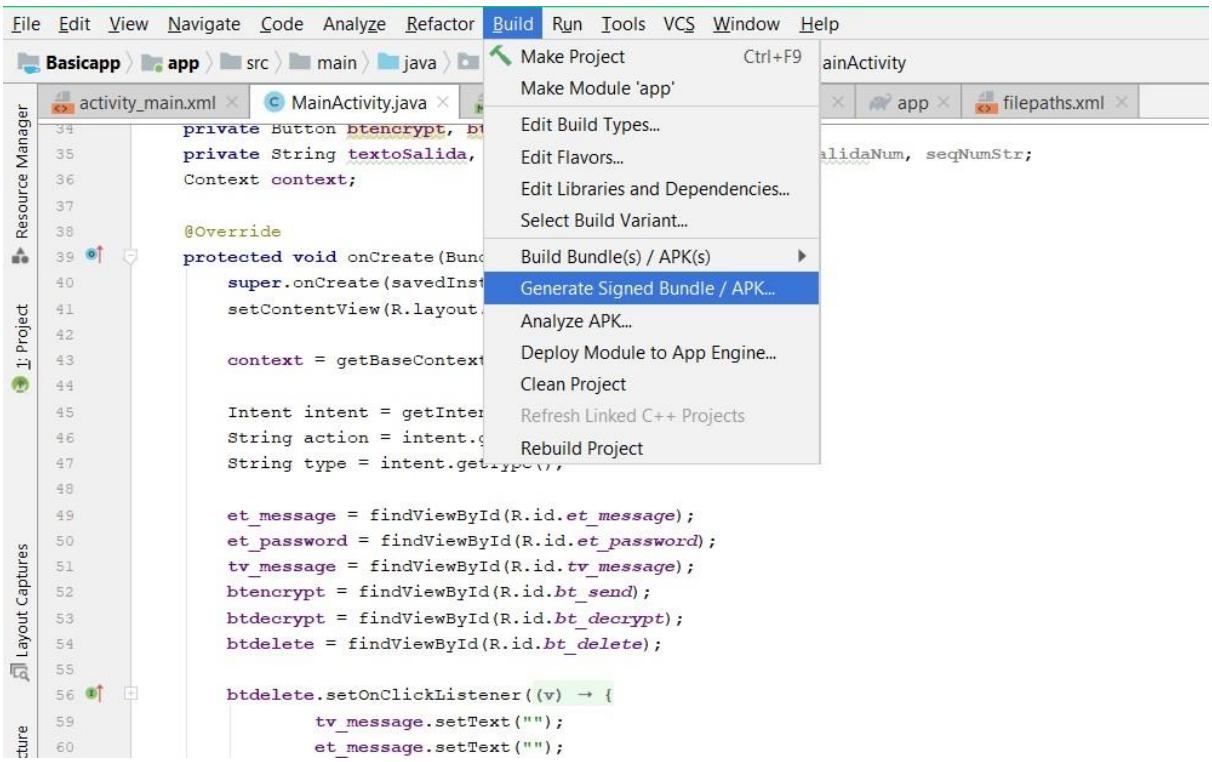

Figura 4-1 Generación de archivo APK (elaboración propia)

El proceso de obtención de este archivo es sencillo. Simplemente hay que, como se muestra en la Figura 4-1, seguir el siguiente camino en Android Studio: *Build* (en el menú de herramientas principal) → *Generate signed bundle/*APK. Esta acción depositará el archivo APK perteneciente al proyecto que nos concierne en la siguiente ruta: \AndroidStudioProjects\MILChat\app\releaseMILChat.apk.

En la mayoría de los dispositivos móviles, la opción de aceptar la instalación de aplicaciones de orígenes desconocidos viene desactivada por defecto, por lo que el usuario que desee usar la aplicación deberá activar esta opción solo durante la instalación de esta. Tras este proceso, se creará un acceso directo en el menú de nuestro dispostivo y ya podremos comenzar a usarla.

4.2 Funcionamiento de la aplicación

La aplicación que nos ocupa, MILChat (ver Figura 4-2), tiene dos funciones principales y complementarias la una de la otra. Por un lado, es capaz de permitir que se esciba un mensaje, encriptarlo y compartirlo con un destinatario a través de una aplicación de IM y, por el otro lado, es capaz de abrir un mensaje recibido en binario generado por la misma aplicación en otro dispositivo y descifrarlo para visualizarlo en pantalla.

Figura 4-2 Icono de MILChat (elaboración propia)

4.2.1 Cifrado y envío de mensaje

Como se ha explicado en el capítulo anterior, la aplicación tiene determinadas funcionalidades que permiten el envío de un mensaje cifrado a través de una aplicación de IM comercial.

Como se puede ver en la Figura 4-3, primero se permite al usuario editar texto, sin límite de caracteres, el cual será el cuerpo del mensaje a enviar. Luego se obliga al usuario a coger una opción de entre tres posibles (*Whatsapp*, *Telegram* u *Other*) para así determinar a través de qué aplicación de IM se enviará la información. Por último, se fuerza al emisor emisor a establecer una contraseña con la que se encripta el mensaje, de forma que su posterior descifrado solo sea posible con esa misma contraseña (esta se presupone compartida por otros medios entre emisor y receptor). No es necesario introducir la clave cada vez que se quiera enviar un mensaje, mientras no se realice otra acción. Mientras no pulsemos el botón "*delete*" o realicemos una desencriptación, la clave que hemos introducido seguirá activa en el ET de la *password*, por lo que se podrá enviar tantos mensajes como se desee sin variar la clave.

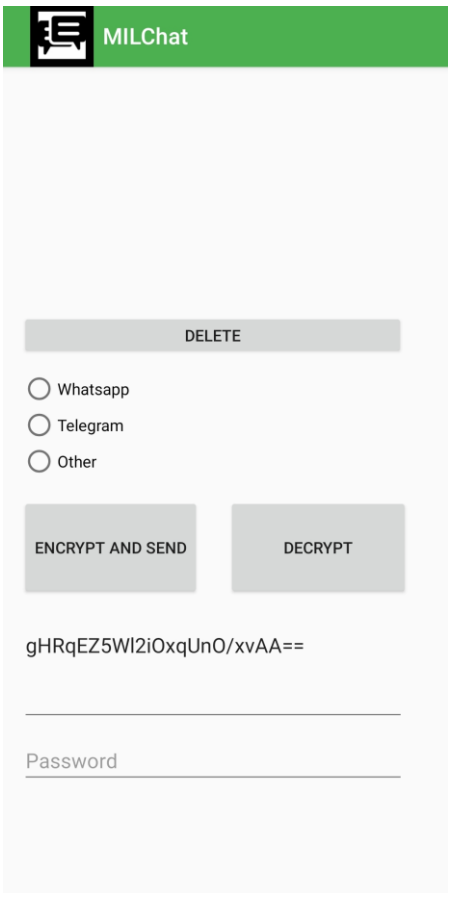

Figura 4-3 Envío de mensaje de ejemplo (elaboración propia)

4.2.1 Recepción y descifrado de mensaje

Tras el envío de un mensaje a un receptor determinado a través de una aplicación de IM comercial elegida por el emisor de este, el usuario de destino obtendrá en su chat de la IM un archivo binario cifrado que contendrá el mensaje original, como se puede ver en la Figura 4-4.

Figura 4-4 Archivo binario en Whatsapp (elaboración propia)

Si el receptor tiene la aplicación MILChat instalada en el dispositivo móvil, al pulsar en el archivo de la conversación de chat, se desplegará nuestra aplicación. Tras la apertura de esta se mostrará el mensaje cifrado en la caja de edición de mensaje, como se puede ver en la Figura 4-5. Si queremos finalmente descifrar y visualizar el mensaje, deberemos introducir la contraseña correcta en el *editText* que hay habilitado para ello y, entonces, si todo se ha desarrollado correctamente se mostrará la información que el emisor redactó en un principio.

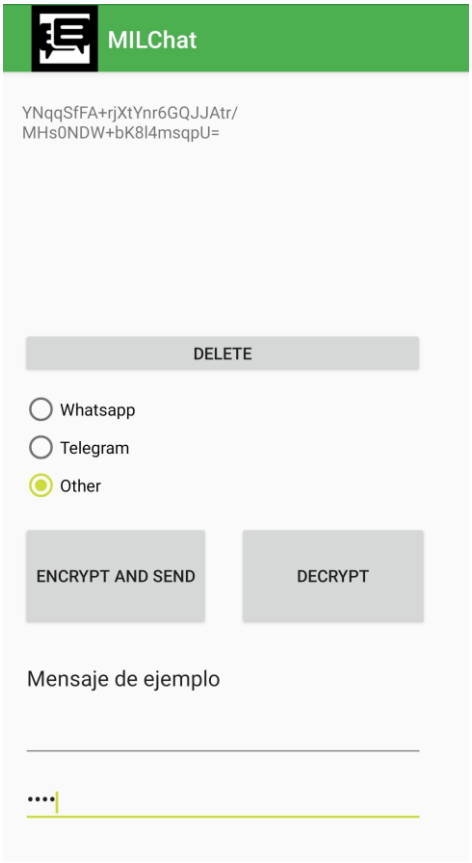

Figura 4-5 Recepción de mensaje de ejemplo (elaboración propia)

4.3 Pruebas realizadas

Aparte del correcto funcionamiento de la aplicación, el cual ha sido probado como explicaremos a continuación, las pruebas han ido encaminadas principalmente a los mecanismos de seguridad que esta ofrece. Además, dado que el enfoque de esta aplicación es destinado a un uso militar o de fuerzas y cuerpos de seguridad, no es interesante pensar en parámetros de diseño que la hagan más atractiva de cara a los usuarios, pues ésta no pretende ser expuesta en la tienda por defecto para usuarios Android, Google Play Store.

Para depurar la aplicación y detectar los posibles errores que pueda tener, así como comprobar que cumple con los objetivos marcados para su correcto funcionamiento, se han hecho pruebas en el emulador que posee Android Studio y, posteriormente, se ha distribuido y probado en varios dispositivos móviles distintos, de compañías diferentes y se han probado todos los posibles casos que se puedan dar en la aplicación. Se procede a detallar la búsqueda metódica de errores.

- Se procede a rotar la pantalla para comprobar la organización de la *MainActivity* en horizontal y vertical.

    - Se descubre que al rotar la pantalla a formato *landscape* no se conserva bien la organización de los botones y cajas de texto, por lo que se decide restringir esta funcionalidad y no permitir la rotación de pantalla, de forma que solo se puede usar en formato *portrait*. Para ello, se introduce en el AndroidManifest.xml la siguiente línea de código de la Figura 4-6:

```
<activity android:name=".MainActivity"
    android:screenOrientation="portrait">
```

Figura 4-6 Fragmento AndroidManifest (elab. prop.)

- Se procede a usar la aplicación sin conexión a internet.

    - Como se esperaba, la aplicación se ejecuta con total normalidad. Simplemente al no tener conexión, ya en la IM comercial que se use en cada caso, el mensaje no se enviará, pero cuando el usuario vuelva a tener conexión se procederá a terminar su emisión.

- Falta de relleno de algún campo.

    - En el envío, si no se rellena algún campo como el de "*type a message*" o el de "*password*", aparecerá un mensaje en el TV (parte superior de la Figura 4-7) exhortando al usuario al correcto uso de la aplicación.

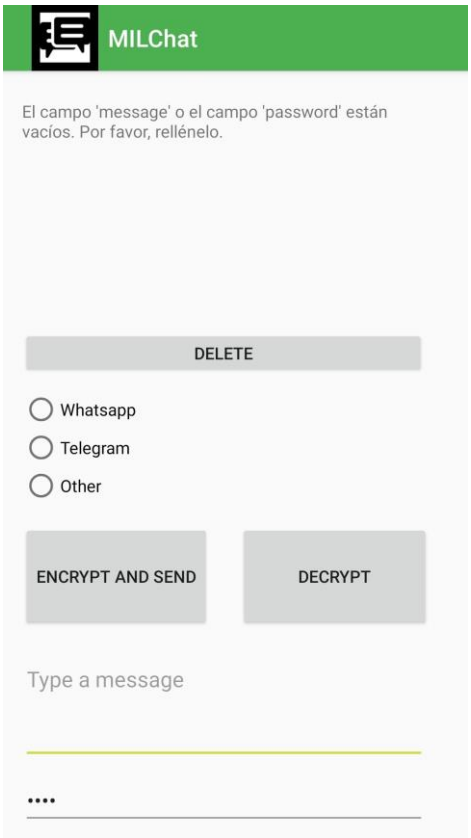

Figura 4-7 Mensaje de error (elaboración propia)

- En el caso que no se seleccione una de las tres opciones de IM, el mensaje se cifrará, pero, evidentemente, no se mandará por ninguna aplicación hasta que se elija una opción.

➢ Error en la contraseña.

- Al principio, no era necesaria la introducción de la contraseña correcta para descifrar un mensaje, pues simplemente con la escritura de un texto ya lo daba como la *password* adecuada. Se corrigió en el método de desencriptación.
- En un origen, la cadena vacía podía servir como contraseña o como mensaje a mandar, sin necesidad de escribir ningún carácter. Se restringió el funcionamiento solo cuando los campos estuviesen rellenos, mostrando un mensaje de advertencia por pantalla para guiar al usuario, igual al mostrado en la Figura 4-7.

➢ Errores según versión de Android o marca del dispositivo móvil.

- Se instala y prueba en un *smartphone* actual (2018), con Android 9.0 Pie (API 28), Xiaomi Mi 8, con una pantalla de 6 pulgadas. En este dispositivo no se detectaron errores relativos a la versión de Android o a la marca del teléfono.
- Se instala y prueba en un *smartphone* actual (2018), con Android 10.0 (API 29), Samsung Galaxy S9 Plus, con una pantalla de 6,2 pulgadas. En este dispositivo tampoco se detectaron errores relativos a la versión de Android o a la marca del teléfono, lo cual es bastante positivo dado que este teléfono usa la última versión de API de Android disponible en el mercado, al comienzo del desarrollo de la *app*.
- Se instala y prueba en un *smartphone* actual (2019), con Android 9.0 Pie (API 28), Motorola G7 Play, con una pantalla de 5,7 pulgadas. En este dispositivo se observan errores de adaptación de la aplicación a la pantalla, al ser ésta algo más pequeña que las demás que se han probado (ver Figura 4-8).

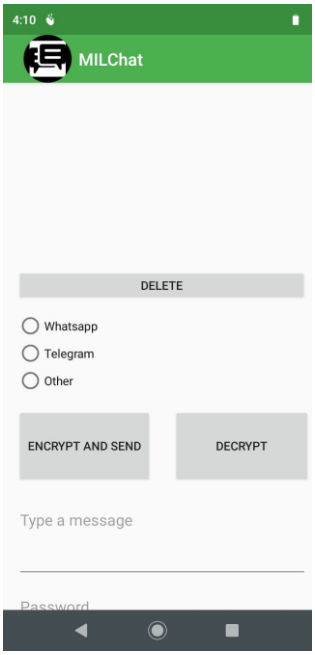

Figura 4-8 MILChat en Moto G7

- Se instala y prueba en un *smartphone* antiguo (2016), con Android 7.0 Nougat (API 24), BQ Aquaris U, con una pantalla de 5 pulgadas. En este dispositivo se vuelven a observar errores de adaptación de la aplicación a la pantalla. Este error se puede resolver de forma sencilla incorporando al archivo "*activity_main.xml*" un comando de ScrollView. Por simplicidad se decide no añadirlo, dejando claro que su implementación sería bastante sencilla.

➢ Permanencia de mensaje en pantalla.

- Se detecta que la información enviada o recibida que se muestra en pantalla, permanece indefinidamente hasta cerrar la aplicación por completo. Para subsanarlo, se habilita un botón "*delete*" que elimina toda la información de las cajas de la interfaz al pulsarlo, tal como se explicó anteriormente en la Figura 3-14.

➢ Integridad del mensaje.

- Se observa que no hay forma de saber si un mensaje ha sido interceptado y modificado en el camino de emisor a receptor. Para ello, se implementa un mecanismo de *hash*, SHA256, que comprueba que la información no ha sido alterada por el camino, en caso afirmativo, se muestra un error en pantalla informando sobre lo acaecido, como se puede apreciar en el siguiente fragmento de código (ver Figura 4-9).

```
// Comparar el texto base64 de los hash para ver si son iguales
if (!strHash.equals(strHashMsg)) {
    strMsg = "ERROR: Hash no coincide";
}
```

Figura 4-9 Error de integridad (elaboración propia)

- Además, para comprobar que este error se disparaba, se seleccionó mal adrede la longitud de *byte* del *hash*, a la hora de separarlo del mensaje original (ver Figura 4-10). Se escribió el rango 0-31, en vez del correcto, 0-32, y como era de esperar nos condujo a un error.

```
byte[] data = getBytes(context, uri);
// Extraer hash de los datos (primeros 32 bytes)
byte[] hashMsg = Arrays.copyOfRange(data, 0, 32);
```

Figura 4-10 Separación de *hash* del mensaje (elaboración propia).

➢ Selección de una aplicación que no sea de IM.

- Se observa que al seleccionar la opción "*other*" antes del envío de un mensaje y seleccionar alguna aplicación que no sea de mensajería instantánea la comunicación es posible. Esto es, el envío de información puede ser llevado a cabo a través de otro tipo de app tales como correo electrónico o bluetooth. No está claro que esto sea un defecto, pues puede llegar a ser una ventaja el poder transmitir el archivo binario ya cifrado a través de otros métodos por si en algún caso alguna de las partes en comunicación carece de acceso a *apps* de IM. Por lo tanto, no se procede a corregir esta funcionalidad y dejar a libre elección de usuario el modo de

transmisión de la información, siempre teniendo en cuenta que la aplicación no se ha diseñado ni optimizado para otros métodos de transmisión.

- No tener instalada alguna de las dos aplicaciones seleccionadas.

  - En el caso de que no se tenga instalada alguna de las dos aplicaciones seleccionadas (*Whatsapp o Telegram*), las cuales han sido elegidas por ser las más utilizadas y conocidas a nivel mundial con una distribución demográfica uniforme, simplemente no se podrá llevar a cabo la transmisión a través de esa aplicación seleccionada y se nos notificará mediante un mensaje de error.

- Apertura automática de la aplicación MILChat.

  - Se descubrió que al pulsar directamente en el archivo binario recibido en el chat de la aplicación de mensajería instantánea usada, no se abría inmediatamente la aplicación MILChat. Por lo tanto, se procedió a crear el código que se puede ver en la Figura 4-11 y se solventó este problema. Ahora directamente al pulsar en el archivo se abre nuestra aplicación.

```
//Entra aquí cuando hacemos clic directamente sobre los datos en WhatsApp
else if ( Intent.ACTION_VIEW.equals(action) && type != null) {
    if ("application/octet-stream".equals(type)) {
        uri = getIntent() != null ? getIntent().getData() : null;
    }
}
```

Figura 4-11 Código de apertura directa de MILChat (elaboración propia)

- Ataque de repetición.

  - En una primera versión de la aplicación, ésta era vulnerable al ataque por repetición, anteriormente explicado (Figura 3-11). Para subsanarlo, se recurrió a un mecanismo que escribe en cada mensaje un sello temporal (*timestamp*) para así de esta forma poder identificar en qué momento fue escrito cada uno (ver Figura 4-12).

```
// Timestamp. 8 bytes del tiempo en segundos que ha pasado
desde el 1 de enero de 1970 hasta ahora
byte[] time = longToBytes(System.currentTimeMillis());
```

Figura 4-12 Creación de sello temporal (elaboración propia)

Tras depurar todos estos errores y fallos comentados en las líneas anteriores y haber instalado la aplicación en distintos dispositivos de empresas distintas y con varias versiones de API, podemos establecer con cierta seguridad que la aplicación se ejecuta de un modo correcto y cumple con la función y los objetivos marcados al inicio del proyecto.

5 CONCLUSIONES Y LÍNEAS FUTURAS

En este capítulo se hará una síntesis de las conclusiones a las que se ha llegado tras la realización del proyecto al completo y se dará unas breves pinceladas sobre las líneas posibles a seguir a partir del mismo.

5.1 Conclusiones

5.1.1 Características MILChat

Tras el esfuerzo y la culminación de todo el proyecto al completo, hemos conseguido generar una aplicación, con un peso de 2.1 MB, que cumple con los objetivos definidos al comienzo de la memoria en la sección 1.3. El usuario puede editar un mensaje y enviarlo cifrado a través de una aplicación que ha resultado ser cómoda, rápida y sencilla en su manejo. Por supuesto, el receptor del mensaje puede descifrarlo y visionarlo con las mismas características: de forma veloz, intuitiva y manejable.

Para hacer un resumen de los parámetros de seguridad que hemos llegado a alcanzar en el proyecto, se expone la siguiente tabla (ver Tabla 5-1 y Tabla 5-2):

	Confidencialidad	**Integridad**	**Autenticidad**	**No repudio**	**Disponibilidad**
MILChat	Sí	Sí	Sí[1]	No	No[2]

Tabla 5-1 Características MILChat 1 (elaboración propia)

	Cifrado extremo-extremo	**Código abierto**	**Recopilación de datos**	**Privacidad**
MILChat	Sí	Sí	No	No

Tabla 5-2 Características MILChat 2 (elaboración propia)

[1] Este parámetro se considera cumplido al mandar el mensaje al contacto que se desee a través de la IM comercial, siendo posible también realizar una vídeoconferencia para comprobar la autenticidad.

[2] En la propia aplicación no se ha hecho nada para asegurar la disponibilidad, pero hay que tener en cuenta que en las plataformas comerciales suele ser alta.

Cabe destacar que algunos de estos parámetros pueden variar según la aplicación a través de la cual se realice el envío del mensaje.

Es el caso de la confidencialidad. Si usamos una aplicación de IM que no asegure esta característica, esta no puede ser controlada por MILChat dado que no tenemos forma de gestionar a qué usuario enviará su mensaje el emisor.

La autenticidad ya queda confirmada, al ser un usuario desde su propio dispositivo móvil y con su cuenta en la aplicación IM que use para el envío, el que finalmente transmite la información. No podemos respaldar el caso de que el dispositivo y la cuenta de dicha IM sea usurpada o robada y se envíe un mensaje desde ésta. De todas formas, en algunas *apps* cabría la posibilidad de hacer una videoconferencia para confirmar la autenticidad de la otra parte, aunque este mecanismo, obviamente no es infalible.

Lo más importante a destacar en este apartado es la base sobre la que ya se partía antes del comienzo del proyecto. Las distintas medidas de seguridad que se han implementado están todas encaminadas a solventar las vulnerabilidades que la aplicación pueda ofrecer. El problema con el que ya se contaba es que no hay forma de, con este planteamiento de proyecto, asegurar que el dispositivo móvil en el que se ejecuta la aplicación sea confiable y no esté comprometido. no podemos asegurar que el terminal originador del mensaje esté comprometido antes de la redacción de la información y, por tanto, ésta pueda ser interceptada antes de que la aplicación cifre la información. Según un estudio de la Universidad de Cambridge [72], "el 87 % de todos los teléfonos Android está expuesto al menos a una vulnerabilidad crítica", por lo que es innegable que el riesgo existe. Cabe mencionar que por "vulnerabilidades críticas" se hace alusión a ciertos tipos de malware como ransomware, spyware, troyanos, etc.

5.1.2 Recomendaciones de uso

Los usuarios potenciales de esta aplicación serán sobre todo personas que trabajen en un entorno militar o policial, dado que existe en ese ámbito una necesidad de comunicación que ha de ser respondida con las características que puede ofrecer la aplicación.

Para llevar a cabo un mejor uso de las comunicaciones y poder evitar ciertos riesgos que puedan surgir, en la Guía STIC 440 [18], se recogen una serie de recomendaciones y consejos a la hora de utilizar una aplicación de mensajería instantánea, de entre los que cabe mencionar los que nos afectan directamente.

- **Evitar el uso de sistemas de IM públicos y/o comerciales:** cuando la información que se va a manejar es sensible, es aconsejable que el servidor que registra los mensajes, o "la nube" en algunos casos, sea interno a la propia organización que necesita realizar las comunicaciones.

- **Establecer una política general de seguridad:** debe haber un organismo interno que defina usos correctos e incorrectos de los sistemas de IM y que pueda controlar la utilización de estos.

- **Política de auditorías:** se debe contar con un registro de todas las comunicaciones realizadas, de manera que se puedan auditar posteriormente.

- **Políticas de concienciación:** en muchos casos los accidentes de seguridad en comunicaciones ocurren por el uso indebido de los IM y no porque el sistema tenga deficiencias y vulnerabilidades informáticas. Por ello, es necesario educar a todo el que vaya a hacer uso de las aplicaciones de mensajería instantánea.

5.2 Líneas futuras

Por supuesto, como en todo proyecto o trabajo, existe margen de mejora y posibles líneas de acción a seguir para enriquecer la aplicación que hemos desarrollado. Bien por tiempo disponible o complejidad no se han realizado, pero es necesario mencionarlas para que puedan servir de apoyo a posibles líneas de desarrollo futuro del proyecto.

- La más importante es la que supliría la vulnerabilidad de que el dispositivo móvil pueda verse comprometido. Se podría diseñar una aplicación, que permitiese editar un mensaje y cifrarlo, con la salvedad de que este proceso debería tener lugar en un dispositivo que no esté conectado a ninguna red y además sea propio de la organización que desee efectuar la comunicación para así asegurar la total confiabilidad en éste. Posteriormente, se debería pasar ese mensaje ya cifrado a algún dispositivo móvil por algún medio que sea difícilmente interceptable (*bluetooth*, óptico, sonido…) para poder enviarlo a través de algún sistema de IM. Finalmente, habría que seguir el mecanismo inverso para el receptor.

- Implementación de más funcionalidades en la aplicación:

    - Mejorar el diseño de la aplicación para hacerla más atractiva para el usuario. No se ha realizado por carecer de interés al no pretender vender la aplicación ni lucrarse de ella en tiendas como Google Play Store.

    - Implementar la función de envío de contenido multimedia como fotos, vídeos y audio e incluso archivos.

    - Diseñar una interfaz en varios idiomas distintos, de forma que sea más intuitiva para usuarios de distintos países.

    - Diseñar una interfaz con *ScrollView* que permita así poder rotar la orientación del dispositivo móvil.

- Investigar en formas de conseguir asegurar los parámetros de seguridad que no han sido alcanzados: no repudio, disponibilidad y privacidad. Para algunos de ellos podría ser de utilidad incorporar técnicas de cifrado asimétrico y firma digital.

- Diseño de una aplicación de mensajería instantánea al completo, interna y propia a la organización que desee explotarla. Así como la UME ha encargado el diseño de una *app* de IM que supla las necesidades de comunicaciones que sus usuarios poseen, la Armada Española, por ser el entorno al que pertenecemos, podría encargar un diseño de una aplicación que cumpla con todas las características de seguridad que hemos mencionada. Esto sería ideal y superaría cualquier mejora que se pueda hacer en este proyecto.

Evidentemente, por límites en presupuesto, tiempo, experiencia y conocimientos en este campo, un trabajo de tal envergadura no resulta asequible para un trabajo de fin de grado, pero sí que resultaría atractivo y útil pensando en solventar las deficiencias en comunicaciones instantáneas que se poseen.

6 BIBLIOGRAFÍA

[1] J. García Nieto, «Millones de dispositivos en Android,» [En línea]. Available: https://www.xatakandroid.com/mercado/android-suma-sigue-2-500-millones-dispositivos-activo-usan-como-sistema-operativo. [Último acceso: 28 enero 2020].

[2] Web de Apple, «App Store,» [En línea]. Available: https://www.apple.com/es/ios/app-store/. [Último acceso: 29 febrero 2020].

[3] Google, «Google Play Store,» [En línea]. Available: https://play.google.com/store/?utm_source=emea_Med&utm_medium=hasem&utm_content=Jun1115&utm_campaign=Evergreen&pcampaignid=MKT-EG-emea-es-all-Med-hasem-py-Evergreen-Jun1115-1%7cONSEM_kwid_43700007365263131&gref=EkUKPQoJCIC06PIFELQBEiwAiknIY6YibN7W82a0sMLM. [Último acceso: 29 febrero 2020].

[4] Azul Web, «Android vs iOS,» [En línea]. Available: https://www.azulweb.net/smartphones-android-vs-ios/. [Último acceso: 11 febrero 2020].

[5] Wikipedia, «PLATO,» [En línea]. Available: https://es.wikipedia.org/wiki/Programmed_Logic_Automated_Teaching_Operations. [Último acceso: 30 enero 2020].

[6] Wikipedia, «ICQ,» [En línea]. Available: https://es.wikipedia.org/wiki/ICQ. [Último acceso: 30 enero 2020].

[7] Techtastico, «Windows Live Messenger,» [En línea]. Available: https://techtastico.com/post/descargar-windows-live-messenger-2009/. [Último acceso: 29 febrero 2020].

[8] L. Castro, «Qué es la mensajería instantánea,» [En línea]. Available: https://www.aboutespanol.com/que-es-im-o-mensajeria-instantanea-y-como-funciona-157567. [Último acceso: 1 febrero 2020].

[9] SecurityArtWork, «Análisis funcionamiento sistemas de mensajería,» [En línea]. Available: https://www.securityartwork.es/2015/02/10/line-android-e-ios-analisis-tecnico-de-la-mensajeria-instantanea/. [Último acceso: 29 febrero 2020].

[10] Ecured, «Protocolos de comunicación en IM,» [En línea]. Available:

https://www.ecured.cu/Protocolo_para_la_mensajer%C3%ADa_instant%C3%A1nea#:~:text=IRC%20es%20un%20protocolo%20de,conectan%20los%20clientes%20o%20no.. [Último acceso: 29 febrero 2020].

[11] R. Asorey y N. Fernández García, *Apuntes de Fundamentos de redes de ordenadores. Ciberseguridad*, 2019/2020.

[12] AVG, «Parámetros de seguridad,» [En línea]. Available: https://www.avg.com/es/signal/secure-message-apps. [Último acceso: 8 febrero 2020].

[13] M. Sanz Romero, «Qué es y para qué sirve la ingeniería inversa,» [En línea]. Available: https://computerhoy.com/reportajes/tecnologia/consiste-ingenieria-inversa-396691. [Último acceso: 8 febrero 2020].

[14] C. C. N. «CCN,» [En línea]. Available: https://www.ccn.cni.es/index.php/es/menu-guias-ccn-stic-es/series-completas-guias-stic-ccn-cert-es. [Último acceso: 28 febrero 2020].

[15] Centro Criptológico Nacional, *CCN-STIC-101 - Procedimiento de seguridad de las TIC*, 2016.

[16] Centro Criptológico Nacional, *CCN-STIC-407 - Seguridad en telefonía móvil*, 2006.

[17] Centro Criptológico Nacional, *CCN-STIC-440 - Seguridad en mensajería instantánea*, 2007.

[18] Centro Criptológico Nacional, *CCN-STIC-440 - Seguridad en la mensajería instantánea*, 2007.

[19] «Whatsapp,» [En línea]. Available: https://www.whatsapp.com/. [Último acceso: 29 febrero 2020].

[20] «Telegram,» [En línea]. Available: https://telegram.org/. [Último acceso: 29 febrero 2020].

[21] «Facebook Messenger,» [En línea]. Available: https://www.messenger.com/. [Último acceso: 29 febrero 2020].

[22] «Skype,» [En línea]. Available: https://www.skype.com/es/. [Último acceso: 29 febrero 2020].

[23] «WeChat,» [En línea]. Available: https://www.wechat.com/es/. [Último acceso: 29 febrero 2020].

[24] Xataka, «Whatsapp implementa cifrado,» [En línea]. Available: https://www.xatakandroid.com/comunicacion-y-mensajeria/whatsapp-para-android-activa-el-cifrado-de-extremo-a-extremo-ahora-tus-conversaciones-son-seguras. [Último acceso: 13 febrero 2020].

[25] Xataka, «Cifrado extremo a extremo de Whatsapp,» [En línea]. Available: https://www.xataka.com/privacidad/whatsapp-ahora-es-seguro-por-completo-llega-el-cifrado-end-to-end. [Último acceso: 13 febrero 2020].

[26] R. Peco, «¿Es Whatsapp seguro?,» [En línea]. Available: https://www.lavanguardia.com/tecnologia/20200130/473210877416/whatsapp-seguro-seguridad-bezos-telegram-signal-cifrado-fallo-vulnerabilidades.html. [Último acceso: 30 enero 2020].

[27] Telegram, «MTProto Protocol,» [En línea]. Available: https://core.telegram.org/mtproto. [Último acceso: 15 febrero 2020].

[28] «Telegram FAQ,» [En línea]. Available: https://telegram.org/faq/es#p-cul-es-la-diferencia-de-los-chats-secretos. [Último acceso: 30 enero 2020].

[29] Wikipedia, «AES,» [En línea]. Available: https://es.wikipedia.org/wiki/Advanced_Encryption_Standard. [Último acceso: 3 febero 2020].

[30] SSL247, «¿Qué es RSA?,» [En línea]. Available: https://www.ssl247.es/certificats-ssl/rsa-dsa-ecc. [Último acceso: 1 febrero 2020].

[31] «Telegram FAQ,» [En línea]. Available: https://telegram.org/faq/es#p-qu-tan-segura-es-telegram. [Último acceso: 30 enero 2020].

[32] Xataka, «Interfaz de Facebook Messenger,» [En línea]. Available: https://www.xatakamovil.com/aplicaciones/facebook-messenger-estrena-interfaz-simple-limpia. [Último acceso: 15 febrero 2020].

[33] Xataka, «Interfaz de Skype,» [En línea]. Available: https://www.xatakawindows.com/telecomunicaciones/la-version-final-de-skype-con-interfaz-redisenada-para-windows-ya-esta-disponible. [Último acceso: 15 febrero 2020].

[34] Wikipedia, «P2P,» [En línea]. Available: https://es.wikipedia.org/wiki/P2P. [Último acceso: 3 febrero 2020].

[35] M. González, «Microsoft permite a la NSA monitorizar sus comunicaciones,» [En línea]. Available: https://www.genbeta.com/actualidad/microsoft-permite-a-la-nsa-interceptar-comunicaciones-de-skype-outlook-y-otros-servicios. [Último acceso: 16 febrero 2020].

[36] Payxpert, «Interfaz WeChat,» [En línea]. Available: https://www.payxpert.com/es/wechat/. [Último acceso: 15 febrero 2020].

[37] Spanish People Daily, «Abusos sexuales en China,» [En línea]. Available: http://spanish.peopledaily.com.cn/31614/8063671.html. [Último acceso: 15 febrero 2020].

[38] Z. Aldama, «WeChat, la app que espía y censura a más de 1.000 millones de personas,» [En línea]. Available: https://retina.elpais.com/retina/2018/08/01/innovacion/1533122021_765780.html. [Último acceso: 3 febrero 2020].

[39] «Stashcat,» [En línea]. Available: https://stashcat.com/. [Último acceso: 1 febrero 2020].

[40] Wikipedia, «Staschat,» [En línea]. Available: https://de.wikipedia.org/wiki/Stashcat. [Último acceso: 4 febrero 2020].

[41] «Wickr,» [En línea]. Available: https://wickr.com/. [Último acceso: 17 febrero 2020].

[42] «Wickr,» [En línea]. Available: https://en.wikipedia.org/wiki/Wickr. [Último acceso: 5 febrero 2020].

[43] Protege, «Guía rápida Wickr,» [En línea]. Available: https://protege.la/wickr-o-wickr-me-guia-rapida/. [Último acceso: 10 febrero 2020].

[44] Signal, «Interfaz de Signal,» [En línea]. Available: https://signal.org/es/. [Último acceso: 12 febero 2020].

[45] Wikipedia, «SMS,» [En línea]. Available: https://es.wikipedia.org/wiki/Servicio_de_mensajes_cortos. [Último acceso: 5 febrero 2020].

[46] Wikipedia, «MMS,» [En línea]. Available: https://es.wikipedia.org/wiki/Servicio_de_mensajer%C3%ADa_multimedia. [Último acceso: 5 febrero 2020].

[47] Xataka, «Todo sobre Signal,» [En línea]. Available: https://www.xatakandroid.com/aplicaciones-android/todo-sobre-signal-por-que-la-app-de-mensajeria-mas-segura-no-ha-evitado-que-se-filtren-los-mensajes-de-puigdemont. [Último acceso: 6 febrero 2020].

[48] Ministerio de Defensa, «UME,» [En línea]. Available: https://www.defensa.gob.es/ume/. [Último acceso: 28 enero 2020].

[49] Opinión Militar, «IM UME,» [En línea]. Available: https://www.opinionmilitar.es/la-ume-contara-con-un-sistema-propio-de-mensajeria-instantanea-para-sus-comunicaciones/. [Último acceso: 5 febrero 2020].

[50] El Confidencial, «UME IM,» [En línea]. Available: https://www.elconfidencialdigital.com/articulo/defensa/ume-contara-sistema-propio-mensajeria-instantanea-comunicaciones/20190718182803128549.html. [Último acceso: 5 febrero 2020].

[51] «Java,» [En línea]. Available: https://www.java.com/es/. [Último acceso: 15 febrero 2020].

[52] «Qué es Java,» [En línea]. Available: https://www.java.com/es/download/faq/whatis_java.xml. [Último acceso: 10 febrero 2020].

[53] Wikipedia, «Historia Java,» [En línea]. Available: https://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n). [Último acceso: 10 febrero 2020].

[54] J. Fernández Guaza y G. F. Norberto, *Desarrollo de un juego para móviles Android que facilite el aprendizaje de las banderas de señales,* Marín, Pontevedra: CUD-ENM, 2019.

[55] Oracle, «Oracle Java Platform Integrator,» [En línea]. Available: https://www.oracle.com/technetwork/java/embedded/overview/ojpi-1972433.html. [Último acceso: 14 febrero 2020].

[56] El Correo, «Versión 11 de Android,» [En línea]. Available: https://www.elcorreo.com/tecnologia/moviles/llegara-android-movil-20200226125418-nt.html?ref=https:%2F%2Fwww.google.com%2F. [Último acceso: 29 febrero 2020].

[57] Wikipedia, «Open Handset Alliance,» [En línea]. Available: https://es.wikipedia.org/wiki/Android. [Último acceso: 8 febrero 2020].

[58] Wikipedia, «iOS,» [En línea]. Available: https://es.wikipedia.org/wiki/IOS. [Último acceso: 8 febrero 2020].

[59] B. Jiménez, «Ok Diario,» [En línea]. Available: https://okdiario.com/economia/google-suspende-sus-negocios-huawei-moviles-chinos-no-podran-actualizar-android-4137268. [Último acceso: 10 febrero 2020].

[60] Wikipedia, «Android,» [En línea]. Available: https://es.wikipedia.org/wiki/Android.

[61] «Qué es Java y para qué sirve,» [En línea]. Available: https://www.java.com/es/download/faq/whatis_java.xml. [Último acceso: 8 febrero 2020].

[62] Xataka, «Android es más seguro pero más atacado,» [En línea]. Available: https://www.xatakandroid.com/sistema-operativo/android-es-mas-seguro-que-ios-pero-es-mucho-mas-atacado-segun-symantec. [Último acceso: 8 febrero 2020].

[63] Wikipedia, «NortonLifeLock,» [En línea]. Available: https://es.wikipedia.org/wiki/NortonLifeLock. [Último acceso: 8 febrero 2020].

[64] Xataka, «GCHQ accede a los móviles,» [En línea]. Available: https://www.xataka.com/seguridad/smurf-suite-snowden-destapa-el-software-de-la-nsa-britanica-para-controlar-cualquier-movil. [Último acceso: 1 febrero 2020].

[65] MuySeguridad, «NSA espía cualquier dispositivo móvil,» [En línea]. Available: https://www.muyseguridad.net/2013/09/09/nsa-dispositivo-movil-acceso/. [Último acceso: 1 febrero 2020].

[66] «Android Studio,» [En línea]. Available: https://developer.android.com/studio. [Último acceso: 5 febrero 2020].

[67] Deustoformacion, «Por qué usar Android Studio,» [En línea]. Available: https://www.deustoformacion.com/blog/desarrollo-apps/por-que-empezar-programar-android. [Último acceso: 9 febrero 2020].

[68] Boxcryptor, «AES y RSA,» [En línea]. Available: https://www.boxcryptor.com/es/encryption/. [Último acceso: 17 febrero 2020].

[69] Wikipedia, «Función hash,» [En línea]. Available: https://es.wikipedia.org/wiki/Funci%C3%B3n_hash. [Último acceso: 18 febrero 2020].

[70] Glosarios - Ciberseguridad, «Ataque de repetición,» [En línea]. Available: https://glosarios.servidor-alicante.com/ciberseguridad/ataque-de-repeticion#:~:text=Ataque%20de%20repetici%C3%B3n,note%20que%20ha%20sido%20capturada.. [Último acceso: 1 marzo 2020].

[71] Ionos, «Estándar UTF-8,» [En línea]. Available: https://www.ionos.es/digitalguide/paginas-web/creacion-de-paginas-web/utf-8-codificacion-para-una-comunicacion-digital-global/. [Último acceso: 21 febrero 2020].

[72] L. K. «Amenazas de seguridad móvil dirigidas a dispositivos Android,» [En línea]. Available: https://latam.kaspersky.com/resource-center/threats/mobile. [Último acceso: 23 febrero 2020].

[73] «Guía Wickr,» [En línea]. Available: https://protege.la/wickr-o-wickr-me-guia-rapida/. [Último acceso: 5 febrero 2020].

[74] «NortonLifeLock,» [En línea]. Available: https://es.wikipedia.org/wiki/NortonLifeLock. [Último acceso: 8 febrero 2020].

[75] A. Alcalde, «Programación Android: Intents - Conceptos básicos,» El baúl del programador, [En línea]. Available: https://elbauldelprogramador.com/programacion-android-intents-conceptos/. [Último acceso: 17 febrero 2020].