Centro de Investigación y
de Estudios Avanzados
del Instituto Politécnico Nacional

Unidad Zacatenco

Departamento de Computación

Servicios de seguridad en la comunicación de los dispositivos móviles para el proyecto ELISA

Tesis que presenta

Israel Buitrón Dámaso

para obtener el Grado de

Maestro en Ciencias en Computación

Director de Tesis

Dr. Guillermo Benito Morales Luna

México, D.F. Febrero 2012

RESUMEN

En esta tesis se trata un problema existente en las comunicaciones móviles en cuanto a servicios seguridad así como también la localización geográfica usando módulos GPS o proveedores de red.

Existe un proyecto llamado ELISA, que tiene como objetivo proveer a un usuario servicios de asistencia en casos de emergencia o contigencias, por medio de alertas de pánico.

Las alertas se envían desde el dispositivo móvil a un servidor que las atiende y procesa; contienen información útil para asistir al usuario, como datos personales, la probable causa de la alerta, datos del entorno físico y la localización geográfica.

Partiendo de este proyecto, las transmisiones de los datos se suelen realizar mediante un canal no seguro, lo que implica que pueden ser suceptibles a ataques por alguna entidad maliciosa.

En esta tesis se propone una implementación para la plataforma *Android* que pueda hacer uso de los módulos de localización geográfica tanto de GPS como los proveedores de red. En lo concerniente a la seguridad de la comunicación, explora una forma para que en dicha comunicación se garantice confidencialidad, autenticación, integridad y no-repudio, por medio del uso de certificados digitales.

Otro problema en este tipo de escenarios es la identificación de los usuarios por parte del servidor, para este caso, mediante un diseño formal, se abordan un par de métodos que utilizan identificadores de usuarios y dispositivos, para solventar dicho problema.

ABSTRACT

The main topic of this thesis is to study the issues related to security services and also the geographical positioning using GPS modules and network providers.

Our development consists of the security core of the so called ELISA project. The main goal of ELISA is to provide assistence services on emergency events using panic alerts.

The alerts are sent from mobile devices to a server to be processed and solved. The above mentioned alerts contain useful information in order to provide users assistance, such as users personal data, environment data, possible causes of the alert and users location.

No un-safe communication channels may be used since these channels are liable to be attacked by malicious entities.

Here, an implementation for *Android* platform is proposed which is able to use location modules as those provided by GPS or other network providers. Regarding secure communication issues, this implementation aims to guarantee confidentiality, integrity and non-repudiation through digital certificates.

An important issue regarding the implemented services is the two-sided authentication. The server should authenticate its users and any user should authenticate the server. For our particular case, two methods were developed within a formal design. These methods rely over a multilevel approach, thus providing users and devices identification credentials.

AGRADECIMIENTOS

Quiero agradecer a todas las personas que me apoyaron y contribuyeron a que realizara y concluyera esta tesis y los estudios de maestría.

Quiero agradecer a mi papa Felipe, a mi mamá Susana y a mi hermano Oscar porque a pesar de todos los problema que hemos afrontado tuve su apoyo para realizar el posgrado y para sobrevivir fuera del círculo familiar, también por ayudarme con tres seres que amo Balam, Lalinka y Moka.

Quiero agradecer a la mujer que amo Grisel, porque juntos caminamos hacía un rumbo en el que todo era incierto y muchas veces adverso. Por compartir todas las experiencias que este camino nos dejó, por ayudarme a sobrevivir, por cuidarme, por alentarme a seguir, por todo el tiempo y esfuerzo que invirtió en mi, por todas las lecciones de vida, por compartir la experiencia de ser padres de tres seres hermosos, por compartir simplemente, todo ello se quedó tatuado en nuestras vidas y memorias. Sin ella, este camino no hubiera sido el mismo y quizá tampoco con el mismo desenlace.

Quiero agradecer a Alex, Tania, Lore, Victor y Lupis, porque encontré en ellos otra familia, por el apoyo que nos dieron a Grisel y a mi, por todas la noches de cobijo, comida, agua, celebraciones, fiestas, viajes, etc.

Quiero agradecer a José (*Chch*), Emiliano (*Emily*), Ricardo (*Richar Tito*), Marcos (*Natas*), Rodrigo (*Viñi*), que fueron de ayuda en momentos de mucha carga académica y hasta económica. Con ellos desde la universidad he tenido la oportunidad de compartir muchas experiencias y lo sigo haciendo... aunque los tenga un poco descuidados.

Quiero agradecer al Dr. Guillermo Morales Luna por haberme elegido como su estudiante, por todo el apoyo que me ofreció durante esta etapa académica. También por todas las charlas que siempre dejaron algo agradable y sobre todo sembraban en mi esa curiosidad por conocer más cosas. Gracias al trabajo en conjunto tuve la oportunidad de conocer algunos lugares dentro de mi país y fuera de él, lo cual es muy importante para mi ya que representa conocer otras culturas, personas, etc. y son de las cosas que más me gustan hacer en la vida. En fin muchas gracias doctor.

Quiero agradecer a los Drs. del Cinvestav Cesar Torres Huitzil, Gregorio Toscano Pulido y Amilcar Meneses Viveros por todas las experiencias enriquecedoras y consejos tanto a nivel personal como académico.

Quiero agracer a Carlos Galindo por su apoyo en mi proyecto de tesis, con su ayuda se hizo posible la realización de un mejor trabajo y mejores resultados.

Por último y no menos importante a Sofia Reza (*Sofy*), ¡ahh, lo que diga aqui sale sobrando!, ella sido la persona que no por nada se gana la mención en todas las tesis, ella es casi como nuestra segunda madre, platicamos, nos ayuda, bromeamos, chismeamos, . . . , en fin. Gracias Sofi por todas las platicas en esa sillita negra, por todos los documentos y trámites *urgentes*, por los avisos, por todo, asi de simple gracias. Si hubiera más gente como tu que está comprometida con su trabajo y da ese *extra* en él, nuestro país sería muy diferente.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE CUADROS

ÍNDICE DE CÓDIGOS

ACRÓNIMOS

3DES Triple Data Encryption Algorithm

AAC Advanced Audio Coding

AES Advanced Encryption Standard

AGPS Assisted GPS

AMR Adaptive Multi-Rate

API Application Programming Interface

AVC Advanced Video Coding

CA Autoridad Certificadora

CINVESTAV-IPN Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional

CUDI Corporación Universitaria para el Desarrollo de Internet

DES Data Encryption Standard

DH Diffie-Hellman

DSA Digital Signature Algorithm

DVM Dalvik Virtual Machine

EDGE Enhanced Data rates for GSM Evolution

ELISA Entorno de Localización Inteligente para Servicios Asistidos

EML Encryption Manager Lite

FTP File Transfer Protocol

GIF Graphics Interchange Format

GPG GNU Privacy Guard

GPS Global Positioning System

GSM Global System for Mobile Communications

HMAC Hash-based Message Authentication Code

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol over Secure Socket Layer

ICYTDF Instituto de Ciencia y Tecnología del Distrito Federal

IDEA International Data Encryption Algorithm

IETF Internet Engineering Task Force

IMEI International Mobile Equipment Identity

IP Internet Protocol

ISO International Organization for Standardization

ITESM Instituto Tecnológico y de Estudios Superiores de Monterrey

JPEG Joint Photographic Experts Group

JKS Java KeyStore

JSSE Java Secure Socket Extension

MAC Message authentication code

MD5 Message-Digest Algorithm

MIME Multipurpose Internet Mail Extensions

MITM Man in the middle

MP3 MPEG-2 Audio Layer III

NNTP Network News Transfer Protocol

OPENGL Open Graphics Library

OPGPM OpenPGP Manager

OSI Open Systems Interconnection

PGP Pretty Good Privacy

PKI Public Key Infraestructure

PNG Portable Network Graphics

RA Autoridad de Registro

RFC Request for Comment

RSA Rivest, Shamir and Adleman

SC SpongyCastle

SHA Secure Hash Algorithm

SIM Subscriber Identity Module

SMTP Simple Mail Transfer Protocol

SMS Short Message Service

SSH Secure Shell

SSL Secure Sockets Layer

TCP Transmission Control Protocol

TCP/IP Transmission Control Protocol and Internet Protocol

TLS Transport Layer Security

URI Uniform Resource Identifier

URL Uniform Resource Location

URN Uniform Resource Names

VPN Virtual Private Network

W3C World Wide Web Consortium

1 INTRODUCCIÓN

1.1 DESCRIPCIÓN DEL PROYECTO

1.1.1 *Antecedentes y motivación para el proyecto*

Algunos países de Latinoamérica y en especial México, viven un ambiente de crisis en el ámbito de seguridad y crimen organizado. La situación vivida día a día, lleva a un grupo de investigación y desarrollo a diseñar una herramienta capaz de ofrecerle al ciudadano localización inmediata.

La Corporación Universitaria para el Desarrollo de Internet (CUDI), en la que participa el Instituto Tecnológico y de Estudios Superiores de Monterrey (ITESM), el Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional (CINVESTAV-IPN) y el Instituto de Ciencia y Tecnología del Distrito Federal (ICyTDF) emprenden el proyecto de nombre ELISA, que tiene como objetivo hacer posible la herramienta antes mencionada.

ELISA es un sistema basado en dispositivos móviles y una red de microsensores que generan una señal de alarma en caso de riesgo inminente. La señal de alarma incluye la ubicación física del usuario, así como la causa que generó dicha señal. Una vez recibida por el servidor, se le da seguimiento y se consulta una base de datos para obtener información relacionada con el propietario del dispositivo que emitió la señal, como pueden ser números de contacto (de los familiares, la unidad médica, seguridad pública), mapas de ubicación, dirección postal, etc. Esta información es utilizada por el servidor para el reenvío subsecuente hacia los contactos obtenidos de la consulta, con el objetivo de notificarlos del evento.

1.1.2 *Conceptos relacionados con la seguridad*

1.1.2.1 *Autenticación*

La *autenticación* puede entenderse como el acto que confirma la personalidad de un elemento, físico o digital. Este concepto puede ser aplicado a objetos o personas. En el enfoque de la seguridad en las redes, es el proceso mediante el cual se decide si una entidad es quien dice ser.

La autenticación se puede realizar mediante tres métodos, algo que:

- se *conoce*, por ejemplo una clave o contraseña,
- se *tiene*, por ejemplo una tarjeta inteligente o dispositivo token,

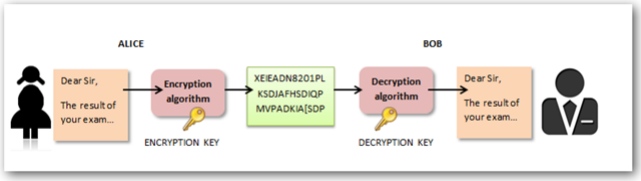


Figura 1.1.: Esquema básico de confidencialidad por medio de criptografía simétrica

- se *es*, por ejemplo patrones biométricos.

1.1.2.2 *Confidencialidad*

Este servicio de seguridad garantiza que la información transmitida entre dos partes no sea accesible para un tercero.

Para garantizar este servicio se hace uso de mecanismos de cifrado, donde tanto el emisor como el destinatario de la comunicación conocen un *secreto compartido* y un tercero no lo conoce. Con dicho secreto, el emisor y el destinatario pueden cifrar y descifrar los mensajes que envían.

Como se puede ver en la figura 1.1, si un tercero intenta acceder al mensaje, se encontrará con el mensaje cifrado y a menos de que consiga el secreto compartido, no podrá acceder al mensaje.

1.1.2.3 *Integridad*

La *integridad* es un servicio de seguridad que tiene por objetivo garantizar que la información enviada no se altere en una transmisión, es decir, garantizar que el mensaje que un emisor envía a un destinatario sea el mismo mensaje que el destinatario reciba.

Esto tiene como consecuencia, la protección del mensaje contra posibles alteraciones durante la transmisión del mismo y en caso de que haya sido alterado, se pueda determinar que dicha alteración fue realizada.

Para este servicio, con frecuencia se hace uso de las funciones *picadillo* (*hash*, en inglés), las cuales toman como entrada un mensaje de longitud variable y devuelven como salida un mensaje de longitud fija[1] conocido como *resumen* (*digest*, en inglés).

Una función picadillo básicamente debe cumplir con las siguientes propiedades:

- Debe ser una función *unidireccional*. Es decir, dado un mensaje m, es fácil computar su resumen $H(m)$, pero no viceversa.

1 La longitud del resumen es constante en la función pero varía de función a función.

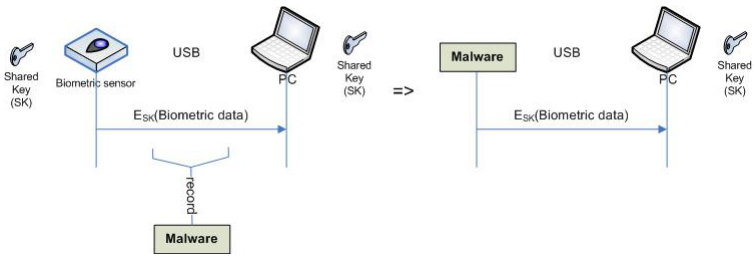


Figura 1.2.: Esquema básico de una ataque de retransmisión en una autenticación

- No debe generar colisiones. Es decir, dado dos mensajes distintos m_1 y m_2, los resúmenes no deben ser iguales: $H(m_1) \neq H(m_2)$[2].

En consecuencia, las funciones picadillo, pueden ser vistas como las generadoras de *huellas digitales*, es decir, si un mensaje m es enviado y se recibe un mensaje m', sólo si el mensaje es *íntegro*, los resúmenes serán iguales:

$$H(m) = H(m') \Rightarrow m = m'.$$

1.1.2.4 *Ataque de Retransmisión*

El ataque de retransmisión (*replay*, en inglés) es una forma de ataque de red, en el cual una transmisión de datos válida es repetida o retardada.

El ataque pretende capturar información y posteriormente reenviarla con el objetivo de falsificar la identidad de uno de los integrantes en la comunicación[2].

Un caso ejemplo puede ser un proceso de autenticación similar a la figura 1.2, donde se capturan los datos para dicha autenticación y en otro momento son reenviados con el objetivo de autenticarse de manera fraudulenta.

1.1.2.5 *Modelo OSI e IP*

El modelo *Open Systems Interconnection (OSI)* es un producto de la *International Organization for Standardization (ISO)*, para dividir los sistemas de comunicación en capas. Una capa es una colección de funciones conceptualmente similares que proveen un servicio a la capa superior y que recibe servicios de la capa inferior. Este modelo está compuesto por 7 capas: física, enlace de datos, red, transporte, sesión, presentación y aplicación, yendo desde la abstracción de nivel más bajo al más alto.

Al igual que el modelo OSI, existe otro modelo para representar los sistemas de comunicaciones mediante capas; este es el modelo *Transmission Control Protocol and Internet Protocol (TCP/IP)*

2 Esta característica resulta imposible en la realidad que se realiza una proyección del espacio de mensajes (infinito) al espacio de los resúmenes (finito) de palabras de longitud k, es decir, un conjunto de 2^k palabras. Por esta razón las colisiones existen, pero deben resultar computacionalmente difíciles de enconrtar.

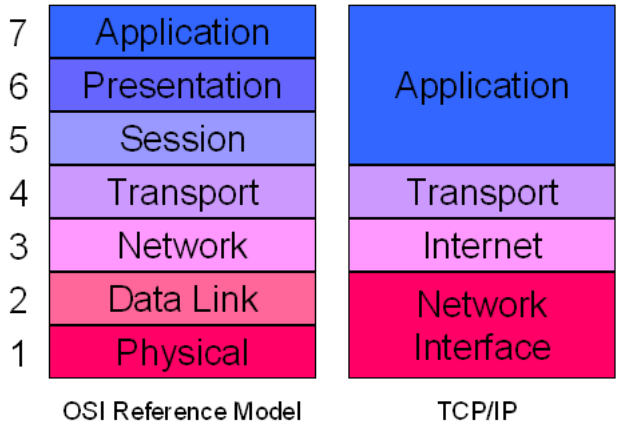


Figura 1.3.: Relación entre el modelo OSI e IP

o *Internet Protocol (IP)*, la relación de capas entre los dos modelos se puede ver en la figura 1.3.

El modelo TCP/IP, es llamado así por dos de los protocolos más importantes en él: *Transmission Control Protocol (TCP)* e IP.

En el modelo IP los protocolos son deliberadamente no rígidos en el diseño como lo son en el modelo OSI. El modelo IP reconoce cuatro capas de funcionalidad desde la perspectiva operativa de los protocolos que contiene: *enlace, Internet, transporte* y *aplicación*.

1.1.2.6 *Protocolo* Transport Layer Security (TLS)

Un protocolo que tiene como objetivo proveer servicios de seguridad en las comunicaciones es TLS. Este protocolo se encuentra ubicado en la capa de presentación y de aplicación en el modelo OSI e IP respectivamente, y es usado por otros protocolos de la misma capa o de capas superiores para establecer comunicaciones seguras.

TLS se encuentra especificado en su última actualización en el documento RFC-5246[3], el cual se basa en especificaciones del protocolo SSL desarrollado por Netscape.

TLS permite a las aplicaciones *cliente-servidor* comunicarse a través de la red en una forma que prevenga la manipulación y el espionaje de los datos transportados. Además proveé autenticación de los nodos extremos y confidencialidad en las comunicaciones a través de Internet usando criptografía.

1.1.2.7 *Protocolo* Hypertext Transfer Protocol over Secure Socket Layer (HTTPS)

El protocolo HTTP, especificado en el documento RFC-2616 [4], fue diseñado para ser usado como protocolo de comunicación en claro. Dado el incremento de su uso en aplicaciones con información crítica o sensible, los mecanismos de seguridad fueron requeridos. Por otra parte SSL y su sucesor TLS fueron diseñados para proveer servicios de seguridad en comunicaciones orientadas a un canal.

Conceptualmente el protocolo HTTPS, descrito en el documento RFC-2818[5], resulta muy simple: *usar HTTP sobre TLS*, de la misma forma en que se usa HTTP sobre TCP. De manera muy general como consecuencia, las comunicaciones HTTP pueden gozar de servicios de seguridad como la autenticación, la confidencialidad, la integridad, etc. de los datos transportados.

1.1.3 *Plataforma de desarrollo*

1.1.3.1 *Android*

Android es un sistema operativo desarrollado en sus inicios por Android Inc. posteriormente esta firma fue comprada por Google en 2005[6]. El anuncio del sistema operativo Android se realiza el 5 de noviembre de 2007, junto con la creación de un consorcio de compañías de hardware, software y telecomunicaciones, dedicadas al desarrollo de estándares abiertos para dispositivos móviles.

Android está basado en una versión modificada del kernel de Linux, sus componentes principales son:

- Aplicaciones: en esta capa se encuentran todas las aplicaciones en general, a las que el usuario tiene acceso y hace uso, como pueden ser: el calendario, los mapas, el navegador, etc.
- *Framework* de aplicaciones: contiene las bibliotecas para creación y administración de las aplicaciones de la capa superior, por ejemplo, la administración de actividades, ventanas, notificaciones, ubicaciones, etc.
- Bibliotecas: contiene bibliotecas desarrolladas en C/C++, para gráficos, 3D, SQLite, medios, etc, y quedan expuestas a los desarrolladores mediante la capa superior.
- *Runtime* de Android: son bibliotecas de base que proporcionan la mayor parte de las funciones disponibles en las bibliotecas de base del lenguaje Java. Cada aplicación Android ejecutará su propio proceso, con su propia instancia de la máquina virtual Dalvik. Dalvik ha sido escrito de forma que un dispositivo puede correr múltiples máquinas virtuales de forma eficiente.
- Núcleo de Linux: Android depende de Linux para los servicios de base del sistema como seguridad, gestión de memoria, gestión de procesos, pila de red, y modelo de controladores. El núcleo también actúa como una capa de abstracción entre el hardware y el resto de la pila de software.

1.2 METODOLOGÍA DEL DESARROLLO DE LA TESIS

Los puntos más importantes que hemos desarrollado son:

1. Un estudio de los protocolos relacionados con la seguridad. El interés está en la protección de los datos que se envían desde un dispositivo móvil, es decir, proveer autenticación, confidencialidad e integridad en el canal de comunicación.

2. Una investigación enfocada a los trabajos previos de programación de los servicios de seguridad en la plataforma Android y iPhone.

3. Un análisis de los procesos de comunicación en los que los dispositivos móviles están involucados dentro del protocolo ELISA, debido a la necesidad de identificar las vulnerabilidades del protocolo a nivel de procesos, así como la forma en que las comunicaciones se establecen.

4. Un estudio de protocolos TLS y HTTPS para su integración en las comunicaciones con dispositivos móviles.

5. Una implementación *prototipo*, para Android.

1.3 ORGANIZACIÓN DE LA TESIS

La presente tesis de maestría está estructurada en seis capítulos. Después de haber presentado el contexto de investigación y de haber planteado el problema que se pretende resolver, exponemos los antecedentes de manera más general en el capítulo 2. Aquí abordamos los conceptos teóricos referentes a los servicios de seguridad, protocolos y la plataforma de estudio, también presentamos un estado del arte en donde se presentan proyectos comerciales sobre alertas de emergencia y proyectos sobre temas de seguridad y criptografía en la plataforma de estudio

Posteriormente en el capítulo 3, se expone un diseño propuesto para la identificación de dispositivos móviles. En este capítulo se planea el problema de identificación al cual se enfrenta algún *servicio* que desea identificar a sus *clientes*, dichos identificadores pueden ser del propio dispositivo o no, de modo que el concepto de *usuario* resulta no trivial definirlo en una implementación dada.

Avanzando en la tesis presentamos en el capítulo 4 los detalles de la implementación del prototipo de solución propuesto. Abordamos los temas referentes a la implementación de la solución de la localización geográfica y la implementación del canal seguro haciendo uso del protocolo HTTPS y el uso de certificados digitales.

Luego de revisar los aspectos de implementación, en el capítulo 5 presentamos las pruebas realizadas a la propuesta de solución. Se documentan las pruebas tanto de localización geográfica como las de conexión con el servidor de ELISA. En las pruebas de localización detallamos cuestiones como tiempo de localización o *hotfix* en diferentes ambientes y mediante el uso del módulo Global Positioning System (GPS) y mediante localización de red.

Por último, en el capítulo 6 se exponen las conclusiones, presentamos algunos puntos de interés y un listado de posibles lí neas de investigación para continuar como trabajo futuro.

En el apéndice A se presenta información relacionada con el código fuente resultado de la implementación realizada en el trabajo de tesis y que se trata en el capítulo 4. Debido a que no resulta práctico publicar dicho código fuente en este documento, se proporciona información para obtenerlo en Internet.

2 ANTECEDENTES

2.1 MARCO TEÓRICO

2.1.1 *Servicios de seguridad*

En los sistemas de información, la materia prima de trabajo es la misma información, son entes aislados por sí mismos que por medio de redes de comunicaciones reciben datos, los procesan, obtienen información para su uso, almacenamiento, etc. Los sistemas necesitan una forma de comunicarse con otros para sacar el mayor provecho posible de dicha información. Con esto surge la necesidad de tener medios y protocolos de comunicación que interconecten los sistemas.

Para ello se crearon protocolos de comunicación como el HTTP, y el *File Transfer Protocol (FTP)*, entre otros, los cuales son útiles para la tranferencia de mensajes y archivos entre dos sistemas de cómputo bajo el modelo *cliente-servidor*. Sin embargo, dentro de su diseño no fueron considerados los posibles ataques o vulnerabilidades a los que la información transmitida estaba expuesta.

La información transmitida entre dos entidades se encuentra expuesta a que una tercera entidad tuviera acceso al canal de comunicación e interceptara la información, para modificarla e incluso para iniciar una conversación en nombre de alguna de las dos entidades "legales". A todas estas acciones se les conoce como *ataques*.

En consecuencia, se necesita una forma para evitar este tipo de ataques o, si acaso no fuese posible establecerla, entonces minimizar la probabilidad de éxito de los ataques.

Los servicios de seguridad comprenden las reglas, los métodos y los protocolos establecidos para proteger la información ante posibles ataques.

En una transmisión se puede contemplar muchos servicios de seguridad básicos: la confidencialidad, la autenticación, el no-repudio, la integridad, la disponibilidad, el anonimato, etc. Sin embargo sólo cuatro ellos son de nuestro interés en el presente trabajo.

2.1.1.1 *Confidencialidad*

Este servicio tiene como objetivo garantizar el acceso a la información sólo a las entidades autorizadas. Los responsables de dicha información deciden quién o quiénes pueden tener acceso a ella. Naturalmente, se trata de evitar que una entidad no autorizada acceda a la información transmitida.

Para garantizar la confidencialidad de un mensaje, se hace uso de mecanismos de cifrado. Cada uno de ellos toma como entrada un mensaje *en claro* y una clave secreta, y lo cifra con lo que obtiene un mensaje *cifrado*, en el cual la información no puede ser accesible.

Dentro de los mecanismos de cifrado de los que hace uso la confidencialidad, también se encuentra el proceso inverso: el *descifrado*, en donde se toma como entrada el mensaje cifrado y la clave secreta, se aplica un proceso inverso al cifrado y se obtiene como salida el mensaje en claro original.

Una parte indispensable en esos procesos es la clave secreta. Esta clave es la que permite que dado un mensaje cifrado pueda ser descifrado y acceder al mensaje en claro.

2.1.1.2 *Integridad*

Dentro de los ataques posibles a la comunicación que dos entidades puedan realizar, se encuentra el hecho en que una entidad no autorizada pueda alterar la información que está siendo transmitida, de modo que alguna de las entidades recibe información no fidedigna.

La *integridad* es un servicio de seguridad que tiene como objetivo garantizar que la información no sea alterada durante una transmisión. Para cumplir este objetivo, se emplea un procedimiento que produzca una partícula relacionada directamente con el mensaje, la cual puede ser vista como una *huella digital*.

La entidad emisora calcula la huella del mensaje y lo envía junto con su huella, posteriormente la entidad receptora toma el mensaje y la supuesta huella, calcula la huella del mensaje recibido y la compara con la huella que recibió. En caso de ser idénticas se puede considerar que el mensaje no fue alterado, es decir, es íntegro.

2.1.1.3 *Autenticación*

La información que se transmite generalmente viaja por un canal y no se tiene la certeza de quién es la entidad que se encuentra en el otro extremo del canal. Por lo tanto es importante tener una forma de asegurarse de la identidad de las entidades con las que se sostiene una comunicación.

Para evitar que un ataque tenga éxito es necesario tener la certeza de la identidad de las partes en la comunicación. Generalmente los ataques del tipo de *Man in the middle (MITM)*, digamos *Persona Enmedio*, son realizados en la ausencia de la autenticación. Una persona enmedio puede permanecer pasivamente, es decir, sólo escuchando la conversación que se sostiene, o actívamente, es decir, haciéndose pasar por alguna de las dos entidades conversadoras, o alterando la información en la conversación.

La idea de autenticación plantea un problema consistente en verificar la identidad digital de una entidad. Generalmente, su solución se fundamenta en tres medios:

- Algo que se *conoce*, por ejemplo una contraseña.
- Algo que se *tiene*, por ejemplo una tarjeta de identidad, un *token*, un certificado digital, etc.
- Algo que se *es*, por ejemplo la verificación de escritura, de las huellas digitales, de los patrones oculares, de los patrones de voz, etc.

2.1.1.4 *No-repudio*

Otro problema en las transferencias de información surge cuando una de las entidades que en algún momento establecieron una comunicación, niega haber enviado un determinado mensaje. Si esta idea se lleva a escenarios como las comunicaciones bancarias, servicios de emergencia, etc. es evidente que esto puede resultar en un ataque a la seguridad de la información.

2.1.2 *Protocolos y estándares de seguridad*

2.1.2.1 *Criptografía simétrica*

La criptografía simétrica es un área de la criptografía que tiene su base en el uso de una sola clave para *cifrar* y *descifrar* un mensaje, generalmente se le llama *clave secreta*. Esto implica que las partes involucradas en la comunicación deben compartir la clave secreta.

Dado que el canal de comunicación se considera inseguro, el proceso de comunicación se realiza de la siguiente manera:

1. El emisor cifra el mensaje usando la clave secreta y posteriormente lo envía.
2. El receptor obtiene el mensaje y lo descrifra usando la clave secreta y tiene acceso al mensaje en claro.

Una de las aportaciones de la criptografía moderna es el principio de que la seguridad de un sistema de cifrado debe recaer en la fortaleza de la clave y no en el algoritmo de cifrado. Es decir la seguridad está en el hecho de que sea *difícil* adivinar, localizar o interceptar la clave.

A medida que transcurre el tiempo, las computadoras tienen una mayor capacidad en sus recursos por lo que realizar el cálculo de todas las claves posibles obliga a que el espacio de claves sea cada vez más grande. Uno de los algoritmos de cifrado más usados tiene un espacio de claves de 2^{56} (aproximadamente $7{,}2 \times 10^{16}$ claves), el cual aparenta ser un número muy grande pero una computadora podría encontrar la clave en un tiempo muy corto, quizá del órden de días u horas.

Los diseños más recientes de cifradores simétricos usan claves de 128 bits (aproximadamente $3{,}4 \times 10^{38}$ claves posibles), bajo este espacio de claves tan grande resulta inviable adivinar la clave correcta.

Los cifradores simétricos existen desde la criptografía clásica con los cifradores de Julio César, de Polibio, de Vigenère, etc. pasando por cifradores como la máquina Enigma, famosa por su uso en la Segunda Guerra Mundial por parte de los alemanes y hasta llegar a los cifradores más recientes como *Data Encryption Standard (DES)* [7], *Triple Data Encryption Algorithm (3DES)* [8], RC5, Blowfish, *International Data Encryption Algorithm (IDEA)* y *Advanced Encryption Standard (AES)* [9].

Uno de los principales problemas de los cifradores simétricos es precisamente el intercambio de las claves, ya que sólo se da por hecho que el intercambio se realiza por un canal seguro. Para una entidad atacante resultaría una mejor opción intentar interceptar la clave secreta en vez de intentar adivinarla teniendo un espacio grande de claves.

Otro problema es el número de claves necesarias para que n entidades se comuniquen. Si cada pareja de entidades necesita una clave para intercambiar mensajes, entonces se necesitan $\frac{n \times (n-1)}{2}$ claves para que cada pareja de entidades pueda comunicarse en privado, lo que coincide con el número de aristas de un grafo completo de n vértices.

2.1.2.2 *Criptografía asimétrica*

En el modelo de criptografía asímétrica se usa un par de claves para realizar el cifrado y descifrado de mensajes, a diferencia de la criptografía simétrica donde se usa una clave.

Las dos claves usadas son llamadas *clave pública* y *clave privada*. Cada entidad es propietaria de una pareja de claves. La clave pública puede ser distribuida libremente mientras que la clave privada sólo debe tenerla su propietario. La clave pública se usa para cifrar mensajes, de modo que cualquier entidad que desee comunicarse puede hacerlo, mientras que solo con la clave privada dicho mensaje puede ser descifrado y solo el propietario podrá tener acceso al mensaje en claro.

La idea base de la criptografía asimétrica es utilizar funciones *trampa* de un solo sentido, es decir, funciones tales que su cómputo sea fácil pero el cómputo de sus inversas resulte difícil y la trampa radica en que sólo si se conoce una de las partes clave del cómputo inverso, éste resulte fácil.

Un ejemplo de dicha idea es la factorización de números primos, donde el cómputo de un número partiendo de su descomposición en números primos resulta fácil. En cambio, resulta difícil obtener la descomposición en números primos de un número dado. Por otro lado si se contase con uno de los factores de la descomposición en primos de un número producto de tan solo dos primos, resulta fácil obtener el otro factor.

Un problema de la criptografía asimétrica es la confianza, ya que es cuestionable que la clave pública realmente pertenezca a la entidad que afirma poseerla, en otras palabras, decidir si una clave pública es *auténtica* o si ha sido alterada o usurpada.

Para los problemas de confianza que surgen de la criptografía asimétrica, existen dos modelos con los que se busca resoverlos. El primero de ellos es un modelo centralizado, como la *Public Key Infraestructure (PKI)*, en la cual se hace uso de terceras entidades a las que se otorga confianza plena y ellas dan fe de la autenticidad de las claves públicas. El segundo es un modelo descentralizado que se le conoce como *maraña de confianza* (*trust web*).

El espacio de claves en criptografía asimétrica es mayor con respecto al de la criptografía simétrica. Mientras que en la criptografía simétrica las claves de 128 bits ofrecen una seguridad suficiente, ya que obliga a comprobar $2^{128} - 1$ claves mediante fuerza bruta lo cual resulta inviable, en la criptografía asimétrica se recomienda al menos de una clave de 1024 bits, ya que la factorización de un número de 1024 bits resulta un problema más fácil que hacer una comprobación por fuerza bruta de un número de dicha longitud.

Se considera que la criptografía asimétrica nació en el año de 1976 con el algoritmo Diffie-Hellman (DH) y desde ese entonces han surgido muchos algoritmos entre los que destacan *Rivest, Shamir and Adleman (RSA)* (así llamado por los nombres de sus autores), *Digital Signature Algorithm (DSA)*, *ElGamal* y la *criptografía de curvas elípticas*. A partir de éstos, existen protocolos que hacen uso de estos algoritmos, tales como DDS, *OpenPGP/Pretty Good Privacy (PGP)*, *GNU Privacy Guard (GPG)*, *Secure Shell (SSH)* y SSL/TLS.

2.1.2.3 *Infraestructura de clave pública*

Una PKI es una combinación de componentes como son hardware, software, políticas de seguridad y entidades que permiten que se realicen operaciones criptográficas con garantías de seguridad. Estas pueden ser el cifrado o el descifrado, la firma digital o el no-repudio de transacciones.

De manera muy general una operación criptográfica que haga uso de una PKI tiene involucradas al menos tres partes:

- una entidad inicia una operación
- otras entidades dan fe de la realización de la operación y garantizan su validez
- una entidad destinataria recibe los datos de la entidad emisora, y de los cuales tiene confianza gracias a que las entidades en las que mutuamente confían han dado fe

El uso de PKI está generalmente relacionado con las siguientes operaciones:

- Autenticación
- Cifrado y descifrado de datos
- Firma de documentos

- Confianza de las comunicaciones
- Garantía de no-repudio de transacciones

Los componentes básicos y más comunes de una PKI son:

- Autoridad Certificadora (CA). Esta entidad es la encargada de emitir y revocar certificados y es la entidad en que las partes tienen confianza y que otorga fe de las operaciones realizadas y la identidad de las entidades que participan en estas.
- Autoridad de Registro (RA). Es la responsable de verificar la identidad de las entidades que se desean realizar su registro y establecer su enlace de identidad, generalmente representado por su clave pública.
- Repositorios. Son entidades que se encargan de todo lo relacionado con el almacenamiento de la información relacionada a la PKI. Los repositorios principales en una PKI son:
  - Lista de Certificados. En los que se encuentran los certificados expedidos y vigentes.
  - Lista de Revocación de Certificados. En esta lista se encuentran los certificados que han sido revocados debido a que su vigencia ha vencido o que su clave privada ha sido comprometida y por lo tanto ya no es de confianza.
- Autoridad de Validación. Esta entidad tiene la responsabilidad de verificar la validez de los certificados digitales.
- Autoridad de Sellado de Tiempo. Esta entidad tiene la responsabilidad de generar un sello de tiempo y firmar documentos con el objetivo de dar fe de su existencia a partir de cierto tiempo.
- Usuarios. Son todas aquellas entidades finales que poseen un par de claves (públicas y privadas) y un certificado que da fe de su clave pública y utilizan un conjunto de aplicaciones para aprovechar la PKI, es decir, para realizar cifrado y firma de documentos, autenticación de otras entidades, etc.

Es importante tener presente algunas consideraciones en el uso de PKI, entre las que destacan:

- Todo certificado válido debe ser emitido por una CA reconocida que de fe de la asociación entre en presunto propietario y el propio certificado.
- El propietario de un certificado es el responsable de la conservación y almacenamiento de la clave privada correspondiente a la clave pública que se certifica.

- Las RAs son las encargadas de verificar la validez y veracidad de los datos que se solicitan para la expedición de un certificado y la administración del ciclo de vida de las peticiones realizadas hacia las CAs.
- El propietario de un certificado válido puede hacer uso del mismo sólo para los usos para los que dicho certificado fue creado y que se estipulan en las politicas de seguridad de la CA.
- Toda operación que realice el propietario de un certificado ha de realizarse de forma presencial por parte del propietario del certificado y con el hardware del cliente.
- Las comunicaciones seguras mediante PKI no requieren que se realice ningún intercambio que implique una clave privada.

2.1.2.4 *Protocolo TLS*

El protocolo TLS permite a las aplicaciones `cliente-servidor` comunicarse a través de una red, previniendo que una entidad externa a la comunicación tenga acceso o pueda alterar los datos que se transmiten sobre el canal de comunicación.

Un cliente y servidor TLS negocian una conexión con estado usando un procedimiento de intercambio de claves conocido como *handshaking*. Durante este procedimiento, el cliente y el servidor acuerdan el uso de parámetros para establecer la seguridad de la conexión.

- El intercambio de claves inicia cuando el cliente se conecta a un servidor con TLS habilitado solicitando una conexión segura y presenta una lista de funciones *hash* y cifradores que pueda usar.
- El servidor seleciona de la lista que el cliente presenta la función *hash* y el cifrador más seguro que pueda usar y notifica al cliente de la decisión.
- El servidor envía su identificación en forma de un certificado digital. El certificado usualmente contiene el nombre del servidor, el certificado de la CA de confianza y la clave pública del servidor.
- El cliente puede contactar al servidor que ha emitido el certificado digital, es decir, la CA de confianza y verificar la validez del mismo antes de proceder.
- Con el fin de generar las claves de sesión a usarse en la conexión segura, el cliente cifra un número aleatorio con la clave pública del servidor y envía la cifra resultante al servidor. Sólo el servidor será capaz de descifrarla con su clave privada.

- Con el número aleatorio elegido, ambas partes generan una clave para el cifrado y descifrado en la conexión segura.

Esto concluye el intercambio de claves e inicia la conexión segura, donde se cifra y se descifra con la clave hasta que la conexión finaliza.

Si alguno de los pasos del intercambio de claves falla, todo el proceso falla y la conexión segura no es creada.

TLS es usualmente implementada sobre cualquiera de los protocolos de la capa de transporte, encapsulando los protocolos de aplicación como HTTP, FTP, *Simple Mail Transfer Protocol (SMTP)*, etc.

Un uso frecuente de TLS es para asegurar el tráfico de HTTP, formando HTTPS. TLS también puede ser usado para establecer un tunel a través de una red y crear una *Virtual Private Network (VPN)*.

TLS cuenta con muchas medidas de seguridad:

- Protección contra un cambio a una versión menor y menos segura del protocolo.
- Uso de un valor *hash* del mensaje mejorado con una clave, por ende sólo los poseedores de esa clave pueden verificar la clave *Message authentication code (MAC)*. La construcción de un *Hash-based Message Authentication Code (HMAC)* usada por la mayoría de los cifradores en TLS están especificados en el documento RFC-2104 [10].
- El mensaje que finaliza el intercambio de claves envía un valor *hash* de todos los mensajes que se transmitieron en el mismo intercambio de claves y visto por ambas partes.
- La función de generación de números seudoaleatorios divide el parámetro de entrada a la mitad y procesa cada mitad por separado usando diferentes funciones *hash* (*Message-Digest Algorithm (MD5)* y SHA-1), después aplica una función `XOR` para obtener la MAC. Esto brinda protección incluso si uno de los algoritmos se encontrase vulnerable.
- SSL versión 3.0 mejora a la versión 2.0 agregando funciones que hacen uso de la función SHA-1 e implementa la autenticación por medio de certificados.

2.1.2.5 *Protocolo HTTP*

HTTP se encuentra ubicado en la séptima capa del modelo OSI, la capa de aplicación. Fue desarrollado en sus inicios por dos organizaciones, *World Wide Web Consortium (W3C)* e *Internet Engineering Task Force (IETF)*, en 1999. Como resultado de esta colaboración se publica un conjunto de documentos *Request for Comments (RFCs)*, entre los que destaca el documento RFC-2616 que contiene la especificación de HTTP versión 1.1.

HTTP es un protocolo ubicado en la capa de aplicación del modelo OSI para sistemas de información multimedio distribuidos, colaborativos.

La primera versión de HTTP conocido como HTTP/0.9 fue un protocolo sencillo para la transferencia de texto a través de Internet. Posteriormente HTTP/1.0 definido en el documento RFC-1945[11], hace mejoras al protocolo permitiendo que los mensajes sean especificados en un formato tipo *Multipurpose Internet Mail Extensions (MIME)*, el cual puede contener metainformación de los datos transferidos y modificadores de la semántica de las peticiones o respuestas, aunque no considera los efectos de *proxies* jerárquicos, el caché de datos, la necesidad de conexiones persistentes o de servidores virtuales.

Los sistemas requieren cada vez de mayor funcionalidad que peticiones sencillas como búsquedas, actualización de vistas y anotaciones. HTTP/1.1, implementa de forma nativa la referencia por medio de identificadores o *Uniform Resource Identifier (URI)* [12], localizaciones o *Uniform Resource Location (URL)* [13] y nombres o *Uniform Resource Names (URN)* [14] para indicar el recurso deseado. Los mensajes son envíados en un formato MIME el cual está definido por el documento RFC-2045[15].

HTTP es un protocolo genérico para comunicación entre usuarios agentes, *proxies* o *gateways* y otros sistemas en Internet incluyendo aquellos que implementan protocolos como SMTP [16], Network News Transfer Protocol (NNTP) [17], FTP [18], Gopher [19] y WAIS [20]. En este sentido HTTP permite acceso multimedio a recursos disponibles de diferentes aplicaciones.

HTTP es un protocolo de petición-respuesta bajo el modelo cliente-servidor. Un cliente envía una petición al servidor en el formato: método de petición, URI, versión del protocolo, seguido por un mensaje en formato MIME que contiene los modificadores de la petición, información del cliente y opcionalmente información sobre la conexión con el servidor. El servidor responde con una línea de estatus, incluyendo la versión del protocolo y un código de éxito o error, seguido por un mensaje en formato MIME que contiene información del servidor, etc.

La mayoria de las comunicaciones HTTP son iniciadas por el agente usuario y consiste en una petición para ser aplicada a un recurso que reside en un servidor.

HTTP define ocho métodos de comunicación, en cada uno indica la acción a realizar durante la transacción sobre un recurso determinado. Los ocho métodos son:

- `OPTIONS`: Este método representa una petición de la información de las opciones disponibles en la comunicación en la cadena de *petición-respuesta*. Este método permite al cliente determinar las opciones y/o requisitos asociados al recurso o las capacidades del servidor, sin implicar el inicio de una petición de recursos.

- `GET`: Este método sirve para obtener cualquier información identificada por una petición de URI. Si la URI hace referencia a un proceso que procese datos, se contestará al cliente con el resultado del proceso y no del código fuente del

proceso. En caso contrario se regresará el texto como la respuesta de la petición.

- `HEAD`: Este método es idéntico al método `GET` excepto que el servidor no regresará al cliente el cuerpo del mensaje, sólo se regresará su encabezado. Este método puede ser usado para obtener metainformación del recurso referenciado por la petición sin transferir el recurso mismo.
- `POST`: Este método es usado para solicitar al servidor que acepte un recurso en la petición. `POST` está diseñado para permitir una forma uniforme para satisfacer las siguientes funciones:
  - Anotación de los recursos existentes.
  - Publicación de mensajes.
  - Proporcionar un bloque de datos como resultado de un formulario a un proceso de manupulación de datos.
  - Medio para alimentación de bases de datos.
- `PUT`: El método `PUT` solicita al servidor que la entidad adjunta a la petición sea almacenada en el recurso asociado en la URI. Si la URI hace referencia a un recurso existente la entidad adjunta debería ser considerada como una versión modificada de la existente en el servidor. Si la URI no hace referencia a un recurso existente y es posible que el URI genere un nuevo recurso en el servidor, entonces debe ser considerado como un recurso nuevo.
- `DELETE`: El método `DELETE` solicita que el servidor elimine el recurso referido por el URI. Este método puede ser reimplementado con alguna función diferente. El cliente no puede tener garantía que la operación se ha llevado acabo o el código retornado por la operación.
- `TRACE`: El método `TRACE` es usado para invocar un mensaje de traza en la capa de aplicación.
- `CONNECT`: Este método está reservado para ser usado con un proxy que podría cambiar dinámicamente un tunel de conexión, por ejemplo como SSL.

2.1.3 *Plataformas de dispositivos móviles*

A continuación mencionamos algunas características de la plataforma de interés.

2.1.3.1 *Android*

Android es una pila de software para dispositivos móviles que incluye un sistema operativo, un *middleware* y aplicaciones [21].

Las características principales son:

- *Framework* de aplicaciones: Permite la reutilización y reemplazo de componentes.
- Máquina virtual *Dalvik*: Optimizada para dispositivos móviles.
- Navegador integrado: Basado en un motor Webkit *open-source*.
- Gráficos optimizados: Contiene bibliotecas de gráficos 2D, gráficos 3D basados en la especificación *Open Graphics Library (OpenGL)* 1.0.
- SQLite: Para almacenamiento estructurado de datos.
- Soporte de formatos multimedia. Incorpora bibliotecas nativas para los formatos de audio, video e imágenes más comunes: MPEG4, H.264 ó *Advanced Video Coding (AVC)*, *MPEG-2 Audio Layer III (MP3)*, *Advanced Audio Coding (AAC)*, *Adaptive Multi-Rate (AMR)*, *Joint Photographic Experts Group (JPEG)*, *Portable Network Graphics (PNG)*, *Graphics Interchange Format (GIF)*, etc.
- Telefonía *Global System for Mobile Communications (GSM)*.
- Bluetooth, *Enhanced Data rates for GSM Evolution (EDGE)*, 3G y WiFi.
- Cámara, GPS, brújula y acelerómetro. La plataforma integra nativamente bibliotecas para el manejo de estos dispositivos aunque son dependientes de cada dispositivo.
- Ambiente integrado de desarrollo. El conjunto de herramientas para desarrollar aplicaciones para la plataforma incluye un emulador de dispositivos, herramientas de depuración, especificación de perfiles para memoria y desempeño y un *plugin* para el desarrollo usando Eclipse.

2.1.3.2 *Arquitectura de Android*

En la figura 2.1 se muestra de manera general los componentes del sistema operativo *Android*, el cual está formado por 5 componentes principales:

- *Applications*
  Android por omisión ya cuenta con un conjunto de aplicaciones como un cliente de correo electrónico, un gestor de mensajería Short Message Service (SMS), un calendario, mapas, un navegador, organizador de contactos, etc. Todas las aplicaciones son escritas con el lenguaje Java.
- *Application Framework*
  Android provee una plataforma abierta y ofrece a los desarrolladores la capacidad de crear aplicaciones muy útiles e inovadoras. Cualquier desarrollador tiene acceso a la misma *Application Programming Interface (API)* usada por las

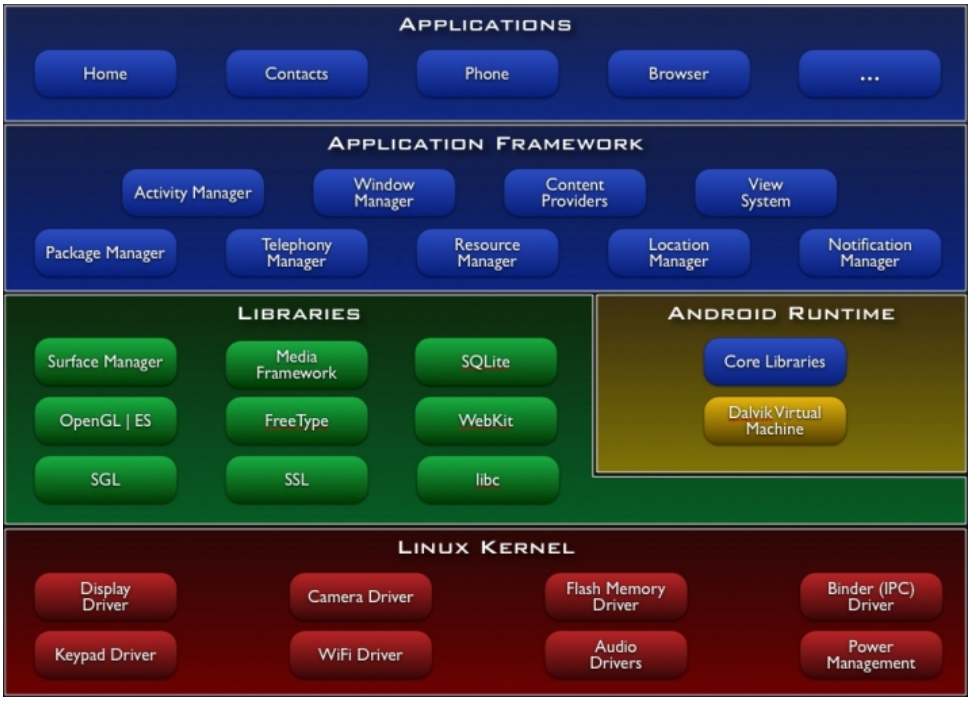


Figura 2.1.: Arquitectura del sistema operativo Android

aplicaciones base. Esta capa está diseñada para simplificar la reutilización de componentes, cualquier aplicación puede hacer públicas sus capacidades y cualquier otra puede hacer uso de ellas (regidas bajo ciertas medidas de seguridad).

Detrás de todas las aplicaciones está un conjunto de servicios y sistemas que incluye:

- Un conjunto de componentes para *vistas* que pueden ser usados para construir aplicaciones que incluyan listas, tablas, campos de texto, botones e incluso un navegador incrustado.
- *Proveedores de contenido* que permiten que las aplicaciones tengan acceso a datos de otras aplicaciones o compartan sus mismos datos.
- *Administrador de recursos* proveen acceso a recursos como cadenas de internacionalización, gráficos y *layouts* (administradores de contenido).
- *Administrador de notificaciones* permiten que cualquier aplicación pueda desplegar alertas en la barra de estado.
- *Administrador de actividades* controla el ciclo de vida de las aplicaciones.

- Bibliotecas

  Android incluye un conjunto de bibliotecas escritas en lenguaje C/C+ las cuales son usadas por los componentes del sistema y son accesibles a través de la capa *Application Framework*. Algunas de las bibliotecas de base son:

- Bibliotecas C del sistema
- Bibliotecas de multimedia
- Administración de la superficie
- Biblioteca web
- Gráficos 2D (SGL)
- Graficos 3D
- FreeType
- SQLite

- *Android Runtime*

  Cada aplicación ejecuta su propio proceso y en una instancia aislada de una *Dalvik Virtual Machine (DVM)*. Dalvik ha sido contruido de manera que cada dispositivo puede ejecutar muchas máquinas virtuales de manera eficiente. DVM ejecuta archivos con el formato de ejecutables Dalvik (`.dex`) que son optimizados para un consumo mínimo de memoria.

- Kernel de Linux

  Android está basado en el kernel de Linux (específicamente la versión 2.6) para todos los servicios base del sistema como la seguridad, la administración de memoria, la administración de procesos, pila de redes, modelos de controladores, etc.

2.2 ESTADO DEL ARTE

2.2.1 *Proyectos similares a ELISA*

Cada día vemos nacer un mayor número de soluciones de servicio de localización; sin embargo, pocas son las que cuentan con un enfoque parecido al propuesto en el proyecto ELISA. A continuación, se describe sólo dos de ellas.

- Nextel iAlarm: Nextel presenta su servicio iAlarm, con el que los usuarios con un equipo i776 Alumina [22, 23] cuentan. El costo del servicio es de $50 pesos por emergencia más el envío de mensajes SMSs, ya que sólo es para usuarios Nextel. A estos costos se le incluye la renta mensual. Se puede seleccionar hasta un máximo de 9 contactos de emergencia, a los cuales se les envía hasta un máximo de 48 mensajes. Sólo está disponible en México con cobertura de Nextel.

- Importta: Se requiere de la compra de un equipo adicional y aunque no se paga renta, se cobra por el envío de SMSs hasta a un máximo de 10 contactos de emergencia. Tiene un enfoque de servicio a los vehículos ya que permite, mediante una instalación, avisar del exceso de velocidad,

arranque del vehículo y en dado caso cortar la corriente eléctrica. Tiene un servicio de micrófono, mediante una llamada telefónica al equipo. Se activa mediante un botón de emergencia de parte del usuario, o un SMS remoto. A continuación, se presenta una tabla comparativa de estas dos soluciones, con fines de apreciar el valor agregado de ELISA.

Los servicios aquí mencionados son comerciales y en sus respectivas fuentes, a la fecha del presente documento, no presentan datos técnicos especificando cuestiones de seguridad en la comunicación.

En cuanto a los proyectos académicos, se encuentra:

- ELISA-TEC: Este proyecto forma parte del mismo proyecto ELISA, sin embargo es desarrollado por otro equipo de trabajo en el ITESM. En este proyecto el foco de interés está en realizar una implementación sobre la plataforma BlackBerry, también incluye el desarrollo de una aplicación web para la administración de los datos en el servidor ELISA, asćomo los servicios que este debe prestar [24].

2.2.2 *Proyectos de seguridad y criptografía*

2.2.2.1 *MemorizingTrustManager*

MemorizingTrustManager es un proyecto que pretende permitir un uso mayor y más seguro de SSL en la plataforma Android [25].

Si un certificado digital desconocido es encontrado por la aplicación, ésta pide al usuario la confirmación para aceptar dicho certificado o de lo contrario se rechaza (ver las figuras 2.2a y 2.2b). De esta manera se pretende prevenir ataques MITM y ayuda a filtar certificados expirados, autofirmados o de dudosa procedencia.

2.2.2.2 *Encryption Manager Lite*

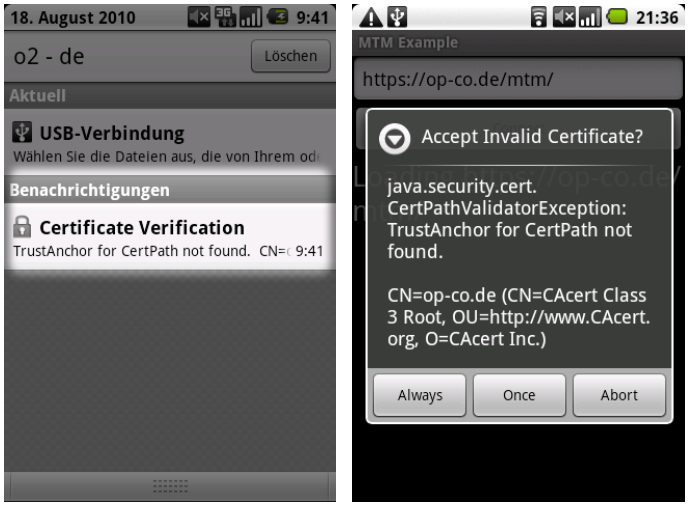
Encryption Manager Lite es un proyecto que provee un administrador de archivos con la posibilidad de usar el cifrador AES para cifrar/descifrar los archivos. De esta manera es como proporciona una forma *segura* para mantener la confidencialidad de los archivos.

Mediante una clave maestra se accede a la aplicación y se cifran las claves que son generadas aleatoriamente.

Entre sus características está el uso de AES o Twofish en sus versiones de 128 y 256 bits de seguridad.

2.2.2.3 *SpongyCastle*

SpongyCastle es un proyecto que reempaqueta el conjunto de herramientas *BouncyCastle* para Android.



(a) Búsqueda de certificados (b) Confirmación de confianza

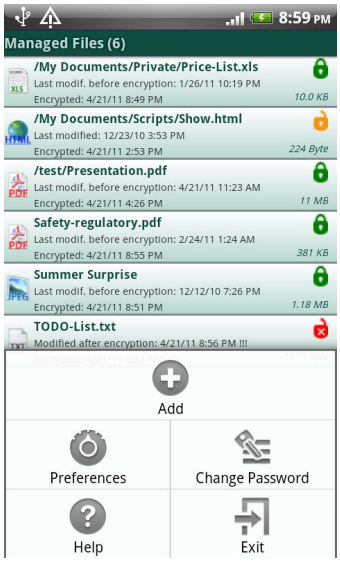
Figura 2.2.: Pantallas de MemorizingTrustManager

Tiene por objetivo ser un reemplazo para las versiones cortas de las bibliotecas *BouncyCastle* que vienen nativas con la plataforma Android.

2.2.2.4 *OpenPGP Manager*

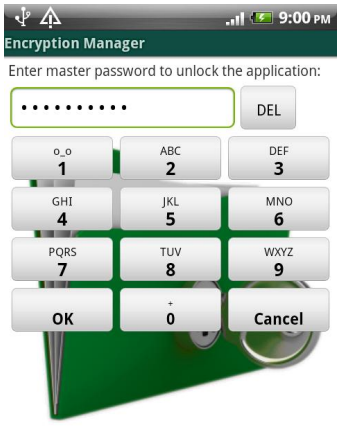
OpenPGP Manager es un proyecto que ofrece un cliente PGP para la plataforma Android. Entre sus capacidades incluye:

- Administración de claves (creación, importación, exportación y publicación).
- Cifrado, descifrado, firma y verificación de correo.
- Cifrado, descifrado, firma y verificación de archivos.
- Subclaves.

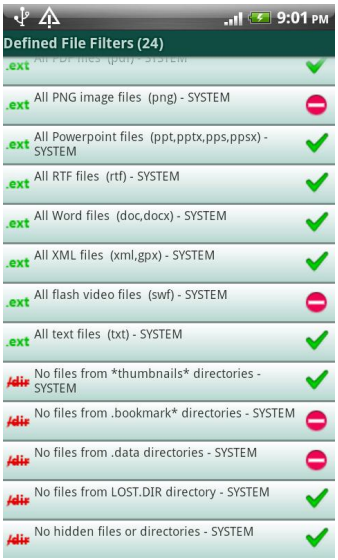


(a) Administrador de archivos de EML

(b) Detalle de archivos en EML

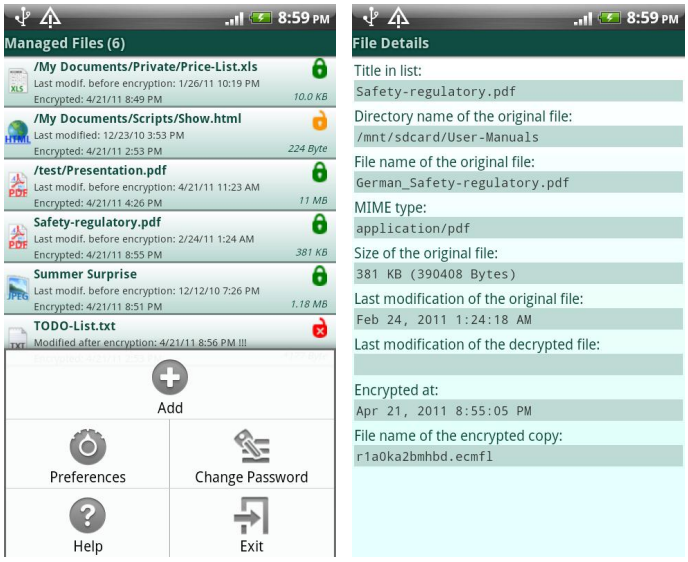


(c) Autenticación en EML



(d) Opciones del filtros en EML

Figura 2.3.: Pantallas de Encryption Manager Lite



(a) Administrador de claves de OPGPM

(b) Envío de correo en OPGPM

Figura 2.4.: Pantallas de OpenPGP Manager

3

DISEÑO FORMAL DE IDENTIFICADORES

3.1 PROBLEMÁTICA EN IDENTIFICACIÓN

Determinar un identificador para servicios que serán consumidos por dispositivos móviles implica cuestionarse: ¿Qué se desea *identificar*?

Dependiendo del sistema donde la identificación se desee implementar, un *usuario* puede ser representado por el dispositivo móvil u otro objeto relacionado al propio usuario.

Para ejemplificar el problema de identificación, se pueden plantear dos escenarios prácticos, en los cuales un dispositivo móvil realiza una solicitud a una computadora que le provee cierto servicio:

1. Un dispositivo envía su identificador adjunto a un requerimiento emitido, estos datos son recibidos en el servidor y es posible identificar al usuario que emitió el requerimiento por medio de dicho identificador ya que existe registrada una relación de pertenencia entre el usuario y el dispositivo en el servidor.

2. Un usuario emite un requerimiento desde un dispositivo diferente al que se encuentra asociado en la relación de pertenencia. Este caso plantea la disyuntiva de decidir si el requerimiento es o no válido.

En estos dos casos surge la necesidad de considerar dos tipos de identificadores: los de usuarios y los de dispositivos.

3.2 DISEÑO FORMAL E IMPLEMENTACIÓN

3.2.1 *Diseño formal*

Sea $\mathcal{U}$ un espacio de *identificadores de usuario*, $\mathcal{D}$ un espacio de *identificadores de dispositivo* y $P(t)$ una relación temporal de *pertenencia*, es decir $P(t) \subset \mathcal{U} \times \mathcal{D}$, que cumple con las siguientes características en todo tiempo t:

- Todo usuario tiene un dispositivo:

$$\forall u \in \mathcal{U} \quad \exists d \in \mathcal{D} : (u, d) \in P(t) \tag{3.1}$$

- Un dispositivo sólo puede ser usado por un usuario:

$$\forall d \in \mathcal{D} \forall u, v \in \mathcal{U} : [(d, u) \in P(t) \& (d, v) \in P(t) \Rightarrow u = v] \tag{3.2}$$

Estas dos propiedades quedan resumidas planteando que $P(t)$ es una función $\mathcal{D} \to \mathcal{U}$ que es suprayectiva.

Consideremos por el momento un conjunto $\mathcal{E}$ de *etiquetas*. Una *función de etiquetado* es del tipo $\phi : \mathcal{U} \times \mathcal{D} \to \mathcal{E}$. Decimos que ϕ *distingue*

- *usuarios* si $\forall (u_0, d_0), (u_1, d_1) \in \mathcal{U} \times \mathcal{D}$:

$$\phi(u_0, d_0) = \phi(u_1, d_1) \Rightarrow u_0 = u_1 \tag{3.3}$$

- *dispositivos* si $\forall (u_0, d_0), (u_1, d_1) \in \mathcal{U} \times \mathcal{D}$:

$$\phi(u_0, d_0) = \phi(u_1, d_1) \Rightarrow d_0 = d_1 \tag{3.4}$$

- *parejas de usuarios-dispositivos* si $\forall (u_0, d_0), (u_1, d_1) \in \mathcal{U} \times \mathcal{D}$:

$$\phi(u_0, d_0) = \phi(u_1, d_1) \Rightarrow u_0 = u_1 \ \& \ d_0 = d_1 \tag{3.5}$$

Sea ahora $B = Bytes$ el alfabeto que coincide propiamente con el ASCII; sea B^* el conjunto de palabras de bytes y $B^+ = B^* - \{nil\}$. Diremos que para $\mathcal{E} = B^+$ una función de etiquetado $\phi : \mathcal{U} \times \mathcal{D} \to B^+$ es una de *identificación*.

Por otra parte, sean $\mathcal{K}_d, \mathcal{K}_e$ dos espacios de *claves privadas* y *claves públicas*, respectivamente. En todo esquema de cifrado de clave pública existe un conjunto $\mathcal{K} \subset \mathcal{K}_e \times \mathcal{K}_d$ consistente de parejas de claves (pública–privada). Para $\mathcal{E} = \mathcal{K}$, una *función de registro* es una de etiquetado $\psi : \mathcal{U} \times \mathcal{D} \to \mathcal{K}$.

La noción de *identificación* puede ser usada como sinónimo de autenticación de una entidad, o alternativamente, también es usada para referirse al proceso de asignación de una identidad, sin que esto conlleve un medio de comprobación de la entidad identificada [26]. Convendremos aquí en llamar *identificación* a esta segunda noción, en tanto que *autenticación* involucra un proceso de verificación de identidad.

Recordamos que un *esquema de cifrado* se construye como sigue: sean $\mathcal{M}$ el espacio de mensajes y $\mathcal{C}$ el espacio de textos cifrados, ambos definidos sobre el alfabeto B. Una función de cifrado E de textos es del tipo $E : \mathcal{M} \times K_e \to \mathcal{C}$, con una correspondiente función de descifrado $D : \mathcal{C} \times K_d \to \mathcal{M}$, tales que $\forall (k_e, k_d) \in \mathcal{K}$ se cumple:

$$\forall m \in \mathcal{M} \ : \ D(E(m, k_e), k_d) = m \tag{3.6}$$

$$\forall c \in \mathcal{C} \ : \ E(D(c, k_d), k_e) = c \tag{3.7}$$

Recordamos también que un *esquema de firma* se construye como sigue: sea $\mathcal{S}$ un espacio de firmas y sea f una función de firmado $\mathcal{M} \times \mathcal{K}_d \to \mathcal{S}$, $(m, k_d) \mapsto s$. Si $s = f(m, k_d) \in \mathcal{S}$ se dice que s es la firma del mensaje $m \in \mathcal{M}$, usando la clave privada k_d. Es convencional que tanto el espacio de mensajes $\mathcal{M}$ como el de firmas $\mathcal{S}$ coincidan con el de palabras en ASCII B^*. En el esquema de firma, también se tiene una función de verificación de firma:

$$v : \mathcal{M} \times \mathcal{S} \times \mathcal{K}_e \to \{0, 1\} \tag{3.8}$$

de manera que $\forall(m, s, k_e) \in \mathcal{M} \times \mathcal{S} \times \mathcal{K}_e$

$$v(m, s, k_e) = 1 \iff s = f(m, k_d) \tag{3.9}$$

donde k_d es la clave privada y k_e su pública correspondiente.

3.2.2 *Políticas de implementación*

3.2.2.1 *Funciones* hash

Una función *hash* es un procedimiento determinístico que toma como parámetro de entrada una cadena de longitud variable o *mensaje* y produce como salida una cadena de longitud constante o *valor hash*, es pues del tipo $h : \{0,1\}^* \mapsto \{0,1\}^n$, donde n es un parámetro fijo de longitud.

Una función *hash* debe cumplir idealmente con las siguientes características:

- Es fácil computar el valor *hash* dado un mensaje:

$$\text{Computable}(m) \Rightarrow \text{Computable}(h(m)) \tag{3.10}$$

- Es inviable computar el mensaje dado su valor *hash* .

$$\text{Computable}(h(m)) \not\Rightarrow \text{Computable}(m) \tag{3.11}$$

- Es inviable modificar un mensaje sin alterar su valor *hash*.

$$m \neq m' \Rightarrow h(m) \neq h(m') \tag{3.12}$$

- Es computacionalmente complejo encontrar dos mensajes con un mismo valor *hash*.

$$h(m_1) = h(m_2) \Leftrightarrow m_1 = m_2 \tag{3.13}$$

  Cuando dos mensajes diferentes que generen un mismo valor *hash*, se conoce como una *colisión* de la función.

Las características anteriores de las funciones *hash* hacen que sean ámpliamente usadas en aplicaciones de seguridad informática, tales como firmas digitales, MAC y otras formas de autenticación.

Se considera *quebrantada* a una función *hash*, cuando se ha encontrado alguna violación a dichas características como el caso de las colisiones.

3.2.2.2 *Identificadores para implementación*

Según se puede observar en la sección 3.2.1, el problema de identificación reside principalmente en definir la función de identificación ϕ a implementar.

La identificación puede ser realizada evaluando o la unión de los dos identificadores o cada uno por separado.

En el caso específico de la plataforma Android, el uso del número de suscriptor como identificador de usuario y el uso

Función	Longitud (bits)
MD5	128
SHA-1	160
SHA-256	256
SHA-512	512

Cuadro 3.1.: Longitud de salida de funciones *hash*

del International Mobile Equipment Identity (IMEI) como identificador de dispositivo, no están restringidos[1], sin embargo existe la posibilidad de que en otras plataformas pueda existir restricción en la transmisión de dichos datos.

3.2.3 *Uso de funciones Hash*

Se propone el uso de funciones *hash* con el objetivo de evitar la transmisión de dichos números pero sin perder la posibilidad de realizar verificación con ellos.

Las funciones *hash* con mayor difusión son MD5 [27, 28] y el grupo de funciones SHA [29].

Se sugiere el uso de ellas con el propósito de verificar estos identificadores, debido a que han sido implementadas nativamente en una gran cantidad de dispositivos y eso facilita la integración con muchos sistemas de cómputo.

La longitud del resultado depende de la función que se aplique, en la tabla 3.1 se puede observar una comparativa de 4 funciones comunes.

En las siguientes dos secciones bosquejamos la implementación de funciones de etiquetado que distinguen dispositivos, usuarios o parejas de dispositivos y usuarios.

3.2.4 *Método AND*

El método AND tiene como objetivo considerar a la unión de los dos identificadores para obtener un resultado y este valor será el objeto de verificación, haciendo uso del diseño especificado en la sección 3.2.1. Para este método la función de etiquetado ϕ ha de distinguir parejas de dispositivos y usuarios, y por lo tanto ha de cumplir con la implicación (3.5).

Se propone aplicar una función *hash* a la suma de los dos identificadores:

$$\text{UID} = \text{Hash}(\text{SusID} + \text{IMEI})$$

La ventaja principal de este método es que se obtiene un identificador (UID) más corto. La desvestaja principal es que la

1 No se encontraron restricciones en políticas y términos de uso de la plataforma al día de redacción de este documento.

ausencia o alteración de cualquiera de los dos identificadores tiene como consecuencia que el UID no verifique.

Aparéntemente eso es un punto favorable, pero en la práctica implica que los requerimientos sólo pueden ser emitidos por el usuario registrado, con su dispositivo registrado, para ser válidas.

A pesar de que en (3.2) del diseño se hace referencia a un modelo ideal, donde un *usuario siempre usa su propio dispositivo*, en la práctica es común que un usuario haga uso de un dispositivo diferente al suyo, por ejemplo, cuando un usuario intenta hacer una llamada usando su tarjeta Subscriber Identity Module (SIM) desde otro dispositivo. Este caso puede ser común y es válido. De modo que se considera hacer una validación de los identificadores por separado.

3.2.5 *Método OR*

Este método se encuentra contemplado en el diseño de la sección 3.2.1, donde la función de identificación ϕ cumple con la implicación (3.3) o con la (3.4) para realizar la verificación de cada identificador, dejando la posibilidad a que sólo un identificador sea válido y delegando la decisión de considerar como válida la alerta a las políticas de operación del proyecto.

Para este caso se propone que el UID se construya mediante la concatenación de los resultados de aplicar las funciones *hash* a los identificadores:

$$\text{UID} = \text{Hash}(\text{SusID})||\text{Hash}(\text{IMEI})$$

El costo de aplicar este método es la longitud del UID (será el doble de la longitud de la salida de la función *hash*), pero a cambio se obtiene la posibilidad de verificar cada identificador por separado.

4 IMPLEMENTACIONES

4.1 DESCRIPCIÓN GENERAL

El trabajo realizado en esta tesis toma como base la arquitectura del proyecto ELISA y se enfoca en la comunicación entre un teléfono celular y el servidor de ELISA que recibe las peticiones.

Se realizó una implementación de una aplicación "cliente" para la plataforma de Android y tiene como objetivo final comunicarse con el servidor mediante un canal seguro como HTTPS y el uso de certificados digitales para el proceso de autenticación en el establecimiento del canal, para enviar hacia dicho servidor toda la información necesaria para la ubicación del usuario y que el protocolo de alertas de ELISA entre en funcionamiento.

4.2 CASO DE USO GENERAL

La aplicación cliente ofrece muchas opciones de configurar para las necesidades específicas al momento de su uso.

Aquí se explica el caso de uso genérico esperado.

1. El teléfono recibe un petición de alerta.
2. El teléfono solicita la ubicación geográfica al módulo GPS o a la red IP o a la red de telefonía celular.
3. Una vez obtenida la ubicación, establece una conexión con el servidor ELISA.
4. Se envía la información relacionada con la alerta y la ubicación.
5. El servidor ELISA recibe la información y cierra la conexión.

De forma general el caso de uso no muestra pasos importantes de la implementación, como el proceso de ubicación geográfica, el protocolo de canal de comunicación, etc.

4.3 PLATAFORMA DE DESARROLLO

En el desarrollo de este trabajo fue necesario un equipo *laptop* para la programación de la aplicación cliente, un teléfono como dispositivo móvil y algunos paquetes de software. Todos ellos se describen a continuación.

El dispositivo usado fue un teléfono *Samsung Galaxy S* (modelo `SAMSUNG-SGH-I897`), el cual tiene las siguientes características:

- Procesador de 1 GHz `ARM Cortex-A8 processor, PowerVR SGX540 GPU`, *Hummingbird chipset*

- Redes de comunicación:
  - GSM (850, 900, 1800, 1900 MHz)
  - HSDPA (850, 1900, 2100 MHz)
- Memoria y almacenamiento:
  - Memoria RAM de 512 MB
  - Memoria flash interna de 16 GB
- Módulo Wi-Fi 802.11 b/g/n; DLNA
- Módulo GPS con soporte de Assisted GPS (AGPS)
- Software:
  - Sistema operativo Android v2.2 (Versión del Kernel `I897UCJI6`)
  - Soporte de mensajería SMS, MMS, *e-mail*, etc.

La plataforma de desarrollo fue un equipo *laptop Dell* (modelo `XPS M1210`) que se describe a continuación:

- Procesador `Intel CPU T2300` a 1.66 GHz
- Memoria RAM de 2.5 GB
- Sistema operativo `Gentoo Linux`

4.4 LOCALIZACIÓN GEOGRÁFICA

Android provee, por medio de su *API*, una forma de consumir servicios de localización geográfica para los dispositivos dotados de esas capacidades en hardware. Para ello existen dos medios de localización: módulos GPS y redes de tipo IP o de tipo celular.

Para la implementación se utilizó la API de localización disponible bajo el paquete `android.location` y la API del servicio de mapas de Google disponible bajo el paquete `com.google.android.maps`

La estructura de la aplicación se muestra en el diagrama de bloques de la figura 4.1 y se compone de tres módulos principales: el *dispositivo*, el *proveedor de mapas* y el *proveedor de localización*.

El primer módulo es el dispositivo y en él operan dos componentes: el administrador de localización (`Location Manager` (LM)) y el componente que despliega los mapas (`MapView` (MV)). La interacción entre estos dos componentes se realiza cuando el LM le envía al MV los datos de localización, en ese momento el MV hace el desplegado del mapa y actualiza el marcador de ubicación.

El LM por su parte se encarga de realizar las peticiones de localización a cualquiera de sus proveedores de localización. Para el caso de esta implementación, la localización se realiza por medio del dispositivo GPS o por medio de la conexión de red IP.

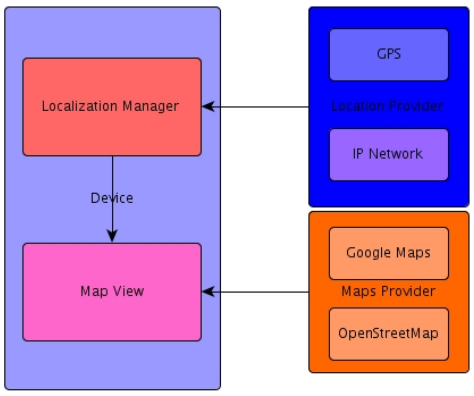


Figura 4.1.: Diagrama de bloques de la aplicación

El segundo módulo es el proveedor de localización y como su nombre lo indica tiene la función de proporcionar todos los detalles de localización posibles.

En caso de usar GPS, se obtiene latitud y longitud, un margen de error en las mediciones y altitud (opcionalmente), y dependiendo de la disponibilidad de los dispositivos complementarios como los acelerómetros, se pueden obtener datos como velocidad de desplazamiento (rapidez y sentido). En caso de usar la red, se obtiene la latitud y longitud a partir de la dirección IP pública a Internet.

Por último el tercer módulo es el proveedor de mapas, el cual proporciona al dispositivo los mapas necesarios para que éste a su vez, los despliegue y ofrezca al usuario de forma gráfica una noción de los datos de localización obtenidos.

Este tercer módulo puede ser considerado en la aplicación como un módulo auxiliar y de bajo acoplamiento, es decir, la aplicación no tiene dependencia del módulo para obtener la localización ya que sólo afecta en el despliegue de los mapas, pero los datos de localización pueden ser usados para otros objetivos (almacenarlos, procesarlos, etc.).

4.5 CANALES HTTPS

HTTPS [5] es la combinación del protocolo HTTP [30] junto con la capa de transporte que provee el protocolo SSL/TLS. La unión de estos protocolos ofrecen un servicio de canal seguro e autenticación de los pares en la comunicación.

Este protocolo garantiza protección en contra de ataques de MITM y espionaje, por medio de cifradores de bloque y certificados digitales los cuales pasan por una verificación y sólo después se les otorga confianza.

La principal característica de HTTPS es la confianza en una tercera entidad: la CA, el cual da fe la de identidad de entidades otorgando certificados que avale dicha fe. Si una entidad es certificada por la autoridad certificadora entonces ésta se vuelve de confianza. El principio básico del protocolo es: *si confío en la autoridad certificadora entonces díme en quién más confío*.

Una conexión se considera *confiable* si y sólo si se cumple:

- El usuario confía en el software que implementa el protocolo HTTPS.
- El usuario confía en las autoridades certificadoras instaladas por omisión [31].
- El servidor envía al cliente un certificado válido y vigente y firmado por una autoridad certificadora de confianza.
- El certificado identifica correctamente al servidor.
- El usuario confía en que el protocolo de cifrado es lo suficientemente *seguro* en contra de ataques de espionaje.

4.5.1 *Modelo de programación*

En la plataforma Android existen muchas clases e interfaces que proveen una abstracción de estos protocolos. Estas están agrupadas en paquetes, y entre ellos aparecen:

- `org.apache.http.impl.client` Este paquete contiene un una API para la comunicación HTTP del lado del cliente.
- `org.apache.http.impl.conn` Este paquete provee interfaces para implementar la creación, administración y cierra de conexiones.
- `javax.net.ssl` En este paquete existen interfaces y clases para implementar una capa de abstracción de sockets seguros compatibles con el protocolo SSL v3.0 y TLS v1.2.
- `java.security.cert` Este paquete contiene recursos para la administración de certificados, revisión de listas de revocación y búsqueda de rutas de certificados.

En HTTPS, el establecimiento de conexiones se puede dividir en tres módulos:

- Configuración de la conexión HTTP.
- Generación de *sockets*.
- Administración de certificados.

Código 4.1: Plantilla de implementación propuesta para DefaultHttpClient

```
public class MyHttpsClient
    extends DefaultHttpClient {
    ...
    protected ClientConnectionManager
        createClientConnectionManager() {}
    ...
}
```

Código 4.2: Declaración de objetos Schema

```
new Scheme("http", new MyPlainSocketFactory(), 80);
new Scheme("https", new MySSLSocketFactory(), 433);
```

Esos módulos por sí solos son independientes funcionalmente, pero pueden trabajar en conjunto, así que fueron programados para ser componentes en bibliotecas, con el objetivo de aprovechar la funcionalidad de cada uno minimizando las dependencias de ellas.

4.5.2 *Configuración de conexión HTTP*

Para establecer conexiones HTTPS es necesario hacer uso (o crear una clase que herede de) la clase `DefaultHttpClient` y en caso de usar herencia se debe sobreescribir el comportamiento del método `createClientConnectionManager` (ver ejemplo en la tabla 4.1).

En este método las conexiones son generadas por llamadas a una clase de tipo *fábrica* (*factories*) que crea los *sockets*.

También es necesario especificar un esquema por medio de la clase `SchemeRegistry`, esto para especificar cuales serán los protocolos que serán aceptados y el puerto asociado a cada uno de ellos.

Para el caso partícular de esta implementación sólo son especificados detalles relativos al protocolo HTTP. Esto no limita a que se puede implementar nuevos protocolos o agregar otros ya existentes para que puedan ser aceptados por la aplicación.

Cada objeto `Scheme` encapsula las especificaciones de los protocolos. En este punto los objetos `SocketFactory` son configurados para crear los sockets para las conexiones en sus puertos (ver ejemplo en la tabla 4.2).

El siguiente paso es especificar los parámetros HTTP (por ejemplo user−agent, content−charset, accept−encoding, accept−language, etc). Para esto Android proveé algunas interfaces como `HttpParams` que define un comportamiento similar a una tabla *hash*. A su vez ésta interfaz es implementada por clases como `AbstractHttpParams`, `BasicHttpParams`, `ClientParamsStack` o `DefaultedHttpParams` (ver ejemplo en la tabla 4.3).

Código 4.3: Parámetros del protocolo HTTP

```
HttpParams hp = getParams();
HttpProtocolParams.setUserAgent(hp, "myUserAgent");
HttpProtocolParams.setContentCharset(hp, "utf8");
...
```

Código 4.4: Extrayendo almacén de claves

```
KeyStore ks = KeyStore.getInstance("JKS");
ks.load(null, "mystorepassword".toCharArray());
SSLSocketFactory factory = new SSLSocketFactory(ks);
```

Por último, se requiere un objeto para administrar las conexiones del cliente. La interfaz `ClientConnectionManager` proveé algunos métodos para manejar esas conexiones. Para construir este objeto es necesario usar el objeto que hereda de la clase `DefaultHttpClient` (o esta clase misma) y la clase de registro de esquemas.

```
new SingleClientConnManager(hp, registry);
```

4.5.3 *Creación de* sockets

En un canal de comunicación, cada par debe tener disponible un *socket* para enviar o recibir datos. Los *sockets* están disponibles en la plataforma de Android por medio de la clase `Socket`. Existen clases que son fábricas de estos objetos como la clase `SSLSocketFactory`, la cual construye objetos de tipo `SSLSockets` que son abstracciones de *sockets* que proveen servicios SSL o TLS.

Esta clase fabrica *sockets* para conexiones SSL/TLS basados en Java Secure Socket Extension (JSSE) [32]. Puede ser usada por un servidor HTTPS para validar las identidades usando una lista de certificados de confianza o para autenticarse usando su propia clave privada.

Esta clase permite la autenticación del servidor mediante un archivo que contenga uno o más certificados de confianza y permita la autenticación del cliente mediante un archivo de claves que contenga una clave pública y un certificado de la clave pública.

Para obtener una instancia de `SSLSocketFactory` existen varios métodos, el método `getInstance(String)` es la forma más fácil. Este requiere un almacén de claves (*keystore*) como parámetro y el formato más común de almacenes de claves es el *Java KeyStore (JKS)* (ver ejemplo en la tabla 4.4).

La fábrica debe ser usada para registrar cualquier esquema de protocolo nuevo, en nuestro caso el esquema de interés es el SSL/TLS.

Código 4.5: Preparación del contexto de comunicación SSL

```
public class MySocketFactory
    implements LayeredSocketFactory {
    ...
    private static SSLContext
        createMySSLContext(String certKey)
        throws NoSuchAlgorithmException,
               KeyManagementException {
        SSLContext context =
            SSLContext.getInstance("TLS");
        context.init(null,
                     new TrustManager[] {},
                     null);
        return context;
    }
    ...
}
```

Existe una alternativa al uso de la clase `SSLSocketFactory`. Este procedimiento es un poco más elaborado y corresponde al que proporciona la interfaz `LayeredSocketFactory`, esto implica sobreescribir los métodos `connectSocket`, `createSocket` e `isSecure`.

4.5.4 *Administración de certificados*

Con la clase `SSLSocketFactory` la administración de certificados se realiza de manera automática y todas las dificultades y detalles son encapsuladas y permanecen ocultas en esta clase y puede ser útilizada sin mayor trabajo.

Si esta clase no es usada, se requiere realizar una implementación de la administración de los certificados lo cual puede ser una tarea complicada.

Si el uso de esta clase fuese evitado, entonces la interfaz `LayeredSocketFactory` debe ser implementada y se debe crear un objeto `SSLContext`. Este a su vez debe ser creado usando un conjunto de objetos `TrustManager` (ver ejemplo en la tabla 4.5).

Un objeto `TrustManager` es el responsable de administrar los datos de confianza para tomar decisiones como entre si se debe o no dar como válidas las credenciales de algún par. Sin embargo, existe otra interfaz con mayores prestaciones llamada `X509TrustManager`. Esta interfaz declara tres métodos para el manejo de certificados de tipo X509. Se dispone de un arreglo de objetos `X509TrustManager` como una cadena de certificados y con ésta se obtiene un control total del proceso de autenticación de usuarios (ver ejemplo en la tabla 4.6).

Código 4.6: Plantilla de implementación propuesta para X509TrustManager

```
public class MyX509TrustManager
  implements X509TrustManager {
  ...
  public void checkClientTrusted(
       X509Certificate[] chain,
       String authType)
    throws CertificateException {
    // Do client authentication
  }

  public void checkServerTrusted(
       X509Certificate[] chain,
       String authType)
    throws CertificateException {
    // Do server authentication
  }

  public X509Certificate[] getAcceptedIssuers() {
    // Return the list of certificate issuer
    // authorities which are trusted for
    // authentication of peers
  }
}
```

5 PRUEBAS Y EXPERIMENTACIÓN

5.1 PRUEBAS DE LOCALIZACIÓN

Las pruebas realizadas fueron enfocadas a los tiempos necesarios para obtener una localización bajo diferentes circunstancias.

Las pruebas se realizaron en espacios abiertos (al aire libre) y cerrados, por otro lado se probó la localización por GPS y por red IP.

En la tabla 5.1 se muestra la precisión máxima y mínima que se registró al obtener localizaciones. De manera general se puede observar que la precisión del GPS es mayor a la obtenida por la red debido a que en el caso del GPS el cálculo se realiza por linea de vista entre los satélites GPS y el dispositivo.

En la tabla 5.2 se muestra el tiempo que se tomó para obtener el primer resultado de localización. Esta prueba se realizó en un espacio abierto en un día despejado para brindar condiciones propicias para el GPS y en el caso de la red, se hizo en un espacio al aire libre con acceso a una red WiFi y el tiempo se tomó una vez que ya se tenía asiganda una dirección IP.

En la tabla 5.3 se muestra el tiempo promedio[1] que tomaron las actualizaciones en realizarse y la precisión obtenida. Como se observa, para el caso de la red prácticamente permanece constante ya que los factores de latencia que pueden afectar la medición resultan despreciables en comparación con los que pueden afectar la medición en el caso del GPS. Es importante señalar que la documentación de la API señala que el tiempo de actualización de cada solicitud de actualización puede ser ajustado, pero no se recomienda que sean valores menores a los 60 segundos para el caso del GPS debido a que afecta el consumo de energía y por otro lado el valor de actualización puede no respetarse [33].

Las pruebas se realizaron sin hacer ajustes adicionales a los valores de actualización. En el caso del GPS, una vez que se encuentra el módulo del GPS ajustado, es decir, tiene en línea de vista a los satélites, las actualizaciones son relativamente constantes. En el caso de la red, las actualizaciones fueron casi constantes con variaciones de un segundo en la mayoría de los

1 Promedio después de 10 mediciones.

Cuadro 5.1.: Precisión en datos de localización

	GPS	Network
Mínima (m)	100	800
Máxima (m)	30	100

Cuadro 5.2.: Tiempo y precisión de la primera localización

	GPS	Network
Tiempo (s)	320	14
Precisión (m)	100	700

Cuadro 5.3.: Tiempo y precisión en actualizaciones de localización

	GPS	Network
Tiempo (s)	15	15 ± 1
Precisión (m)	30-50	cte

casos y la precisión permaneció constante a la que se obtuvo en la primera vez en la misma red.

5.2 PRUEBAS DE CONEXIÓN HTTPS

Recordando uno de los objetivos principales de proyecto era integrar servicios de seguridad en las comunicaciones cuando se emiten alertas y para este caso sobre un envío de datos bajo HTTP, por ende el establecimiento de un canal de transmisión HTTPS es la solución más adecuada para el envío de alertas ya que cumple con los servicios de seguridad deseados y la implementación conserva una compatibilidad con estándares tanto comerciales o no comerciales.

Para verificar que las conexiones se realizaban por medio de HTTPS se usó la biblioteca OpenSSL [34], con ella se pudieron obtener datos como los certificados tanto del servidor como del cliente, datos enviados durante la fase de intercambio (*SSL handshaking*), datos de la sesión SSL (como el protocolo, el cifrador, etc.) (ver en las tablas 5.1, 5.2 y 5.3).

Los resultados mostrados hacen evidente que la transmisión de datos efectivamente se realiza en un canal cifrado de HTTPS.

5.3 PRUEBAS DE COMPATIBILIDAD ANDROID

Como es bien conocido uno de lo mayores problemas que tienen las aplicaciones diseñadas para la plataforma de Android es la fragmentación [35] [36].

Este problema fue tomado en cuenta en la implementación realizada. La implementación usó un nivel de compatibilidad mínima con la versión 4 ó también conocida con el nombre clave *Donut*. Con ello se obtiene una compatibilidad en cuanto a uso de la API con el 99 % de los usuarios del sistema operativo (ver imagen 5.1[2]).

2 Este muestreo está actualizado al 2 de septiembre de 2011.

Código 5.1: Datos de conexión SSL

```
CONNECTED(00000003)
SSL_connect:before/connect initialization
SSL_connect:SSLv2/v3 write client hello A
SSL_connect:SSLv3 read server hello A
depth=0 C = MX, ST = Distrito Federal, L = Mexico,
 O = Israel Buitron, CN = Israel Buitron,
 emailAddress = israel.buitron@gmail.com
verify error:num=18:self signed certificate
verify return:1
depth=0 C = MX, ST = Distrito Federal, L = Mexico,
 O = Israel Buitron, CN = Israel Buitron,
 emailAddress = israel.buitron@gmail.com
verify return:1
SSL_connect:SSLv3 read server certificate A
SSL_connect:SSLv3 read server key exchange A
SSL_connect:SSLv3 read server certificate request A
SSL_connect:SSLv3 read server done A
SSL_connect:SSLv3 write client certificate A
SSL_connect:SSLv3 write client key exchange A
SSL_connect:SSLv3 write change cipher spec A
SSL_connect:SSLv3 write finished A
SSL_connect:SSLv3 flush data
SSL3 alert read:fatal:handshake failure
SSL_connect:failed in SSLv3 read server session ticket A
3074164392:error:14094410:SSL routines:SSL3_READ_BYTES:
 sslv3 alert handshake failure:s3_pkt.c:1193:SSL alert
 number 40
3074164392:error:140790E5:SSL routines:SSL23_WRITE:
 ssl handshake failure:s23_lib.c:177:
---
Certificate chain
 0 s:/C=MX/ST=Distrito Federal/L=Mexico/O=Israel Buitron
     /CN=Israel Buitron
     /emailAddress=israel.buitron@gmail.com
   i:/C=MX/ST=Distrito Federal/L=Mexico/O=Israel Buitron
     /CN=Israel Buitron
     /emailAddress=israel.buitron@gmail.com
---
```

Código 5.2: Datos de conexión SSL

```
Server certificate
-----BEGIN CERTIFICATE-----
MIIF/DCCA+SgAwIBAgIJAOAJ9kB5ciUDMA0GCSqGSIb3DQEBDQUAMIGUM
QswCQYDVQQGEwJNWDEZMBcGA1UECAwQRGlzdHJpdG8gRmVkZXJhbDEPMA
0GA1UEBwwGTWV4aWNvMRcwFQYDVQQKDA5Jc3JhZWwgQnVpdHJvbjEXMBU
GA1UEAwwOSXNyYWVsIEJ1aXRyb24xJzAlBgkqhkiG9w0BCQEWGGlzcmFl
bC5idWlocm9uQGdtYWlsLmNvbTAeFwoxMTAxMjYwNDU3MzdaFwoxMjAxM
jYwNDU3MzdaMIGUMQswCQYDVQQGEwJNWDEZMBcGA1UECAwQRGlzdHJpdG
8gRmVkZXJhbDEPMA0GA1UEBwwGTWV4aWNvMRcwFQYDVQQKDA5Jc3JhZWw
gQnVpdHJvbjEXMBUGA1UEAwwOSXNyYWVsIEJ1aXRyb24xJzAlBgkqhkiG
9w0BCQEWGGlzcmFlbC5idWlocm9uQGdtYWlsLmNvbTCCAiEwDQYJKoZIh
vcNAQEBBQADggIOADCCAgkCggIADeuV+T3b2nwkOnnZD2Cx6PQQ4JsZ9w
tC1npZAjGS90vBFCJaKIaP40SI/06edQVduLXob8GQB5lVCH30CWbuU2Z
J50LS0lggDX/GtCSZbky7We6FeRBB8G6X0/0OIBKkaT8dhP8EEbQEe05V
VYAdrw2Jyi6Mtyt1hfdpXyPLDJhU4CQWemgTjDA4jkEYMG6Pq1jRuumRD
mtV7qrWtsYdPXlxaVadrOuy7cZfsP2UquyQyuvsw7apm2c41Jasa+kC3s
ADKOWveQ6zff5zbi53qpmauVXst+skytqSBh9elJQdI1HqV7cbXuI80AT
Zd83bWGYKoen/TXhza1GzBEKcbkP7dSwJYk+heZiZw5FJNjKrwDTp4zHL
bORM0SPO0jmw1spt5kUYXZdINfjmD4ZvUpvxxecFKptLzQqVwbkdxrQab
zN9cKXLZYDWXk6vIgUpHfTbmaDLXFjMH9SWMLaRaQUdc87+qyhGM2fwRG
k2gq7PxJgiw2JKHlJENbofa/UdAUKNhZ9mFIdon/g06i12vOFo9alfkNu
HAggFBvknToyLLjZAIlckasTnj+xO7SOehWUepiZBxZBBQZQzDKm1MHoi
PfYAOxDgyS+NRNNhYLg1K8GGVj7Qc88comP6gUYH3xMIEmnxIi7fHk495
h5j3yr8SvLTNMsRxQIIRle7Iq8CAwEAAaNQME4wHQYDVR0OBBYEFJXmRz
brmLAo/nnbTjdXfVHyo/gxMB8GA1UdIwQYMBaAFJXmRzbrmLAo/nnbTjd
XfVHyo/gxMAwGA1UdEwQFMAMBAf8wDQYJKoZIhvcNAQENBQADggIBAAN2
TOyBjlJkTksHpVR+FjG1Gq/IaMCNhWALS8HmJBU880ZqjSUpou5YKaVt/
beFsQWrEfwz3kxjMZY/OCQY5XdPqFMkqBJnz/xTlQbm5mFforoNkq0MrF
7cw9wNqCPVNDPCxYK6o+kEZxflTwJMGdBuB2uMkB3xKjAjNlSe5EgVPBP
Mj4hKbN9W5AtM0gahotdxl5VTo6gTSYX8sjnjCyxr8KDYQZ1F70ZLBvk9
K0YtHkuOjBKGjpsQp5Q7QAnquUTZFaAxkJlvXQ+42tXMg+mASIcJMSBFM
VvDuK/D1kfVEaGTf9FefJWN0881rl4XFSiPSQefgRbzDbalUXa1zQM/W1
sHvfc2GyEGk3ByRpXlTTZgpJBWlbcZc58Z/+CuM8B3Pcoco40Edqgt3gU
qaM9gdFi6kZPMFEz6ori43peaxx42jKQWIwXxFGxJgPfD2WCD1Iq6MlM8
0/rWCnolT5V59Ajjc2Q4NmXgFzTSNVxpI0MrcdWlyp/XO2+yv4baz9eHe
WIRP/RTRbJeE3QaL7AciCIXDcBEaH0AAeFY7w30G1w8P7/0N7NlaN6tQO
0PxcPA+Skzkaye0XihRGujmzA2+p/2ifyUtGAPiXG9802HaZ5h5ag0yfM
f1GL6Y74k36MwPoItcnZwfdSDtNzO904HWv9wWjLcxpyeFRhRmPa9
-----END CERTIFICATE-----
subject=/C=MX/ST=Distrito Federal/L=Mexico
        /O=Israel Buitron/CN=Israel Buitron
        /emailAddress=israel.buitron@gmail.com
issuer=/C=MX/ST=Distrito Federal/L=Mexico/O=Israel
    Buitron
        /CN=Israel Buitron
        /emailAddress=israel.buitron@gmail.com
---
No client certificate CA names sent
---
```

Código 5.3: Datos de conexión SSL

```
SSL handshake has read 2423 bytes and written 210 bytes
---
New, TLSv1/SSLv3, Cipher is DHE-RSA-AES256-SHA
Server public key is 4092 bit
Secure Renegotiation IS supported
Compression: zlib compression
Expansion: NONE
SSL-Session:
    Protocol  : TLSv1
    Cipher    : DHE-RSA-AES256-SHA
    Session-ID:
    Session-ID-ctx:
    Master-Key: EF8CCD73968F1C6A73CD13617E82FD3CC752880
                E6AA28CF8546729C49835A7C4506C869D373306
                B480834AFB68F17969
    Key-Arg   : None
    Krb5 Principal: None
    PSK identity: None
    PSK identity hint: None
    Compression: 1 (zlib compression)
    Start Time: 1315837353
    Timeout   : 300 (sec)
    Verify return code: 18 (self signed certificate)
---
```

images/chart_sep_2_2011.png

Figura 5.1.: Distribución de versiones de Android [1]

Este nivel de compatibilidad tiene como consecuencia que nuestra implementación pueda ser instalada en casi cualquier dispositivo con la plataforma Android. Por otro lado si en un futuro esta implementación fuese modificada (para agregar mejoras, corregir posibles fallos o especialice esta implementación),

mientras el desarrollador conserve este nivel de compatibilidad se sigue garantizando esta prestación.

La aplicación fue instalada con éxito en los siguiente modelos:

- Samsung Galaxy S (usando Android 2.1, 2.2 y Cyanogenmod 7.1[3])
- HTC Wildfire [38] (usando Android 2.1)
- HTC Hero [39] (usando Android 1.6)

3 Cyanogenmod es una distribución alterna no oficial que esta basado en la versión Android 2.3. [37]

6 CONCLUSIONES

6.1 INTERÉS AL NIVEL DE MAESTRÍA

En la sección de 1 se describe el proyecto ELISA y se expone su problemática. Por un lado se requiere una aplicación que se ejecute sobre la plataforma Android en la cual se pueda obtener la localización de un dispositivo y este a su vez envíe una alerta y por otro lado, la comunicación que se establece hacia el servidor no goza de servicios de seguridad que protejan los datos transmitidos.

El presente trabajo presenta una solución integrando diferentes tecnologías y protocolos para ofrecer una solución óptima, al mismo tiempo se explora en la tecnología de los dispositivos móviles que hacen uso de la plataforma Android como su sistema operativo.

Se propone un modelo de programación para el uso de los módulos de localización geográfica como los dispositivos GPS integrados en los teléfonos móviles. También se explotan recursos como las APIs de bibliotecas criptográficas para establecer canales de comunicación seguras.

Al momento de la realización de este documento no se encontraron publicadas aplicaciones similares dentro de los repositorios de proyectos de código fuente libre. Es deseable que existan proyectos similares que compitan entre sí para retroalimentar a su vez a cada uno de ellos y proveer un mejor servicio a los usuarios.

También se expone una forma de autenticar los pares de una conexión usando certificados digitales. En este punto se expuso una forma de realizar una supuesta autenticación en la que se logra establecer una conexión exitósamente, sin embargo, el proceso de autenticación puede dar plena confianza del par contrario y por ende se podrían realizar ataques por este medio. Con esto se pretende evidenciar que las comunicaciones por HTTPS pueden no ser seguras, en este caso es un riesgo dependiente de la implementación realizada por el programador de la aplicación cliente en uso.

6.2 TRABAJO FUTURO

Este proyecto fue construido con una idea de ser actualizado y mejorado por lo que se proponen las siguientes lineas de investigación:

- Desarrollo de una PKI para realizar autenticación tanto de los dispositivos móviles como del servidor de ELISA. En este

punto se puede analizar la implementación de una PKI de un proveedor externo[1].

- Implementar la autenticación del usuario frente al servidor de ELISA en el momento de establecer el canal HTTPS.
- Explorar servicios de seguridad en servicios como SMS y correo electrónico como alternativas de comunicación con el servidor de ELISA.
- Buscar la mejora en los tiempos de adquisición y la precisión de la información de localización geográfica.
- Realizar un estudio para proveer alternativas para los usuarios que no poseen teléfonos móviles de *gama alta* o con recursos de hardware limitados.
- Realizar una implementación del cliente en otras plataformas de dispositivos móviles.
- Explorar formas en las que la autenticación para obtener el servicio de alertas pueda realizarse mediante algún otro factor que no fuese el dispositivo móvil. Inclusive quizá que se pudiese realizar delegando a un usuario no registrado.
- Desarrollar un protocolo de identificación de dispositivos mediante la implementación de identificadores de dispositivos e identificadores de usuarios, con el objetivo de proveer una o varias vías de identificación de usuarios del servicio.

1 Existen proveedores de certificados digitales como VeriSign, CAcert, Thawte, etc. que proveen certificados válidos para servicios especificos y con niveles de confianza diferidas.

A

CÓDIGO FUENTE

Durante el desarrollo de la aplicación se hizo uso de un sistema de control de versiones para administrar de forma más eficiente el código fuente generado, también proporcionó una forma de generar y probar prototipos de la aplicación de forma alterna.

El sistema de control de versiones usado fue Git[1] ya que proveé muchas ventajas como una administración distribuida, accesibilidad por medio de protocolos como SSH y HTTPS, posibilidad de desarrollo no lineal, etc.

A.1 REPOSITORIO PÚBLICO

La aplicación desarrollada se encuentra disponible en un repositorio de control de versiones llamado Github.

GitHub es un servicio de hospedaje basado en web para proyectos de desarrollo de software que usan el sistema de control del versiones Git, además proporciona herramientas como un foro tipo Wiki y una página para cada proyecto, gráficos de comparación en donde se puede consultar de manera fácil el avance de los colaboradores.

El repositorio en Github del proyecto se puede acceder mediante la siguiente URL:

`https://github.com/neoriddle/locationshare`

A.2 ACCESO AL CÓDIGO FUENTE

El código puede ser obtenido mediante la herramienta `git clone`[2] (ver ejemplo en el cuadro A.1).

1 Para mayor información sobre el conjunto de herramientas Git, se recomienda visitar su página oficial [40].

2 En este punto se da por hecho que Git se encuentra instalado y accesible en el sistema.

Código A.1: Obtención del código mediante `git clone`.

```
git clone git://github.com/neoriddle/locationshare.git
Cloning into locationshare...
remote: Counting objects: 228, done.
remote: Compressing objects: 100% (99/99), done.
remote: Total 228 (delta 65), reused 228 (delta 65)
Receiving objects: 100% (228/228), 91.56 KiB, done.
Resolving deltas: 100% (65/65), done.
```

BIBLIOGRAFÍA

[1] A. Developers, "Android platform version," septiembre 2011. Página web: http://developer.android.com/resources/dashboard/platform-versions.html.

[2] P. Syverson, "A Taxonomy of Replay Attacks," *Computer Security Foundations Workshop VII*, pp. 187–191, 1994.

[3] T. Dierks and E. Rescorla, "RFC 5246: The Transport Layer Security (TLS) Protocol Version 1.2," agosto 2008. Página web: http://tools.ietf.org/html/rfc5246.

[4] U. Irvine, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "RFC 2616: Hypertext Transfer Protocol – HTTP/1.1," junio 1999. Página web: http://tools.ietf.org/html/rfc2616.

[5] E. Rescorla, "HTTP over TLS," tech. rep., RTFM, Inc., mayo 2000. Página web: http://tools.ietf.org/html/rfc2818.

[6] B. Elgin, "Google buys android for its mobile arsenal," *Bloomberg Businessweek*, agosto 2005. Página web: http://www.businessweek.com/technology/content/aug2005/tc20050817_0949_tc024.htm (09-dic-2010).

[7] N. I. of Standards and Technology, "Data encryption standard," diciembre 1993.

[8] P. M. P. Karn and W. Simpson, "RFC 1851: The ESP Triple DES Transform," septiembre 1995. Página web: http://tools.ietf.org/html/rfc1851.

[9] N. I. of Standards and Technology, "Aes algorithm (rijndael) information," diciembre 1993. Página web: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

[10] M. B. H. Krawczyk and R. Canetti, "RFC 2104: HMAC: Keyed-Hashing for Message Authentication," febrero 1997. Página web: http://tools.ietf.org/html/rfc2104.

[11] R. F. T. Berners-Lee and H. Frystyk, "RFC 1945: Hypertext Transfer Protocol – HTTP/1.0," mayo 1996. Página web: http://tools.ietf.org/html/rfc1945.

[12] T. Berners-Lee, "RFC 1630: Universal resource identifiers in www," junio 1994. Página web: http://tools.ietf.org/html/rfc1630.

[13] L. M. T. Berners-Lee and M. McCahill, "RFC 1738: Uniform Resource Locators (URL)," diciembre 1994. Página web: http://tools.ietf.org/html/rfc1738.

[14] K. Sollins and L. Masinter, "RFC 1737: Functional Requirements for Uniform Resource Names," diciembre 1994. Página web: http://tools.ietf.org/html/rfc1737.

[15] N. Freed and N. Borenstein, "RFC 2045: Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies," november 1996. Página web: http://tools.ietf.org/html/rfc2045.

[16] J. Postel, "RFC 821: Simple Mail Transfer Protocol," agosto 1982. Página web: http://tools.ietf.org/html/rfc821.

[17] B. Kantor and P. Lapsley, "RFC 977: Network News Transfer Protocol," febrero 1986. Página web: http://tools.ietf.org/html/rfc977.

[18] J. Postel and J. Reynolds, "RFC 959: File Transfer Protocol," octubre 1985. Página web: http://tools.ietf.org/html/rfc959.

[19] F. Anklesaria, M. McCahill, P. Lindner, D. Johnson, D. Torrey, and B. Alberti, "RFC 1436: The Internet Gopher Protocol (a distributed document search and retrieval protocol)," marzo 1993. Página web: http://tools.ietf.org/html/rfc1436.

[20] F. Davis, B. Kahle, H. Morris, J. Salem, T. Shen, R. Wang, J. Sui, and M. Grinbaum, "WAIS Interface Protocol Prototype Functional Specification," abril 1990.

[21] G. Inc., "Google projects for android," febrero 2011. Página web: http://tools.ietf.org/html/rfc1851.

[22] Motorola, "Motorola méxico - celulares - i1776 especificaciones," 2008. Página web: http://www.motorola.com/Consumers/MX-ES/Consumer-Products-and-Services/Mobile-Phones/ci.i776_MX-ES.alt.

[23] Motorola, "Motorola i176 guía de usuario," 2008. Página web: http://www.nextel.cl/pdf/Manual-i776.pdf.

[24] J. C. Árias, "Sistema de localización y asistencia inmediata," Master's thesis, Instituto Tecnológico y de Estudios Superiores de Monterrey, Atizapán de Zaragoza, Estado de México, noviembre 2011.

[25] G. Lukas, "Memorizingtrustmanager - the android trustmanager - github," febrero 2011. Página web: https://github.com/ge0rg/MemorizingTrustManager.

[26] H. C. van Tilborg, *Encyclopedia of Cryptography and Security*. 233 Spring Street, New York, NY 10013, USA: Springer Science+Bussiness Media Inc., 2nd ed., 2005.

[27] R. Rivest, "The MD5 Message-Digest Algorithm," tech. rep., MIT Laboratory for Computer Science and RSA Data Security, Inc., April 1992.

[28] S. Turner and L. Chen, "Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms," tech. rep., IECA and NIST, March 2011.

[29] N. I. of Standards and Techonology, "Secure hash standard," tech. rep., Federal Information Processing Standards Publication, october 2008. This standard specifies five secure hash algorithms - SHA-1, SHA-224, SHA-256, SHA-384, and SHA-512 - for computing a condensed representation of electronic data.

[30] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1," tech. rep., UC Irvine and Compaq and W3C and MIT and Xerox and Microsoft, junio 1999. Página web: http://tools.ietf.org/html/rfc2818.

[31] Mozilla, "Included certificate list." Website, marzo 2007. Página web: http://www.mozilla.org/projects/security/certs/included/.

[32] Oracle, "Java SE Security," 2011. Página web: http://www.oracle.com/technetwork/java/javase/tech/index-jsp-136007.html.

[33] Android.com, "Locationmanager class," 2010. Página web: http://developer.android.com/reference/android/location/LocationManager.html#requestLocationUpdates(java.lang.String,long,float,android.location.LocationListener).

[34] J. Weller, "Handy https check with openssl." Website Forum, septiembre 2011. Página web: http://jimweller.com/2008/07/handy-https-check-with-openssl.html.

[35] H. McCracken, "What android fragmentation problem?." WebSite, mayo 2010. Página web: http://www.pcworld.com/article/196985/what_android_fragmentation_problem.html.

[36] S. Kovach, "Yes, android fragmentation is still a huge problem. this chart proves it," septiembre 2011. Página web: http://www.businessinsider.com/android-phone-upgrade-report-2011-9.

[37] S. Kondik, "Cyanogenmod | android community rom based on gingerbread." Website, septiembre 2011. Página web: http://www.cyanogenmod.com/.

[38] H. Corporation, "Htc - productos - htc wildfire - especificación," septiembre 2011. Página web: http://www.htc.com/es/product/wildfire/specification.html.

[39] H. Corporation, "Htc - productos - htc hero - especificación," septiembre 2011.

[40] S. Chacon, "Git - fast version control system," septiembre 2011. Página web: http://git-scm.com, consultado 11-oct-2011.