Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional

Unidad Zacatenco

Departamento de Computación

Creación de una infraestructura de clave pública para dispositivos móviles

Tesis

que presenta

Javier Silva Pérez

para obtener el grado de

Maestro en Ciencias en Computación

Director de tesis

Dr. Guillermo Morales Luna

México, D.F. Noviembre de 2012

Agradecimientos

Al finalizar este trabajo no puedo evitar pensar en todas las personas que me brindaron su apoyo, ya que sin ellas no hubiera sido posible concluir esta etapa de mi vida.

Primero que nada, quiero agradecer a esas personas que desde siempre me han brindado su apoyo y confianza incondicional, a esas personas que han inculcado en mi todos los valores que poseo, a esas personas que han dado todo para que pueda llegar hasta donde estoy, a esas personas a las que les debo todo lo que soy, a esas personas a las que es un orgullo llamar... mis padres.

A mis hermanas, que siempre estuvieron ahí cuando las necesite y cuando no también. Gracias.

A Cyn, por saber siempre como levantarme el ánimo, por regalarme una sonrisa en esos momentos difíciles, por nunca dudar de mí y siempre impulsarme para convertirme en una mejor persona. Gracias por el apoyo que siempre me diste y la ayuda que siempre me ofreciste. Pero sobre todo, gracias por el amor incondicional que siempre me mostraste. Te amo.

A mi asesor Dr. Guillermo Morales, gracias por los consejos siempre útiles que me brindo, por todo el conocimiento que me regalo, pero sobre todo gracias por su confianza y paciencia a lo largo de este trabajo.

A mis sinodales Dr. Arturo Díaz y Dr. Dominique Decouchant, les agradezco haberse tomado el tiempo para leer este trabajo y por los comentarios que me aportaron. Gracias.

Y claro, no puedo dejar de agradecer a todas aquellas personas que de una u otra forma ayudaron a que este día llegara, a mis compañeros de generación, a mis profesores, a Sofi y al personal del departamento. Gracias por el apoyo desinteresado.

Por último, quiero agradecer al CONACYT por la beca brindada. Gracias

Índice general

1. Introducción **1**
- 1.1. Antecedentes y contexto de la investigación . . . 1
- 1.2. Planteamiento del problema . . . 3
- 1.3. Objetivos del trabajo de investigación . . . 5
  - 1.3.1. Objetivo general . . . 5
  - 1.3.2. Objetivos específicos . . . 6
- 1.4. Metodología . . . 6
- 1.5. Organización de la tesis . . . 7

2. Marco general de una PKI y estado del arte **9**
- 2.1. Mecanismos criptográficos de seguridad . . . 9
  - 2.1.1. Servicios de seguridad . . . 9
  - 2.1.2. Criptografía simétrica . . . 10
  - 2.1.3. Funciones "picadillo" . . . 10
  - 2.1.4. Criptografía asimétrica . . . 12
  - 2.1.5. Resumen de mecanismos de seguridad . . . 14
- 2.2. Infraestructura de clave pública (PKI) . . . 14
  - 2.2.1. Clave pública . . . 15
  - 2.2.2. Componentes básicos . . . 16
  - 2.2.3. Estructuras de datos . . . 18
  - 2.2.4. Modelos de Confianza . . . 20
- 2.3. Revisión del estado de arte . . . 26

3. Diseño de la PKI **33**
- 3.1. Proceso de diseño . . . 34
  - 3.1.1. Definir requerimientos de certificación . . . 34
  - 3.1.2. Definir modelo de confianza de la infraestructura . . . 36
  - 3.1.3. Definir opciones de configuración de los certificados . . . 39
  - 3.1.4. Definir plan de administración de certificados . . . 42
- 3.2. Descripción y características de la PKI . . . 44
  - 3.2.1. Requerimientos . . . 45
  - 3.2.2. Modelo de confianza . . . 47
  - 3.2.3. Opciones de configuración de certificados . . . 52

3.3. Declaración de políticas de certificación 55
3.3.1. Introducción 56
3.3.2. Repositorios y publicación de la información 59
3.3.3. Obligaciones de los titulares 59
3.3.4. Identificación y autenticación de los titulares de certificados 60
3.3.5. Requisitos operacionales para el ciclo de vida de los certificados . . 61
3.3.6. Perfiles de los certificados y CRL 69

4. Implementación y pruebas sobre la plataforma móvil 73
4.1. Diseño del API 73
4.1.1. Diagrama "entidad-relación" 74
4.1.2. Diagrama de paquetes 75
4.1.3. Diagrama de clases 77
4.2. Pruebas de la API 90
4.2.1. Características de la API 91
4.2.2. Pruebas funcionales 95
4.2.3. Pruebas de rendimiento 96

5. Aplicación de prueba para Android 107
5.1. Android para dispositivos móviles 107
5.1.1. Características 108
5.1.2. Arquitectura 109
5.1.3. Componentes de una aplicación 111
5.2. Aplicación de prueba 112
5.2.1. Características principales 113
5.2.2. Diagrama de Paquetes 115

6. Conclusiones y trabajo a futuro 117
6.1. Conclusiones 117
6.2. Trabajo a futuro 120

Bibliografía 121

A. Manual de Uso 125
A.1. Características de diseño 125
A.2. Interfaces de la aplicación 131

Índice de figuras

1.1. Disco de cifrado utilizado en la Guerra Civil de EUA. 2

2.1. Esquema básico de criptografía simétrica. 11
2.2. Esquema básico de funciones picadillo. 11
2.3. Esquema básico de funciones HMAC. 12
2.4. Esquema básico de criptografía asimétrica - Cifrado. 13
2.5. Esquema básico de criptografía asimétrica - Firma Digital. 13
2.6. Relación entre componentes de una PKI. 17
2.7. Estructura de un certificado X.509. 19
2.8. Cadena de certificación de dos niveles. 20
2.9. Estructura de una CRL. 21
2.10. Ejemplo de una estructura jerárquica. 21
2.12. Ejemplo de un modelo jerárquico múltiple. 22
2.11. Ejemplo de un modelo jerárquica. 22
2.13. Ejemplo de un modelo de malla. 23
2.14. Ejemplo de un modelo mixto. 24
2.15. Ejemplo de un modelo con puente. 25
2.16. Ejemplo de una red de confianza. 25

3.1. Proceso para diseñar una PKI. 34
3.2. Subproceso para definir los requerimientos de certificación. 35
3.3. Subproceso para establecer el modelo de confianza. 37
3.4. Subproceso para definir las opciones de configuración de los certificados. . . 40
3.5. Subproceso para definir el plan de administración de certificados. 43
3.6. Ejemplo de un modelo de confianza jerárquico de la PKI. 49
3.7. Ejemplo de un modelo de red de confianza de la PKI. 50
3.8. Esquema implementado de criptografía asimétrica - Firma digital. 54
3.9. Esquema implementado de criptografía asimétrica - Cifrado. 55
3.10. Arquitectura general de la PKI. 56
3.11. Proceso de solicitud de certificado con claves propias. 62
3.12. Proceso de solicitud de certificado con generación de nuevas claves. 63
3.13. Proceso de renovación remota de certificados. 66
3.14. Proceso de renovación presencial de certificados. 67
3.15. Proceso de revocación de certificados. 69

4.1. Diagrama de Entidad-Relación. 75
4.2. Diagrama de paquetes - General. 76
4.3. Diagrama de paquetes - `db`. 76
4.4. Diagrama de paquetes - `cryptography`. 77
4.5. Diagrama de clases - `ec`. 79
4.6. Diagrama de clases - `algorithm`. 80
4.7. Diagrama de clases - `cert`. 81
4.8. Diagrama de clases - `exception`. 81
4.9. Diagrama de clases - `key` - RSA. 82
4.10. Diagrama de clases - `key` - EC. 83
4.11. Diagrama de clases - `utils` - Criptografía. 85
4.12. Diagrama de clases - `utils` - X.509. 86
4.13. Diagrama de clases - `utils` - Otros. 87
4.14. Diagrama de clases - `exception`. 87
4.15. Diagrama de clases - `db`. 88
4.16. Diagrama de clases - `dao`. 89
4.17. Diagrama de clases - `controller`. 90

5.1. Modelo de capas para desarrollo de software. 107
5.2. Arquitectura de *Android*. 109
5.3. Diagrama de paquetes - Aplicación de prueba. 116

A.1. Gestos soportados en la aplicación. 125
A.2. Botón *home* e icono de progreso. 126
A.3. Navegación en la aplicación - Tipo 1. 127
A.4. Navegación en la aplicación - Tipo 2. 128
A.5. Elementos *Action Bar*. 129
A.6. Cambio de orientación utilizando la *Action Bar*. 129
A.7. Soporte de interfaz para múltiples pantallas - Claves. 130
A.8. Soporte de para múltiples idiomas. 131
A.9. Interfaz principal. 132
A.10. Interfaz - Detalles de clave. 133
A.11. Cuadro de dialogo para contraseñas. 134
A.12. Operaciones adicionales - Claves. 135
A.13. Interfaz para importar claves. 136
A.14. Interfaz para crear certificados. 136
A.15. Interfaz para importar certificados. 137
A.16. Interfaz - Detalles de certificado. 138
A.17. Interfaz - Proceso de firma de certificado. 139
A.18. Interfaz - Proceso de verificación de certificado. 140
A.19. Interfaz para agregar certificado a lista de confianza. 141
A.20. Interfaz - Certificado en lista de confianza. 142
A.21. Interfaz - Operaciones criptográficas. 142

Índice de tablas

2.1. Servicios de seguridad por mecanismo. 14
2.2. Servicios de seguridad contemplados. 31
2.3. Características de la aplicación. 31
2.4. Elementos de una PKI considerados. 31

3.1. Equivalencia para tamaño de llaves (bits). 41
3.2. Recomendación del NIST para tamaño de claves. 41
3.3. Extensiones propietarias de la PKI. 70

4.1. Opciones de cifrado y descifrado - AES. 91
4.2. Parámetros para funciones picadillo. 92
4.3. Parámetros para funciones cifrado y descifrado - RSA. 93
4.4. Parámetros para funciones cifrado y descifrado - EC. 93
4.5. Formatos de clave soportados. 94
4.6. Formatos para exportación e importación de claves. 94
4.7. Algoritmos de firma para de certificados X.509 soportados. 96
4.8. Características dispositivos de prueba. 97
4.9. Resultados pruebas de rendimiento - AES. 99
4.10. Resultados pruebas de rendimiento - RSA. 102
4.11. Resultados pruebas de rendimiento - EC. 103
4.12. Resultados pruebas de rendimiento - Exportar/Importar. 105
4.13. Resultados pruebas de rendimiento - X.509. 106

Lista de Acrónimos

PKI Infraestructura de Clave Pública (siglas del inglés *Public Key Infrastructure*)

VPN Red Privada Virtual (siglas del inglés *Virtual Private Network*)

API Interfaz para el desarrollo de aplicaciones (siglas del inglés *Application Programming Interface*)

SO Sistema Operativo

PGP Pretty Good Privacy

PKCS Estándar de Criptografía de Clave Pública (siglas del inglés *Public-Key Cryptography Standards*))

CRL Lista de Revocación de Certificados (siglas del inglés *Certificate Revocation List*)

CA Autoridad de Certificación (siglas del inglés *Certification Authority*)

RA Autoridad de Registro (siglas del inglés *Registration Authority*)

IPRA Autoridad de Políticas de Registro de Internet (siglas del inglés *Internet Policy Registration Authority*)

PCA Autoridad de Políticas de Certificación (siglas del inglés *Policy Certification Authorities*)

DER Reglas Distinguidas de Codificación (siglas del inglés *Distinguished Encoding Rules*)

AES Estándar Avanzado de Cifrado (siglas del inglés *Advanced Encryption Standard*)

DES Estándar de Cifrado de Datos (siglas del inglés *Data Encryption Standard*)

RSA Rivest, Shamir y Adleman (por las iniciales de sus autores)

EC Curva Elíptica (siglas del inglés *Elliptic Curve*)

ECC Criptografía de Curvas Elíptica (siglas del inglés *Elliptic Curve*)

SDK Kit de Desarrollo de Software (siglas del inglés *Software Development Kit*)

XML Lenguaje de Marcas Extendibles (siglas del inglés *eXtensible Markup Language*)

NDK Kit de Desarrollo Natal (siglas del inglés *Native Development Kit*)

JNI Interfaz Natal de Programación en Java (siglas del inglés *Java Native Interface*)

HMAC Código de Autenticación de Mensajes (siglas del inglés *Hash-based Message Authentication Code*)

NIST Instituto Nacional de Estándares y Tecnología (siglas del inglés *National Institute of Standards and Technology*)

ETSI Instituto Europeo de Normas de Telecomunicaciones (siglas del inglés *European Telecommunications Standards Institute*)

OID Identificador de Objeto (siglas del inglés *Object Identifier*)

DPC Declaración de Políticas de Certificación

PC Políticas de Certificación

DN Nombre Distintivo (siglas del inglés *Distinguished Name*)

Resumen

La creación de sistemas de cómputo en ambientes móviles va de la mano con el uso de medios inalámbricos para transmitir la información necesaria. Dichos medios son ampliamente vulnerables a diversos ataques informáticos que atentan (entre otros) contra la confidencialidad, la integridad y la autenticación tanto de los datos como de los participantes en la comunicación. Debido a esto, es necesario crear mecanismos que permitan una comunicación segura. Una de las técnicas más ampliamente utilizadas para ofrecer este servicio es sin duda la Criptografía, la cual permite el desarrollo de diversos bloques de seguridad dependiendo de las necesidades del sistema, específicamente la Infraestructura de Clave Pública (siglas del inglés *Public Key Infrastructure*) (PKI) ofrece servicios como integridad y autenticación. En este trabajo se realiza un análisis de la importancia del desarrollo de una PKI tomando en cuenta un ambiente de dispositivos móviles, considerando ventajas y desventajas. En el análisis se hace una comparación entre los diferentes desarrollos existentes en la actualidad para la plataforma de desarrollo *Android*, la cual ha ido ganando una enorme popularidad en los últimos años.

Palabras clave: PKI, Android, Dispositivos móviles, Seguridad, Criptografía.

Abstract

The creation of computer systems in mobile environments is closely related to the use of wireless technologies for information transmission. However, the technologies are vulnerable to several attacks threatening confidentiality, integrity and authentication of both data and participants. Thus, the creation of mechanisms for secure communication is capital. Definitely one of the most widely used mechanisms for providing this service is Cryptography, which allows the use of several security measures; in fact, public key infrastructure (PKI) offers services such as integrity and authentication. This paper presents an analysis of the importance of developing a PKI environment, taking into account mobile devices to analyze their advantages and disadvantages. The analysis will present a comparison among existing applications for the *Android* platform which has gained enormous popularity in recent years.

Keywords: PKI, Android, Mobile Devices, Security, Cryptography.

Capítulo 1

Introducción

1.1. Antecedentes y contexto de la investigación

Por siglos, reyes, reinas, jefes de estado y generales militares han confiado en comunicaciones eficientes para gobernar sus territorios y comandar sus ejércitos. Al mismo tiempo todos ellos han estado conscientes de las consecuencias que se pueden generar si los mensajes caen en manos hostiles, revelando así secretos importantes. Ésto fue lo que motivó en un inicio el desarrollo de diversos códigos y cifradores, los cuales son técnicas para disfrazar los mensajes de tal manera que la única persona capaz de leerlos sea la persona a la que se enviaron dichos mensajes. El deseo de secrecía ha hecho que las naciones creen departamentos desarrolladores de códigos y cifradores, que son los responsables de garantizar la seguridad de las comunicaciones. Al mismo tiempo se han ido generando diversos ataques para quebrantar estos mecanismos y revelar los secretos. La historia de códigos y cifradores es la historia de la batalla antagónica entre criptógrafos y criptoanalistas, la cual ha tenido un impacto dramático en el curso de la historia [Singh, 2001]. A grandes rasgos, codificar es colocar un mensaje en un alfabeto apropiado para garantizar su transmisión fiel, en tanto que cifrar es transformar un mensaje en otro de manera que el mensaje original no sea recuperable del cifrado a menos de contar con la clave correcta.

El desarrollo de códigos puede ser visto como una lucha evolutiva, ya que un código está siendo atacado constantemente por criptoanalistas, una vez que éstos han encontrado una debilidad en el código, éste tiene dos opciones, evolucionar hasta convertirse en un código más fuerte o extinguirse [Singh, 2001]. Esta evolución es la que ha llevado a los códigos a transformarse, desde sus inicios con la criptografía clásica y el uso de diversos mecanismos como el disco de cifrado mostrado en la Figura 1.1, en algoritmos más complejos y fuertes para resistir los diversos ataques del mundo moderno, hasta llegar a los algoritmos y esquemas que actualmente predominan en el mundo de la criptografía como lo son los algoritmos de clave simétrica y los algoritmos clave pública. Los cuales se valen de la complejidad para resolver ciertas funciones matemáticas para garantizar la seguridad del código.

Figura 1.1: Disco de cifrado utilizado en la Guerra Civil de EUA.

A medida que la información se convierte en un producto cada día más valioso, las técnicas de comunicación y tecnologías de la información van cambiando a la sociedad hasta convertirse en una parte fundamental de la misma, por lo que es de vital importancia crear conciencia de que a pesar de todos los beneficios que estas tecnologías nos ofrecen, también hay riesgos al transmitir cualquier información a través de medios no seguros como lo son, ya en nuestros tiempos, Internet o las redes inalámbricas. Es por esto que el proceso de convertir mensajes, conocido como cifrado, jugará un papel cada día más fundamental en la vida diaria, ayudando a garantizar la integridad de los datos, la autenticad tanto del mensaje como de las partes que participan en la comunicación y la confidencialidad de la información que se transmite dentro de una red.

Por otro lado, la necesidad de acceder a dicha información a cualquier hora y en cualquier lugar ha sido el principal impulsor del crecimiento de diversas tecnologías entre las cuales destacan los dispositivos móviles. El cómputo móvil aparece con el objetivo de proveer ambientes de cómputo ubicuos para usuarios móviles [BFar, 2005]. Hoy en día, las empresas buscan que sus empleados y usuarios tengan acceso a la información que requieren de forma rápida y sin importar el lugar en el que se encuentren. Es aquí donde los dispositivos móviles han jugado un papel protagónico en las actividades cotidianas de todas las personas, el cual se ha ido incrementando rápidamente en los últimos años sustituyendo en gran medida a las computadoras personales [Crook et al., 2011], esto debido a que es práctico, a su gran capacidad de movilidad y a que facilita el acceso a fuentes de información mediante redes de comunicación inalámbrica. Lo cual es gran riesgo para la

seguridad de toda la información transmitida si ésta no se protege de la manera adecuada.

Los ambientes de cómputo móvil se caracterizan por tener restricciones importantes de recursos y cambios frecuentes en las condiciones de operación [BFar, 2005]. Por otra parte, la rápida evolución, tanto de las capacidades de cómputo como de los sistemas operativos que hoy en día se pueden encontrar en los dispositivos, propicia la creación de todo tipo de sistemas, pensados y diseñados específicamente para funcionar en un dispositivo móvil, planteando a los diseñadores y desarrolladores retos adicionales en la creación de dichos sistemas.

1.2. Planteamiento del problema

En el mundo actual las tecnologías de la información y comunicación forman una parte fundamental en la vida cotidiana de todas las personas, por lo que es de vital importancia hacer conciencia de que a pesar de todos los beneficios que estas tecnologías nos pueden traer, también hay riesgos al realizar cualquier transacción, principalmente si ésta se realiza por un medio no seguro como lo es Internet o una infraestructura inalámbrica. Adicionalmente es importante mencionar que las nuevas tecnologías computacionales marcan una tendencia muy importante en la que día a día la computación va evolucionando de un equipo estático, a equipos móviles, los cuales ingresan a la red mediante infraestructuras inalámbricas lo que supone un gran riesgo en cuanto a seguridad se refiere.

La conectividad de los dispositivos móviles ha generado un crecimiento exponencial en el tráfico de todo tipo de contenido digital a través de redes inalámbricas inseguras, por lo que la protección, autenticación e integridad de esos datos debe ser considerada una prioridad fundamental en el diseño de cualquier sistema informático. El problema se acrecienta en aplicaciones empresariales donde la información que viaja por medios inseguros puede ser sumamente delicada y con un gran valor comercial. Debido a esto, el uso de esquemas criptográficos se vuelve algo fundamental en todo sistema informático si se quiere estar protegido contra diversos ataques.

La infraestructura de clave pública, o PKI, ofrece diversos servicios de seguridad como lo son: integridad, confidencialidad, autenticación, no-repudio. Éstos pueden ser cubiertos asignándole una clave única e infalsificable a cada usuario. De esta manera los usuarios pueden firmar digitalmente la información, el receptor puede verificar dicha firma y comprobar su validez. Adicionalmente una PKI puede ofrecer servicios de cifrado para asegurar la confidencialidad de la información, de manera que únicamente el destinatario original de un mensaje pueda descifrarlo [Kuhn et al., 2001, Schmeh, 2003].

Por otra parte, la Red Privada Virtual (siglas del inglés *Virtual Private Network*) (VPN) se ha convertido en una de las tecnologías más utilizadas. Las principales razones de éste éxito son la reducción de costes en las comunicaciones entre oficinas, la mejora del acceso remoto de teletrabajadores y oficinas remotas, así como de clientes y proveedores. Antes de la proliferación de Internet, si las compañías que querían que las redes de sus

empresas trascendieran más allá del ámbito de la oficina e incluyeran a los trabajadores y centros de información de otros edificios, ciudades, estados o incluso otros países, tenían que invertir en hardware y servicios de telecomunicaciones costosos y proporcionales a las distancias implicadas para crear redes amplias de servicio, además tomando en cuenta el gran crecimiento en el uso de dispositivos móviles como terminales dentro de las empresas, hace pensar que la conexión vía VPN de un dispositivo móvil a las redes empresariales es una idea factible y muy rentable ya que estos dispositivos poseen un costo reducido en comparación con equipos de cómputo personales.

En adición a lo anterior, la siguiente generación de sistemas operativos abiertos no será para equipos de escritorio, sino, para pequeños dispositivos móviles, llevados por los usuarios en cualquier momento. La apertura de estos nuevos ambientes ha llevado a la creación de nuevas aplicaciones y mercados, adicionalmente tales ambientes han permitido una mayor integración con servicios en línea existentes. Sin embargo, a medida que la importancia de los datos comunicados y de los servicios proporcionados por los dispositivos crece, también lo hacen las vulnerabilidades [Enck et al., 2009]. Debido a esto, es esencial contar con mecanismos que ayuden a garantizar la seguridad de los datos intercambiados en cualquier transmisión entre el dispositivo móvil y el servidor, éste es el principal tema del presente trabajo, en el cual se propone implementar una PKI empresarial la cual será diseñada y desarrollada en el ambiente de dispositivos móviles, particularmente utilizando *Android* como Sistema Operativo (SO) para el desarrollo de esta tesis, debido a su gran popularidad la cual asciende a las 300,000 activaciones diarias colocándolo como el sistema operativo más popular durante el cierre del 2010 y 2011 [Crook et al., 2011].

Es importante mencionar que la PKI creada toma en cuenta el uso de una conexión entre el dispositivo móvil y la empresa, de tal manera que el dispositivo podrá acceder a fuentes de información que se encuentren dentro de la misma, las cuales pueden contener información sensible, por lo que la PKI contempla todos los servicios de seguridad, los cuales serán explicados a detalle en el siguiente capítulo.

Para el desarrollo de esta tesis se utilizó un enfoque de software libre, por lo que se analizaron y probaron diversos desarrollos de PKIs existentes, con la finalidad de observar las fortalezas y debilidades de estos desarrollos y obtener como resultado una PKI eficiente para *Android* con de la cual se puedan realizar comunicaciones seguras entre un servidor y sus clientes con dispositivos móviles.

Como se mencionó anteriormente una de las principales características de los dispositivos móviles, es que poseen capacidades limitadas tanto de cómputo como de energía, por lo que estas restricciones fueron consideradas en la selección de la Interfaz para el desarrollo de aplicaciones (siglas del inglés *Application Programming Interface*) (API), de manera que la PKI resultante es eficiente y útil en la creación de aplicaciones móviles seguras. Ya que como se sabe, un sistema criptográfico es como una cadena: "tan fuerte como su eslabón más débil".

Debido a lo anterior se plantean los siguientes requerimientos básicos, los cuales tienen una mayor relación con el ambiente de desarrollo móvil, estos son complementados

en la sección 3.2.1 con los requerimientos de seguridad y certificación de la PKI.

- La PKI estará diseñada e implementada en un ambiente de dispositivos móviles, por lo que es necesario que se tomen en cuenta las siguientes características de dichos dispositivos:
  - Recursos limitados en los dispositivos principalmente de procesamiento y almacenamiento.
  - Acceso a redes móviles e inalámbricas no seguras.
  - Tamaños de pantalla reducidos
  - Conciencia de la ubicación.
  - Soporte de una gran variedad de interfaces de usuario (voz, sensores, pantalla, entre otros).
- La implementación de la PKI se hará en *Android*.
- Como parte de la PKI implementada se deberá proporcionar una API para el desarrollo de aplicaciones.
- La API creada deberá contener toda la funcionalidad de la PKI.
  - La API generada deberá permitir tanto usuarios expertos como a usuarios con conocimientos básicos de seguridad hacer uso completo de la funcionalidad ofrecida en el mismo.
  - Debido a lo anterior la API deberá contar con funciones sencillas, con las cuales no se requiera de conocimientos muy avanzados en seguridad para poder explotar al máximo las capacidades de la PKI.
  - Por otra parte, también se requiere que existan funciones más complejas, con las cuales un usuario con conocimientos más avanzados pueda personalizar los parámetros de entrada de los algoritmos y de esta manera se pueda obtener el funcionamiento deseado de la PKI.

1.3. Objetivos del trabajo de investigación

Como punto de partida del trabajo de tesis se formularon los siguientes objetivos que han sido clasificados en generales y específicos.

1.3.1. Objetivo general

Diseñar e implementar una PKI que contemple los servicios de seguridad básicos (integridad, autenticación, confidencialidad y no-repudio) en el diseño de la misma. Adi-

cionalmente, es de vital importancia pensar en un ambiente de usuarios móviles a lo largo del desarrollo y diseño de la PKI, los cuales utilizan dispositivos restringidos como lo son teléfonos inteligentes y tablets, particularmente utilizando *Android* como plataforma de desarrollo y pruebas.

Utilizar herramientas y APIs de software libre para el desarrollo de la PKI, por lo que es necesario llevar a cabo un análisis detallado de las opciones existentes.

Proporcionar como parte de la PKI un API para el desarrollo de aplicaciones que requieran servicios de criptografía para garantizar comunicaciones seguras. Al finalizar la construcción de la API crear una aplicación demostrativa en la cual se expongan las fortalezas de la misma.

1.3.2. Objetivos específicos

1. Implementar una PKI, principalmente el estándar X.509, de acuerdo con estándares y convenciones actuales.
2. Generar certificados compatibles con X.509, Pretty Good Privacy (PGP) y/o PKCS utilizando la PKI generada.
3. Utilizar APIs y desarrollos de software libre que proporcionen una base sólida en cuanto a algoritmos criptográficos de clave pública.
4. Generar una API en *Android* para el desarrollo de aplicaciones seguras, que ofrezca entre otros, generación de certificados digitales, asignación de permisos de acceso en base a una estructura jerárquica, cifrado y firma digital.
5. Validar la funcionalidad de la PKI mediante la implementación de una aplicación en *Android* que utilice la API generada, esta aplicación deberá permitir al usuario proveer autenticación de usuarios móviles, ofrecer servicios de firma y verificación, así como cifrado y descifrado de archivos o mensajes dentro del dispositivo, para que estos puedan ser enviados a través de redes no seguras.

1.4. Metodología

1. *Construcción del estado del arte*: en esta sección se llevó a cabo la revisión de diferentes fuentes de información como artículos de revista, proyectos de software libre y libros relacionados con PKI , así como sus diferentes implementaciones, con el fin de establecer la base teórica sobre la que se sustenta el trabajo y el estado actual de las investigaciones en esta área. Al finalizar esta etapa se obtuvo un análisis comparativo de diferentes diseños e implementaciones.
2. *Elección de proyectos base y APIs a utilizar*: una vez que se tuvo el listado y las características de las APIs y proyectos disponibles se realizó un estudio a detalle de

éstas, para observar las ventajas y desventajas de cada una y así elegir las opciones que más se adaptaron al enfoque deseado para ésta tesis.

3. *Elección y diseño del esquema PKI*: con la base criptográfica elegida se estudiaron los diferentes enfoques existentes para la implementación de una PKI, con la finalidad de diseñar un esquema que se adaptara a las características de movilidad deseadas en esta tesis.

4. *Pruebas de funcionamiento y posibles adaptaciones*: posteriormente se pasó a una fase de adaptación de los desarrollos elegidos previamente, en esta fase se realizaron las modificaciones necesarias para que el funcionamiento dentro de *Android* fuera el mejor posible. En esta fase se realizaron pruebas tanto en un simulador como en diversos dispositivos móviles para garantizar que se obtuviera la funcionalidad deseada con un rendimiento óptimo.

5. *Creación del API para la PKI* :utilizando los bloques seleccionados en las fases anteriores y siguiendo el diseño realizado se implementó un API de desarrollo para facilitar la construcción de aplicaciones en *Android* con las capacidades de la PKI creada. Cada uno de los componentes fue probado por separado utilizando pruebas unitarias para garantizar su correcto funcionamiento.

6. *Creación de aplicación de prueba*: posteriormente se desarrolló una aplicación en *Android* la cual sirvió como modelo para probar la funcionalidad y facilidad de uso de la API construida, esta aplicación contiene interfaces que le permitan al usuario explotar al máximo las capacidades de la API dentro de un ambiente móvil.

7. *Validación de la aplicación*: finalmente se estableció una plataforma de pruebas controladas, que permitieron evaluar el correcto funcionamiento y desempeño del trabajo realizado.

1.5. Organización de la tesis

La tesis consta de seis capítulos, los cuales están organizados de la siguiente manera:

El capítulo 2 contiene un marco teórico general de las PKI, así como la revisión del estado del arte. En este capítulo se presenta la definición y los conceptos básicos de las PKI: servicios de seguridad, componentes básicos, arquitecturas y esquemas criptográficos básicos. Adicionalmente se destacan las características deseables en una PKI, a partir de las cuales se realiza un estudio comparativo con los diversos esquemas e implementaciones analizados.

En los siguientes capítulos se encuentra la información que refleja los principales aportes de esta tesis. En el capítulo 3 se describe la estructura y requerimientos de la PKI a implementar así como la declaración de políticas de certificación que deberán seguir los usuarios, estas políticas serán descritas conforme al estándar "*ETSI*

TS 101 042: Policy requirements for certification authorities issuing public key certificates"[Barreira and Compans, 2011]. Estas políticas establecen entre otras cosas, el uso que se le debe dar a los certificados, los procesos internos a seguir para emitir, revocar, solicitar y almacenar certificados, así como características particulares de los certificados.

El capítulo 4 presenta en detalle la implementación de la PKI y la API, así como diversos diagramas UML que describen la funcionalidad implementada. En este capítulo se encuentran los aspectos de diseño y la descripción de los principales módulos de la API. Al final de este capítulo presentamos, en las tablas 4.9–4.13, los tiempos de ejecución de los módulos criptográficos principales utilizados por la API construida. Los tiempos demuestran que todas las operaciones criptográficas pueden llevarse a cabo dentro del dispositivo móvil sin mayores retrasos.

A continuación, en el capítulo 5 se describen los detalles de la implementación de una aplicación de prueba sobre *Android*, que se construyó para verificar el correcto funcionamiento de la PKI y la API previamente explicadas, esta sección contiene los aspectos de diseño y tiempos de ejecución obtenidos al probar la aplicación en diversos dispositivos móviles con diferentes características entre ellos.

Finalmente, el capítulo 6 provee las conclusiones acerca de este trabajo, así como algunas ideas de trabajo a futuro que podrían mejorar o ampliar la funcionalidad ofrecida por la PKI.

Capítulo 2

Marco general de una PKI y estado del arte

El uso de una PKI puede acelerar y simplificar diversos procesos y servicios dando enfoques electrónicos seguros a procesos que dependan de la integridad y autenticidad de los datos. Ambos se pueden cumplir ligando una firma digital única a un individuo garantizando que ésta no puede ser falsificada. De esta manera un individuo podría firmar datos y enviarlos por un canal no seguro, mientras que el receptor de los mismos podría verificar al emisor y que el contenido del mensaje no fue modificado. Adicionalmente una PKI puede proveer servicios de cifrado para garantizar, también, la privacidad de los datos.

Así como en diversos aspectos de la seguridad informática, el entendimiento y uso de una PKI requiere de algunos conocimientos previos, los cuales se plantean a lo largo del presente capítulo, en el que se presenta una breve descripción de los principales mecanismos de seguridad y posteriormente una introducción a los conceptos principales de una PKI, para finalizar con la revisión del estado del arte referente a implementaciones realizadas para *Android* de una PKI.

2.1. Mecanismos criptográficos de seguridad

Como introducción a este capítulo se definirán los principales mecanismos y servicios de seguridad que la criptografía tiene para ofrecer a sus usuarios. Se describirán brevemente la criptografía de clave simétrica, las funciones picadillo (*hash functions*) y la criptografía de clave asimétrica.

2.1.1. Servicios de seguridad

Los cuatro servicios básicos de seguridad que toda PKI debe ofrecer [Kuhn et al., 2001, Schmeh, 2003] son:

Integridad: este servicio está dirigido a evitar la modificación no autorizada o accidental de la información. Esto incluye la inserción, la eliminación y la modificación de los datos originales. Para asegurar la integridad, el sistema debe ser capaz de detectar modificaciones no autorizadas. La meta principal es que el receptor sea capaz de verificar si acaso la información fue alterada durante la comunicación.

Confidencialidad: este servicio restringe el acceso a la información sensible únicamente a los usuarios que estén autorizados a consultarla. Impide la revelación no autorizada de información sensible de una empresa o persona.

Autenticación: establece la validez del mensaje y de los participantes en la comunicación. La meta es permitir a los participantes de la comunicación determinar si el origen del mensaje es válido o no.

No-repudio: ofrece protección a un usuario o entidad frente a la posibilidad de que otro usuario niegue posteriormente que se realizó cierta transacción, por ejemplo, haber hecho un pedido a una empresa.

2.1.2. Criptografía simétrica

Los algoritmos de criptografía simétrica son aquellos donde dos partes, Alicia y Beto, comparten una clave secreta, son utilizados primordialmente para lograr confidencialidad y son altamente eficientes y seguros. En la figura 2.1 se puede observar el proceso básico de un algoritmo simétrico, Alicia transforma el texto en claro en texto cifrado utilizando el algoritmo de cifrado y una clave, por su parte Beto transforma el texto cifrado en texto en claro utilizando el algoritmo de descifrado y la misma clave utilizada por Alicia. De manera similar estos algoritmos pueden ser utilizados para autenticar y verificar la integridad de los datos, para esta tarea Alicia usa la clave para generar un texto cifrado, manda el texto cifrado y el texto en claro a Beto, éste utilizando la clave secreta compartida previamente vuelve cifrar el texto en claro y si los textos cifrados coinciden, Beto puede saber que el texto no fue alterado y que fue Alicia la que envío el mensaje [Kuhn et al., 2001].

A pesar de la gran eficiencia y seguridad de este tipo de algoritmos, tienen una gran desventaja, ya que para que el proceso se lleve a cabo correctamente, es necesario que Beto y Alicia compartan una clave secreta previamente, lo cual puede llegar a ser un problema, ya que si la clave secreta es comprometida, todo el proceso lo estará.

2.1.3. Funciones "picadillo"

Las funciones picadillo, o funciones *hash*, toman un flujo de datos y lo reducen a un tamaño fijo a través de funciones matemáticas. Al resultado de esta operación se le conoce como picadillo y puede ser visto como una huella digital de los datos. Este picadillo puede ser reproducido por cualquier parte que tenga el flujo original, sin embargo es virtualmente imposible crear un mismo picadillo de dos flujos diferentes [Kuhn et al., 2001]. Un picadillo

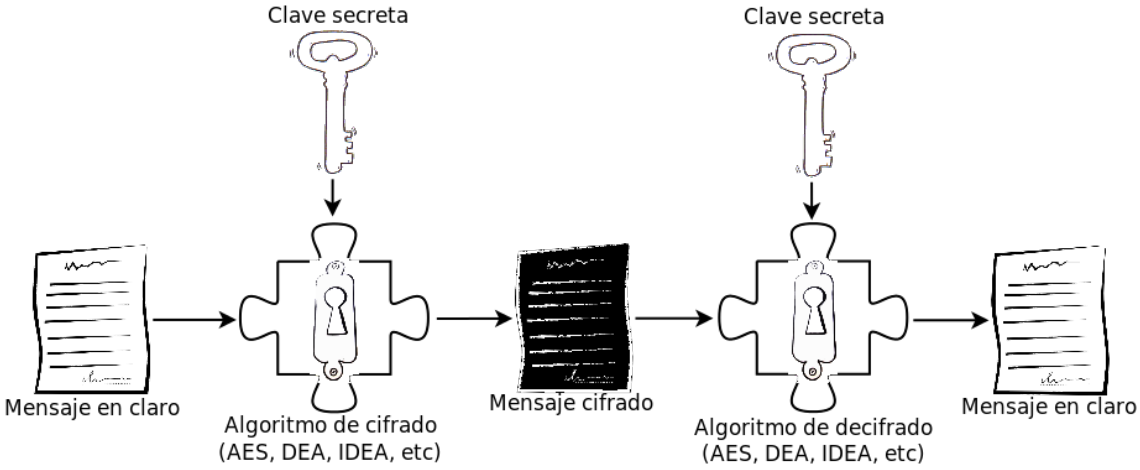


Figura 2.1: Esquema básico de criptografía simétrica.

puede ser usado para proveer integridad. Como se muestra en la figura 2.2 un mensaje entra a la función picadillo y como salida de ésta será una serie de bits aleatorios de tamaño fijo en función al mensaje de entrada.

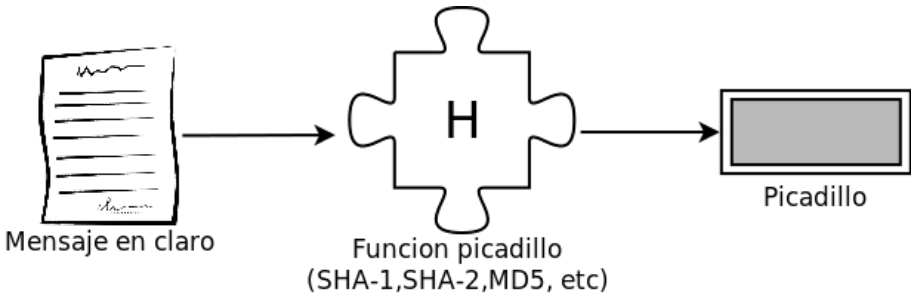


Figura 2.2: Esquema básico de funciones picadillo.

Una noción de estas funciones es la de Código de Autenticación de Mensajes (siglas del inglés *Hash-based Message Authentication Code*) (HMAC), siempre y cuando Alicia y Beto compartan una clave secreta. Si Alicia envía un mensaje a Beto junto con su HMAC, el puede reconstruir el HMAC para verificar que se no hayan hecho cambios en los datos de ningún tipo, ya que si por ejemplo Carlos intercepta el mensaje de Alicia y reemplaza éste con un nuevo mensaje, no hay forma de que genere un HMAC correcto sin conocer la clave secreta de Beto y Alicia. Adicionalmente Beto podría aceptar el HMAC como un mecanismo de autenticación de la identidad de Alicia, ya que en teoría únicamente ellos conocen la clave secreta para generar el código [Kuhn et al., 2001]. En la figura 2.3 se puede observar que el proceso para generar es muy similar al proceso de generar un picadillo normal, ya que únicamente se agrega la clave secreta como entrada a la función picadillo. Es importante mencionar que se pueden utilizar las funciones picadillo estándar para generar un HMAC [Eastlake and Hansen, 2011].

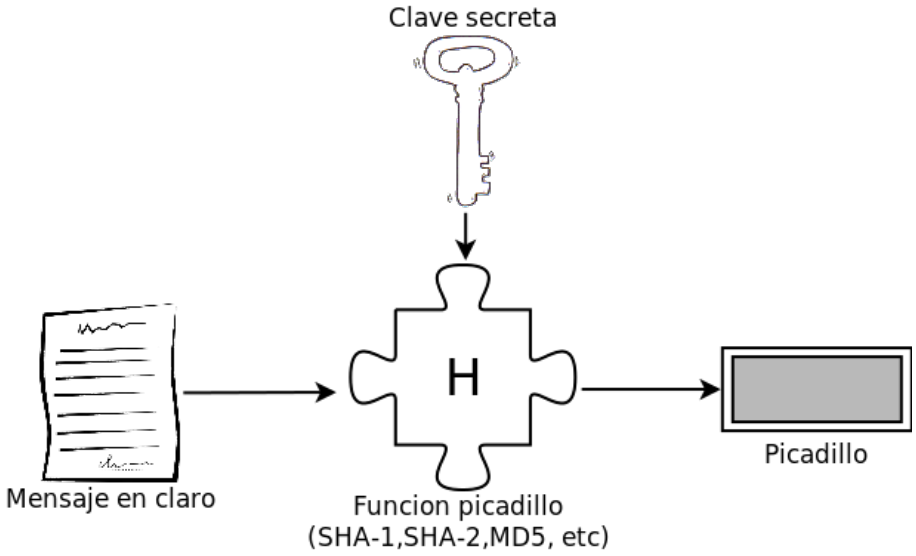


Figura 2.3: Esquema básico de funciones HMAC.

2.1.4. Criptografía asimétrica

La criptografía asimétrica también es conocida como criptografía de clave pública, este tipo de criptografía utiliza algoritmos en los cuales cada uno de los participantes tiene un par de claves, una pública y una privada, la clave pública es distribuida y todos la conocen. Se basa en que los datos cifrados con una clave, pueden ser descifrados utilizando la otra. Estos algoritmos no son la mejor opción para procesar mensajes muy largos, ya que son relativamente lentos, sin embargo estos algoritmos son utilizados para proveer autenticación, integridad, no-repudio y confidencialidad, utilizando principalmente dos operaciones [Kuhn et al., 2001].

Cifrado

El proceso de cifrado se realiza utilizando la clave pública del receptor del mensaje, ésto se puede observar en la figura 2.4 en la cual, Beto quiere cifrar un mensaje para Alicia, para lo que obtiene su clave pública y utilizando el algoritmo de cifrado genera el texto cifrado, posteriormente Beto transfiere el mensaje cifrado a Alicia y ésta, utilizando su clave privada y el algoritmo de correspondiente descifra el mensaje.

Como se ha mencionado anteriormente los algoritmos de clave pública no son recomendables para cifrar grandes cantidades de datos, ya que son lentos en comparación con los algoritmos simétricos, sin embargo pueden ser utilizados para transportar de manera segura claves simétricas [Kuhn et al., 2001] o en protocolos de intercambio de llaves (por ejemplo el protocolo Diffie-Hellman [Diffie and Hellman, 1976]).

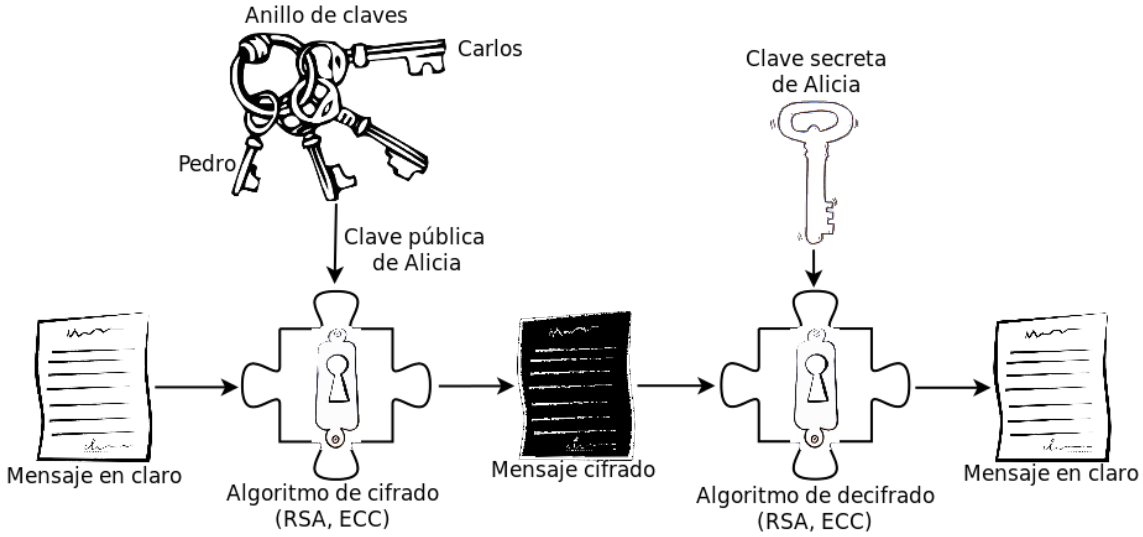


Figura 2.4: Esquema básico de criptografía asimétrica - Cifrado.

Firma digital

De la manera más general, Alicia puede generar una firma digital utilizando su clave privada y un flujo de datos, por su parte Beto puede verificar la firma de Alicia utilizando el mensaje original y la clave pública de Alicia, si una clave privada diferente fue utilizada para generar la firma, la verificación fallara. Este proceso se puede observar en la figura 2.5.

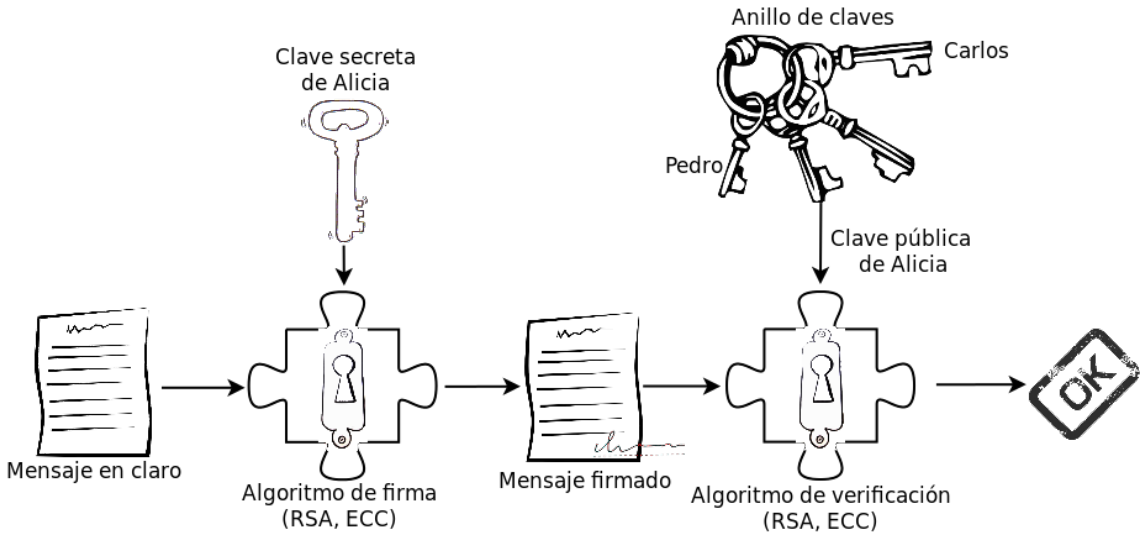


Figura 2.5: Esquema básico de criptografía asimétrica - Firma Digital.

En contraste con las firmas manuscritas, las firmas digitales proveen también integridad de los datos, ya que si éstos han sido modificados, la verificación sera incorrecta. Por otra parte, también pueden proveer no-repudio ya que (en teoría) únicamente Alicia tiene acceso a su clave privada, por lo que no puede alegar que ella no generó la firma.

Otro de los servicios que pueden proveer las firmas digitales, es la autenticación mediante protocolos de reto-respuesta, en los cuales, el sistema genera un reto aleatorio, Alicia firma el reto y si la firma es verificada de manera correcta, el sistema puede estar seguro de que es Alicia con quien se está comunicando.

2.1.5. Resumen de mecanismos de seguridad

Los mecanismos de seguridad son utilizados para proveer el conjunto completo de servicios de seguridad, ya que cada tipo de algoritmo posee sus fortalezas y debilidades.

Los algoritmos simétricos, como el Estándar Avanzado de Cifrado (siglas del inglés *Advanced Encryption Standard*) (AES), son utilizados principalmente para brindar confidencialidad. Sin embargo pueden ser utilizados para proveer cierto grado de integridad y autenticación, sin embargo no es lo mejor para hacerlo. Sin duda alguna el punto débil de este tipo de algoritmos es la distribución de claves en un sistema.

Por otra parte las funciones picadillo y las HMAC proporcionan las bases para proveer integridad de datos, a su vez son una forma muy básica para la autenticación, por otra parte estas funciones no proveen de ninguna manera confidencialidad.

Finalmente los algoritmos de criptografía asimétrica son altamente recomendables para ofrecer integridad, autenticación y no repudio mediante el uso de las firmas digitales, mientras que es posible utilizar cifrado para la confidencialidad aun que esto no es muy recomendable ya que estos algoritmos son lentos en comparación con los algoritmos simétricos.

Lo anterior se puede resumir en la tabla 2.1, en la cual se muestra el listado de los mecanismo de seguridad explicados en esta sección y los servicios de seguridad que cada uno puede proveer.

Mecanismo		Integridad	Confidencialidad	Autenticación	No-repudio
Criptografía simétrica	Cifrado	×	✓	×	×
Funciones picadillo	Picadillo	✓	×	×	×
	HMAC	✓	×	✓	×
Criptografía asimétrica	Firma Digital	✓	×	✓	✓
	Cifrado	×	✓	×	×

Tabla 2.1: Servicios de seguridad por mecanismo.

2.2. Infraestructura de clave pública (PKI)

Una vez explicados los mecanismos y servicios básicos a considerar en una PKI, en esta sección se describen los conceptos y fundamentos básicos que son necesarios para tener un buen entendimiento de la infraestructura de clave pública. Se presenta la definición, así como los componentes básicos y principales estructuras de cualquier PKI.

2.2.1. Clave pública

La criptografía de clave pública fue propuesta por Diffie y Hellman en 1976 en su célebre artículo "New Directions in Cryptography" [Diffie and Hellman, 1976], en el cual se propone, para cada usuario, un par de claves (pública y privada) y su utilidad para demostrar la posesión de un secreto (clave pública) mediante alguna operación entre los datos y la clave pública. Por sí misma, la criptografía de clave pública únicamente provee una serie de operaciones matemáticas asimétricas, pero no ofrece una conexión a aplicaciones o ambientes. Para establecer esta conexión se necesitan piezas o componentes, las cuales forman la PKI, una "infraestructura" que hace que la tecnología de clave pública esté disponible a aplicaciones y ambientes que deseen utilizarla [Adams and Just, 2004].

Desde sus inicios, han surgido diversas implementaciones de PKIs y en dichas implementaciones existen variaciones en la forma de definir cada uno de los componentes que conforman una infraestructura, principalmente por la forma de ligar la identidad del usuario de la PKI con su clave pública, ya que las partes que utilicen la criptografía de clave pública dependerán de esta liga para asociar la clave con una entidad. Diffie y Hellman propusieron un modelo en el cual las claves públicas son entregadas por un repositorio seguro, pero no fue hasta 1978 que en [Kohnfelder, 1978] la definición de certificado fue propuesta, donde la clave pública y el identificador son almacenados en una estructura de datos y firmados por Autoridad de Certificación (siglas del inglés *Certification Authority*) (CA).

De acuerdo con Kuhn [Kuhn et al., 2001] una PKI es aquella que se encarga de ligar una clave pública con una entidad, permitiendo así a otras entidades que verifiquen esta liga, adicionalmente provee los servicios necesarios para la administración y distribución de claves. Una PKI permite a sus usuarios realizar diversas operaciones, con la confianza de que:

- La persona o proceso identificado como emisor de una transacción es realmente quien dice ser.
- La persona o proceso receptor es el receptor deseado.
- La integridad de los datos no ah sido comprometida.

Por lo que la infraestructura de clave pública o PKI se puede definir como la combinación de software, tecnologías de cifrado y servicios que permiten a sus usuarios proteger sus comunicaciones y transacciones. Una PKI integra firmas digitales, criptografía de clave pública y autoridades certificadoras en una estructura de seguridad completa. Típicamente una PKI contempla la emisión de certificados digitales, registro de usuarios, herramientas para el manejo, renovación y revocación de certificados y los servicios que soporten estas operaciones [Kuhn et al., 2001].

2.2.2. Componentes básicos

Una PKI contiene diversos componentes funcionales para cumplir con las funciones deseadas, a continuación se describirá cada uno de ellos, las relaciones entre estos elementos quedan bosquejadas en la figura 2.6 [Cooper et al., 2008].

Autoridad de Certificación (siglas del inglés *Certification Authority*) (CA) La autoridad certificadora o CA es el bloque básico de toda PKI, es la colección de hardware, software y las personas que los operan. La CA es conocida por dos cosas: su nombre y su clave pública. Su deber es proveer cuatro funciones básicas [Kuhn et al., 2001]:

- Emitir certificados.
- Mantener información del estatus de los certificados y emitir Lista de Revocación de Certificados (siglas del inglés *Certificate Revocation List*) (CRL).
- Publicar los certificados emitidos y las CRL.
- Mantener el estatus y la información de los certificados expirados.

Estas funciones pueden ser delegadas a terceras partes autorizadas por la CA. Una CA puede emitir certificados a usuarios o a otras CAs y su deber es verificar la identidad del titular sea correcta y en caso de ser necesario (cuando el par de claves no es generado por la CA), que el titular tiene posesión de la clave privada correspondiente a la clave pública incluida en el certificado [Cooper et al., 2008].

La CA debe insertar su nombre y firma en cada certificado emitido. Para verificar que un certificado es genuino, los usuarios deben verificar esta firma usando la clave pública de la CA.

Autoridad de Registro (siglas del inglés *Registration Authority*) (RA) Es la designada a verificar el contenido de un certificado para la CA, dicha información puede ser presentada por el solicitante del certificado o puede ser información obtenida en una fuente de externa. Cada CA debe mantener la lista de las RA autorizadas [Kuhn et al., 2001].

Repositorio Es un sistema o colección de sistemas distribuidos que se encargan de almacenar certificados y CRLs, con el objetivo de distribuir estos datos a los usuarios [Cooper et al., 2008].

Usuarios de la PKI Son usuarios o individuos que usan la PKI, pero no emiten certificados. Ellos deben confiar en los demás componentes de la PKI para que ésta opere de manera correcta. Los usuarios deben confiar en que la identidad contenida en un certificado es válida, siempre y cuando este certificado esté firmado por la CA [Kuhn et al., 2001].

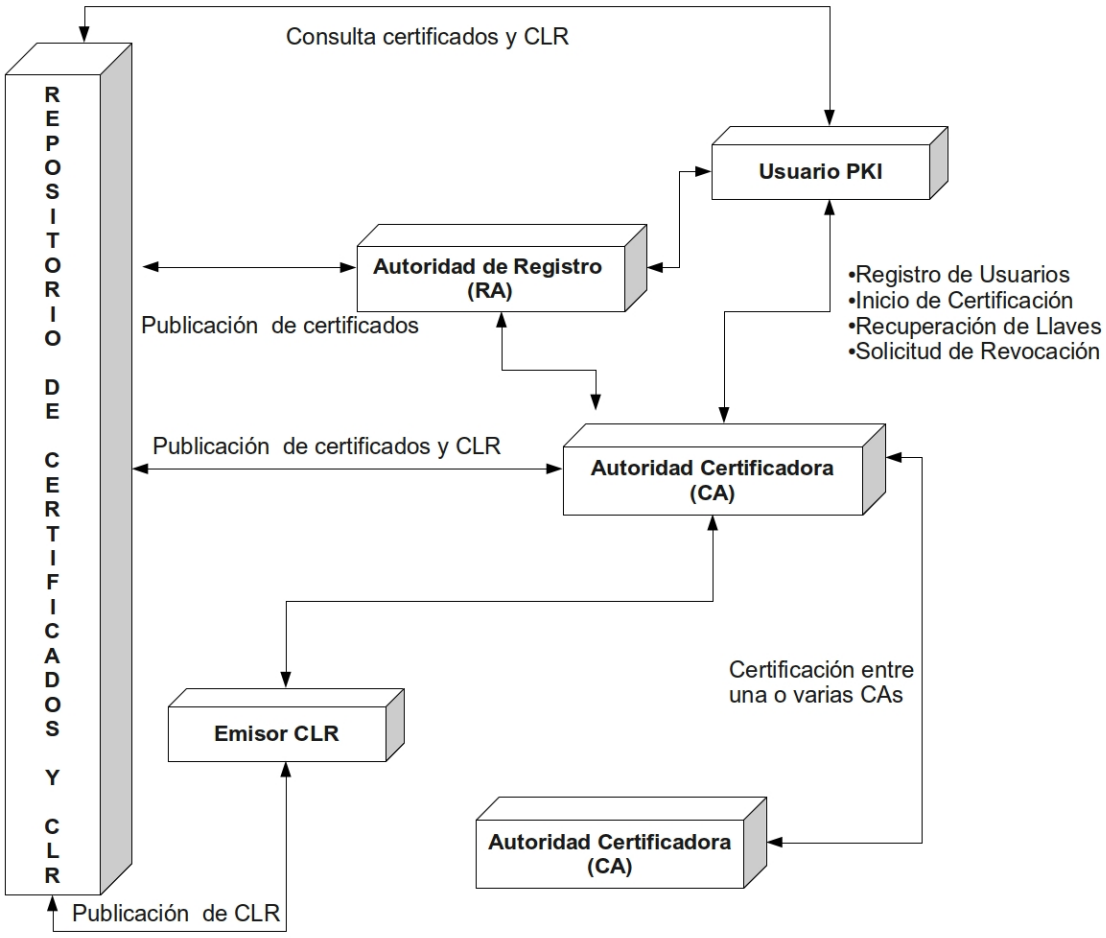


Figura 2.6: Relación entre componentes de una PKI.

2.2.3. Estructuras de datos

Dos tipos de estructuras de datos básicas son utilizadas en una PKI, estas estructuras son los certificados de claves públicas y las Lista de Revocación de Certificados (siglas del inglés *Certificate Revocation List*) (CRL). En esta sección se explicarán a detalle estas estructuras.

Certificado Digital

Un certificado de clave pública, es archivo digital que contiene la clave pública de un individuo y está firmado por una entidad de confianza (comúnmente conocida como CA), los certificados son utilizados para evitar la suplantación de la identidad y se pueden encontrar en repositorios públicos. Un certificado no contiene únicamente la clave pública del individuo y la firma de la autoridad de confianza, si no que, contiene una serie de información adicional, como puede ser la dirección, nombre, organización, entre otros de tal manera que se pueda ligar la identidad del individuo con la clave pública. Esta información debe ser verificada al momento de emitir el certificado por la CA [Schneier, 1996].

En esta tesis se hace uso de certificados digitales X.509, los cuales han evolucionado en un mecanismo de almacenamiento de información muy flexible y poderoso, mucha de esta información es opcional, sin embargo también existen campos obligatorios señalados en los diferentes estándares, en esta tesis se sigue el estándar presentado en [Cooper et al., 2008].

Los certificados X.509 se encuentran protegidos por las firmas digitales del emisor, de esta manera los usuarios de una PKI pueden validar que la información contenida en éste no ha sido alterada ya que la firma puede ser verificada en cualquier momento. La versión 3 del estándar señala que comúnmente existen seis campos obligatorios y cuatro opcionales. Los campos obligatorios son: número de serie, identificador del algoritmo de firmado, nombre del emisor, periodo de validez, clave pública, nombre del titular. Mientras que los campos opcionales son: versión, identificadores únicos y las diversas extensiones [Kuhn et al., 2001, Cooper et al., 2008] estos campos se pueden ver gráficamente en la figura 2.7. A continuación se describe brevemente cada uno de estos campos:

- Versión: Describe la versión del certificado, si este campo es omitido, se considera que el certificado fue codificado utilizando la versión 1 del estándar.
- Número de Serie: Es un entero asignado por el emisor del certificado, debe ser único para cada certificado generado por un emisor en particular, de manera que la combinación del nombre del emisor y el número de serie pueda identificar de manera única a cualquier certificado.
- Firma: Este campo indica el algoritmo de firma utilizado (por ejemplo `RSAwithMD5`) para proteger el certificado.
- Emisor: Contiene el nombre distinguido (en formato X.500) del emisor del certificado.

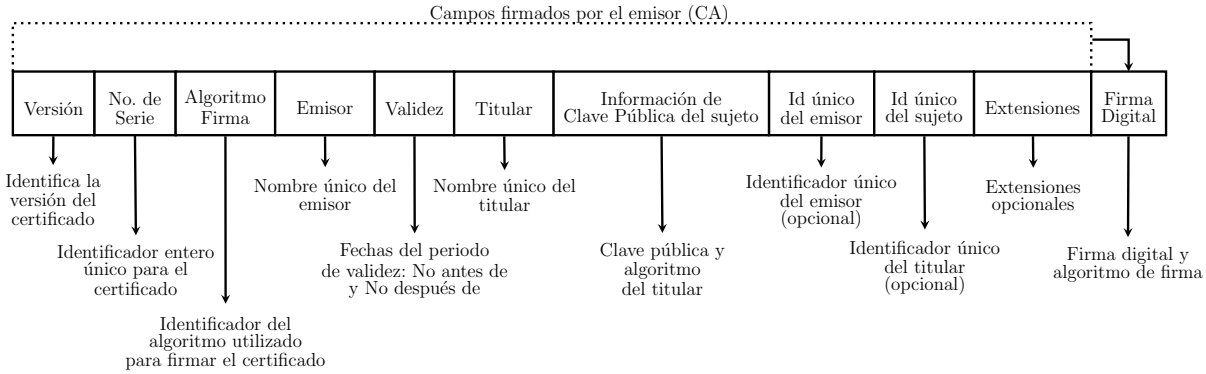


Figura 2.7: Estructura de un certificado X.509.

- Validez: Indica la fecha a partir de la cual el certificado es válido y la fecha en la cual expira.
- Titular: Contiene el nombre distinguido (en formato X.500) del poseedor de la clave privada correspondiente a la clave pública almacenada en el certificado.
- Información de la clave pública: Este campo almacena información acerca de la clave pública como los parámetros opcionales y el identificador del algoritmo. La clave pública de este campo junto con la información adicional es utilizada por los algoritmos de criptografía asimétrica para verificar una firma digital o cifrar un conjunto de datos. Si el certificado corresponde a una CA, esta clave es utilizada para verificar la firma en otros certificados.
- Id único de titular y emisor: Estos campos contienen identificadores únicos de las claves de estas entidades, únicamente aparece en las versiones 2 y 3 del estándar. El objetivo es permitir el reuso de nombres a lo largo del tiempo.
- Extensiones: Este campo únicamente aparece en los certificados versión 3. Si se encuentra presente puede contener una o más extensiones, cada una incluye un identificador, una bandera para indicar si es critica o no y el valor de la extensión. Las extensiones comunes han s sido definidas por ISO y ANSI.

Una forma de definir verificar el estatus de un certificado, es la construcción de una cadena de certificación, la cual es construida a partir del certificado final hasta llegar a un certificado raíz que es de confianza para el usuario. En la figura 2.8 se puede ver una cadena de certificación de ejemplo la cual tiene dos niveles a partir del certificado raíz y hasta llegar al certificado del usuario final.

CRL

A pesar de que un certificado contiene un periodo de validez, existe la posibilidad de que la información contenida en éste se vuelva poco fiable antes de que la fecha de expiración llegue, por lo que es necesario contar con un mecanismo que actualice el estatus

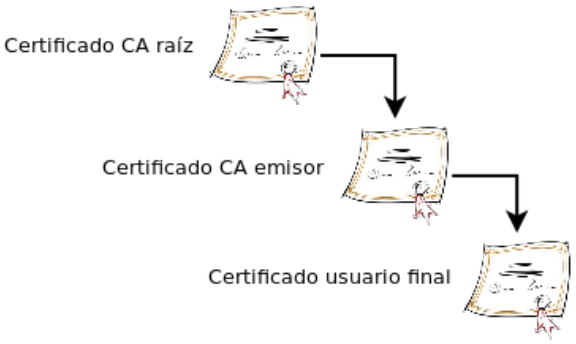


Figura 2.8: Cadena de certificación de dos niveles.

de los certificados emitidos. Uno de estos mecanismos son las Lista de Revocación de Certificados (siglas del inglés *Certificate Revocation List*) (CRL) [Kuhn et al., 2001].

Al igual que los certificados digitales, las CRL están protegidas por la firma digital del emisor, de manera que cualquier usuario puede verificar que el contenido de la misma no ha sido alterado desde su generación.En la figura 2.9 se puede observar gráficamente la estructura de una CRL, la cual contiene los siguientes campos [Kuhn et al., 2001, Cooper et al., 2008]:

- Versión: Campo opcional que señala la sintaxis de la lista.
- Firma: Contiene el algoritmo utilizado para firmar la CRL.
- Emisor: Nombre distinguido del emisor de la CRL en formato X.500.
- Fecha de emisión: Fecha en la cual fue emitida la lista.
- Siguiente actualización: Fecha que indica la liberación de la próxima CRL.
- Lista de certificados revocados: Lista estructurada de certificados revocados, cada entrada en la lista contiene el numero de serie del certificado, la fecha de la revocación y extensiones opcionales.
- Extensiones: Campo utilizado para dar información extra acerca de una CRL.

2.2.4. Modelos de Confianza

Cuando se verifica un certificado, la aplicación que realiza la verificación no conoce a la CA que emitió el certificado, por lo que es necesario introducir el concepto de *Confianza*, la cual es una palabra complicada, ya que puede significar tantas cosas en tantos contextos diferentes. Sin embargo, en este caso se utilizará la definición estricta de la palabra. La *Confianza* es el mecanismo utilizado para decidir si un certificado es válido o no [Callas, 2008]. Los modelos de confianza (también conocidos como estructuras de confianza o arquitecturas) de una PKI son modelos utilizados para determinar el nivel de confianza que se tiene en un certificado determinado y pueden variar desde una relación de confianza directa o

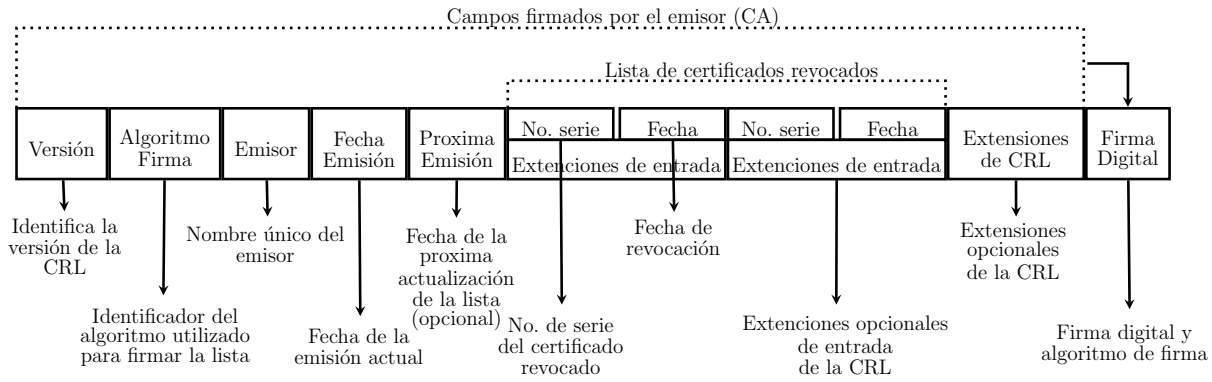


Figura 2.9: Estructura de una CRL.

una estructura jerárquica simple hasta arquitecturas de malla muy complicadas. El tipo de arquitectura seleccionada por la PKI define el tipo de camino de certificación, que se debe construir y validar por las aplicaciones [Cooper et al., 2005, Callas, 2008]

Confianza directa

El modelo más sencillo, sin duda es la confianza directa (figura 2.10), en la cual un usuario confía en el certificado de otro, ya que, éste último le proporcionó su certificado personalmente. Ésta es la mejor estructura de confianza y sin duda alguna el más utilizado en la vida cotidiana, por ejemplo, al entregar una tarjeta de presentación y al dar un correo electrónico [Callas, 2008].

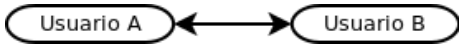


Figura 2.10: Ejemplo de una estructura jerárquica.

El problema con este modelo, es que no puede ser escalable a algo como Internet en donde no es posible que todos los usuarios se conozcan entre sí.

Jerárquico

Uno de los modelos más comunes y utilizados es sin duda el modelo jerárquico, en la cual las entidades se encuentran bajo una CA raíz. La jerarquía puede tener diferentes niveles, esto se presenta cuando una CA emite certificados a CAs intermedias, también conocidas como subordinadas. En la figura 2.11 se puede ver un ejemplo de este modelo, donde una CA raíz, emite certificados a 3 CA intermediarias y éstas a su vez emiten certificados a usuarios finales o a otras CA [Cooper et al., 2005, Kuhn et al., 2001].

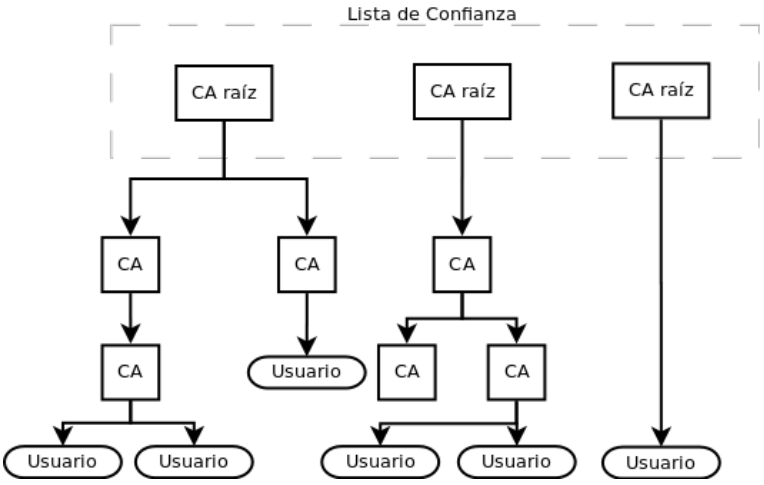


Figura 2.12: Ejemplo de un modelo jerárquico múltiple.

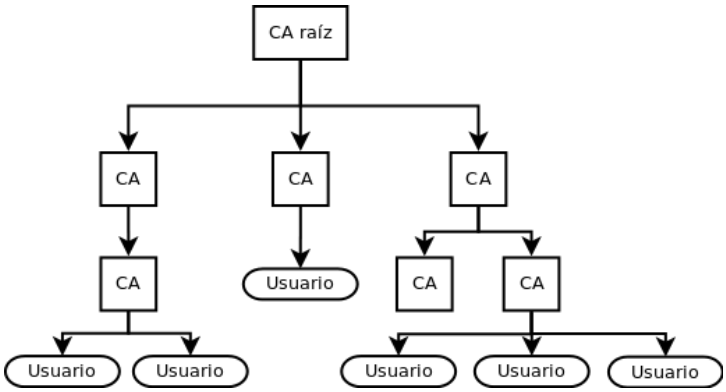


Figura 2.11: Ejemplo de un modelo jerárquica.

En este modelo, una CA no puede emitir certificados para otras CA que se encuentren en un nivel superior en la jerarquía. Típicamente, sólo una CA superior puede certificar cada CA subordinada. Adicionalmente cualquier certificado puede ser verificado validando sucesivamente los certificados hasta llegar a la CA raíz.

Una variación al modelo jerárquico clásico se muestra en la figura 2.12, en este caso, se tienen varias CA raíz, que a su vez pertenecen a un lista de confianza. El modelo jerárquico se mantiene, el cambio principal es al momento de verificar un certificado, ya que al final de la ruta de validación se tienen que comprobar todas las CA pertenecientes a la lista de confianza [Cooper et al., 2005].

Malla

En este modelo cada entidad confía en su propia CA y no existe una entidad raíz, por lo que todas las CA tienen el mismo nivel, es decir no hay CA intermediarias o subor-

dinadas. Sin embargo en la malla, cada CA puede certificar a las demás y viceversa, esto se conoce como certificación cruzada, resultando en una relación de confianza entre CAs. En la figura 2.13 se muestra como las diferentes CA se certifican entre sí (unidireccional o bidireccionalmente) y su vez cada una de ellas puede certificar a los usuarios finales de la PKI [Cooper et al., 2005, Kuhn et al., 2001].

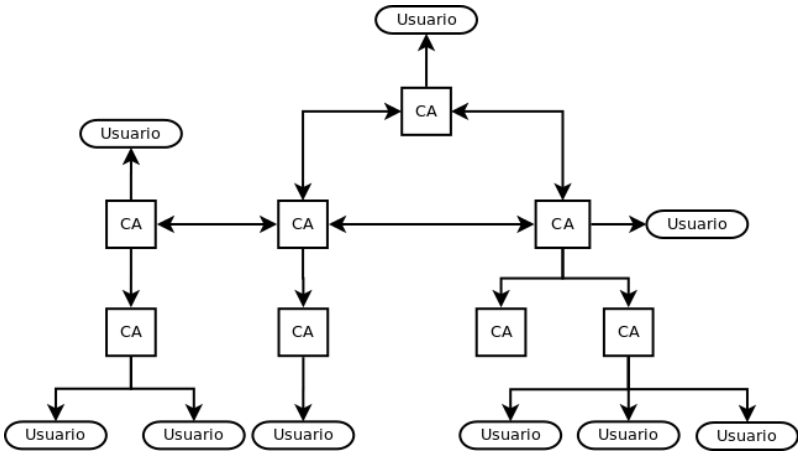


Figura 2.13: Ejemplo de un modelo de malla.

El camino de certificación en este modelo es más complejo que en el modelo jerárquico, debido a que pueden existir múltiples caminos entre las diferentes relaciones de confianza entre las CA, incluso se pueden caer en ciclos infinitos al momento de verificar un certificado [Cooper et al., 2005].

Mixto

La certificación cruzada puede ser utilizada para unir dos o más PKI, esto se realiza haciendo que las CA raíz las estructuras se certifiquen entre sí y de ésta manera generar una relación de confianza entre ellas. En el caso del modelo de malla, se selecciona una CA de manera arbitraria para que esta sirva como enlace con otra PKI. En la figura 2.14 se puede ver como dos estructuras, la PKI A y B se "conectan" cuando la CA raíz de la PKI A y una CA seleccionada arbitrariamente de la PKI B se certifican mutuamente [Cooper et al., 2005].

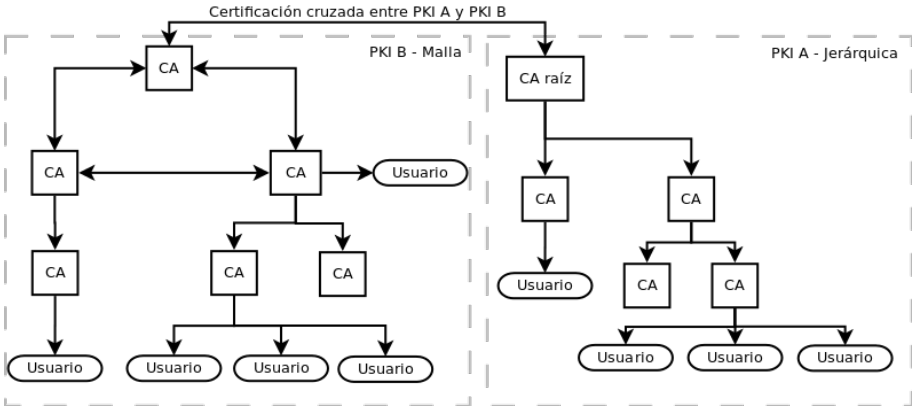


Figura 2.14: Ejemplo de un modelo mixto.

Puente

Este modelo fue diseñado como otra opción para conectar diferentes PKI entre sí, independientemente de su propia estructura, esto se logra introduciendo una nueva CA llamada, CA puente, la cual tiene como único propósito establecer relaciones entre diferentes PKI. A diferencia del modelo de malla, la CA puente, no puede emitir certificados a los usuarios finales. La CA puente únicamente puede relacionarse con una CA por cada PKI, como resultado, el número de relaciones de certificación cruzada en esta estructura crece linealmente mientras que en la malla, estas relaciones pueden crecer exponencialmente. En la figura 2.15 se puede ver como una CA puente conecta tres PKI diferentes entre sí.

Red de Confianza

También llamada 'Confianza acumulativa', es el modelo en el que se toman en cuenta una serie de factores para decidir si un certificado es válido. En este modelo no se utiliza como tal una CA, en su lugar cada entidad de la red certifica el vínculo entre las claves públicas y sus propietarios para otras entidades. Por ejemplo, una entidad A puede pensar que tiene buen conocimiento de la entidad de B y por lo tanto está dispuesto a firmar la clave pública de A. Todos los certificados emitidos en el sistema forman un grafo como el mostrado en la figura 2.16 llamado 'Red de confianza' [Mawloud et al., 2012].

PGP es el ejemplo de red de confianza más utilizada, en la cual una entidad asigna un nivel de confianza a otras entidades, conocidas como *'trusted introducers'* (que en realidad tienen la función de una CA), en PGP se existen cuatro niveles de confianza para estas entidades [Prohic, 2005, Callas, 2008]:

- No es de confianza: señala que una entidad no se confianza para firmar otros certificados, por lo que al detectar un certificado firmado por ésta, no es posible determinar la validez del mismo.

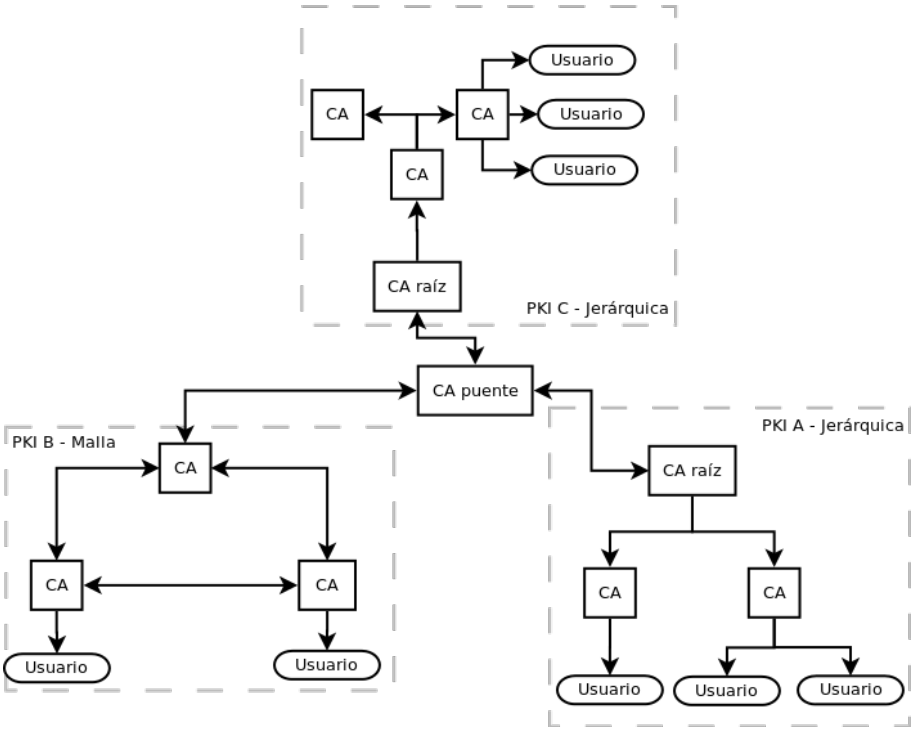


Figura 2.15: Ejemplo de un modelo con puente.

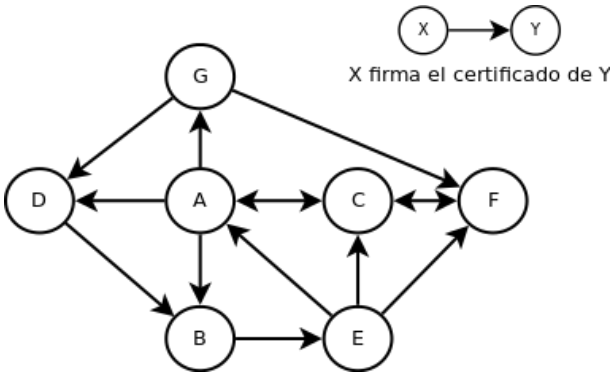


Figura 2.16: Ejemplo de una red de confianza.

- Confianza parcial: si se le asigna este nivel a una entidad, implica que al firmar un certificado, debe existir al menos otra entidad (con al menos el mismo nivel) que también haya firmado dicho certificado para poder determinarlo como válido.
- Confianza total: los certificados expedidos por certificados con este nivel son considerados válidos.
- Confianza definitiva: este nivel está reservado para entidades que posean el par de claves (pública y privada), por lo que al encontrar un certificado firmado por dicha entidad se debe considerar como válido, ya que al poseer ambas claves, implica que es un certificado auto-firmado.

En PGP se utiliza una un sistema basado en puntos en el cual se le asigna un valor en puntos a cada uno de los niveles y un umbral para determinar si un certificado es válido o no, por ejemplo al nivel de "confianza parcial" comúnmente se le asigna un punto y se utiliza un umbral de dos puntos por lo que como mínimo se requiere que dos certificados con ese nivel avalen la validez de un tercer certificado para que éste sea considerado como válido [Callas, 2008].

2.3. Revisión del estado de arte

A continuación se hará una revisión de los diferentes desarrollos existentes de PKI sobre dispositivos móviles, nos enfocaremos particularmente en las aplicaciones y los sistemas que utilizan *Android* debido a su gran popularidad la cual asciende a las 300,000 activaciones diarias colocándolo como el sistema operativo más popular durante el cierre del 2010 y 2011 [Crook et al., 2011].

Actualmente existen diversas propuestas e implementaciones de PKIs, las cuales han surgido para posteriormente convertirse en estándares, por lo que si se desea que una PKI sea aceptada y utilizada ampliamente es necesario que tenga compatibilidad con al menos alguna de ellas. Entre los esquemas más utilizados destacan dos:

X.509 [Cooper et al., 2008] Es un estándar para infraestructuras de clave pública que especifica entre otras cosas, los formatos para certificados (codificados utilizando ANSI X9), los algoritmos de validación para rutas de certificación y el formato para las listas de revocación de certificados. La primera versión del estándar surgió en 1988 y se basa en una estructura estrictamente jerárquica de las autoridades de certificación. Actualmente se encuentra en su tercera versión.

En la primera versión del estándar existían restricciones estructurales impuestas para asociar claramente la cadena de certificación. Ahí se requería una estructura jerárquica en donde todos las rutas de certificación iniciaban con la llamada IPRA, en el segundo nivel se encontraban las PCA y por último las CA's. La tercera versión es mucho más flexible ya que se pueden utilizar extensiones de certificados, sin la necesidad de una estructura

de certificación como tal, permitiendo además que un usuario pueda ser tanto una entidad como una autoridad certificadora.

Los certificados X.509 pueden tener diferentes extensiones entre las que destacan:

- `.CER`- Certificado codificado en CER (mnemónico por certificado), algunas veces es una secuencia de certificados.
- `.DER`- Certificado codificado en DER.
- `.PEM`- Certificado codificado en Base64, encerrado entre
  `-----BEGIN CERTIFICATE-----`
  y
  `-----END CERTIFICATE-----`.
- `.P7C`- Estructura PKCS#7, sin datos, solo certificado(s) o CRL(s).
- `.P12`- PKCS#12, puede contener certificado(s) (público) y claves privadas (protegido con clave)

PGP [Callas et al., 2007] *Pretty Good Privacy* fue desarrollado inicialmente por Phil Zimmermann con la finalidad de proteger la información distribuida a través de Internet utilizando criptografía de clave pública. Este esquema sirvió como base para el desarrollo de OpenPGP y posteriormente GnuPG. Éste se basa en la creación de redes de confianza utilizando servidores de claves, también puede utilizar certificados X.509.

Por otra parte las implementaciones en ambientes móviles son muy limitadas, aún más si se quiere hablar de los desarrollos hechos específicamente para *Android*. En la industria privada existen aplicaciones que ponen a disposición de los usuarios diversas operaciones criptográficas como cifrar y descifrar ciertos tipos de información dentro del dispositivo, tales como: archivos, correos electrónicos, mensajes [Messerman and Mustafic 2011, Echoworx, 2011, Rodriguez, 2011, Thialfihar, 2011, SANDBOX, 2010], o bien, firmar y verificar información digital contenida en el dispositivo [Thialfihar, 2011, SANDBOX, 2010]. Sin embargo, la mayoría de estas aplicaciones ofrecen únicamente funciones de criptografía simétrica, particularmente utilizando el AES como esquema de cifrado/descifrado. En el caso de las aplicaciones que utilizan esquemas de clave pública, éstas carecen de la definición de uno o de varios de los componentes esenciales de una PKI o bien dan lugar a algunas vulnerabilidades en la definición de los mismos. A continuación se presenta una breve reseña de las características, ventajas y desventajas de algunas de las aplicaciones disponibles en *Android Market*.

DroidCrypt [Messerman and Mustafic 2011] Es una aplicación que permite cifrar y descifrar archivos dentro del dispositivo, utilizando criptografía simétrica, particularmente AES como bloque de cifrado y descifrado. Una de las innovaciones más atractivas de esta aplicación es el uso de tres tipos de contraseñas para las operaciones criptográficas:

palabra clave, orientación del dispositivo y gesto táctil. Es capaz de manejar cualquier tipo de contenido digital como imágenes, música, documentos, etc.

Esta aplicación es eficiente en cuanto a recursos y cumple con el objetivo de cifrar archivos en el dispositivo, sin embargo el uso de criptografía simétrica hace que sea complicado enviar los archivos cifrados a otros usuarios (debido al manejo de claves). Adicionalmente, la aplicación no hace uso de ningún estándar, por lo que la compatibilidad con aplicaciones existentes es muy baja, con lo que el compartir archivos o claves generadas con esta aplicación es sumamente complicado. Por último, esta aplicación está disponible en dos versiones dentro del *Android Market*, una de prueba con funcionalidad limitada y otra con un costo para obtener la funcionalidad completa de la aplicación.

mobilEncrypt [Echoworx, 2011] Forma parte de una plataforma de cifrado para correo electrónico creada por Echoworx, por lo que únicamente es útil si se compra toda la plataforma, ya que por sí sola esta aplicación sólo tiene la funcionalidad de enviar y recibir correos electrónicos, los cuales son cifrados y firmados. Sin embargo esta aplicación tiene algunos puntos importantes a su favor:

- Es capaz de utilizar tanto esquemas de cifrado simétrico como asimétrico, utilizando estándares como RSA-1024, RSA-2048, y AES-256 entre otros
- Tiene soporte para certificados X.509, por lo que podría interactuar con otras aplicaciones que utilicen este tipo de certificados, sin embargo en la página oficial de la aplicación no se menciona ninguna.
- La plataforma completa de Echoworx conforma una PKI ya que las aplicaciones de escritorio tienen utilidades para la administración de certificados, listas de revocación de certificados, manejo de claves, validación y búsqueda de certificados, entre otras. No obstante estas opciones no están disponibles para la aplicación móvil, la cual únicamente tiene un cliente de correo electrónico seguro.

Por otro lado, al no ser una aplicación de software libre no se tiene acceso al código original de la aplicación ni la API utilizada para la implementación de estos servicios, por lo que la generación de aplicaciones empresariales particulares con estas características no seria posible.

OpenPGP Manager [Rodriguez, 2011] Actualmente se encuentra en la versión 1.44, es una implementación de PGP para *Android* y es compatible con la versión de escritorio. Desde su versión inicial se ha ido agregando funcionalidad poco a poco, e incorpora entre otros servicios: la creación de claves OpenPGP (utilizando RSA, DSA o El-Gammal), importar y exportar las claves desde OpenPGP (las cuales pueden estar almacenadas en el dispositivo o en servidores de claves), adicionalmente provee un módulo de administración de claves y ofrece una variedad de algoritmos de cifrado simétrico para que el usuario elija el que desee.

Esta aplicación tiene la funcionalidad de cifrar, descifrar, firmar y verificar tanto mensajes de correo electrónico como archivos dentro del dispositivo. Una de las ventajas de esta aplicación es que después de su instalación es posible realizar cualquiera de las operaciones disponibles sin la necesidad de estar conectado a una red, ya que realiza todas las operaciones criptográficas dentro del dispositivo móvil. A pesar de que las funciones de esta aplicación son muy atractivas, su interfaz de usuario es muy sencilla y nada estilizada, lo cual puede desanimar a un usuario a explotar al máximo las capacidades de la aplicación.

Por otra parte, esta aplicación también tiene un costo por instalación en el dispositivo, además de esto, en la página del desarrollador no hay acceso al código fuente o información de la API utilizada, por lo que la reutilización de esta aplicación no es posible.

Android Privacy Guard (APG) [Thialfihar, 2011] Al igual que la aplicación anterior, ésta es una implementación del estándar PGP que permite cifrar, descifrar, firmar y verificar tanto archivos dentro del dispositivo como correos electrónicos, con la diferencia de que esta aplicación es software libre, por lo que el código fuente esta disponible para su uso y modificación. Esta aplicación utiliza la API de desarrollo SpongyCastle [Tyley, 2012] como capa criptográfica base. Es importante mencionar que esta API es una migración de la implementación de Bouncy Castle [BouncyCastleJava, 2012], la cual es una biblioteca criptográfica ampliamente utilizada disponible para varios lenguajes de programación. APG tiene una interfaz de usuario más estilizada y útil que la aplicación anterior, por lo que permite al usuario aprovechar al máximo las capacidades de la API desarrollada, adicionalmente tiene un amplio soporte de algoritmos criptográficos (asimétricos, simétricos y funciones de síntesis (hash)) y es totalmente compatible con la versión de escritorio de OpenPGP en cuanto a claves y archivos (cifrados/firmados) se refiere.

Uno de los puntos a favor de esta aplicación es que realiza todo el procesamiento requerido en el dispositivo, por lo que si se requiere hacer una operación sobre archivos muy grandes, puede ser algo tardado pero en general tiene un desempeño muy bueno en diversos dispositivos móviles. A pesar de sus buenas cualidades, este software aún está lejos de constituir una PKI como tal, debido a que no contempla ninguna definición o implementación de una autoridad certificadora, la cual es el corazón de toda PKI.

PKI Webtop [SANDBOX, 2010] Fue desarrollada como parte de un proyecto financiado por el gobierno de España. Es la única disponible en *Android* que define un esquema completo de PKI dentro de este sistema y además que contempla el uso de un dispositivo móvil como parte del diseño. Para utilizar dicha aplicación se requiere instalarla en el dispositivo y registrarse en la página de los desarrolladores (sin costo alguno). Para hacer el registro, únicamente solicita que se ingrese un nombre de usuario y contraseña, los cuales se utilizan en el dispositivo para ingresar a la aplicación. Al finalizar el registro se genera el par de claves, las cuales se encuentran almacenadas en un solo archivo, el cual se puede descargar. Este archivo utiliza el estándar PKCS #12 [RSA, 1999], en el cual se incluye tanto la clave privada como la pública, protegidas utilizando la contraseña ingresada en el registro y criptográfica simétrica.

Tiene su versión para equipos de escritorio y ambas versiones utilizan el enfoque de cómputo de nube, en el cual las aplicaciones se ven como terminales "tontas" y toda la información es almacenada en la nube. La versión para *Android* solamente permite al usuario firmar y verificar archivos digitales que se encuentren en el dispositivo sin tomar en cuenta la confidencialidad de los mismos

Una de las principales desventajas de esta aplicación es que se requiere de una conexión a Internet en todo momento, ya que desde que se abre la aplicación se le solicita al usuario que se identifique en el portal de *Webtop* utilizando las credenciales correspondientes. Una vez dentro de la aplicación se muestra una lista con las opciones disponibles. Por otra parte, esta aplicación permite registrar y contestar peticiones de firmas digitales de archivos en la nube, es decir, solicitar a otro usuario que se firme algún documento, lo cual es algo innovador y atractivo.

Adicionalmente, una de las características más destacadas de esta PKI también se puede ver como su mayor debilidad, ya que al utilizar el enfoque de cómputo de nube, la aplicación almacena una copia de la clave privada en el servidor. Además de esto, las operaciones criptográficas no se realizan dentro del dispositivo, sino que se hacen en el servidor, por lo que al realizar cualquier operación el archivo es enviado al servidor (en claro) y al terminar se retorna la versión firmada del archivo. Otro punto en contra es que el servidor almacena una copia de los archivos firmados, por lo que es indispensable confiar ciegamente en el servidor, para que la PKI funcione correctamente.

Además de ésto, en la PKI no se encuentra definida una autoridad certificadora como tal (sin embargo, la nube podría desempeñar este papel), tampoco se menciona ningún mecanismo para revocar certificados, manejar tiempo de validez de los mismos o algún tipo de liga entre el certificado y la identidad del usuario, lo más parecido a esto es el identificador que se registró en la página.

Por último, en la página web del desarrollador no se da información acerca de los algoritmos utilizados ni si acaso se usa algún estándar aparte del PKCS #12. A pesar de que esta aplicación no tiene ningún costo, no se tiene acceso ni al código original de la aplicación ni información sobre el uso de una API basada en la implementación realizada.

A continuación se presentan una serie de tablas con la intención de comparar las principales características de las aplicaciones descritas en este capítulo, adicionalmente en estas tablas se pueden observar las características de la PKI propuesta en este trabajo.

Para comenzar la tabla 2.2 muestra los servicios de seguridad contemplados en estas aplicaciones, ahí observamos que algunas aplicaciones como OpenPGP Manager y APG, al igual que la PKI propuesta proveen todos los servicios de seguridad deseables.

En la tabla 2.3 se listan las características funcionales de las aplicaciones, entre las cuales destacan sus grados de compatibilidad respecto a otras aplicaciones en el mercado y la reusabilidad de dicha aplicación para la creación de otras. Como se puede observar la PKI propuesta tiene un alto grado de compatibilidad y reusabilidad gracias al API implementado, la cual ayuda en la creación de aplicaciones móviles seguras como la presentada

	Confidencialidad	Integridad	Autenticación	No-Repudio
X.509	✓	✓	✓	✓
PGP	✓	✓	✓	✓
DroidCrypt	✓			
mobilEncrypt	✓	✓	✓	
OpenPGP Manager	✓	✓	✓	✓
APG	✓	✓	✓	✓
PKI Webtop		✓	✓	✓
PKI Propuesta	✓	✓	✓	✓

Tabla 2.2: Servicios de seguridad contemplados.

en el capítulo 5 de este trabajo. Adicionalmente la tabla muestra los tipos y la variedad de algoritmos disponibles para los usuarios en estas aplicaciones, en el caso de OpenPGP Manager, APG y la PKI propuesta tienen disponibles algoritmos tanto de criptografía simétrica como asimétrica.

	Uso de estándares	Compatibilidad	Criptografía simétrica	Criptografía asimétrica	Versatilidad algorítmica	Reusabilidad	Costo
X.509	Alto	Alto	✓	✓	Alta	Alta	Gratuito
PGP	Alto	Media	✓	✓	Alta	Media	Gratuito
DroidCrypt	Bajo	Ninguna	✓		Baja	Nula	Alta [1]
mobilEncrypt	Bajo	Ninguna	✓		Baja	Nula	Bajo
OpenPGP Manager	Medio	Media	✓	✓	Alta	Nula	Medio
APG	Alto	Alta	✓	✓	Alta	Alta	Gratuito
PKI Webtop	Bajo	Media		✓	Media	Nula	Gratuito
PKI Propuesta	**Alto**	**Alto**	✓	✓	**Alta**	**Alta**	**Gratuita**

Tabla 2.3: Características de la aplicación.

Por último, en la tabla 2.4 se muestran los elementos deseables en una PKI y cuáles de éstos están definidos en el diseño de la aplicación. En algunos casos, como en el de *mobilEncrypt*, estos elementos no están definidos por sí solos para la aplicación disponible en Android, sino que se encuentran definidos en la suite de Echoworx.

	Autoridad Certificadora	Autoridad de Registro	CRL	Repositorio	Usuario de PKI
X.509	Definida	Definida	Definida	Definido	Definido
PGP	Considerada	No definida formalmente	Considerada	Considerado	Definido
DroidCrypt	No definida	No definida	No definida	No definido	Definido
mobilEncrypt	No definida [2]	No definida [2]	No definida [2]	No definido [2]	Definido
OpenPGP Manager	Considerada	No definida formalmente	Considerada	Considerado	Definido
APG	Considerada	No definida	No definida	No definido	Definido
PKI Webtop	No definida formalmente	Definida	Definida	Definido	Definido
PKI Propuesta	**Definida**	**Considerada**	**Definida**	**Definida**	**Definido**

Tabla 2.4: Elementos de una PKI considerados.

[1] Disponible una versión de prueba gratuita

[2] No se encuentra definida en la aplicación móvil, sin embargo en el paquete completo si lo está

Capítulo 3

Diseño de la PKI

Las organizaciones usan una variedad de soluciones tecnológicas para habilitar procesos empresariales esenciales, como lo son las compras en línea, intercambio de información y acceso remoto. Una Infraestructura de Clave Pública (siglas del inglés *Public Key Infrastructure*) (PKI) provee los medios para que las organizaciones puedan realizar estos procesos de manera segura [Microsoft, 2004].

En este capítulo se describe el proceso que se ha seguido para implementar la PKI desarrollada en este trabajo, sus requerimientos y sus características principales. Es importante mencionar que el diseño realizado en este trabajo contempla el estudio de dos tipos de modelos de confianza: una estructura jerárquica y una red.

Presentaremos aquí el diseño general de nuestro sistema, de la PKI propia que implementamos así como las políticas de certificación adoptadas. Éstas coinciden con la declaración de políticas de certificación que rigen el funcionamiento y operaciones de una PKI, las cuales fueron establecidas siguiendo el estándar "*ETSI TS 101 042: Policy requirements for certification authorities issuing public key certificates*" [Barreira and Compans, 2011].

Con el propósito de que nuestra experiencia sea replicada abundaremos en detalles acaso repetitivos. Es nuestra intención que cualquier desarrollador pueda tomar el presente documento como autocontenido de manera que se aproveche al máximo nuestra experiencia.

Presentamos nuestro diseño en tres partes: En la primera aparecen las características generales adoptadas para nuestra PKI, entre éstas están los tamaños de claves y los campos recomendados en el estándar X.509, así como los modelos de confianza y la gestión de certificados; en la segunda se describe la PKI con todo detalle; y finalmente se describen las políticas de certificación que potencialmente pueden ser implementadas en la PKI presentada.

Así pues, comenzando con la sección 3.1, presentamos el proceso de diseño, en la sección 3.2 se listan las características generales del diseño de la PKI que puede implementar tanto una red de confianza como un modelo jerárquico, con algunos cambios menores en

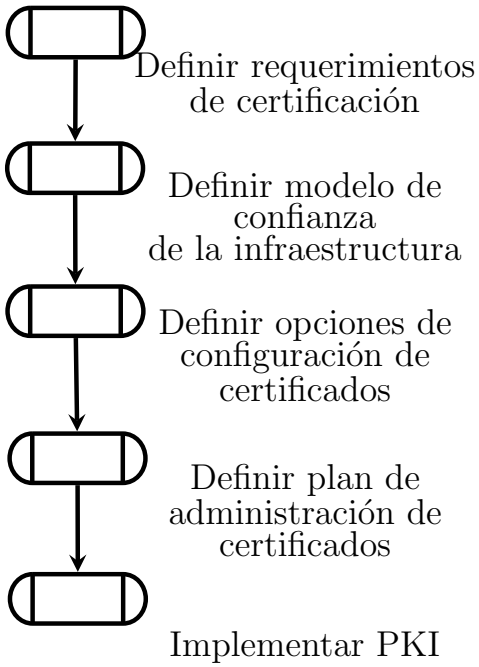


Figura 3.1: Proceso para diseñar una PKI.

la estructura base. En la sección 3.3 se plantean las políticas de certificación para un caso de estudio puramente teórico el cual utiliza un modelo de confianza jerárquico, con el fin de plantear las bases para trabajos a futuro. Mientras que en el capítulo 5 se presenta el caso de estudio práctico, en el cual se utilizó el diseño aquí planteado para elaborar una aplicación de prueba en *Android*, la cual contempla un modelo de red de confianza.

3.1. Proceso de diseño

Diseñar una PKI conlleva una serie de pasos en los cuales se definen características como, requerimientos de los certificados, diseño de la arquitectura, plan de administración de certificados y por último la implementación de la misma. En esta sección se describirá el proceso que se ha seguido para realizar el diseño que se presenta en esta tesis, tomando como base el proceso propuesto en [Microsoft, 2004]. Éste se puede observar en la Figura 3.1

3.1.1. Definir requerimientos de certificación

En este paso del proceso se definen las características que los certificados deben tener, para esto es necesario tomar en cuenta diversos factores como lo son: el valor de la información que se va a proteger, el costo que involucra la implementación de los sistemas de

seguridad y el impacto deseado en la seguridad del sistema. Para esto se recomienda seguir el subproceso descrito en la figura 3.2, el cual se explicará a continuación [Microsoft, 2004].

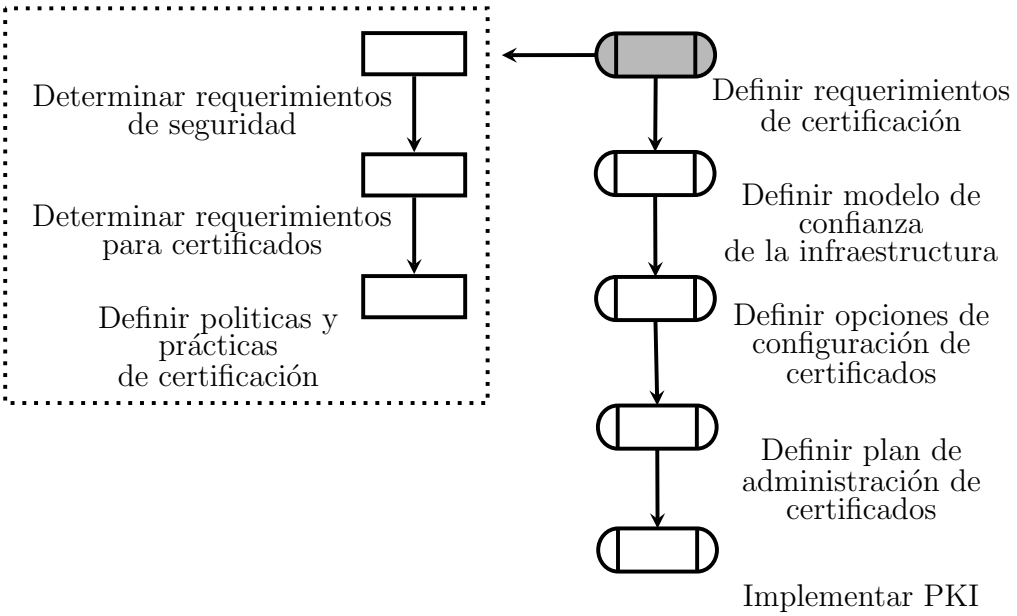


Figura 3.2: Subproceso para definir los requerimientos de certificación.

Al inicio de este subproceso, se deben determinar los requerimientos de seguridad de la aplicación, es decir, es necesario reconocer los servicios que se desean proveer y seleccionar los mecanismos de seguridad adecuados para esto. Por ejemplo, si se requiere verificar la integridad de archivos y mensajes dentro de una red, seleccionar cuales algoritmos de firma digital o funciones picadillo se utilizarán. Adicionalmente es deben determinar las tecnológicas que se utilizarán para cubrir estos requerimientos.

El siguiente punto es identificar los tipos de usuarios que harán uso de la infraestructura y con base en estos determinar las características que los certificados deberán cumplir, entre las cuales destacan:

- Tipo de certificados a emitir: esta característica determina si los certificados serán emitidos por una CA interna o una externa, si serán certificados auto-firmados y los estándares que se deberán seguir en la creación y validación de dichos certificados.
- Número de certificados por usuario: en algunos casos un certificado puede cumplir todas las necesidades dentro de una empresa, sin embargo en algunos casos puede ser que se requieran de múltiples certificados para cumplir con los requisitos de seguridad.
- Nivel de seguridad requerido por los certificados: esta característica es importante ya que dependiendo de la información que se manejara dentro de la PKI los requerimientos de seguridad pueden variar, por ejemplo el tamaño de clave y función picadillo a utilizar.

- Requerimientos de registro: determinar si es necesario que los usuarios presenten algún tipo de documento para comprobar su identidad antes de que se emita un certificado o únicamente se requiere de una solicitud electrónica para solicitar un certificado.

Finalmente, el último paso de este subproceso es definir las políticas y prácticas de certificación para la PKI que se desea implementar, este paso es uno de los más laboriosos ya que diseñar una PKI involucra factores como lo son: certificados, autoridades de certificación y el desarrollo de procesos que ayuden en la administración de la infraestructura. Por lo que es necesario contar con procesos y políticas claramente definidos para que interactúen con la parte tecnología para asegurarse que los servicios ofrecidos provean el nivel de seguridad deseado por la aplicación.

Para la generación de estas políticas existe el estándar "*ETSI TS 101 042: Policy requirements for certification authorities issuing public key certificates*" presentado en [Barreira and Compans, 2011], mediante el cual se dan los lineamientos necesarios para generar un documento de políticas y prácticas de certificación para una PKI. De manera general una política de certificación debe contener los siguientes puntos:

- Como se autentican los usuarios ante la CA.
- Detalles legales, como responsabilidades de la CA si se llega a comprometer o se utiliza con otros propósitos a los especificados.
- Propósitos del(los) certificado(s) emitidos.
- Manejo de llaves privadas, como por ejemplo, forma de almacenamiento y forma de recuperación.
- Requerimientos para los usuarios, estos incluyen las obligaciones de los usuarios hacia la PKI
- Procedimientos de renovación y revocación de certificados.
- Procedimientos de registro y solicitud de certificados.
- Tamaño mínimo en el tamaño de llaves.

3.1.2. Definir modelo de confianza de la infraestructura

El siguiente paso en el proceso de diseño de la PKI es definir lo referente al modelo de confianza que se utilizará dentro de la misma, ya que este modelo será el encargado para establecer el vínculo necesario entre las CA de la aplicación y los usuarios, para que la PKI pueda ofrecer servicios confiables a los usuarios, además de definir las capacidades de escalabilidad que tendrá la PKI en el futuro mientras se mantiene un nivel óptimo de seguridad para la aplicación. Al igual que el subproceso anterior, este subproceso incluye una serie de pasos que ayudan al diseñador a establecer las características deseadas de la PKI, estos pasos se pueden observar en la figura 3.3 y serán descritos a continuación.

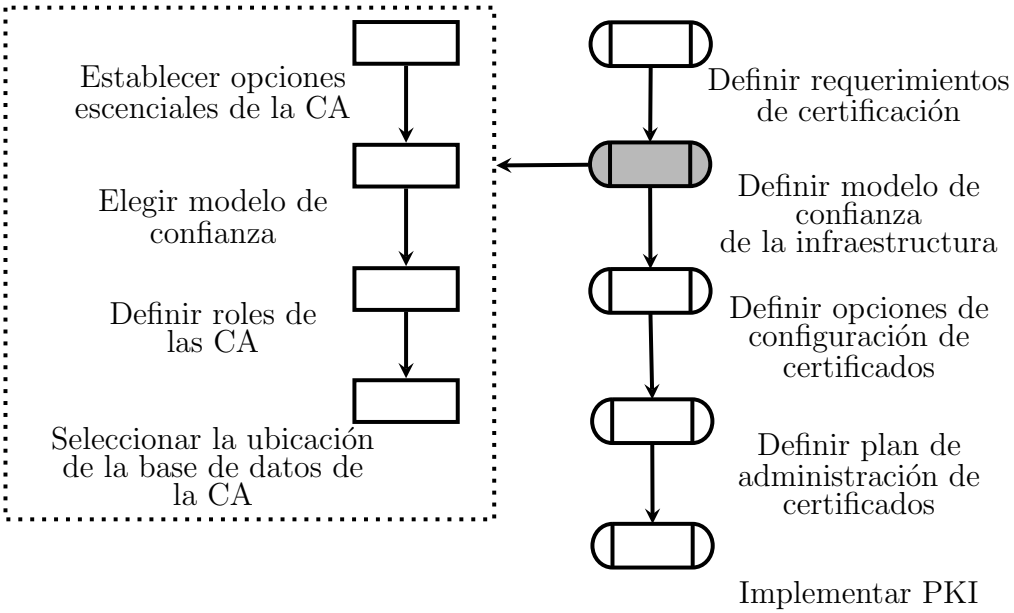


Figura 3.3: Subproceso para establecer el modelo de confianza.

Antes de definir el modelo de confianza que tendrá la PKI es importante establecer las opciones principales de la CA, de estas opciones o característica depende que la infraestructura cumpla con los requisitos de seguridad y certificación que se requieren en la aplicación. Las opciones que se deben considerar al momento de diseñar el modelo de confianza son las siguientes:

- Designar a la autoridad certificadora raíz: en el caso de los modelos jerárquicos es necesario designar una autoridad de certificación final o CA raíz, esta entidad es la encargada de certificar a otras CA para que puedan emitir los certificados necesarios en la infraestructura, además de ser la encargada de administrar y publicar los certificados de la PKI. Para poder designar a la CA raíz se requiere que:
  - Determinar quien administrara a esta entidad, por ejemplo, decidir si es responsabilidad del equipo de TI de la organización, o si será administrada por el director general de la misma.
  - Establecer quien tendrá el control sobre las claves de la CA.
  - Decidir si la CA únicamente se dedicara a certificar a otras CA o si también emitirá certificados a usuarios finales
  - Determinar como se resguardaran las claves de la CA.
- Utilizar una CA interna o una externa: esta decisión es fundamental para el diseño de una PKI, ya que dependiendo de las necesidades y presupuesto de cada aplicación son diferentes, por lo que determinar si se utilizará una CA raíz interna, externa o una mezcla de ambas tendrá repercusiones directas en las demás opciones de la PKI:

  - CA Interna: entre las principales ventajas de este tipo de autoridad destaca que: se tiene total control sobre las políticas de certificación a establecerse, se puede extender la funcionalidad a medida que se vaya requiriendo y que el costo de estas extensiones es relativamente bajo. Sin embargo también existen algunas desventajas que es importante considerar, como: se tiene que tener un manejo interno de los certificados expedidos, es responsabilidad de la organización manejar problemas derivados del mal uso de los certificados que se expidieron en la aplicación y debido al trabajo que conlleva, el tiempo de desarrollo de la solución suele llevar más tiempo del que tomaría si se utiliza una solución externa.

  - CA Externa: por otra parte utilizar los servicios de una autoridad certificadora externa también tiene sus ventajas como: el uso de certificados expedidos por entidades externas y reconocidas, puede hacer que los clientes y socios de las organizaciones tengan mayor confianza en sus certificados, se cuenta con la experiencia de profesionales en el ramo para ofrecer estos servicios, es posible utilizar estos certificados mientras se desarrolla una solución propia. A pesar de estas ventajas, utilizar una CA externa también trae consigo algunas desventajas, por ejemplo: Los certificados emitidos por estas empresas, típicamente tienen un costo alto por certificado, es posible que se requiera la definición de dos tipos de políticas, una para certificados internos y otra para los externos, las opciones de configuración que se permiten en este tipo de soluciones es muy limitada, la generación de certificados auto-firmados no es posible.

- Modelo de administración de la PKI: es importante definir este modelo al inicio del proceso de diseño de la PKI, para asegurarse que al comprometerse un solo individuo de la PKI, no sean comprometidos sus servicios, una recomendación para esto es distribuir los roles de administración entre diferentes individuos de manera que ninguno tenga un poder total sobre la CA. Algunas de los roles que se deben distribuir son: creación y manejo de CAs, administración de plantilla de los certificados, emisión y revocación de certificados. Decidir que tanto se distribuirán las tareas de la CA raíz, depende en el nivel de seguridad que se requiera para cada aplicación. Por ejemplo, en una aplicación la CA raíz podría ser la encarga de emitir certificados a CA subordinadas, las cuales pueden tener roles específicos como creación y administración de plantillas de certificados, distribución de CRL, generación de certificados a usuarios finales, registro y autenticación de usuarios, de ésta manera al comprometerse cualquiera de estas CA la PKI puede continuar su funcionamiento general mientras se repara el daño o se genera una nueva CA que ocupe el lugar de la entidad comprometida.

- Número de CAs que se utilizarán: por último el número de CAs que se utilizarán dentro de la PKI es diferente en cada caso, ya que esto tendrá repercusiones directas en otras características como en el almacenamiento de los certificados emitidos, la forma de publicación de los mismos, o en caso de las redes de confianza como se verificará el estatus de los certificados.

Una vez establecidas las opciones que se tienen en las CA, se debe definir el modelo de confianza que se utilizará en la PKI, como se explicó en la sección 2.2.4 de este documento, del modelo seleccionado dependerá la forma de verificar un certificado. Por ejemplo, en un modelo jerárquico los certificados pueden ser localizados y verificados relativamente fácil, ya que por naturaleza se construye un árbol de certificados y las consultas se hacen a través de este árbol, en el peor de los casos se requiere subir hasta la CA raíz para poder determinar la validez de un certificado, sin embargo el algoritmo de verificación se va haciendo más complejo a medida que se utilizan modelos más complejos, por ejemplo en el modelo con puente, es necesario brincar de una CA a otra para determinar la validez de los certificados, por último en una red de confianza la complejidad para determinar si un certificado es válido o no, puede ser muy compleja ya que no existe como tal una entidad raíz, por lo que se utilizan otros mecanismos como el propuesto en [Callas, 2008] basado en puntos y listas de confianza.

Cuando el modelo de confianza se haya establecido, es necesario definir claramente (en caso de ser necesarios) los roles y tipos de CAs que se emplearán, ya que para cada aplicación los requisitos de certificación pueden ser diferentes, por lo que un solo tipo de CA podría no ser suficiente, por ejemplo, la CA raíz podrá emitir certificados de otras CAs, pero no de usuarios finales, o una CA puede encargarse únicamente de la emisión de las CRLs.

Por último una de las cosas más importantes a definir en el diseño de una PKI es ubicación de la o las bases de datos que utilizará la CA de la PKI, es decir si se utilizará una base de datos centralizada o una distribuida, en su caso, quién sera el encargado de administrarla, adicionalmente es necesario definir que se almacenará en ésta, por ejemplo, si se almacenara el certificado como tal o únicamente una referencia a su ubicación física.

3.1.3. Definir opciones de configuración de los certificados

Al llegar a este punto, ya se tienen definidos los requisitos de seguridad, los servicios que se necesitan ofrecer, las opciones que se desea que tengan las CA de la infraestructura y se encuentra claramente definido el modelo de confianza que se utilizará para realizar la validación de certificados. Ahora el siguiente paso en el proceso diseño presentado en la figura 3.1 es definir las opciones de configuración de los certificados que se expedirán en la PKI, esto quiere decir, que en paso se definirán las característica que tendrán los certificados, éstas incluyen entre otras, nivel de seguridad, algoritmos a utilizar, tamaños de clave y campos y extensiones utilizadas dentro en los certificados. En la figura 3.4 se ilustra el subproceso a seguir para poder definir las opciones de configuración anteriormente mencionadas.

Como se puede observar en la figura 3.4 este subproceso tiene únicamente dos pasos, en el primero se definirá la plantilla que se utilizará para crear los certificados, está platilla debe contener un listado de los campos que se utilizarán dentro del certificado, para

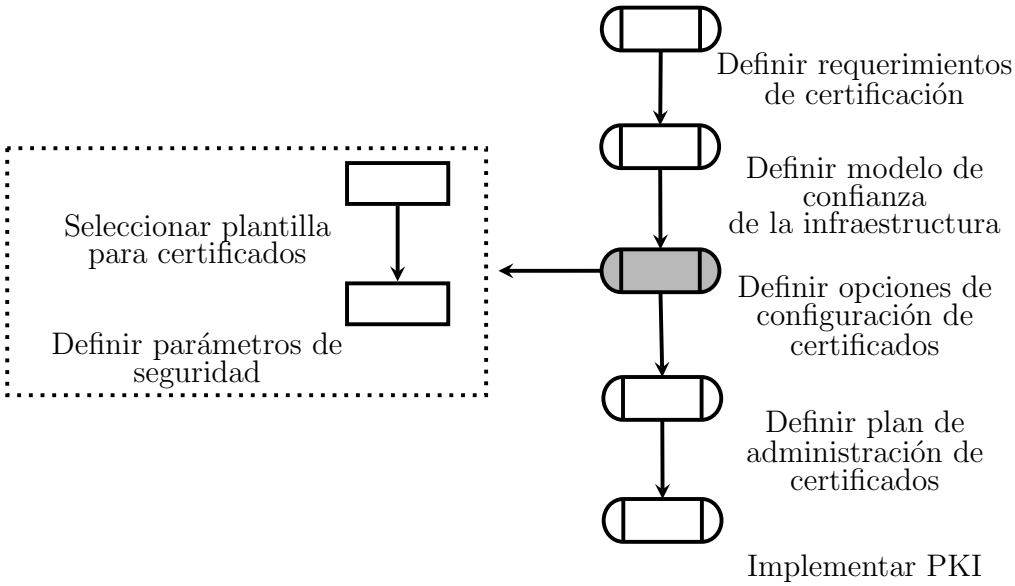


Figura 3.4: Subproceso para definir las opciones de configuración de los certificados.

elaborar la plantilla se debe tomar como base el esqueleto de un certificado estándar, por ejemplo los certificados X.509 en su versión 3, los cuales contienen una gran variedad de campos para satisfacer la mayoría de los requerimientos de una PKI. A continuación se da un listado de los campos obligatorios en este estándar, estos campos son descritos con mayor detalle en la sección 2.2.3 de este documento:

- Titular del certificado.
- Emisor del certificado.
- Número de serie.
- Clave pública del titular.
- Periodo de validez.
- Extensiones opcionales.
- Firma digital del emisor.

Una vez definida la plantilla para los certificados que se emitirán, se necesita determinar cuáles serán los parámetros de seguridad que se utilizarán para cada uno de los tipos de certificados dentro de la PKI, los parámetros que se deben ser definidos son los siguientes:

- Algoritmos criptográficos: se recomienda que los algoritmos utilizados en la PKI sigan los diversos estándares, para que los certificados emitidos sean compatibles con otras aplicaciones. A su vez mientras más algoritmos sean soportados, más compatibilidad se tendrá al importar y exportar tanto certificados como los mensajes cifrados por la aplicación que está siendo diseñada, por ejemplo, si una PKI incluye soporte

Bits de seguridad	Algoritmos de clave simétrica	Diffie-Hellman, DSA RSA	ECC
80	2TDEA	1024	160-223
112	3TDEA	2048	224-255
128	AES-128	3072	256-383
192	AES-192	7680	384-511
256	AES-256	15360	512+

Tabla 3.1: Equivalencia para tamaño de llaves (bits).

<table>
<tr><th colspan="2" rowspan="2">Seguridad</th><th colspan="3">Periodo de tiempo</th></tr>
<tr><th>2011 al 2013</th><th>2014 al 2030</th><th>Más allá de 2031</th></tr>
<tr><td rowspan="2">80</td><td>Aplicado</td><td>Desaprobado</td><td colspan="2">No permitido</td></tr>
<tr><td>Procesado</td><td colspan="3">En caso de ser necesario</td></tr>
<tr><td rowspan="2">112</td><td>Aplicado</td><td rowspan="2">Aceptable</td><td rowspan="2">Aceptable</td><td>No permitido</td></tr>
<tr><td>Procesado</td><td>En caso de ser necesario</td></tr>
<tr><td>128</td><td rowspan="3">Aplicado y Procesado</td><td>Aceptable</td><td>Aceptable</td><td>Aceptable</td></tr>
<tr><td>192</td><td>Aceptable</td><td>Aceptable</td><td>Aceptable</td></tr>
<tr><td>256</td><td>Aceptable</td><td>Aceptable</td><td>Aceptable</td></tr>
</table>

Tabla 3.2: Recomendación del NIST para tamaño de claves.

tanto para el esquema de criptografía asimétrica Rivest, Shamir y Adleman (por las iniciales de sus autores) (RSA) como para el de Curva Elíptica (siglas del inglés *Elliptic Curve*) (EC) la gama opciones para los usuarios será mucho más variada de forma que puedan existir certificados firmados con una clave RSA y firmados utilizando una clave de EC o utilizar el mismo algoritmo en ambos casos, de esta manera los usuarios tendrán mayor de libertad para elegir la opción que más les convenga.

- Tamaño de claves: el tamaño de las claves está en parte definido por el tipo de algoritmo que se elija, ya que cada uno tiene parámetros propios y por lo tanto ofrecen un nivel de seguridad diferente. En la tabla 3.1 se muestran los valores recomendados por el Instituto Nacional de Estándares y Tecnología (siglas del inglés *National Institute of Standards and Technology*) (NIST) para proteger las claves utilizadas en algoritmos de cifrado comunes como lo son DES y AES y los tamaños de claves para RSA, Diffie-Hellman y EC necesarios para proveer una seguridad equivalente [Barker et al., 2012].

  Es importante considerar un balance entre seguridad y eficiencia al momento de elegir el tamaño de claves que se utilizarán, ya que mientras más grande sea la clave, más complicadas computacionalmente serán las operaciones criptográficas. En [Barker et al., 2012] se proporciona una recomendación en cuento al tamaño de claves que se debería utilizar en aplicaciones seguras para proteger información sensible, estas recomendaciones se pueden ver en la tabla 3.2.

Esta tabla está dividida en 2 columnas principales, la primera se encuentra dividida a su vez en 2 subcolumnas, la primera para especificar la seguridad en bits que se desea obtener y la segunda para especificar si el proceso criptográfico está siendo aplicado a los datos (por ejemplo, cifrado y firma) o si los datos protegidos mediante el proceso criptográfico están siendo procesados (por ejemplo, descifrado y verificación). La segunda de las columnas principales de la tabla 3.2 indica los periodos de tiempo durante los cuales el nivel de seguridad es aceptable, no aceptable, no permitido o su uso se recomienda solo en caso de ser necesario. "Aceptable", indica que el nivel de seguridad es considerado como seguro durante ese periodo de tiempo. "Desaprobado", indica que se debe utilizar ese nivel de seguridad bajo el riesgo de que puedan existir ataques que vulneren la seguridad del sistema. "En caso de ser necesario", indica que el nivel de seguridad puede ser utilizado para mantener la compatibilidad con datos viejos, ya que de otra manera no podrían utilizarse. Por último "No permitido" indica que ese nivel de seguridad no debe ser utilizado en aplicaciones seguras [Barker et al., 2012].

- Periodos de validez: otro punto de suma importancia es definir cuanto durara una clave o certificado siendo válido, esto depende de un gran número de factores, de manera general se puede pensar que mientras más alto sea el nivel de seguridad empleado para generar una clave, más largo puede ser su periodo de validez [Microsoft, 2004].

3.1.4. Definir plan de administración de certificados

Una vez se tengan establecidas las opciones de los configuración de los certificados emitidos, es necesario crear el plan de administración para estos certificados, el plan de administración es aquel en que se define cada una de las etapas del ciclo de vida de un certificado, es decir debe contener los procesos para la solicitud, emisión, renovación y revocación de los certificados. En la figura 3.5 se muestran los pasos a seguir para poder definir un plan de administración de certificados, de manera general este plan debe contener mecanismos para la solicitud y renovación de certificados, la manera de establecer un vínculo entre el titular del certificado y sus certificados, definir políticas de revocación de certificados y en caso de ser necesario establecer el plan para la recuperación de claves.

Como se puede ver en la figura 3.5 este paso inicia mediante la definición de los mecanismos que se utilizarán para la solicitud y renovación de certificados, la solicitud de un certificado involucra diferentes acciones las cuales se tienen que documentar en el plan de administración establecido en [Barreira and Compans, 2011], este plan puede variar dependiendo si se va a expedir un certificado para una nueva CA o para un usuario final, puede involucrar la participación de una RA para verificar cada solicitud de registro y determinar si la identidad proporcionada corresponde a la verdadera identidad del titular del certificado mediante el uso de documentos físicos u cualquier otra información que la organización señale como necesaria. El plan de administración debe contener todos mecanismos mediante los cuales un sujeto pueda obtener un certificado, por ejemplo, para certificados auto-firmados, para certificados de CA, para certificados firmados por una

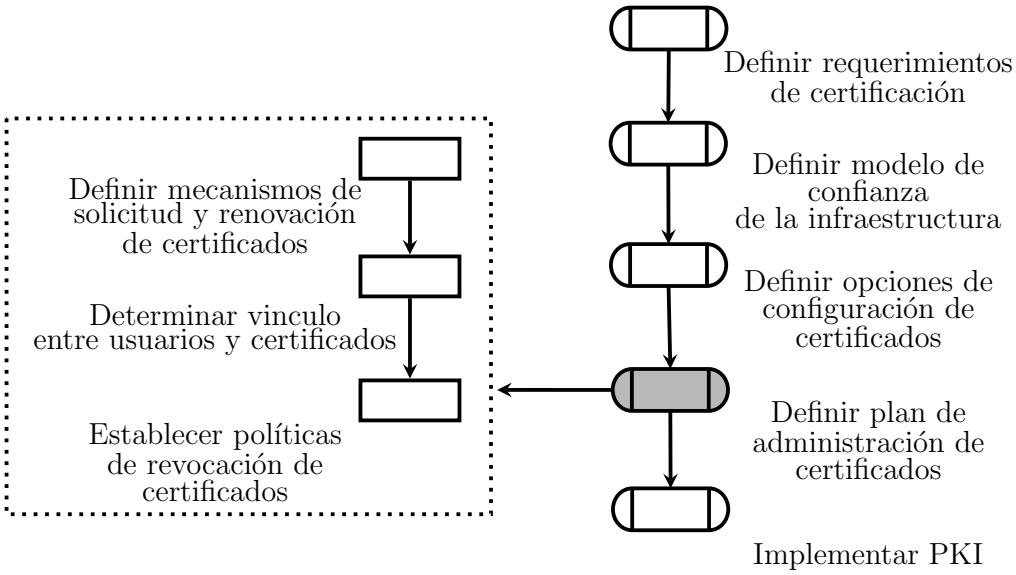


Figura 3.5: Subproceso para definir el plan de administración de certificados.

CA, etc. En [Barreira and Compans, 2011] se señala que el mecanismo de solicitud debe contemplar lo siguiente:

- Solicitud de información de identificación del solicitante, en caso de ser requerido por la organización.
- La CA debe proporcionar a los solicitantes la información de sus obligaciones.
- En caso que la CA no genere las claves del sujeto, está se debe asegurar que el sujeto tenga posesión de la clave privada.
- La especificación paso a paso del proceso para solicitar un nuevo certificado.

Por otra parte en el caso de la renovación, el estándar señala que el mecanismo debe contemplar:

- La verificación de la existencia y validez del certificado a ser renovado, así como que la información referente a la identidad del titular sea aun válida.
- Comunicación de cambios a los términos y condiciones de la CA al titular, solo en caso de que estos hayan cambiado.
- En caso de que algún atributo haya cambiado o el certificado previo haya sido revocado, debe incluir la validación de la información de identidad al igual que en el proceso registro.
- Que el proceso de renovación debe expedir un certificado nuevo utilizando la clave pública previa, si y solo si la seguridad criptográfica de la misma aun es válida y no existen pruebas de que la clave privada haya sido comprometida.

Una vez que se tiene definido como se obtendrán los certificados dentro de la PKI es

necesario definir el mapeo entre los usuarios de la PKI y sus correspondientes certificados, esto es necesario por ejemplo, para proveer mecanismos de autenticación más seguros y eficientes ya que se provee una manera de ligar a un certificado con una cuenta de usuario en el sistema.

El último paso del subproceso de definición del plan de administración de certificados, es establecer las políticas de revocación de certificados, en este punto es importante ya que a pesar que los certificados tienen un tiempo de vida establecido, en algunas ocasiones pueda ser necesario invalidar un certificado antes de que éste termine su ciclo de vida de forma normal debido a diferentes factores. En el estándar propuesto por la Instituto Europeo de Normas de Telecomunicaciones (siglas del inglés *European Telecommunications Standards Institute*) (ETSI) señala que la sección de política de renovación de certificados debe contener al menos:

- Quien puede solicitar la revocación.
- Procedimiento para solicitar la revocación.
- Requerimientos para poder solicitar la revocación.
- Razones por las cuales se puede revocar un certificado.
- Mecanismos de distribución de las CRL
- Tiempo máximo de retraso entre la petición de revocación y la distribución de la información del cambio de estatus de revocación.

Adicionalmente se recomienda utilizar el estatus de "Suspendido" para los certificados que estén en proceso de revocación, es decir, para certificados para los cuales se haya solicitado una revocación, pero la CA correspondiente aun se encuentre validando dicha solicitud. Además de lo anterior, en este estándar, señalan que es recomendable que la PKI incluya la implementación del protocolo OCSP (*Online Certificate Status Protocol*) para que terceros puedan verificar el estatus de un certificado en cualquier momento sin necesidad de recurrir o esperar a que una CRL sea publicada. Todo lo anterior es debido a que un certificado podría ser considerado como "válido" por un usuario de la PKI durante el tiempo transcurrido entre la solicitud de revocación y la publicación de la CRL, lo cual supone un riesgo para la seguridad de la PKI.

3.2. Descripción y características de la PKI

En esta sección se darán los detalles de la PKI diseñada siguiendo los pasos descritos en el proceso explicado en la sección 3.1, como se mencionó al inicio de este capítulo, el diseño que se presenta es un diseño base para poder implementar diferentes modelos de confianza, lo cual dependerá de las necesidades que tenga la aplicación que empleará la API presentada como resultado final de este diseño para poder ofrecer los servicios de seguridad deseados.

3.2.1. Requerimientos

Esta sección tiene como finalidad presentar la definición de los requerimientos de la PKI que se presenta, es decir una descripción detallada en lenguaje natural de la funcionalidad que la PKI ofrece. Los requerimientos describen las características deseables de la PKI de una manera general, es decir, los requerimientos deben establecer que debe hacer, pero no como lo debe hacer.

Requerimientos de seguridad

Como se mencionó en la sección 3.1.1 los requerimientos de seguridad, son aquellos que definen que servicios se necesitan ofrecer dentro de la PKI, así como los algoritmos y esquemas criptográficos que se utilizarán para ofrecerlos.

- Se deben contemplar los cuatro servicios de seguridad básicos: integridad, confidencialidad, autenticación, no-repudio.
- La PKI deberá utilizar alguna biblioteca criptográfica disponible, la cual contenga una implementación eficiente de los algoritmos que se utilizarán para la construcción de los servicios de seguridad.
- Cada uno de los servicios de seguridad requeridos deben ser pensados tanto para proteger tanto el intercambio de archivos como de mensajes entre dispositivos.
- Para proveer confidencialidad, se requiere contemplar el uso de AES como algoritmo de clave simétrica para la PKI.
- A su vez para los servicios de integridad, autenticación y no-repudio, se deberán utilizar esquemas de criptográfica asimétrica.
- RSA y ECC deberán ser contemplados como esquemas de criptográfica asimétrica en el diseño de la PKI.
- La PKI deberá permitir a los usuarios de la misma la creación de claves para ambos esquemas de criptográfica asimétrica utilizando cualquiera de los tamaños de clave mostrados en la tabla 3.1, estando esto limitado únicamente por las capacidades del dispositivo de pruebas.

Requerimientos de certificación

Ahora se mencionarán las característica que deberán tener los certificados emitidos dentro de la PKI que implemente este diseño:

- La PKI deberá apegarse a uno o varios estándares para el desarrollo de la misma, como lo son X.509, PKCS y/o PGP.

- La PKI deberá permitir el uso de diversos modelos de confianza dejando al usuario final la decisión de que modelo utilizar, dependiendo de los requerimientos particulares de su aplicación. Particularmente se deberá contemplar los modelos jerárquico y de red de confianza.
- Los certificados creados con la API deberán seguir el estándar X.509 en su tercera versión, por lo cual la PKI deberá soportar las extensiones especificadas en éste.
- La PKI debe contemplar tanto el uso de CA internas como externas.
- Dentro de la PKI se debe contemplar el uso de certificados auto-firmados como mecanismo de comprobación de posesión de las claves tanto pública como privada.
- En el diseño de PKI se contemplará el uso de una base de datos para almacenar información de la PKI como: certificados, claves, CRLs y alias de usuarios.
- La PKI deberá permitir a los usuarios importar y exportar tanto las claves como los certificados utilizando estándares y codificaciones existentes para este propósito.
- Los certificados y llaves exportadas deberán ser compatibles con aplicaciones externas las cuales sigan los estándares utilizados.
- La PKI debe contemplar la delegación de responsabilidades, mediante el uso de certificados digitales. Por lo que será posible que un mismo usuario tenga más de un certificado, ya que cada uno de sus certificados podrá tener diferentes roles asignados.
- Los certificados creados por esta PKI deben tener un nivel de seguridad mínimo equivalente a 80 bits.
- En el esquema base de la PKI presentada, no se requiere de manera obligatoria la presentación de documentos físicos para poder emitir un certificado, sin embargo es recomendable que esto sea considerado como parte de las prácticas y políticas de certificación de un modelo jerárquico.
- El diseño de la PKI deberá contener un plan detallado para la administración de los certificados, tomando como base un modelo de confianza jerárquico, se recomienda que el plan presentado sea seguido por los usuarios de la infraestructura para optimizar su funcionamiento.
- El plan de administración contendrá los procesos de solicitud, emisión, revocación de certificados, además de la publicación de las listas de revocación de certificados y la generación de claves por parte de una CA.
- Adicionalmente a este plan de administración se deberá presentar como parte del diseño una lista de recomendaciones a seguir en la elaboración de aplicaciones que utilicen un modelo de confianza diferente, particularmente para la implementación de una red de confianza.

3.2.2. Modelo de confianza

A continuación se explicarán brevemente los dos modelos seleccionados, el modelo de red de confianza para el caso práctico de la aplicación de prueba y el modelo jerárquico para la presentación teórica de las políticas y prácticas de certificación. Para iniciar la definición de estos modelos de confianza es necesario definir las opciones de las CA que se utilizarán.

Dentro de la red de confianza no existe como tal una autoridad de certificación raíz, ya que en este modelo los usuarios son certificados entre sí y la validez de estos certificados es definida en base a otros factores como la confianza entre los diversos usuarios de la red. Por otra parte para el modelo jerárquico es básico designar quien será la autoridad certificadora raíz, el diseño de la PKI contempla tres tipos de CA:

- CA raíz: ésta es la encargada de emitir los certificados a las demás CA de la infraestructura.
- CA intermedia: las CA intermedias tienen los privilegios para emitir certificados tanto a CA emisoras como a los usuarios finales.
- CA emisora: por su parte estas entidades podrán emitir certificados únicamente a los usuarios finales de la PKI.

La elección de quien administrará la CA raíz de cada implementación de esta PKI queda a responsabilidad de los administradores de las aplicaciones que utilizarán el presente diseño, sin embargo como parte de este diseño se dan algunas recomendaciones:

- No dar el control completo de las claves de la CA raíz a una sola persona, es preferible dividir esta responsabilidad entre varias.
- Dentro de las responsabilidades a repartir están:
  - Emisión de certificados.
  - Emisión de CRL.
  - Análisis y aceptación de solicitudes de revocación.
  - Administración de plantilla de certificados.
- Utilizar un tamaño de equivalente a 192 bits de seguridad para las claves de CA raíz.
- Utilizar 128 bits de seguridad para las claves de las diversas CA intermedias.
- Finalmente utilizar al menos 112 bits de seguridad para las CA finales.

Como se mencionó en los requisitos de la PKI se debe contemplar el uso de certificados propios o certificados externos dentro de la infraestructura, teniendo entendido que los certificados externos tendrán algunas limitaciones en cuanto a las extensiones propias que los certificados creados dentro de la infraestructura si tendrán, estas extensiones contienen en su mayoría información referente al dispositivo móvil que fue utilizado para la

generación y firma de un certificado, por lo que el uso de certificados externos se recomienda únicamente, dentro de una PKI que utiliza el modelo jerárquico, para una CA raíz, de manera que los certificados emitidos por ésta sean válidos fuera de la aplicación de prueba, sin embargo en el caso de la red de confianza, el uso de estos certificados podría limitar las capacidades de verificación si el emisor de dicho certificado no es reconocido dentro de la red.

Una vez establecido lo anterior en la figura 3.6 se muestra un ejemplo de una estructura jerárquica diseñada utilizando la PKI que se elaboró, en el diagrama se puede observar cómo la CA raíz puede emitir diferentes tipos de certificados tanto para CA intermedias o para diversas entidades como por ejemplo una RA, la cual tiene la funcionalidad de realizar el proceso de registro dentro de la infraestructura. A su vez, las CA intermedias pueden emitir certificados a las CA del siguiente nivel (CA emisoras), a su propia RA o a usuarios finales, dependiendo de las necesidades de la organización. Además, se pueden observar a una serie de CA emisoras certificando a los usuarios finales. Es importante mencionar que en este diseño se contempla la delegación de responsabilidades ya que dentro de los certificados emitidos por una CA, ésta pueda incluir uno o varios de sus permisos dentro de la organización y de esta manera otorgar nuevos permisos al titular del certificado utilizando un determinado periodo de validez para esto. Por último, es importante notar que dentro de la PKI los usuarios pueden tener dispositivos asociados a un certificado, lo cual puede servir dentro de la implementación de la PKI como información adicional para los mensajes o archivos procesados dentro de la PKI, por ejemplo obtener la ubicación de donde se generó un certificado o un determinado archivo y verificar si los archivos recibidos fueron procesados en el dispositivo utilizado para crear el certificado o en algún otro dispositivo.

En el modelo jerárquico la validación de un certificado es relativamente sencilla, ya que como se puede ver en la figura 3.6, al irse formando la estructura, ésta forma un árbol, por lo que para validar un certificado únicamente se debe ir subiendo nivel a nivel hasta llegar a la CA raíz, si esto ocurre, se puede determinar que el certificado es válido.

Por otra parte en la figura 3.7 se puede observar un ejemplo de un modelo basado en una red de confianza, en el cual los usuarios se certifican entre sí, por lo que no existe una CA raíz como tal, sin embargo es posible utilizar los diferentes niveles de CA incluidos en este diseño al emitir un certificado para otro usuario y de esta manera evitar que el titular del nuevo certificado propague la firma del emisor a través de la red. Las diferentes entidades de la red emiten certificados a otras y de esta manera se puede considerar que el emisor avala que la clave contenida en el certificado efectivamente corresponde a la identidad almacenada en el mismo. En este ejemplo de red de confianza se pueden observar algunos casos en que las entidades se emiten certificados entre sí, lo cual es completamente válido en este tipo de redes.

En este tipo de modelo, la validación de los certificados es más complicada, por lo que a continuación se explica el mecanismo propuesto para realizar esta operación.

Este mecanismo toma como base la validación de claves de OpenPGP [Callas, 2008], donde una clave puede almacenar N firmas de entidades que avalan la autenticidad de la

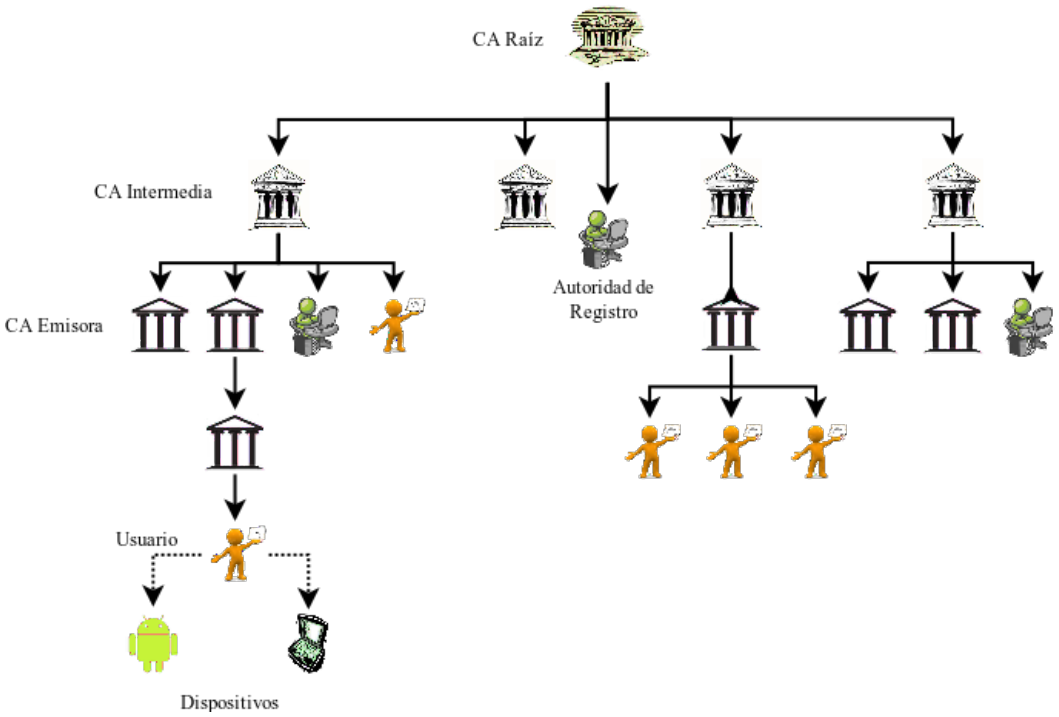


Figura 3.6: Ejemplo de un modelo de confianza jerárquico de la PKI.

misma con respecto a la identidad del usuario al cual pertenece. Sin embargo, el diseño propuesto utiliza certificados X.509, los cuales pueden almacenar únicamente una firma digital, por lo que solo se puede almacenar la información de un emisor, lo que hace al modelo de validación de PGP inadecuado para este diseño. Debido a lo anterior el mecanismo de validación propuesto se basa en la creación de listas de confianza, en las cuales cada usuario asigna un nivel de confianza a cada certificado almacenado en la base de datos.

Este nivel de confianza señala que tanto se confía en dicho certificado como emisor de otros, es decir, este nivel determina que tan válidos serán los certificados emitidos utilizando este certificado. OpenPGP señala el uso de 3 niveles básicos, sin embargo en [Caronni, 2000] se menciona que al utilizar una escala mayor, por ejemplo, de cero a diez es posible obtener un nivel de validez aun mejor, adicionalmente en [Guo et al., 2011] se hace mención a que el uso de una escala positiva para evaluar un certificado no es del todo apropiado, ya que no es posible marcar un certificado como inválido, o en este caso a un emisor como desconfiable, lo cual en el mundo real es muy útil si se llega a detectar que un emisor es utilizado para crear certificados falsos y esto es deseable que se refleje en la validez de un certificado. En base a lo anterior se propone que el mecanismo de validación tenga un intervalo de $[-10, 10]$ como los posibles valores para el nivel de confianza de un emisor en la lista. En este diseño se considera que un emisor tiene el nivel de confianza igual a cero por defecto, es decir si no se especifica por el propietario de la lista, ya que solo éste podrá modificar el valor.

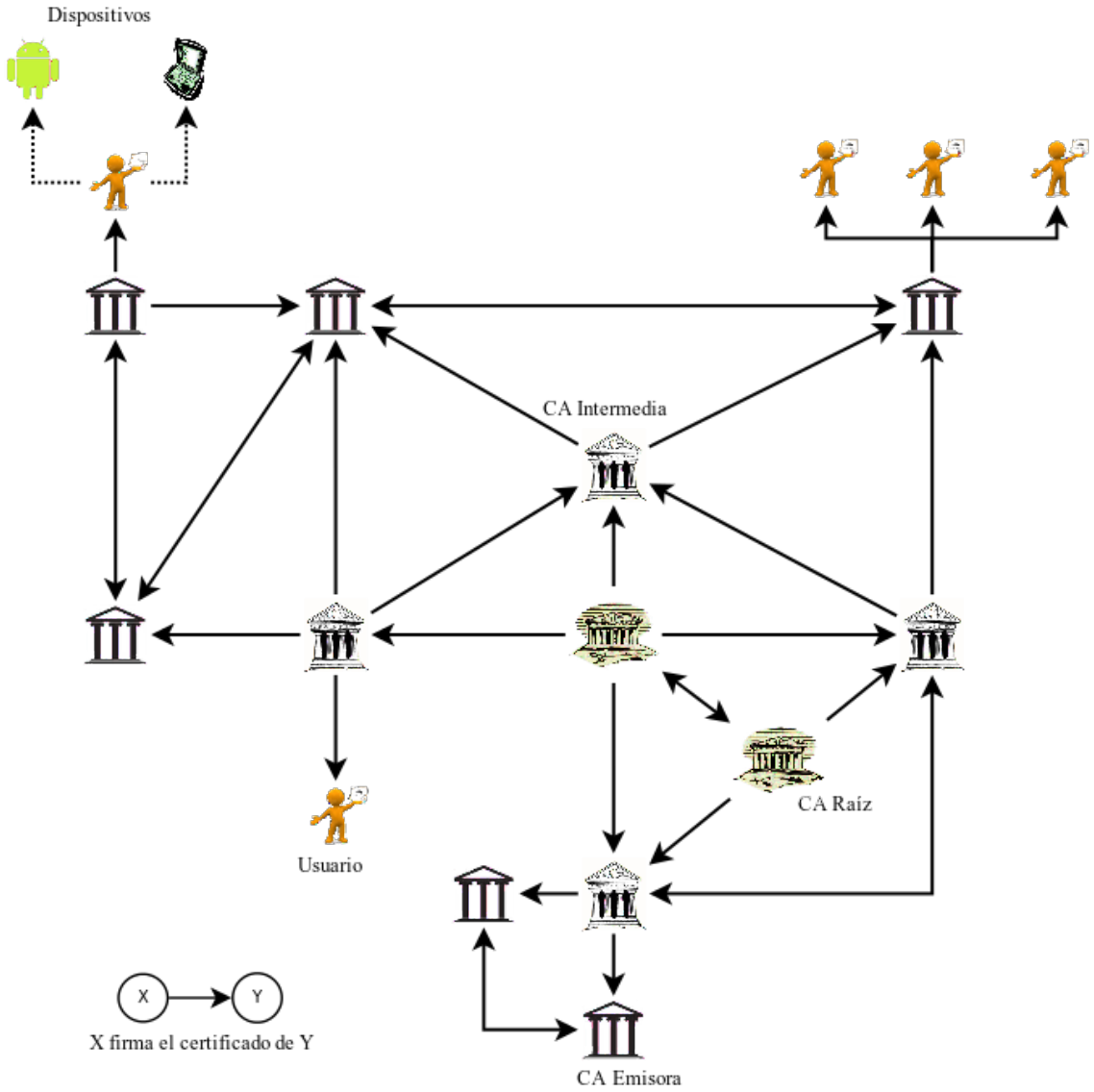


Figura 3.7: Ejemplo de un modelo de red de confianza de la PKI.

Como se mencionó anteriormente en PGP las claves pueden contener una o varias firmas las cuales avalan su autenticidad, sin embargo en el presente diseño se utilizan certificados X.509, los cuales no tienen esta bondad, por lo que es necesario tener otro mecanismo para verificar la autenticidad del certificado. Por otra parte como se ha mencionado, en la PKI es posible que un titular tenga diferentes certificados para una misma clave, teniendo como diferencia principal, al emisor del certificado y la información que éste pueda agregar al certificado antes de firmarlo. Tomando esto en cuenta, la propiedad de los certificados PGP puede ser emulada, si se busca dentro de la base de datos todos los certificados que contengan la misma clave y se hace una comparación utilizando los campos del certificado para verificar que se trate del mismo titular. Una vez se tiene esta lista de certificados, es posible obtener una lista con todos los emisores de estos certificados y al combinar esta última con la lista de confianza explicada anteriormente es posible determinar el nivel de confianza total de un certificado. Para obtener el nivel de confianza total de un certificado se deben obtener los valores de confianza para cada uno de los emisores que han firmado la clave y calcular un promedio entre estos valores. El valor obtenido representa el nivel de confianza que una entidad dentro de la PKI sobre el certificado en cuestión.

Finalmente, como se mencionó en la sección 3.1.2 el último paso para tener un modelo de confianza terminado es determinar donde se almacenará la información de dicho modelo, en este diseño se contempla el uso de una base de datos relacional en la cual sean almacenados todos los certificados utilizados dentro de la PKI, así como la información adicional a ellos, como por ejemplo, CRL, información de titulares, claves públicas y privadas (si es necesario) y las listas de confianza de cada usuario. Se recomienda almacenar todo el contenido los objetos utilizados como lo son certificados, claves y CRL en lugar de solo referencias a la ubicación física de los mismos para evitar errores por factores ajenos a la PKI, como por ejemplo, que los archivos cambien de ubicación por una reinstalación del sistema. Adicionalmente es importante mencionar que estos objetos serán almacenados en forma de cadenas en `Base64`, utilizando diferentes codificaciones según sea el caso:

- Certificado: codificados utilizando DER.
- Clave pública: DER como codificador de la clave.
- Clave privada: utilizando el estándar Estándar de Criptografía de Clave Pública (siglas del inglés *Public-Key Cryptography Standards*)) (PKCS)#8 con AES256 para proteger la clave .
- CRL: al igual que los certificados, se utiliza DER como codificador.

En el caso de la red de confianza, esta base de datos sera almacenada de manera local en cada dispositivo que instale la aplicación, utilizando *SQLite* como manejador de la misma y de esta forma cada usuario tendrá su pequeña red de confianza de manera local y la podrá ir extendiendo al intercambiar archivos con otros dispositivos. Para el caso del modelo jerárquico se recomienda que la información de la PKI sea almacenada en una base de datos centralizada utilizando *MySQL* como manejador ,la cual se encuentre en

posesión de la CA raíz y los clientes de la PKI únicamente tengan una pequeña copia de la base general, solo con los información que cada uno vaya requiriendo, seguir este esquema es un poco más complicado que el propuesto para la red de confianza, ya que se deben contemplar factores como la sincronización y actualización de datos entre los dispositivos móviles y el servidor, para lo cual se recomienda la implementación de servicios web con las operaciones necesarias.

3.2.3. Opciones de configuración de certificados

Una vez definido el modelo de confianza para la PKI, se definió la plantilla para crear los certificados, la cual tiene como base un certificado X.509 [Cooper et al., 2008], por lo que contiene los campos base del mismo y algunas extensiones propias de esta PKI.

Plantilla para certificados

Los certificados emitidos por la PKI propuesta deben contener al menos los siguientes campos:

- Versión: estándar X.509 versión 3.
- Número de Serie: entero positivo, de 20 octetos de longitud, este número conforma una tupla de identificación única para el certificado en conjunto con la CA y el dispositivo de firma.
- Algoritmo de firma: identificador del algoritmo criptográfico usado por el emisor para firmar el certificado. Los algoritmos de firma soportados aparecen listados en [Polk et al., 2002, Leontiev and Shefanovski, 2006, Schaad et al., 2005]. Sin embargo otros algoritmos pueden ser utilizados.
- Emisor: identifica a la entidad que firmó y emitió el certificado. Este campo debe contener un nombre distinguido (DN) no vacío. El nombre está compuesto por una serie de atributos, como por ejemplo, país y su correspondiente valor (MX). El estándar señala que las implementaciones deben reconocer al menos los siguientes atributos para los campos de emisor y titular:
  - País,
  - organización,
  - unidad organizacional,
  - calificador DN,
  - estado o provincia,
  - nombre común y
  - número de serie

Adicionalmente se recomienda que las implementaciones de este estándar reconozcan los siguientes atributos:

- Localidad,
- titulo,
- apellido,
- nombre de pila,
- iniciales,
- seudónimo y
- calificador generacional (ej., "Jr.", "3rd", or "IV").

- Validez: el periodo de validez, es un periodo de tiempo durante el cual la CA garantiza que mantendrá información acerca del estatus del certificado. Este campo representa una secuencia de dos fechas: la fecha en la cual el periodo de validez inicia (No antes de) y la fecha en la cual el periodo termina (No después). Ambas fechas deben ser codificadas utilizando *UTCTime*.
- Titular: este campo identifica a la entidad asociada con la clave pública almacenada en el certificado. el nombre del titular puede encontrarse tanto en el campo de titular como en la extensión *subjectAltName*. Si el titular es una CA raíz, el campo debe contener un nombre distinguido (DN) igual al contenido en el campo de emisor.
- Información de clave pública: este campo es utilizado para almacenar la clave pública y el identificador del algoritmo con el cual la clave se utiliza.
- Extensiones: este campo debe aparecer en certificados de la versión 3, si esta presente es una secuencia de extensiones, el formato y contenido de las extensiones utilizadas en esta PKI se encuentran definidas en la sección 3.3.6 de este documento.
- Firma: contiene la firma digital del certificado. Al generar esta firma, el emisor certifica la validez de la información contenida en el certificado, en particular certifica el vínculo entre la clave pública y la información del titular.

Una vez definida la plantilla del certificado, es necesario establecer los parámetros de seguridad que serán utilizados en los certificados de la PKI.

Parámetros de seguridad

Los certificados utilizados en la PKI pueden utilizar dos mecanismos criptográficos: RSA y EC. En ambos casos, estos algoritmos serán utilizados en conjunto con otros para realizar las operaciones de firma/verificación y cifrado/descifrado. Particularmente se utilizan en conjunto con AES y con funciones picadillo para hacer más eficientes y robustas las operaciones. En la sección 2.1.4 de este documento se presentaron esquemas básicos

para estas operaciones, a continuación se explicarán los esquemas utilizados en la PKI para llevar a cabo dichas operaciones.

En la figura 3.8 se puede observar el esquema de firma digital utilizado en esta PKI. Tiene algunos cambios con respecto al esquema original: en este esquema, si Alicia desea firmar un documento o de forma general un flujo de datos cualquiera, es necesario que primero se obtenga el picadillo de este flujo y este último sera la entrada (junto con la clave privada de Alicia) al algoritmo de firma digital, el cual entrega como salida el picadillo firmado. Por otra parte si Beto desea verificar la firma de Alicia, debe obtener primero el picadillo del flujo de datos original, este picadillo será la entrada, junto con la clave pública de Alicia al algoritmo de verificación, el cual comparará el picadillo obtenido de la firma con el picadillo obtenido del flujo original. Si éstos son iguales, se puede concluir que la firma es correcta.

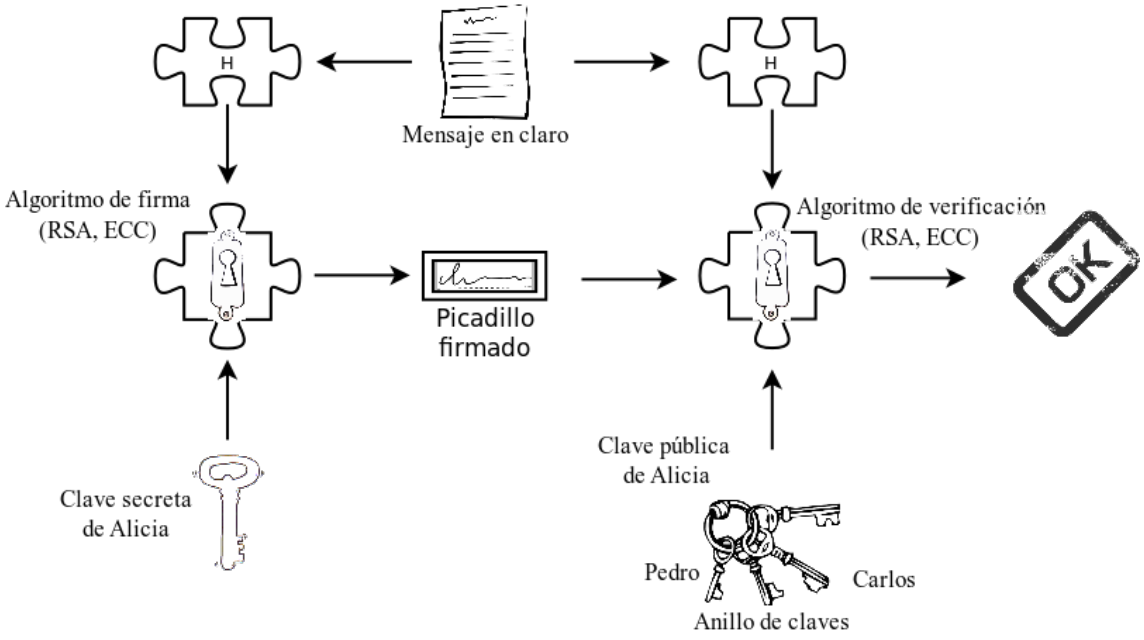


Figura 3.8: Esquema implementado de criptografía asimétrica - Firma digital.

De igual manera en la figura 3.9 se presenta el esquema de cifrado asimétrico utilizado en la PKI, en este caso se utiliza una combinación entre un algoritmo asimétrico y uno simétrico para obtener las ventajas de ambos. En este esquema es necesario generar una clave de sesión, la cual será utilizada como entrada para el algoritmo simétrico (AES para el caso de la PKI) que junto con el mensaje en claro generará el mensaje cifrado. Por otra parte la clave de sesión será cifrada utilizando el algoritmo asimétrico y la clave pública del destinatario, en este caso Alicia. Es importante mencionar que Beto necesita enviar tanto la clave de sesión cifrada como el mensaje cifrado a Alicia, para que ésta a su vez descifre la clave de sesión utilizando su clave pública y el algoritmo de descifrado asimétrico y una vez obtenga la clave de sesión original, utilizar el algoritmo simétrico de descifrado para obtener finalmente el mensaje en claro.

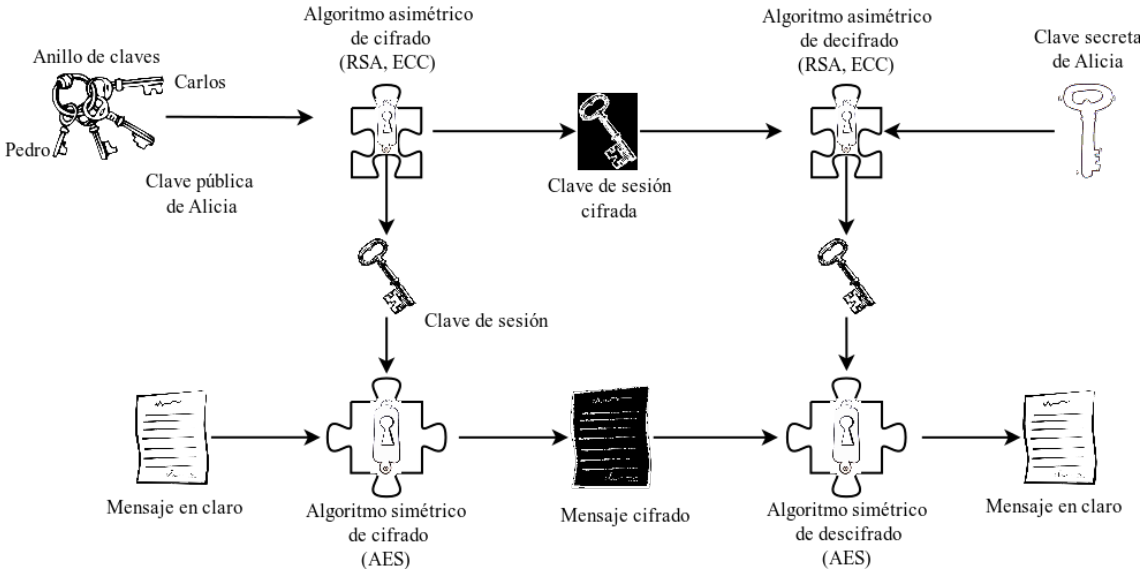


Figura 3.9: Esquema implementado de criptografía asimétrica - Cifrado.

Adicionalmente de los esquemas de seguridad a utilizarse dentro de la PKI, es necesario definir el tamaño de claves que se utilizaran en dichos esquemas, como se mencionó en los requerimientos de está PKI, no existe ninguna restricción impuesta en el diseño en cuanto al tamaño de claves que los usuarios podrán utilizar, siempre y cuando se respeten las recomendaciones para las diferentes tipos de CA mencionadas al inicio de está sección. De manera similar se deja a consideración de los emisores seleccionar el periodo de validez más apropiado al momento de firmar un certificado, ya que este periodo de validez deberá establecerse en base cuando tiempo se desea que el titular del certificado tenga acceso a los permisos otorgados en el mismo.

3.3. Declaración de políticas de certificación

En esta sección se presenta una propuesta de Declaración de Políticas de Certificación (DPC) las cuales tiene como finalidad regir el funcionamiento y operaciones de la PKI jerárquica presentada en este trabajo. Sin embargo las normas y políticas establecidas en esta sección tienen como objetivo principal servir como base para la elaboración de políticas propias, por lo cual podrán ser modificadas y adaptadas según las necesidades de la implementación final de la PKI.

Esta DPC se aplica a todos los que intervienen relacionados con la jerarquía de la PKI, incluyendo a las autoridades de certificación, autoridades de registro, solicitantes, titulares y terceros aceptantes.

3.3.1. Introducción

La presente DPC, se ha estructurado conforme a lo dispuesto por el estándar europeo "*ETSI TS 101 042: Policy requirements for certification authorities issuing public key certificates*" [Barreira and Compans, 2011] con el fin de dar uniformidad al documento y de esta manera facilitar su análisis y lectura. Este documento declara las políticas de servicios, la declaración del nivel de garantía ofrecido, mediante la descripción de las medidas técnicas y organizativas establecidas para garantizar el nivel de seguridad de la PKI así como todos los mecanismos de gestión de certificados durante su ciclo de vida.

En la redacción de esta DPC se asume que el lector ha leído los conceptos básicos presentados en el capítulo 2, por lo cual conoce los conceptos de PKI, certificado y firma electrónica; en caso contrario se recomienda la lectura de dicho capítulo, con el fin de tener un mejor entendimiento de esta sección. En la figura 3.10 se muestra la arquitectura general, a nivel jerárquico, de la PKI en la cual se pueden ver cuatro entidades principales, cada una con capacidades y obligaciones diferentes entre sí.

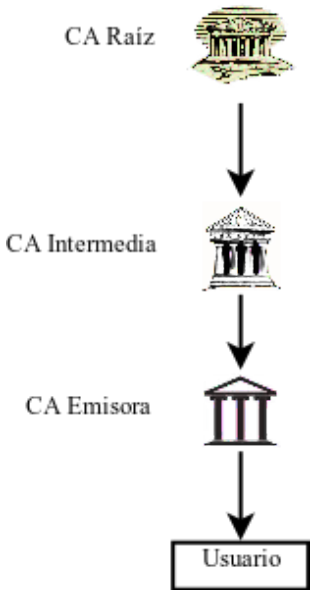


Figura 3.10: Arquitectura general de la PKI.

Entidades y personas que intervienen

Las entidades y personas que intervienen son:

- La Autoridad de Administración de Políticas (APP): es la organización establecida como responsable de la administración de la declaración de políticas de certificación y de las políticas de la PKI. Asimismo, la APP es la encargada (en caso de que se tenga que evaluar la posibilidad de que una CA externa interactúe con la PKI) de

determinar la adecuación de la declaración de políticas de la CA externa a la Política de Certificación afectada. Finalmente, es responsable de analizar los informes de las auditorías, totales o parciales, que se hagan de PKI, así como determinar, en caso necesario, las acciones correctoras a ejecutar.

- Las Autoridades de Certificación: son entidades encargadas de la emisión de certificados. Así mismo tienen la responsabilidad de realizar la renovación y revocación de los mismos y en caso de ser necesario de la generación de las claves pública y privada de los usuarios finales. Las Autoridades de Certificación que son contempladas en esta PKI en cada uno de sus niveles son:

  - CA Raíz: Autoridad de Certificación de primer nivel. Esta CA podrá emitir certificados únicamente para sí misma, sus CA subordinadas y para su RA. Únicamente deberá estar en funcionamiento durante la realización estas operaciones.

  - CA intermedia: Autoridad de Certificación subordinada de la CA raíz. tiene la obligación de emitir los certificados tanto para las CA emisoras como a usuarios finales.

  - CA emisora: es la CA de más bajo nivel y solo emitir certificados a usuarios finales de la PKI.

- Las Autoridades de Registro (RA): son las entidades encargadas de la verificación de la identidad de los solicitantes de certificados y, en caso de ser necesario, de los atributos asociados a los mismos. Las RA llevarán a cabo la identificación de los solicitantes de certificados conforme a las normas de la declaración de políticas de certificación y el acuerdo suscrito con la CA.

- El archivo de Claves: las políticas de certificación podrán establecer la existencia de un archivo de claves, el cual permite el almacenamiento recuperación de las claves privadas de los titulares. Dicho archivo deberá garantizar la confidencialidad de la clave privada y su recuperación deberá exigir, como mínimo, la intervención de dos entidades.

- Los solicitantes y titulares de los certificados emitidos por PKI: un solicitante es toda aquella entidad que realice una petición para la generación de un certificado dentro de la PKI, por otra parte se denomina titular de un certificado a toda entidad para la que se emite un certificado en la PKI.

- Los terceros aceptantes de los certificados emitidos por la PKI: son las personas o entidades diferentes del titular que deciden aceptar y confiar en un certificado emitido por la PKI.

Uso de los certificados

Los certificados emitidos dentro de la PKI solamente podrán ser utilizados por el titular del mismo, para autenticarse contra los sistemas diseñados dentro de la PKI, generar firmas electrónicas y cifrar información de forma que ésta sea accesible solo para el titular del mismo.

Los certificados de usuarios finales, no pueden utilizarse para actuar ni como Autoridad de Registro ni como Autoridad de Certificación, es decir, no se pueden utilizar para firmar certificados de clave pública a otras entidades dentro de la infraestructura, ni listas de revocación de certificados (CRL).

Limitaciones y restricciones en el uso de los certificados Los certificados únicamente se utilizarán para el fin contemplado en el apartado anterior. La utilización de los certificados emitidos dentro de la PKI podrán ser utilizados en otras infraestructuras, quedado este uso restringido por las normativas impuestas en dichas infraestructuras, así mismo el emisor del certificado en cuestión no tiene responsabilidad alguna sobre el uso de los certificados, cuando éste sea utilizado fuera de la PKI.

Administración de las políticas

Persona de contacto La persona de contacto así como el administrador de las políticas de certificación deberá ser definido claramente para cada implementación de la PKI propuesta, es necesario que incluya al menos una dirección de correo electrónico, teléfono y dirección de contacto de dicha entidad.

Determinación de la adecuación de la DPC de una CA externa a las Políticas de Certificación de PKI En el caso de que se tuviese que evaluar la posibilidad de que una CA externa interactúe con la PKI, la Autoridad de Administración de Políticas es la responsable de determinar la adecuación de la DPC de la CA externa a las políticas afectadas. Los procedimientos para determinar la adecuación se recogen en la PC que tenga prevista la posibilidad de operar con otras CA.

Procedimiento de aprobación de esta DPC Es deber de las implementaciones finales de la PKI establecer el mecanismo de aprobación para su propia declaración de políticas de certificación, ya que como se ha mencionado, la presente DPC tiene como intención servir como base para la creación de políticas propias en cada implementación.

Definiciones y acrónimos Las definiciones y acrónimos requeridos se pueden encontrar en el capítulo 2 y en la sección respectivamente.

3.3.2. Repositorios y publicación de la información

Repositorios

El repositorio de la PKI está compuesto por una base de datos relacional centralizada, un sistema de acceso mediante servicios web y una página Web con acceso libre, la ubicación y mecanismos de acceso deberán ser definidos para cada implementación.

Publicación de información de certificación

Es obligación de las CA pertenecientes a la PKI notificar a la CA raíz la información relativa a los certificados y CRL emitidos, para que ésta pueda actualizar la información en el repositorio central. Esta información deberá ser de carácter público, por lo que deberán existir mecanismos de acceso y consulta a está información. El mecanismo de consulta propuesto es la publicación de servicios web mediante los cuales entidades externas e internas puedan acceder a información relevante como lo es: el estatus de un certificado, CRLs y los certificados emitidos.

Controles de acceso a los repositorios

El acceso para la lectura a las DPC es abierto, pero sólo la entidad designada dentro de la PKI está autorizada a modificar, sustituir o eliminar información de su repositorio y sitio web. Para ello la PKI deberá establecer controles que impidan a personas no autorizadas manipular la información contenida en los repositorios.

3.3.3. Obligaciones de los titulares

Dentro de la DPC de la infraestructura es necesario establecer las obligaciones para los titulares de los certificados emitidos, éstas son:

1. Suministrar información exacta, completa y veraz con relación a los datos que los encargados de su verificación les soliciten para realizar el proceso de registro.
2. Informar a los responsables de la PKI de cualquier modificación de esta información.
3. Conocer y aceptar las condiciones de utilización de los certificados, en particular las contenidas en la DPC, así como las modificaciones de las mismas.
4. Solicitar inmediatamente la revocación de un certificado en el caso de detección de inexactitudes en la información contenida en el mismo o tener conocimiento o sospecha del compromiso de la clave privada correspondiente a la clave pública contenida en el certificado.

5. No monitorizar, manipular o realizar actos de ingeniería inversa sobre la implantación técnica (hardware y software) de los servicios de certificación.
6. No transferir ni delegar a un tercero sus responsabilidades sobre un certificado que le haya sido asignado.

3.3.4. Identificación y autenticación de los titulares de certificados

Nombres

Los certificados emitidos por la PKI contienen Nombre Distintivo (siglas del inglés *Distinguished Name*) (DN) conforme al estándar X.500, tanto del emisor como del titular del certificado en los campos *issuer name* y *subject name* respectivamente.

En todos los casos los nombres distintivos de los certificados han de ser significativo, el DN de los certificados no puede estar repetido. La utilización del código único de usuario garantiza la unicidad del DN.

Validación de la identidad inicial

En caso de que el par de claves sea generado por el solicitante del certificado, la posesión de la clave privada, correspondiente a la clave pública para la que solicita que se genere el certificado, quedará probada mediante el envío de la petición de certificado (CSR), la cual consiste de un certificado auto-firmado el cual contiene la clave pública firmada mediante la clave privada asociada.

El mecanismo de autenticación de la identidad inicial, deberá ser especificado por cada implementación de esta PKI, quedando como base dos mecanismos principales: autenticación presencial, en la cual el solicitante se deberá presentar físicamente con la RA para iniciar la solicitud del certificado; autenticación remota, la cual se recomienda únicamente para entidades que cuenten con un certificado válido previo, esta autenticación se hará mediante la firma electrónica de la información solicitada por la RA.

Información no verificada sobre el solicitante

Toda la información recabada en el apartado anterior ha de ser verificada.

Comprobación de las facultades de representación

No estipulado al no estar contemplada la emisión de certificados para personas jurídicas.

Criterios para operar con CA externa

Antes de establecer relaciones de autoridades de certificación externas es necesario establecer ciertos requisitos básicos que esta CA debe cumplir para garantizar que los certificados emitidos por ésta no afectaran el funcionamiento general de la PKI, los requisitos básicos son:

- La CA externa ha de proporcionar un nivel de seguridad en sus los certificados, a lo largo de su ciclo de vida, como mínimo, igual al de la PKI.
- Confirmar la autenticidad de la CA en cuestión.

3.3.5. Requisitos operacionales para el ciclo de vida de los certificados

En esta sección se presenta el plan de administración de certificados, el cual contempla los procesos de solicitud, emisión, renovación y revocación de certificados.

Solicitud de certificados

La solicitud del certificado no implica su obtención si el solicitante no cumple los requisitos establecidos en la DPC. El Administrador de la PKI podrá recabar del solicitante la documentación que considere oportuna.

El registro de las solicitudes se hace cuando el solicitante completa un formulario del cual no todos los datos aparecerán en el certificado, pero quedan almacenados en el repositorio central de la PKI. En caso de que el solicitante tenga su par de llaves, deberá incluir la llave pública en la solicitud.

El proceso general de registro de una solicitud para la obtención de un certificado cuenta con dos mecanismos principales, queda a consideración del usuario elegir el más adecuado en cada caso y realizar las modificaciones pertinentes. El primer mecanismo es la solicitud de certificado donde el solicitante tiene un par de claves, este proceso se muestra en la figura 3.11 y se detalla a continuación:

1. El solicitante llena el formulario de solicitud y lo envía a la RA correspondiente.
2. La RA lleva a cabo la autenticación de la identidad del solicitante y activa una solicitud de pre-registro en la PKI.
3. La PKI deberá efectuar una autorización a dos pasos, en el primero envía una solicitud de comprobación al solicitante, el cual deberá responder mediante el envío de un certificado auto-firmado, el cual comprueba su posesión de la clave privada asociada a la clave pública que desea agregar en el certificado, en el segundo paso la

PKI deberá solicitar la autorización a la CA correspondiente para la expedición del certificado solicitado.

4. Una vez recibidas ambas autorizaciones, la PKI deberá generar el certificado con la información recabada por la RA y deberá enviar dicho certificado a la CA correspondiente junto con una firma digital propia para garantizar que el certificado no sea modificado durante el proceso.

5. La CA firma el certificado correspondiente y lo envía a la PKI para que sea almacenado.

6. La PKI envía el nuevo certificado al solicitante, el cual en este momento se convierte en titular.

7. Por último el titular del certificado regresa a la PKI un acuse de recibo correspondiente al certificado que acaba de recibir, este acuse deberá estar firmado digitalmente.

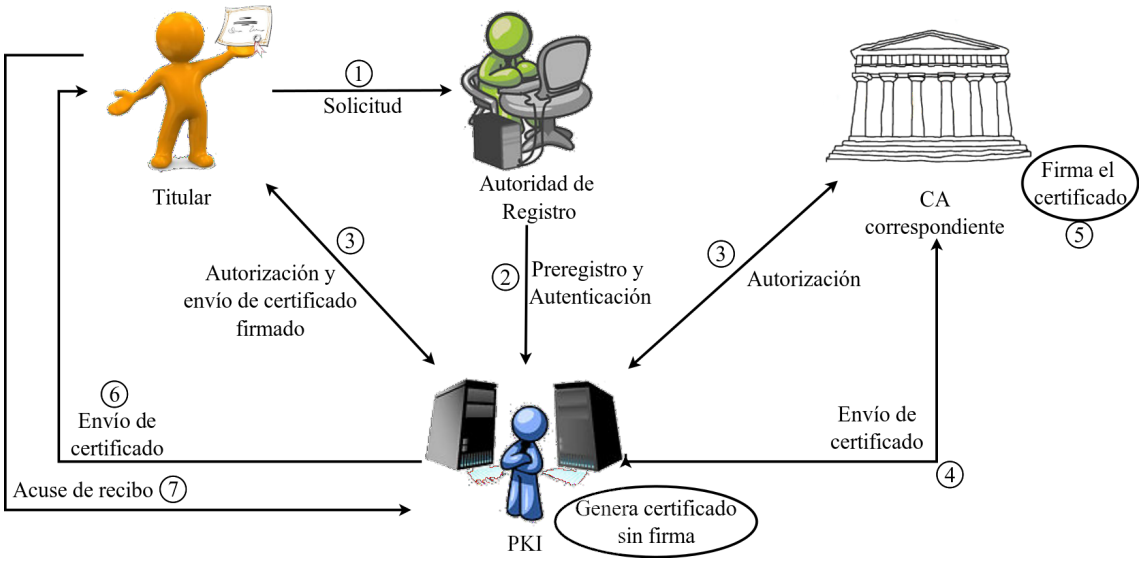


Figura 3.11: Proceso de solicitud de certificado con claves propias.

El segundo proceso corresponde a la solicitud de certificados donde el solicitante requiere que se genere un par de claves a la par se crea su certificado, se puede observar en la figura 3.12, este proceso es muy similar al anterior, teniendo como diferencia principal el la forma de autenticar la solicitud con el solicitante y la generación de claves que se realiza en la PKI, a continuación se detalla este proceso:

1. El solicitante llena el formulario de solicitud y lo envía a la RA correspondiente.

2. La RA lleva a cabo la autenticación de la identidad del solicitante y activa una solicitud de pre-registro en la PKI. En este punto la RA deberá proporcionar al solicitante un número de solicitud y pedirle que seleccione una contraseña para iniciar el proceso.

3. La PKI deberá efectuar una autorización a dos pasos, en el primero envía una solicitud de comprobación al solicitante, el cual deberá responder en el portal web de la PKI utilizando la contraseña que ingreso al iniciar el proceso, en el segundo paso la PKI deberá solicitar la autorización a la CA correspondiente para la expedición del certificado solicitado.
4. Al momento de recibir la autorización del solicitante, la PKI deberá generar el par de claves, las cuales serán almacenadas en formato PKCS#12 utilizando la contraseña ingresada por el solicitante, para garantizar su confidencialidad.
5. Una vez generadas las claves y sea recibida la autorización de la CA, la PKI generará el certificado con la información recabada por la RA y enviará dicho certificado a la CA correspondiente junto con una firma digital propia para garantizar que el certificado no sea modificado durante el proceso.
6. La CA firma el certificado correspondiente y lo envía a la PKI para que sea almacenado.
7. La PKI envía el nuevo certificado al solicitante, el cual en este momento se convierte en titular.
8. Por último el titular del certificado regresa a la PKI un acuse de recibo correspondiente al certificado que acaba de recibir, este acuse deberá estar firmado digitalmente.

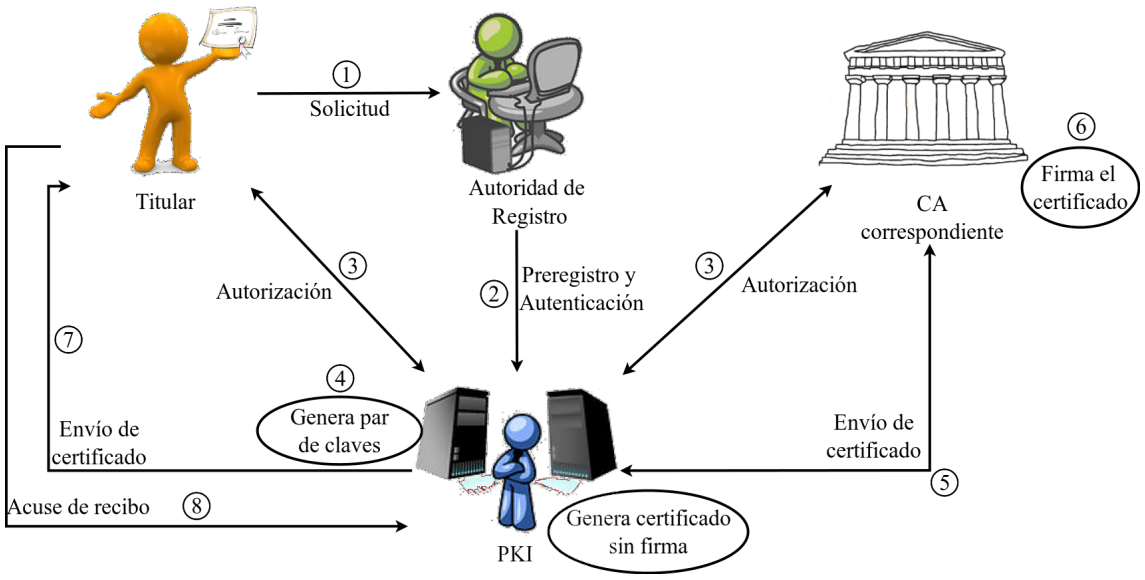


Figura 3.12: Proceso de solicitud de certificado con generación de nuevas claves.

Tramitación de solicitudes de certificados

- Realización de las funciones de identificación y autenticación: puede ser de dos maneras en función del tipo de solicitud.

  - Emisión inicial o renovación por pérdida: en todos estos casos la identificación y autenticación la realiza la RA.
  - Renovación remota: la identificación y autenticación se efectúa electrónicamente utilizando el certificado en vigor del titular. Este tipo de renovación se alterna con la anterior.

- Aprobación o denegación de las solicitudes de certificados: la emisión del certificado tendrá lugar una vez que la PKI haya llevado a cabo las verificaciones necesarias para validar la solicitud de certificación. La CA puede negarse a emitir un certificado de cualquier solicitante basándose exclusivamente en su propio criterio, sin que ello implique contraer responsabilidad alguna por las consecuencias que puedan derivarse de tal negativa.

Emisión de certificados

La emisión del certificado implica la autorización definitiva de la solicitud por parte de la CA. Cuando la CA de la PKI emita un certificado de acuerdo con una solicitud de certificación efectuará las notificaciones correspondientes. Todos los certificados iniciarán su vigencia en el momento de su emisión, salvo que se indique en los mismos una fecha y hora posterior a su entrada en vigor, que no será posterior a los 15 días naturales desde su emisión. El periodo de vigencia estará sujeto a una posible extinción anticipada, temporal o definitiva, cuando se den las causas que motiven la suspensión o revocación del certificado. El solicitante conocerá la disponibilidad del certificado mediante correo electrónico.

Aceptación del certificado

Al aceptar un certificado el titular debe señalar que esta de acuerdo con los términos y condiciones del mismo y da inicio sus obligaciones con respecto a la PKI, el certificado se publicará en el repositorio de la PKI.

Par de claves y uso del certificado

- Clave privada: el titular sólo puede utilizar la clave privada y el certificado para los usos autorizados en la política de certificación y de acuerdo con lo establecido en los campos “Key Usage” y “Extended Key Usage” del certificado. Del mismo modo, el titular solo podrá utilizar el par de claves y el certificado tras aceptar las condiciones de uso establecidas en la DPC y sólo para lo que éstas establezcan.
- Clave pública: los terceros aceptantes sólo pueden depositar su confianza en los certificados para aquello que establece la política de certificación y de acuerdo con lo establecido en el campo “Key Usage” del certificado. Los terceros aceptantes han de realizar las operaciones de clave pública de manera satisfactoria para confiar en el

certificado, así como asumir la responsabilidad de verificar el estado del certificado utilizando los medios que se establecen en la DPC y en la política de certificación. Asimismo, se obligan a las condiciones de uso establecidas en estos documentos.

Renovación de certificados

La renovación la debe solicitar el titular del certificado. Existen dos tipos de renovación dentro de la PKI:

- Sin cambio de claves: No soportada.
- Con cambio de claves: un certificado de puede ser renovado, entre otros, por alguna de las siguientes causas, expiración del periodo de validez, cambio de datos contenidos en el certificado, claves comprometidas o pérdida de fiabilidad de las mismas, cambio de formato, entre otras.

La CA comprobará en el proceso de renovación que la información utilizada para verificar la identidad y atributos del titular es todavía válida. Si alguna información del titular ha cambiado ésta deberá ser verificada y registrada con el acuerdo del titular. La identificación y autenticación para la renovación de un certificado de autenticación contempla de forma general, dos casos:

- Renovación por caducidad del certificado siendo la renovación anterior presencial: en este caso la renovación se podrá realizar de forma remota identificándose mediante un certificado en vigor de la PKI.
- Renovación por caducidad del certificado siendo la renovación anterior en línea o renovación por otras causas: en este caso la renovación se solicitará de forma presencial en los puestos de registro que se establezcan, de igual forma que en el caso de la emisión inicial.

A continuación se describen de forma detallada estos dos procesos mencionados, el figura 3.13 se observa la renovación remota, los pasos de este tipo de renovación son los siguientes:

1. La PKI generará y enviará una notificación por medio de un correo electrónico al titular informando que su certificado está por expirar, 90 días antes de que esto suceda.
2. El titular confirmará la renovación autenticándose en la PKI y solicitando la renovación.
3. La PKI generará el nuevo certificado, en este paso la PKI se comunica con la CA correspondiente para solicitar la firma digital del nuevo certificado.
4. La PKI enviará por correo electrónico el nuevo certificado al titular.
5. Al recibir el acuse de recibo por parte del titular, la PKI publicará el certificado en el repositorio.

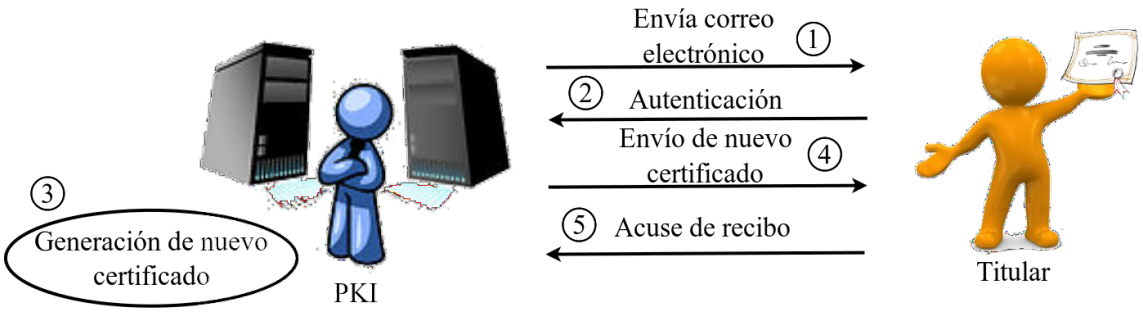


Figura 3.13: Proceso de renovación remota de certificados.

Por otra parte en la renovación presencial también participa la RA como se puede ver en la figura 3.14, el proceso para este tipo de renovación se especifica en lo siguientes pasos:

1. La PKI generará y enviará una notificación por medio de un correo electrónico tanto al titular como a la RA, informando que el certificado está por expirar, 90 días antes de que esto suceda.
2. El titular deberá presentarse con la RA y realizar una autenticación presencial.
3. La RA comprobará la identidad del titular y aprueba o deniega la renovación.
4. La PKI generará el nuevo certificado, en este paso la PKI se comunica con la CA correspondiente para solicitar la firma digital del nuevo certificado.
5. La PKI enviará por correo electrónico el nuevo certificado al titular.
6. Al recibir el acuse de recibo por parte del titular, la PKI publicará el certificado en el repositorio.

Modificación de certificados

Todas las modificaciones de certificados realizadas en el ámbito de esta política de certificación se tratarán como una renovación de certificados, por lo que son de aplicación los apartados anteriores al respecto.

Revocación y suspensión de certificados

La revocación de un certificado es el acto por el cual se inválida un certificado antes de su caducidad. El efecto de la revocación de un certificado es la pérdida de vigencia del mismo, originando el cese permanente de su operatividad conforme a los usos que le son propios y en consecuencia, de la prestación de los servicios de certificación. La revocación de un certificado impide el uso legítimo del mismo por parte del titular.

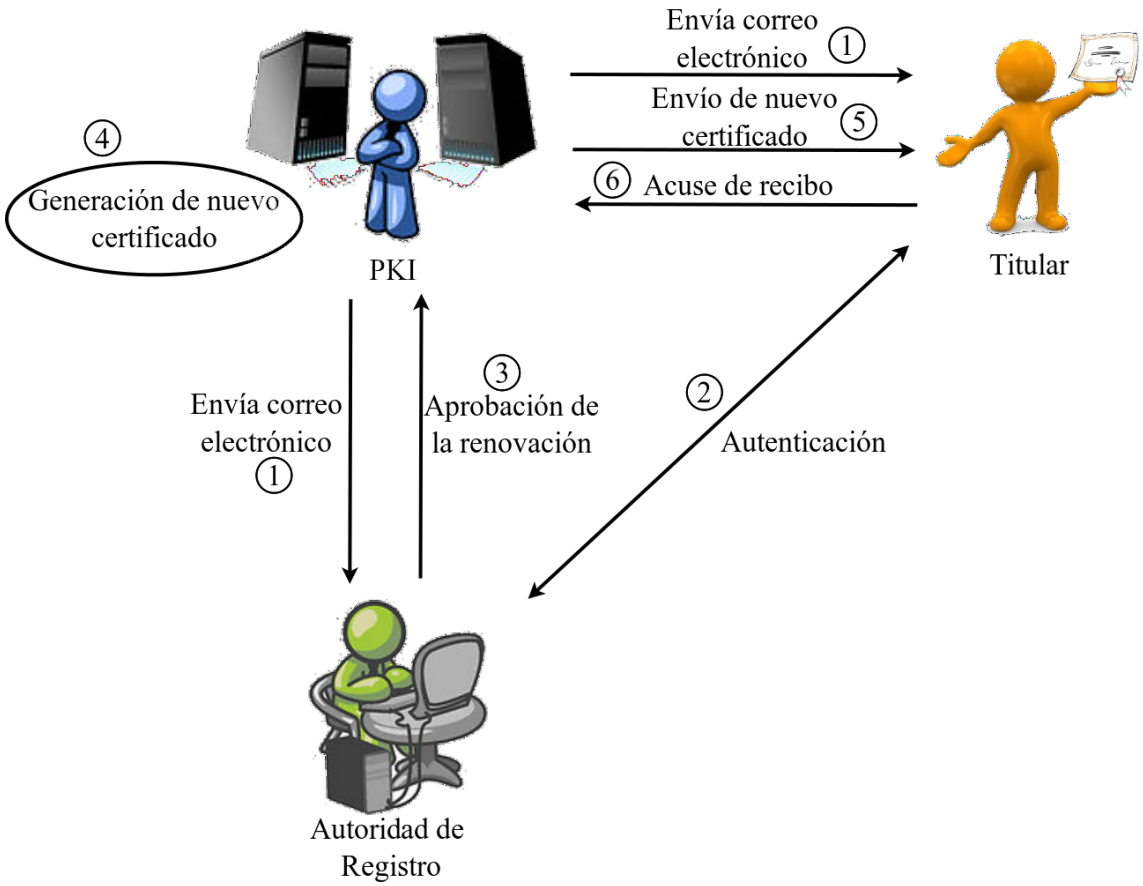


Figura 3.14: Proceso de renovación presencial de certificados.

La revocación de un certificado implica su publicación en la Lista de Revocación de Certificados (CRL) de acceso público. Un certificado puede ser revocado por:

- El robo, pérdida, revelación, modificación, u otro compromiso o sospecha de compromiso de la clave privada del titular.
- El mal uso deliberado de claves y certificados, o la falta de observancia o contravención de los requerimientos operacionales contenidos en el formulario de aceptación de las condiciones de los servicios de certificación de la autoridad de certificación, en la declaración de políticas de certificación.
- El titular de un certificado deja de pertenecer a la PKI.
- Cese de la actividades de la PKI.
- Emisión defectuosa de un certificado debido a que:
  - No se ha cumplido un requisito material para la emisión del certificado.
  - La creencia razonable de que un dato fundamental relativo al certificado es o puede ser falso.

  - Existencia de un error de entrada de datos u otro error de proceso.
- El par de claves generado por un titular se revela como "débil".
- La información contenida en un certificado o utilizada para realizar su solicitud es inexacta.
- Por solicitud formulada por el titular o por un tercero autorizado.
- El certificado de una CA superior en la jerarquía de confianza del certificado es revocado.

La revocación tiene como principal efecto sobre el certificado la terminación inmediata y anticipada del periodo de validez del mismo, convirtiendo a este certificado en no válido. La revocación no afectará a las obligaciones subyacentes creadas o comunicadas por la política de certificación ni tendrá efectos retroactivos.

Adicionalmente, los certificados revocados serán eliminados del directorio en el que estaban publicados. Cualquiera de las CA que componen la PKI pueden solicitar la revocación de un certificado si tuvieran el conocimiento o sospecha del compromiso de la clave privada del subscriptor, o cualquier otro hecho que recomendara emprender dicha acción.

Asimismo, los titulares de certificados también podrán solicitar la revocación de sus certificados, debiendo hacerlo de acuerdo con las condiciones especificadas.

El procedimiento de solicitud de revocación se resume en la figura 3.15 en la cual se pueden observar los siguientes pasos:

1. El titular o persona que solicite la revocación la debe presentar ante la PKI, identificándose e indicando la causa de la solicitud.
2. La CA tramitará las solicitudes de revocación de aquellos titulares que tenga asignados frente a la PKI.
3. La PKI deberá atender las solicitudes de revocación con la máxima celeridad y la revocación se llevará a cabo de forma inmediata a la tramitación de cada solicitud verificada como válida
4. La PKI enviará una notificación al solicitante cuando la revocación se haya realizado.

Las Listas de Revocación de Certificados (CRLs)

La verificación de las revocaciones es obligatoria para cada uso de los certificados de la PKI. Por lo que es responsabilidad de los terceros aceptantes comprobar la validez de la CRL antes de hacer uso de ella y en caso de ser necesario descargar la nueva CRL del repositorio de la PKI al finalizar el periodo de validez de la que posean.

La PKI proporciona una sección donde se encuentran las CRLs para la verificación del estado de los certificados que emite. Asimismo, existe deberá implementar el protocolo OSCP, para la verificación del estado de los certificados que se emiten en la infraestructura.

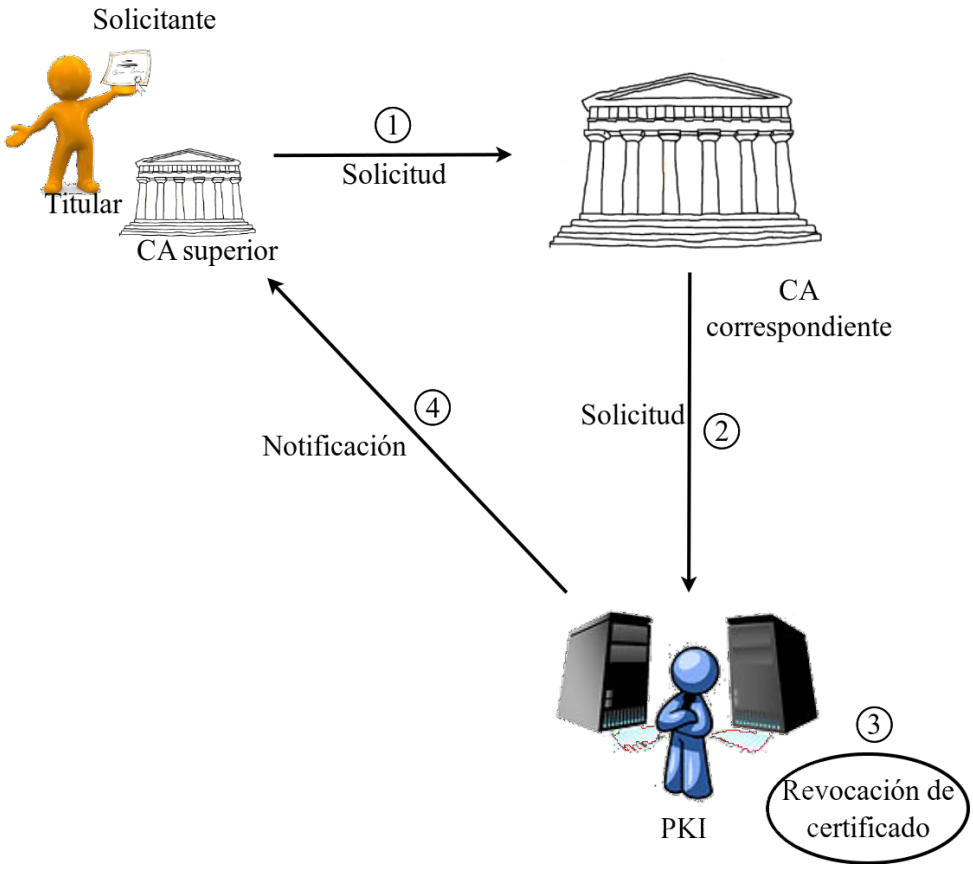


Figura 3.15: Proceso de revocación de certificados.

3.3.6. Perfiles de los certificados y CRL

En esta sección se presentaran los perfiles de los certificados y CRL utilizados en la PKI, los perfiles señalan de manera detallada los campos que tienen cada uno de estos elementos.

Perfil de certificado

La PKI soporta y utiliza certificados X.509 versión 3 (X.509 v3), las extensiones utilizadas de forma genérica son:

- *KeyUsage.* Calificada como crítica.
- *BasicConstraints.* Calificada como crítica.
- *SubjectAlternativeName.* Calificada como no crítica.
- *Subject Key Identifier.* Calificada como crítica.
- *Authority Key Identifier.* Calificada como crítica.

OID	Nombre	Descripción
1.3.6.1.4.1.40611.1	Id dispositivo	Identificador único del dispositivo donde se creo el certificado
1.3.6.1.4.1.40611.6	Id dispositivo de firma	Identificador único del dispositivo donde se firmó el certificado
1.3.6.1.4.1.40611.2.1	Latitud posición de creación	Latitud de la coordenada GPS que señala donde se creo el certificado
1.3.6.1.4.1.40611.2.2	Longitud posición de creación	Longitud de la coordenada GPS que señala donde se creo el certificado
1.3.6.1.4.1.40611.4	Documento de identificación	Documento utilizado para validar la identidad del titular (IFE, Cartilla, etc.)
1.3.6.1.4.1.40611.5	Permisos de usuario	Lista de permisos que tiene el titular en el sistema central
1.3.6.1.4.1.40611.3	Id de usuario	Identificador del titular en el sistema central
1.3.6.1.4.1.40611.7.1	Número de serie de la CA	Campo utilizado para identificar al emisor del certificado en una red de confianza
1.3.6.1.4.1.40611.7.2	Dispositivo de firma de la CA	Identificador único del dispositivo utilizado para firmar el certificado de la CA en la red de confianza
1.3.6.1.4.1.40611.7.3	Id clave de firma de la CA	Identificador de la clave utilizada para firmar el certificado de la CA

Tabla 3.3: Extensiones propietarias de la PKI.

- *extKeyUsage.* Calificada como no crítica.
- *Auth. Information Access.* Calificada como no crítica.

Las políticas de certificación de la PKI pueden establecer variaciones en el conjunto de las extensiones utilizadas por cada certificado.

La PKI tiene definida una política de asignación de Identificador de Objeto (siglas del inglés *Object Identifier*) (OID)'s dentro de su rango privado de numeración por la cual el OID de todas las extensiones propietarias de certificados de la PKI comienzan con el prefijo 1.3.6.1.4.1.40611. Este rango privado fue solicitado a la *The Internet Assigned Numbers Authority* (IANA), que es la organización encargada de la asignación de estos identificadores. La tabla 3.3 muestra las extensiones propietarias que están definidas en la PKI.

Identificadores de objeto (OID) de los algoritmos Identificador de Objeto (OID) de los algoritmos criptográficos soportados por los certificados son:

- SHA-1 with RSA Encryption (1.2.840.113549.1.1.5)
- SHA-224 with RSA Encryption (1.2.840.113549.1.1.14)

- SHA-256 with RSA Encryption (1.2.840.113549.1.1.11)
- SHA-384 with RSA Encryption (1.2.840.113549.1.1.12)
- SHA-512 with RSA Encryption (1.2.840.113549.1.1.13)
- MD2 with RSA Encryption (1.2.840.113549.1.1.2)
- MD5 with RSA Encryption (1.2.840.113549.1.1.4)
- RIPEMD-128 with RSA Encryption (1.3.36.3.3.1.3)
- RIPEMD-160 with RSA Encryption (1.3.36.3.3.1.2)
- RIPEMD-256 with RSA Encryption (1.3.36.3.3.1.4)
- SHA-1 with ECDSA (1.2.840.10045.4.1)
- SHA-224 with ECDSA (1.2.840.10045.4.3.1)
- SHA-256 with ECDSA (1.2.840.10045.4.3.2)
- SHA-384 with ECDSA (1.2.840.10045.4.3.3)
- SHA-512 with ECDSA (1.2.840.10045.4.3.4)

Formatos de nombres Los certificados emitidos por la PKI contienen el *distinguished name* X.500 del emisor y del titular del certificado en los campos *issuer name* y *subject name* respectivamente.

Restricciones de los nombres Los nombres contenidos en los certificados están restringidos a *distinguished names* X.500, que son únicos y no ambiguos.

Identificador de objeto (OID) de la Política de Certificación A definir en cada Política de Certificación. La PKI tiene definida una política de asignación de OID's dentro de su rango privado de numeración por la cual el OID de todas las Políticas de Certificación de la PKI comienzan con el prefijo 1.3.6.1.4.1.40611.

Perfil de CRL

La PKI soporta y utiliza CRLs X.509 versión 2 (v2). Las extensiones utilizadas en las listas de revocación son:

- *CRLNumber.* Calificada como no crítica.
- *AuthorityKeyIdentifier.* Calificada como no crítica.
- *IssuingDistributionPoint.* Calificada como no crítica.

Capítulo 4

Implementación y pruebas sobre la plataforma móvil

La implementación del diseño se realizó en una plataforma móvil, para ser más especifico, se utilizó *Android* como plataforma principal de desarrollo y pruebas.

En el presente capítulo se darán los detalles referentes a la implementación de una red de confianza conforme al diseño explicado en el capítulo 3, primero se describirán los diferentes diagramas que se utilizaron a lo largo del trabajo, como lo son los diagramas de paquetes, diagramas de clases y diagrama entidad relación. En estos diagramas se verán reflejados los aspectos del diseño presentado así como algunos detalles importantes acerca de la implementación de la PKI. Y finalmente, se detallarán las pruebas de funcionalidad y rendimiento que se hicieron para validar el diseño presentado.

Este capítulo está dirigido, por su puesto a un lector de interés general, pero de manera principal a un lector con experiencia en programación que quisiera tomar el desarrollo aquí presentado y realizar una ampliación posterior.

4.1. Diseño del API

La etapa de diseño consiste en traducir los requerimientos del sistema a una representación de software. Es el primer paso en la fase de desarrollo de cualquier producto o sistema. En esta sección se describirán cada uno de los diagramas utilizados durante el diseño de la API, la cual representa la implementación del diseño explicado en el capítulo anterior. Estos diagramas se realizaron utilizando el modelo UML.

4.1.1. Diagrama "entidad-relación"

Para comenzar se encuentra el diagrama entidad-relación, que es una herramienta que permite representar las entidades relevantes de un sistema así como sus relaciones y atributos.

En la figura 4.1, representa el modelo entidad-relación de la PKI en el dispositivo móvil. Cada una de las entidades mostradas en este diagrama representa una tabla en la base de datos creada, con sus atributos y relaciones. Como se puede observar en el diagrama, la base de datos consta de cinco tablas:

- `PersonalKey`: en esta tabla se almacenan todas las claves del sistema, se pueden almacenar claves públicas o privadas tanto de RSA como de EC. Dentro de sus atributos se encuentra `dat_key` en el cual se almacena la clave en sí, en forma de una cadena en `Base64`, adicionalmente en esta tabla se almacena la clave única de la clave (`kid_key`), el tipo de clave (`typ_key`), el identificador en la base de datos `ide_key`, un comentario (`com_key`) y la fecha en que se agrego la clave a la base de datos (`dte_key`).

- `CRL`: sirve para almacenar las CRL que sean importadas o creadas en el dispositivo. En la columna `crl_crl` se almacena una cadena codificada en `Base64` la cual representa a la CRL. Además esta tabla almacena el número de serie de la lista en `num_crl`, la fecha de creación `dat_crl`, una breve descripción de la lista (`des_crl`), entre otros.

- `Certificate`: esta es la entidad central del modelo, ya que representa a los certificados almacenados en la base de datos. Al igual que las entidades anteriores, cuenta con un atributo para almacenar el certificado en forma de una cadena en `Base64`, este campo se llama `dat_cer`, adicionalmente, esta entidad cuenta con atributos para el número de serie (`num_cer`), identificador en la base de datos (`ide_cer`), identificador del certificado de la CA (`ca_cer`), un campo para almacenar un estatus (`sta_cer`), el cual es actualizado cada vez que se verifica un certificado, por lo que es necesario almacenar la fecha de la última actualización de este estatus (`upd_cer`), para finalizar existen dos campos extra, el primero para almacenar el identificador del dispositivo utilizado para crear el certificado (`dev_cer`) y el último para guardar el *subject id* del certificado (`key_cer`).

- `Subject`: esta entidad sirve para almacenar todos los sujetos o usuario de la PKI dentro de un dispositivo, esta entidad almacena el alias o nombre del sujeto en el campo `nam_sub`, el identificador del dispositivo (`dev_sub`) y un campo para verificar si ese usuario está activo o no.

- `TrustedCertificate`: por último esta entidad, sirve para almacenar las listas de confianza creadas en la red de confianza, como se puede ver en el diagrama, relaciona a un sujeto (propietario) con un certificado (entrada en la lista) y le asigna un nivel de confianza, el cual se almacena en el campo `lvl_trc`.

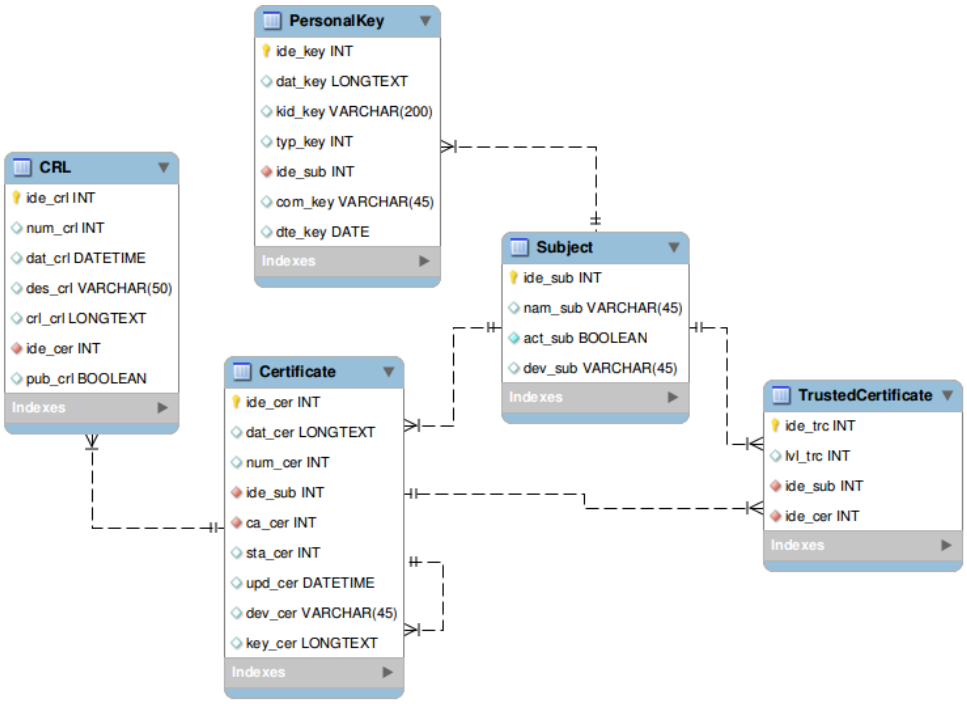


Figura 4.1: Diagrama de Entidad-Relación.

4.1.2. Diagrama de paquetes

En UML, un diagrama de paquetes muestra cómo un sistema está dividido en agrupaciones lógicas mostrando las dependencias entre esas agrupaciones. Dado que normalmente un paquete representa un directorio, los diagramas de paquetes suministran una descomposición de la jerarquía lógica de un sistema.

En la figura 4.2 se muestra el diagrama de paquetes general de la API, en el cual se pueden observar 3 paquetes principales, es importante mencionar todos estos paquetes se encuentran dentro del paquete general de la API llamado `cinvestav.android.pki`:

- `utils`: contiene clases que sirven de utilerías para los demás paquetes, dentro se pueden encontrar dos clases, `DataBaseDictionary` y `LogUtil`, la primera almacena variables estáticas que contienen los nombre de todos los elementos referentes de la base de datos utilizada en el sistema, por otra parte la clase `LogUtil` es una abstracción de la bitácora que se utilizará en la API para registrar los eventos.

- `db`: en este paquete se pueden encontrar todas las clases referentes a la capa de acceso a datos almacenados en el dispositivo. Como se puede observar en la figura 4.3 este paquete se encuentra dividido en otros, los cuales serán explicados más adelante. Este paquete refleja lo establecido en las secciones 3.2.2 y 3.3.2, en las cuales se establece que la PKI cuenta con una base de datos relacional centralizada. En esta base de datos serán almacenados todos los certificados, claves públicas y privadas

que sean generadas en la PKI siguiendo las normas establecidas en la sección 3.2.2.

- `cryptography`: el último paquete es en el que están todas las clases que implementan alguna funcionalidad criptográfica dentro de la API. Este paquete puede ser considerado como el corazón del sistema, y esta divido en 6 paquetes adicionales.

Figura 4.2: Diagrama de paquetes - General.

La figura 4.3 se muestra el diagrama de paquetes del paquete `db`, en el cual se encuentran los siguientes paquetes:

- `dao`: este paquete tiene las clases que abstraen los elementos de la base datos, cada una de las clases de este paquete representa una tabla en la base de datos, por lo que estas clases sirven como enlace entre la base de datos y la API.
- `db`: contiene las clases que representan la capa de acceso a datos ya que son las encargadas de interactuar directamente con la base de datos del dispositivo, todas las clases de este paquete heredan de `DataBaseHelper` la cual contiene las funciones necesarias para realizar la conexión y para ejecutar consultas en la base de datos.
- `controller`: siguiendo el modelo vista-controlador, las clases de este paquete sirven como enlace entre la capa de acceso a datos y las demás capas de la API, de manera que si hay algún cambio en la forma de almacenar los datos en el dispositivo, esto sea transparente a las demás capas del sistema.
- `exception`: almacena las clases referentes a excepciones para el manejo de errores en la capa de acceso a datos, actualmente contiene una única clase.

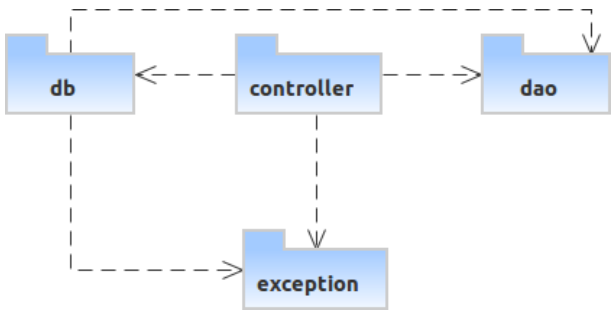


Figura 4.3: Diagrama de paquetes - `db`.

Por otra parte la división del paquete `cryptography` se puede ver en la figura 4.4, en la cual se pueden observar los siguientes paquetes:

- `ec`: contiene todas las clases que implementan a los elementos de curvas elípticas, como lo son: puntos (`ECPointFp` y `ECPointF2m`), curvas (`ECCurveFp` y `ECCurveF2m`), campos (`ECFieldElementFp` y `ECFieldElementF2m`) y parámetros de dominio.
- `cert`: este paquete contiene clases auxiliares para el manejo de certificados digitales y listas de revocación de certificados.
- `key`: almacena las clases que implementan la funcionalidad de las diferentes claves utilizadas en la API, contiene clases para claves de Criptografía de Curvas Elíptica (siglas del inglés *Elliptic Curve*) (ECC) y RSA.
- `algorithm`: contiene la implementación de los diferentes algoritmos utilizados en la PKI, como lo son EC, AES y RSA.
- `exception`: este paquete incluye clases referentes a excepciones que se pueden presentar durante las diferentes operaciones criptográficas de la PKI. Estas clases serán utilizadas para realizar el manejo de los errores que se pueden presentar durante la ejecución de estas operaciones.
- `utils`: finalmente el paquete `util` es el paquete que servirá como enlace entre la API y las aplicaciones que deseen utilizarla, ya que contiene las clases que implementan todas las funciones que se ofrecen en la PKI, estas funciones están divididas en 4 clases principales: `AsymmetricCryptoUtils`, `DigestCryptoUtils`, `SymmetricCryptoUtils` y `X509Utils`.

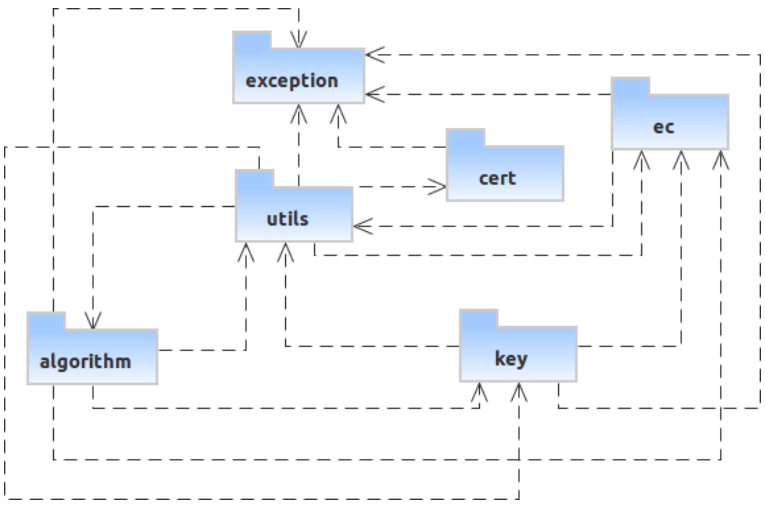


Figura 4.4: Diagrama de paquetes - `cryptography`.

4.1.3. Diagrama de clases

Los diagramas de clases describen la estructura de un sistema mostrando sus clases con atributos, funciones y las relaciones entre las mismas. Estos diagramas son utilizados

durante el proceso de análisis y diseño de los sistemas, donde se crea el diseño conceptual de la información que se manejará en el sistema, los componentes que se encargaran del funcionamiento y la relación entre uno y otro.

A continuación se presentan los diagramas de clases los paquetes principales del sistema, divididos en los paquetes correspondientes.

Paquete `cryptography`

En este paquete se verán reflejados la gran mayoría de los aspectos de diseño explicados en la sección 3.2, ya que contiene las clases más importantes del API.

El primer diagrama de clases representa las clases del paquete `ec`, este diagrama se puede observar en la figura 4.5, en éste se encuentran las implementaciones de los elementos de una curva elíptica como lo son puntos, curvas y parámetros de dominio. Las clases de este paquete heredan su funcionalidad de la biblioteca *Spongy Castle* [Tyley, 2012], en la cual se encuentra la implementación como tal de las operaciones de cada una de estas clases. Las clases de este paquete contienen las funciones necesarias para abstraer y ofrecer un formato uniforme de estos elementos a las demás clases de la API, ya que tanto en biblioteca utilizada como en el SDK de *Android* existen diferentes abstracciones de los elementos de una curva elíptica, lo cual hace que su uso sea complicado a lo largo de la implementación. Utilizando las clases de este paquete es posible pasar de un formato a otro dependiendo de las necesidades que se tengan en el sistema.

De manera similar al diagrama anterior, el paquete `algorithm` cuenta con clases que hacen uso de *Spongy Castle* [Tyley, 2012] para implementar su funcionalidad principal, sin embargo las clases de este paquete, véase figura 4.6, encapsulan la complejidad de los algoritmos utilizados (AES, ECC y RSA) mediante la sobrecargada de funciones, las cuales van desde una sencilla función que requiere los parámetros mínimos para la su ejecución hasta funciones más complejas, en las cuales se piden más parámetros, haciendo que la funcionalidad requerida pueda ser configurada por los usuarios de la API con conocimientos más avanzados. De manera general las clases de este paquete ofrecen funciones para: la generación de las claves necesarias, cifrado/descifrado y firma/verificación.

El diagrama de la figura 4.7, corresponde al paquete `cert` y contiene 3 clases:

- `CertificateInformationKeys`: esta clase es un diccionario en el cual se encuentran definidas todas las llaves válidas para un mapa de información que se utiliza a lo largo de la API para almacenar la información del titular de un certificado. En esta clase se encuentran declaradas todas las extensiones particulares definidas por la PKI en la sección 3.3.6 junto con sus respectivos OIDs.
- `X509CRLRevokedCertificateEntry`: abstracción de una entrada en la CRL, almacena el número de serie del certificado, la fecha de revocación y la razón de revocación.
- `X509RevokedCertificateReason`: clase que se utiliza como diccionario mediante variables estáticas y finales con las posibles razones de revocación de un certificado.

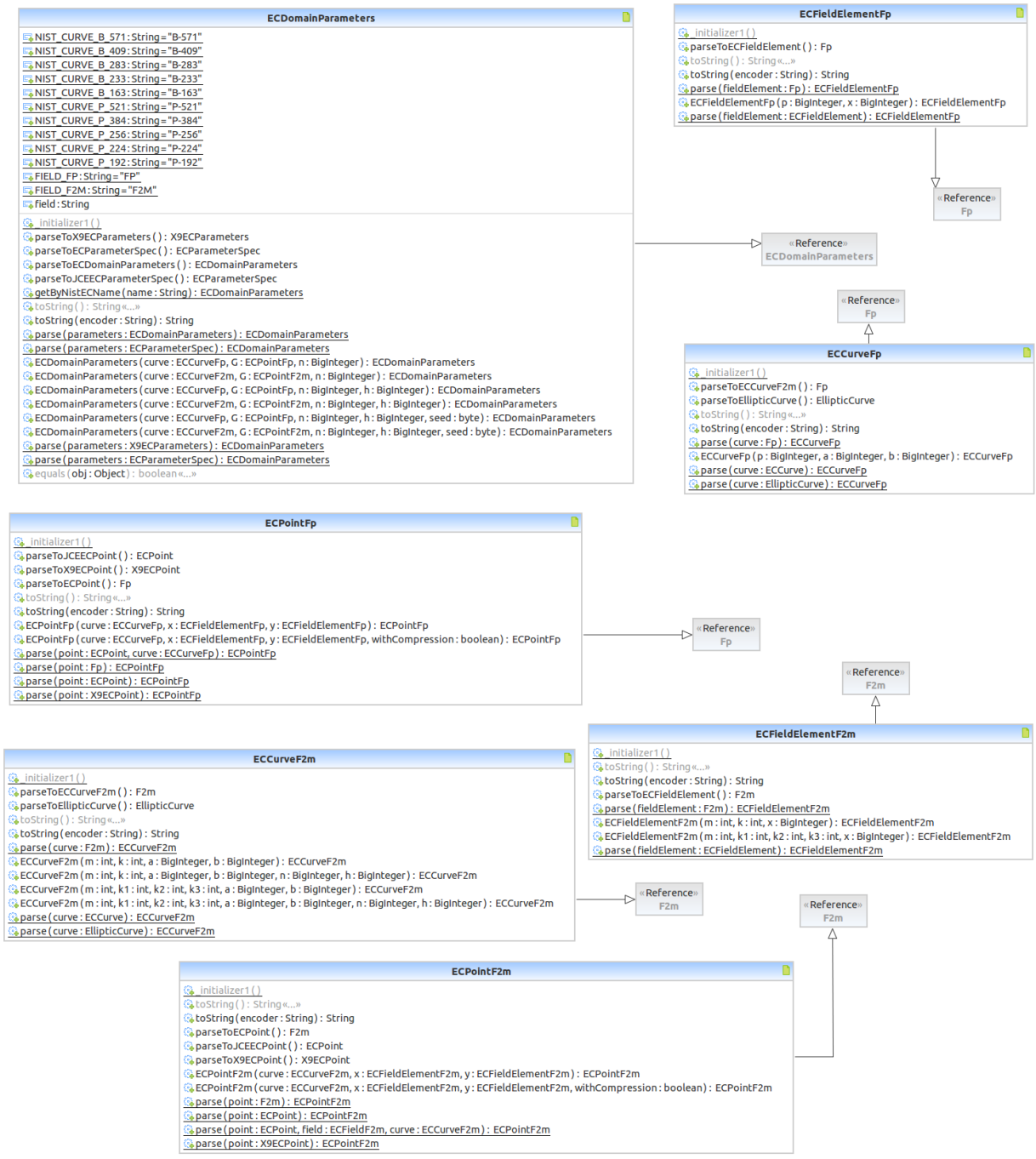


Figura 4.5: Diagrama de clases - `ec`.

El siguiente diagrama de la figura 4.8 que es el correspondiente al paquete `exception`, en este diagrama se pueden observar solo dos clases: `CryptoUtilsException`, utilizada para manipular los errores ocurridos durante las diversas operaciones criptográficas de la

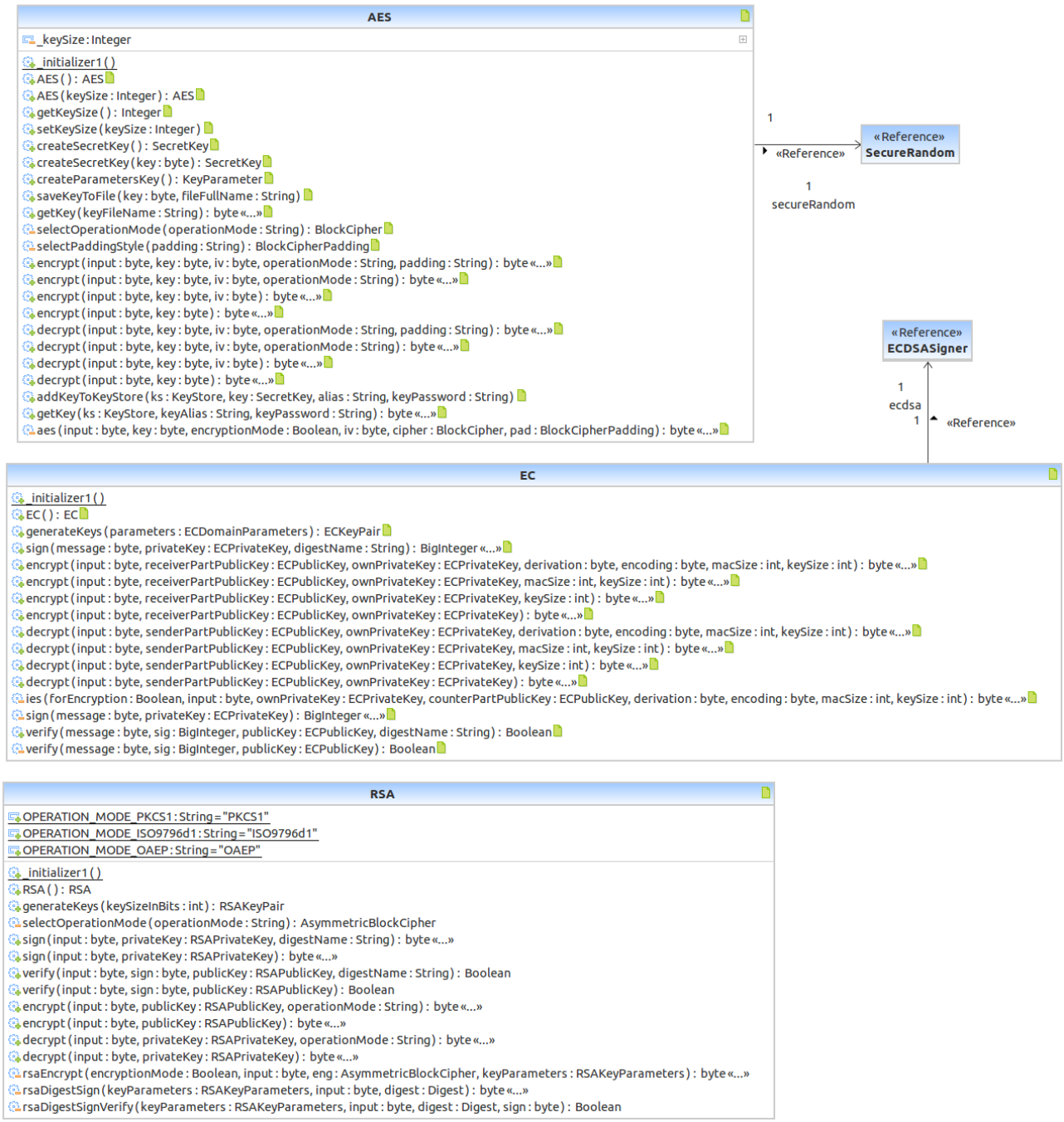


Figura 4.6: Diagrama de clases - `algorithm`.

API, como lo son: cifrar, firmar, creación de certificados, codificación de elementos, entre otras. Por otra parte la clase `CryptoUtilsX509ExtensionException`, es utilizada para manejar los errores referentes a las extensiones de los certificados X.509.

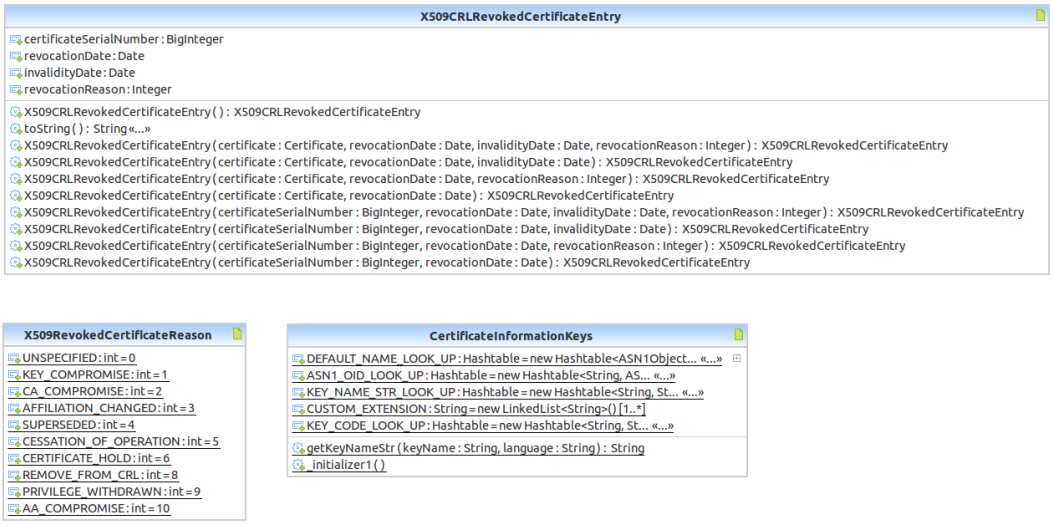


Figura 4.7: Diagrama de clases - `cert`.

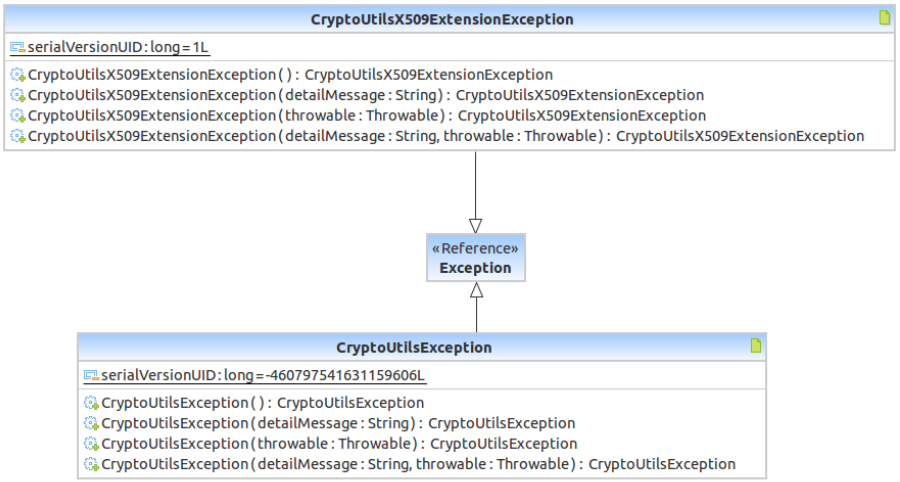


Figura 4.8: Diagrama de clases - `exception`.

En el caso del paquete `key` el diagrama de clases fue divido en dos secciones, para una mejor visualización del mismo. La primera se muestra en la figura 4.9, en la que se pueden observar las clases que implementan la funcionalidad de las claves pública y privada de RSA. De igual manera que en el caso de los elementos de una curva elíptica, estas clases contienen funciones para cambiar de formato de la clave según sea requerido, esto es necesario ya que existe una gran variedad de implementaciones para claves RSA tanto en *Spongy Castle* como en *Android*, lo cual dificulta su utilización sin la presencia de un

formato común entre ellas. Adicionalmente estas clases contienen las funciones necesarias para exportar e importar las claves a archivos dentro del dispositivo, así como codificarlas como una cadena de caracteres para su manipulación.

Figura 4.9: Diagrama de clases - `key` - RSA.

La segunda sección del diagrama correspondiente al paquete `key` se puede ver en la figura 4.10, en este diagrama se ven las clases correspondientes a las claves pública y privada para EC. En éste se pueden observar 3 clases principales: `ECKeyPair`, `ECPublicKey`

y `ECPrivateKey`. La primera de éstas tiene como atributos objetos del tipo de las otras dos clases y las funciones de esta clase ayudan a la exportación e importación del par de claves utilizando archivos PKCS#12. Por otra parte las otras dos clases corresponden a la implementación de una clave pública y privada de EC respectivamente, ambas clases tienen funciones para importar y exportar los datos de la curva a un archivo dentro del dispositivo así como funciones para cambiar de formato según sea necesario a lo largo de la API. Como se puede observar en el diagrama, ambas clases tienen como atributo un objeto de tipo `ECDomainParameters` perteneciente al paquete `ec`, el cual representa los parámetros de dominio de la curva que se utiliza en esa clave. Adicionalmente la clase `ECPrivateKey` tiene como atributo un entero grande `d`, el cual representa la parte privada de una clave, mientras que en la clase `ECPublicKey` se almacena un punto en la curva, el cual representa la parte pública de una clave.

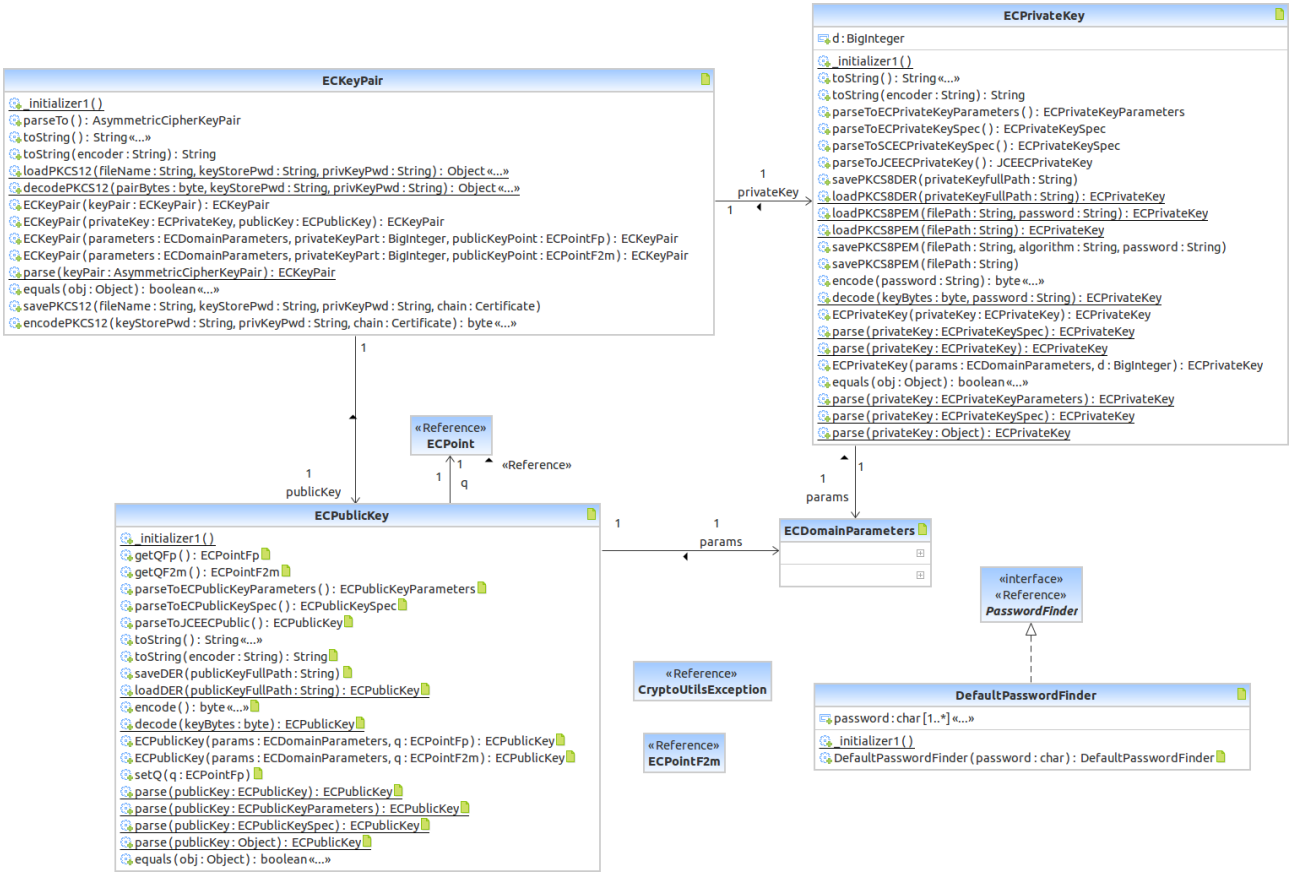


Figura 4.10: Diagrama de clases - `key` - EC.

El último paquete dentro de `cryptography`, es el llamado `utils`, al igual que el diagrama anterior, este diagrama se dividió en 3 secciones para mejorar su visibilidad y entendimiento. La primera sección contiene las clases que implementan funciones criptográficas, la segunda representa las clases con funciones correspondientes a certificados X.509 y por último la tercera sección contiene algunas clases extras que son utilizadas por las demás clases de la API. La primera sección del diagrama se puede observar en la figura 4.11, en

este diagrama se observan dos clases: `AsymmetricCryptoUtils` y `SymmetricCryptoUtils`, cada una de estas clases implementa una interfaz en la cual se definen todas las funciones que se deben implementar. De manera particular la clase `AsymmetricCryptoUtils` contiene las funciones correspondientes a operaciones criptográficas asimétricas, utilizando tanto RSA como EC. Como se puede observar en el diagrama, esta clase esta conformada por 4 funciones criptográficas: cifrar, descifrar, firmar y verificar, sin embargo se realizó una sobrecarga de estas funciones, de manera que la API final cuente con diversas opciones tanto para usuarios con conocimientos básicos de criptografía, como para usuarios con conocimientos más avanzados, los cuales requieran parámetros más específicos para lograr la funcionalidad deseada de la operación que se desea realizar. Por otra parte la clase `SymmetricCryptoUtils` contiene las funciones necesarias para cifrar y descifrar flujos de datos o cadenas utilizando AES, así como para generar y salvar las claves que se requieren para realizar estas operaciones. Es en estas clases donde se reflejan los requerimientos referentes a los parámetros de seguridad que se requieren de la PKI, por ejemplo la creación de claves para las CA garantizando estas claves tengan la seguridad mínima presentada en la tabla 3.1, adicionalmente las funciones de cifrado, decifrado, firma y verificación de la clase `AsymmetricCryptoUtils` siguen los esquemas planteados en la sección 3.2.3 para realizar estas operaciones.

La figura 4.12 muestra el diagrama correspondiente a la segunda sección del paquete `utils`, en el cual se encuentra la clase correspondiente a funciones con certificados X.509, esta clase es llamada `X509Utils`, en la cual se encuentran las funciones para crear, importar, exportar, codificar, decodificar y verificar tanto certificados como CRL X.509. Al igual que las clases que se han explicado anteriormente, esta clase contiene una gran sobrecarga de funciones, principalmente para la función encargada de crear certificados X.509 en su versión 3, con esta sobrecarga es posible generar diferentes tipos de certificados, entre los cuales destacan: certificados auto-firmados y certificados firmados por una CA, cada uno de estos utilizando exclusivamente claves RSA y EC o una combinación de estas. De esta manera es posible generar una gran variedad de certificados con diferente parámetros, algoritmos y tipos de clave. Como se puede ver en el diagrama, esta clase también implementa la funcionalidad definida en una interfaz. En esta clase recaen la gran mayoría de las características centrales de la PKI explicadas en el sección 3.2, por ejemplo los tipos de CA planeados en el modelo de confianza, son utilizados en las funciones de creación y validación de certificados y CRL para verificar si el certificado de la CA tiene privilegios suficientes para realizar la acción correspondiente. Adicionalmente como fue señalado en la sección 3.2.2 la PKI requiere que se contemple el uso de certificados internos y externos, por lo que la clase `X509Utils` cuenta con funciones para importar y exportar certificados X.509. Por otra parte esta clase sigue la plantilla para certificados presentada en la sección 3.2.3. Por lo anterior, se puede considerar que esta clase representa a la CA de la PKI y cumple con las características planteadas en el diseño.

Por otra parte la clase `CryptoUtils` es una clase utilizada por las demás de este paquete para definir parámetros utilizados en los algoritmos como: tipos de codificación, funciones picadillo y modos de operación.

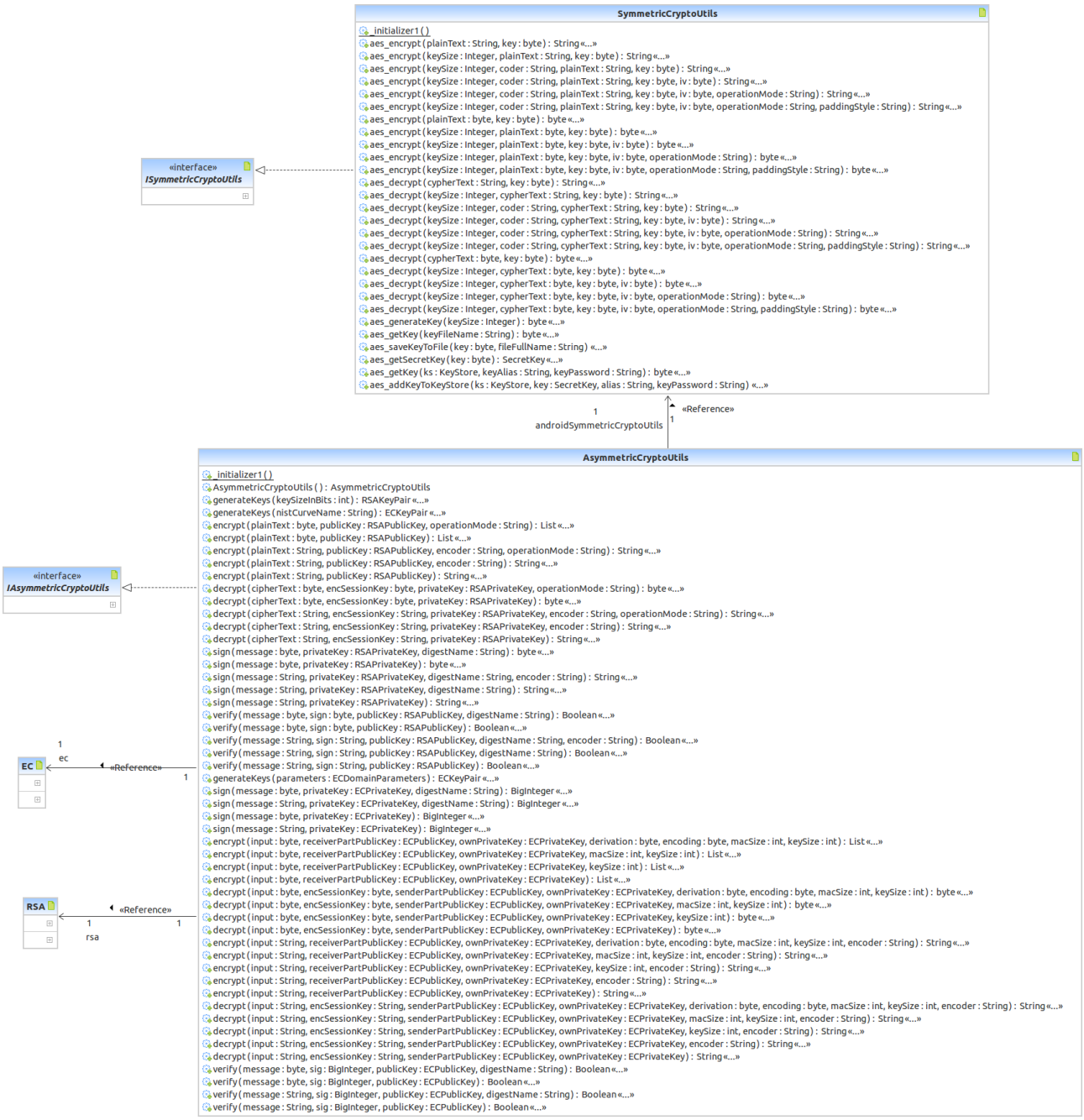


Figura 4.11: Diagrama de clases - `utils` - Criptografía.

Por último, el diagrama de la figura 4.13 muestra dos clases, una con las funciones para generar picadillos utilizando diferentes algoritmos (`DigestCryptoUtils`) y la segunda clase es un diccionario donde se almacenan todas las constantes utilizadas para certificados o CRL X.509, como lo son: posibles estatus, nombre de algoritmos de firma tanto para RSA como para EC, valores para la extensión *key usage*, entre otros. En esta clase se encuentran declarados todos los algoritmos compatibles con la PKI conforme a la sección 3.3.6.

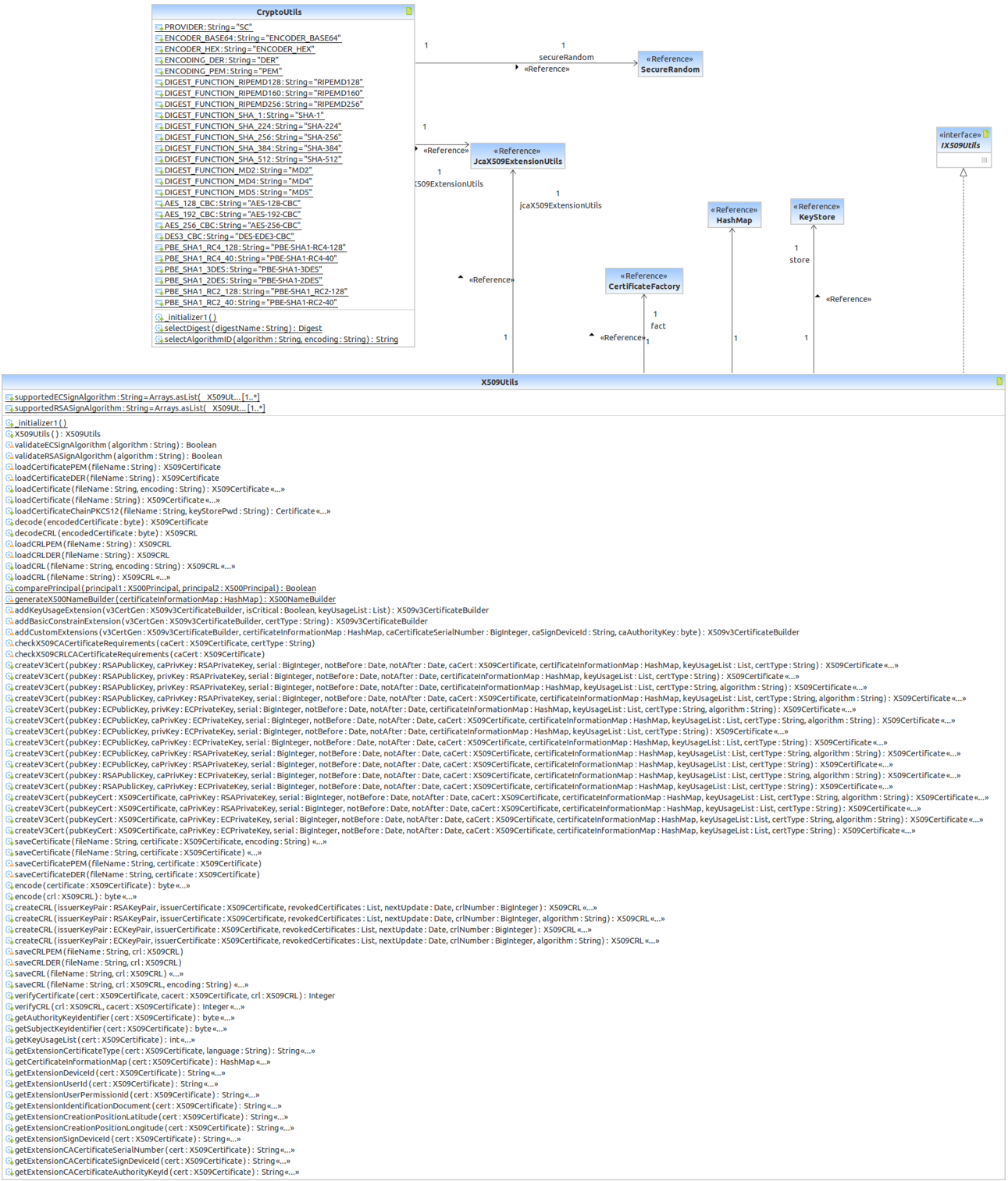


Figura 4.12: Diagrama de clases - `utils` - X.509.

Paquete `db`

Una vez terminadas de explicar las clases principales del paquete `cryptography`, se mostraran los diagramas correspondientes a los paquetes que componen al paquete `db`. De

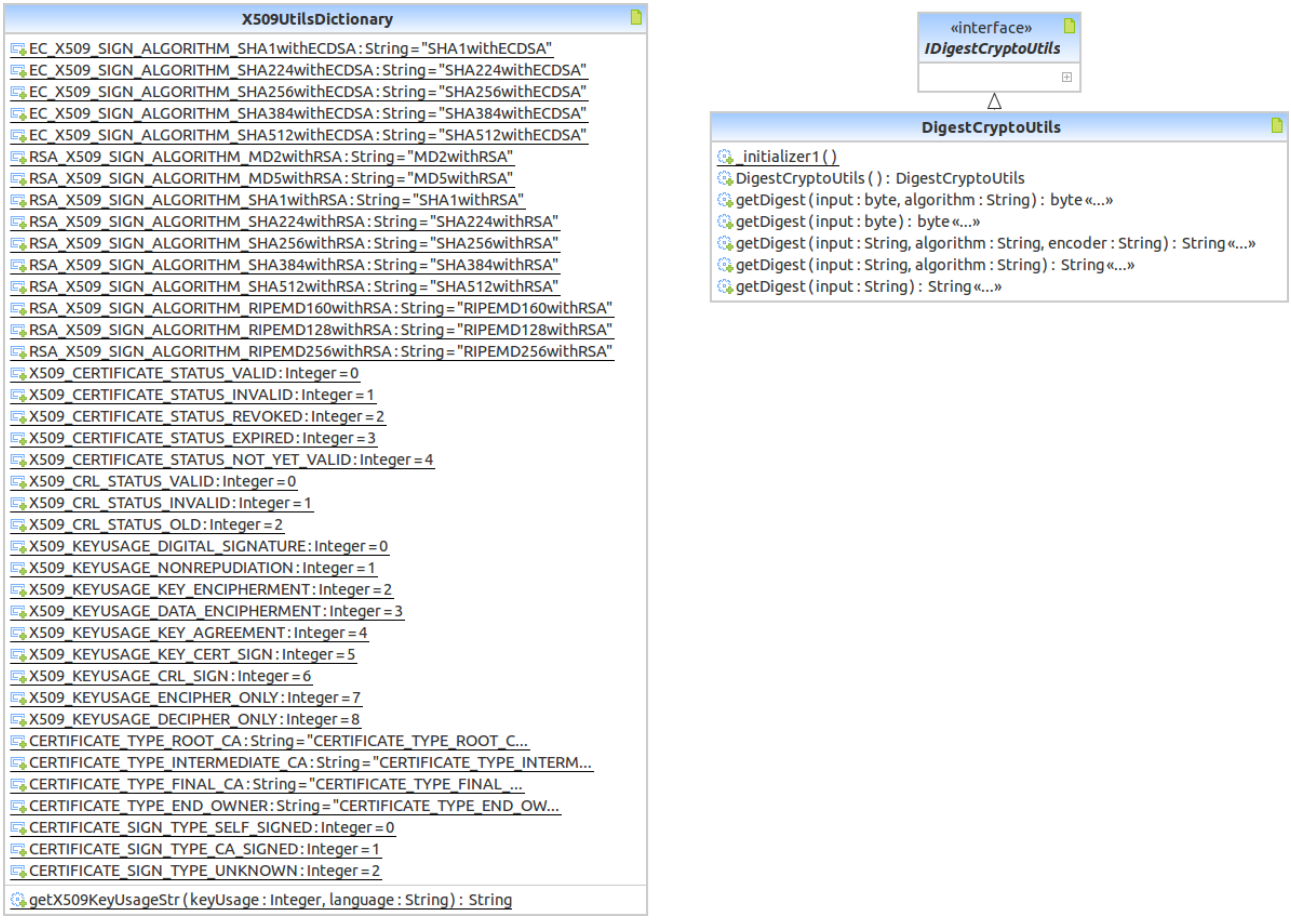


Figura 4.13: Diagrama de clases - `utils` - Otros.

manera general las clases de este paquete representan al repositorio de la PKI diseñada, ya que mediante ellas se accede a la base de datos para recuperar los certificados, claves y listas almacenadas en la misma.

El primer diagrama de este paquete es el correspondiente a las excepciones utilizadas para manejar los errores referentes a la base de datos en la API. Este diagrama consta únicamente de una clase, la cual se puede ver en la figura 4.14.

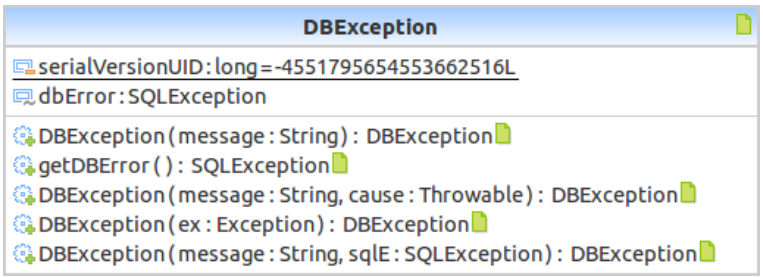


Figura 4.14: Diagrama de clases - `exception`.

El segundo diagrama se muestra en la figura 4.15, el cual corresponde al paquete `db`, las clases de este paquete representan la capa de acceso a datos, ya que son las encargadas de interactuar directamente con la base de datos. Como se puede observar en el diagrama, existe una clase para cada una de las clases del paquete `dao` y cada una de éstas hereda de la clase `DataBaseHelper`, en la que se encuentran las funciones para crear la base de datos, hacer una copia de la misma, realizar la conexión y ejecutar consultas con la base de datos. De manera general las demás clases de este paquete tienen las funciones necesarias realizar las operaciones básicas con elementos en la base de datos las cuales son: insertar, eliminar, actualizar y consultar, para el caso de las consultas todas las clases tiene 2 funciones principales: `getAll` y `getByAdvancedFilter`, la primera obtiene todos las entradas en la base de datos correspondientes al objeto que representa la clase, por ejemplo, en clase `CertificateDB` la función `getAll` obtiene todos los certificados almacenados en la base de datos, mientras que esta misma función pero en la clase `PersonalKeyDB` regresa una lista con todas las claves guardadas en la base. Por otra parte la función `getByAdvancedFilter` obtiene los elementos de la base de datos pero utilizando un filtro de búsqueda, el cual se crea dinámicamente a partir de un mapa que contiene el tipo, valor y nombre del filtro.

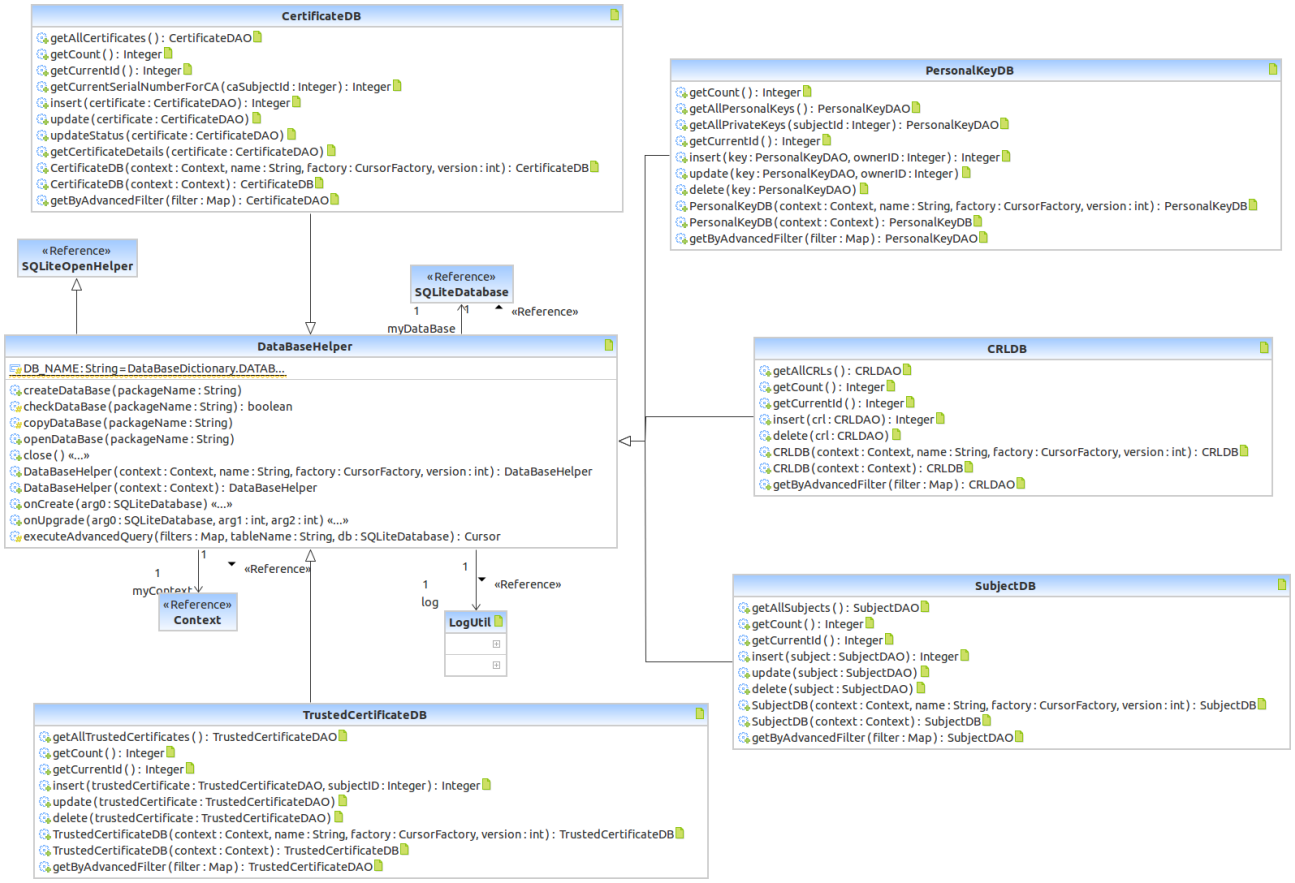


Figura 4.15: Diagrama de clases - `db`.

El siguiente diagrama corresponde al paquete `dao`, las clases que se muestran en la figura 4.16, son clases que representan a cada una de las tablas de la base de datos, estas

clases están compuestas únicamente por funciones `set` y `get`.

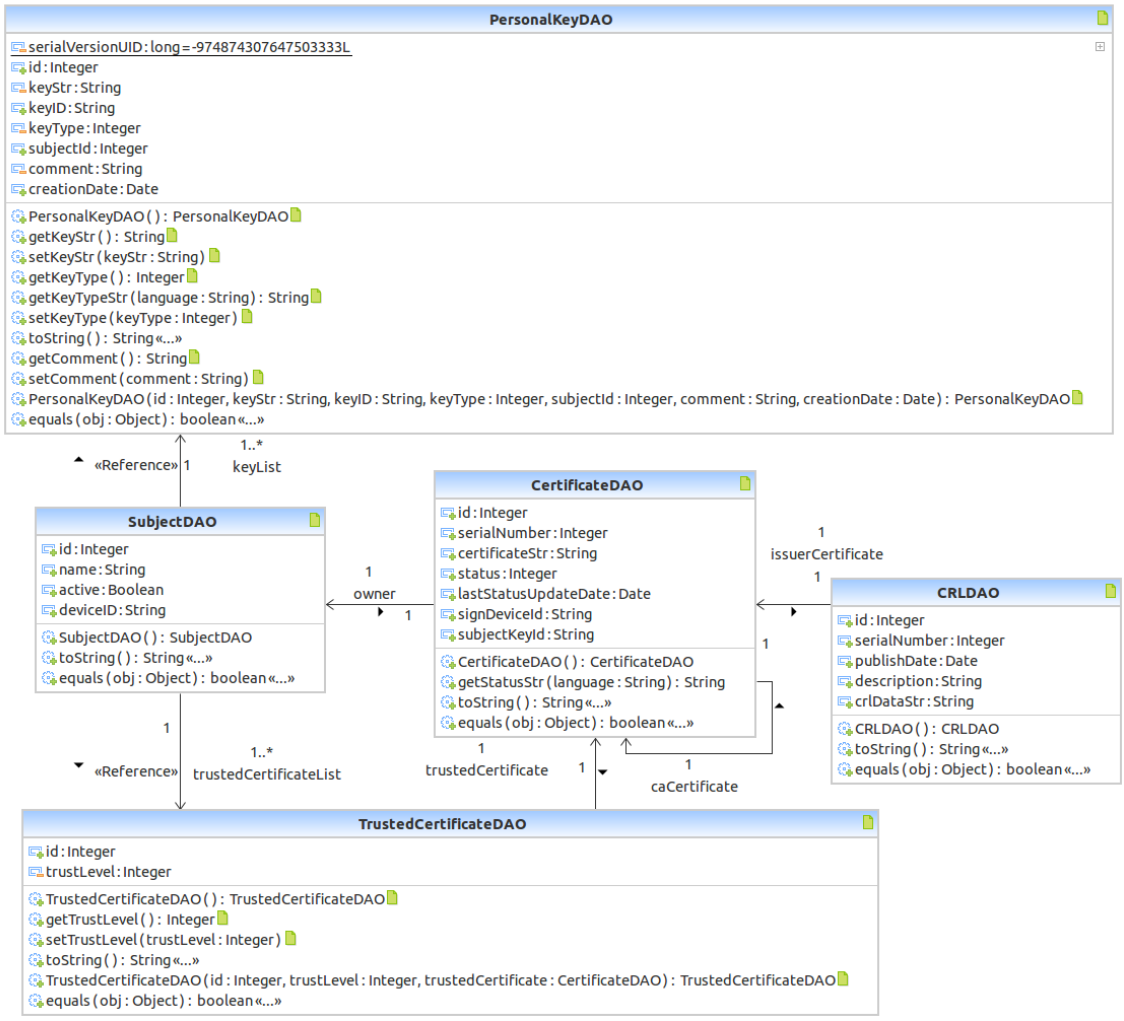


Figura 4.16: Diagrama de clases - `dao`.

Por último el paquete `controller` se muestra en la figura 4.17, al igual que en `db`, en este paquete se encuentra una clase para cada una de las almacenadas en `dao`, estas clases sirven como enlace entre las capas superiores y la capa de acceso a datos, de manera que se abstrae la complejidad de realizar las funciones de acceso a datos para las capas superiores, por lo que si existe algún cambio en el almacenamiento de éstos sea transparente para las demás capas. De manera general todas las clases de este paquete tienen las funciones que las clases de `db` con algunas funciones adicionales, mediante las cuales se realizan operaciones de consulta a la base de datos utilizando algunos filtro predefinidos.

Figura 4.17: Diagrama de clases - `controller`.

4.2. Pruebas de la API

El uso de los dispositivos móviles se ha incrementado notablemente en los últimos años, una de las principales ventajas que poseen los dispositivos móviles es el acceso a cualquier tipo de información en cualquier momento y desde cualquier lugar. Es por esto que este trabajo esta orientado al uso de este tipo de dispositivos.

Aunque las capacidades de hardware pueden variar de un dispositivo a otro, básicamente todos los dispositivos móviles están sujetos a limitaciones en cuanto a capacidad de almacenamiento, procesamiento y uso de la batería. Es por esto que las aplicaciones móviles deben ser diseñadas para optimizar el uso de los recursos disponibles.

En este apartado se describen las características implementadas en la API de la PKI para la plataforma móvil, posteriormente se presentan las pruebas de rendimiento de las principales operaciones de la API realizadas en diversos dispositivos.

4.2.1. Características de la API

En este apartado del documento se detallarán todas las características y funciones con las que cuenta la API desarrollada, de manera general la funcionalidad se puede dividir en cuatro categorías:

- Criptografía simétrica
- Funciones picadillo
- Criptografía asimétrica
- X.509

Criptografía simétrica

Para iniciar se detallarán las funciones con las que se cuentan en la categoría de criptografía simétrica, en la cual se utiliza únicamente AES como algoritmo de cifrado y en base a este se hicieron las funciones para: cifrar, descifrar, crear clave, guardar e importar claves. Cada una con diferentes parámetros y opciones.

En el caso de las funciones de cifrar y descifrar, sus parámetros y opciones se pueden resumir en la tabla 4.1, en la cual se muestran los diferentes tamaños de claves que pueden ser utilizados con AES, así como los diferentes modos de operación y tipos de relleno (también conocido como *padding*), adicionalmente en la siguiente columna se muestran los tipos de flujos de entrada, que puede ser cadenas o arreglos de bytes, en el caso de cadenas es posible utilizar tanto codificación hexadecimal como `Base64`. Es importante mencionar que todas estas opciones se pueden combinar para obtener el cifrado que se desea, por ejemplo se puede generar un cifrado AES-192 utilizando CBC con un relleno ISO7816d4 para una cadena codificada en `Base64`.

<table>
<tr><th>Algoritmo</th><th>Tamaño de clave</th><th>Modo de Operación</th><th>Relleno (Padding)</th><th>Tipo de entrada</th><th>Codificador</th></tr>
<tr><td rowspan="4">AES</td><td rowspan="2">128</td><td rowspan="3">CBC
CFB
OFB</td><td rowspan="4">PKCS7
ISO10126d2
ISO7816d4
X932
ZeroByte
Sin relleno</td><td>Arreglo de bytes</td><td>——</td></tr>
<tr><td rowspan="3">192</td><td rowspan="3">Cadena</td><td>Base64</td></tr>
<tr><td rowspan="2">256</td><td rowspan="2">OPENPGP</td><td>Hexadecimal</td></tr>
</table>

Tabla 4.1: Opciones de cifrado y descifrado - AES.

Adicionalmente en la tabla la API ofrece un función para generar claves de cifrado simétrico, la cual solo requiere indicar el tamaño de la clave que se utilizará, como se mencionó anteriormente este tamaño puede ser de 128, 192 o 256 bits. Para finalizar la API ofrece funciones para exportar e importar claves AES, actualmente solo se dan

opciones para hacer esto: Exportar a un archivo de texto y a un archivo de tipo `KeyStore`, siendo este último protegido por una contraseña para garantizar la confidencialidad de la clave.

Funciones "Picadillo"

Por otra parte la API ofrece una serie de funciones para generar un picadillo, los parámetros para estas funciones se pueden observar en la tabla 4.2. En la primera columna de esta tabla, se puede observar un listado con todos los algoritmos que pueden ser utilizados para generar el picadillo de una entrada, la cual puede ser (como se muestra en la segunda columna) o una cadena o un arreglo de bytes, para el caso de las cadenas, el picadillo es regresado en otra cadena, la cual puede estar codificada ya sea en hexadecimal o utilizando `Base64`.

<table>
<tr><th>Algoritmo</th><th>Tipo de entrada</th><th>Codificador</th></tr>
<tr><td rowspan="4">MD2
MD4
MD5
RIPEMD128
RIPEMD160
RIPEMD256
SHA-1
SHA-224
SHA-256
SHA-384
SHA-512</td><td>Arreglo de bytes</td><td>——</td></tr>
<tr><td rowspan="3">Cadenas</td><td>Base64</td></tr>
<tr><td>Hexadecimal</td></tr>
</table>

Tabla 4.2: Parámetros para funciones picadillo.

Criptografía asimétrica

El caso de las funciones de criptografía asimétrica es más complicado, ya que como se ha mencionado a lo largo de este documento, la API tiene la versatilidad de utilizar RSA y EC como algoritmos para este tipo de funciones. De manera general la API ofrece funciones para: cifrar, descifrar, firmar, verificar, generar claves, exportar e importar claves y para cambiar el formato de las mismas según sea necesario. En la tabla 4.3 se muestran los diferentes parámetros y opciones que tiene la API para cifrar y descifrar una entrada utilizando RSA como algoritmo.

De manera similar la tabla 4.4 muestra los parámetros para las mismas funciones, pero utilizando EC como algoritmo. De igual manera que el caso de AES, el usuario puede combinar estos parámetros para obtener el comportamiento deseado de cualquiera de estos dos algoritmos.

Tamaño de clave	Modo de operación	Tipo de entrada	Codificador
1024	PKCS1	Arreglo de bytes	——
2048	ISO9796d1	Cadenas	Base64
4096	OAEP		Hexadecimal

Tabla 4.3: Parámetros para funciones cifrado y descifrado - RSA.

Nombre Curva	Tipo de entrada	Codificador
B-163	Arreglo de bytes	——
B-233		
B-283		
B-409		
B-571		
P-192	Cadena	Base64
P-224		
P-256		Hexadecimal
P-384		
P-521		

Tabla 4.4: Parámetros para funciones cifrado y descifrado - EC.

Por otra parte, en el caso de las funciones de firma digital y verificación ambos algoritmos tienen la versatilidad de generar la firma utilizando cualquiera de los algoritmos listados en la tabla 4.2 y tomar como entrada un arreglo de bytes o una cadena de caracteres, teniendo como principal diferencia que RSA, regresará una cadena codifica en hexadecimal o `base64` según se le indique a la función, mientras que EC regresará siempre una arreglo de enteros grandes.

Las funciones encargadas de generar el par de claves para ambos algoritmos requieren que se indique el tamaño de clave para RSA y el nombre de la curva (siguiendo la nomenclatura propuesta por NIST) en el caso de EC, adicionalmente para este último algoritmo, es posible crear claves para curvas personalizadas, indicando los parámetros de la misma utilizando la clase `ECDomainParameters`.

Otro grupo de funciones importante, es el correspondiente a las funciones para cambiar de formato a las claves utilizadas en la API, ya que en la biblioteca utilizada como base (*Spongu Castle*) y en el SDK de *Android* existen una gran variedad de clases utilizadas representar una clave, por lo que es necesario contar con un formato común entre ellas y de esta manera simplificar su uso en las demás funciones del API. En la tabla 4.5 se muestra el listado de todos los formatos disponibles en la API para las claves privadas y públicas de RSA y EC.

Por último las funciones para exportar e importar las claves de criptografía asimétrica ofrecen una variedad de formatos y archivos de salida dependiendo el tipo de clave, en la tabla 4.6 se resumen los posibles formatos de clave que la API puede generar. En la primera columna se lista el algoritmo de la clave, mientras que la segunda columna muestra el tipo

Algoritmo	Tipo de clave	Formato soportado
RSA	Pública	`org.spongycastle.crypto.params.RSAKeyParameters`
		`org.spongycastle.asn1.pkcs.RSAPublicKey`
		`java.security.interfaces.RSAPublicKey`
	Privada	`org.spongycastle.crypto.params.RSAPrivateCrtKeyParameters`
		`org.spongycastle.asn1.pkcs.RSAPrivateKey`
		`java.security.interfaces.RSAPrivateCrtKey`
EC	Pública	`org.spongycastle.crypto.params.ECPublicKeyParameters`
		`org.spongycastle.jce.spec.ECPublicKeySpec`
		`java.security.interfaces.ECPublicKey`
	Privada	`org.spongycastle.crypto.params.ECPrivateKeyParameter`
		`org.spongycastle.jce.spec.ECPrivateKeySpec`
		`java.security.spec.ECPrivateKeySpec`
		`java.security.interfaces.ECPrivateKey`

Tabla 4.5: Formatos de clave soportados.

de clave, es importante mencionar que "Par" se refiere al conjunto de clave pública y privada en el mismo archivo, la siguiente columna muestra el estándar que se utiliza para generar el archivo de salida y la codificación que se utilizo para este mismo propósito, por último la columna de seguridad muestra como se protege la clave dentro del archivo generado.

Algoritmo	Tipo de clave	Estándar	Codificación	Seguridad
RSA	Pública	PKCS#8	DER	——
	Privada	PKCS#8	DER	——
			PEM	——
				Contraseña
	Par	PKCS#12	——	Contraseña
EC	Pública	PKCS#8	DER	——
	Privada	PKCS#8	DER	——
			PEM	——
				Contraseña
	Par	PKCS#12	——	Contraseña

Tabla 4.6: Formatos para exportación e importación de claves.

En el caso de las claves privadas, la API permite al usuario seleccionar el algoritmo que se utilizará para proteger dicha clave, los algoritmos soportados son los siguientes:

- AES-128-CBC
- AES-192-CBC
- AES-256-CBC
- DES-EDE3-CBC
- PBE-SHA1-RC4-128

- PBE-SHA1-RC4-40
- PBE-SHA1-3DES
- PBE-SHA1-2DES
- PBE-SHA1-RC2-128
- PBE-SHA1-RC2-40

X.509

Por último las funciones de X.509 se pueden agrupar a su vez en dos tipos: funciones para certificados y para CRL. Las funciones para certificados incluyen una gran variedad de opciones para la creación de los mismos, además de que permiten la exportación e importación de certificados a archivos dentro del dispositivo, adicionalmente la API incluye un conjunto de funciones para obtener la información de un certificado y realizar la validación de estos. De manera similar para CRL, la API incluye funciones para su creación, validación, exportación e importación.

Las funciones para la creación de certificados están sobrecargadas de manera que el usuario de la API puede generar un certificado digital X.509 en su versión 3 con diferentes parámetros, por ejemplo la tabla 4.7 muestra un listado de todos los algoritmos de firma soportados por el API dependiendo del tipo de clave que el emisor posea, adicionalmente la API permite al usuario generar certificados auto-firmados o certificados firmados por otra entidad, los certificados auto-firmados son aquellos en los cuales un usuario firma su propio certificado utilizando la clave privada correspondiente a la clave pública almacenada en el certificado en cuestión. Adicionalmente la API ofrece la posibilidad de crear certificados mixtos, en los cuales una entidad puede utilizar claves RSA y la otra EC.

Por otra parte la API permite exportar un certificado tanto en formato `DER` como en `PEM`, mientras que la validación de un certificado se puede hacer utilizando el certificado que se desea validar, el certificado de CA que emitió el certificado (para verificar la firma del certificado) y una CRL, esta última es necesaria para verificar que el certificado no se encuentre en dicha lista.

Para las CRL, la API permite a los usuarios crear una nueva lista utilizando los algoritmos de la tabla 4.7 para firmar la lista y lo único que se requiere es la lista de los certificados a revocar, cada entrada de esta lista utiliza la clase `X509CRLRevokedCertificateEntry` para definir sus parámetros de revocación.

4.2.2. Pruebas funcionales

Las pruebas funcionales de la API se realizaron mediante la implementación de una pequeña aplicación para *Android*, en la cual se importa la API generada y se crearon clases con diferentes funciones, las cuales actuaron como pruebas unitarias para cada una de las

Clave Emisor	Algoritmo de firma
RSA	MD2withRSA
	MD5withRSA
	SHA1withRSA
	SHA224withRSA
	SHA256withRSA
	SHA384withRSA
	SHA512withRSA
	RIPEMD160withRSA
	RIPEMD128withRSA
	RIPEMD256withRSA
EC	SHA1withECDSA
	SHA224withECDSA
	SHA256withECDSA
	SHA384withECDSA
	SHA512withECDSA

Tabla 4.7: Algoritmos de firma para de certificados X.509 soportados.

funciones con las que cuenta la API. Una prueba unitaria es una forma de comprobar el correcto funcionamiento de un modulo de código, esto sirve para asegurar que cada uno de los módulos funcione correctamente por separado. Para que una prueba unitaria sea buena se deben cumplir los siguientes requisitos:

- Automatizable: no debería requerirse una intervención manual de parte del usuario para correr la prueba, para esto es necesario contar con datos de prueba confiables.
- Completas: deben cubrir la mayor cantidad de código.
- Repetibles o Reutilizables: no se deben crear pruebas que sólo puedan ser ejecutadas una sola vez.
- Independientes: la ejecución de una prueba no debe afectar a la ejecución de otra.
- Profesionales: las pruebas deben ser consideradas igual que el código, con la misma profesionalidad, documentación, etc.

Los resultados de estas pruebas arrojaron que todas las funciones de la API funcionan correctamente, sin embargo los resultados de estas pruebas no fueron incluidos en este documento ya que son demasiado extensos debido a la cantidad de funciones y opciones posibles dentro de la API. Por otra parte en la siguiente sección se muestran las pruebas de rendimiento de una gran cantidad de estas funciones.

4.2.3. Pruebas de rendimiento

Las pruebas de rendimiento son las pruebas que se realizan para determinar la rapidez en que se realiza una tarea un sistema en condiciones particulares de trabajo. También

puede servir para validar y verificar otros atributos de la calidad del sistema, tales como la escalabilidad, fiabilidad y uso de los recursos.

Las pruebas presentadas en está sección corresponden a los módulos más importantes con los que cuenta la API, entre los cuales destacan: criptografía simétrica, criptografía asimétrica (RSA y EC) y X.509. Es importante mencionar que la implementación se hizo tanto en un dispositivo móvil como en una computadora de escritorio esto con el fin de comprobar su correcto funcionamiento. Aunque se cuenta con ambas implementaciones en este capítulo solo se reportan los resultados obtenidos en el dispositivo móvil, ya que es la motivación central del presente trabajo.

Cada una de las pruebas de rendimiento fueron realizadas en 3 dispositivos diferentes, para mostrar los cambios en el desempeño de las mismas de acuerdo a las capacidades del dispositivo de prueba, en la tabla 4.8 detallan las características de cada uno de estos.

Caracteristicas	Dispositivo 1 D-1	Dispositivo 2 D-2	Dispositivo 3 D-3
Marca	Xperia	Xperia	Xperia
Modelo	Arc	Ray	X8
Sistema Operativo	Android 2.3	Android 2.3	Android 2.1
Procesador	Qualcomm MSM8250 Snapdragon 1GHz	Qualcomm MSM8255 1GHz	Qualcomm 600MHz
Pantalla	480 x 854 pixels, 4.2”	480 x 854 pixels, 3.3”	320 x 480 pixeles, 3”
Memoria RAM	512MB	512MB	168 MB
Memoria Interna	320MB	300MB	128 MB
Batería	Li-Po 1500 mAh	Li-Ion 1500 mAh	Li-Po 1200 mAh

Tabla 4.8: Características dispositivos de prueba.

Es importante mencionar que los tiempos reportados a continuación son el resultado de realizar un gran número de pruebas a cada función y calcular el tiempo promedio de ejecución, de manera general se realizaron al menos 1000 pruebas por función para obtener tiempos más confiables.

La primera prueba que se presenta es la correspondiente a AES, algoritmo de criptografía simétrica utilizado en la PKI. La tabla 4.9 muestra el listado de las pruebas de rendimiento que se realizaron a este algoritmo, de manera general se pueden observar 3 pruebas: cifrado/descifrado, generación de clave y exportación/importación de clave. La primera se realizó variando los parámetros de:

- Tamaño de clave entre los posibles valores (128,192 y 256)
- Tamaño de entrada, este se refiere a la cantidad de bits que fueron procesados durante la operación, para la prueba se utilizaron los valores de 1024 y 3072.
- Modo de operación, se utilizaron todos los modos de operación al menos una vez
- Relleno o *Padding*, cada uno de los tipos de relleno se intercaló con algún modo.

Por lo que, la tabla 4.9 muestra un total de 30 pruebas para de cifrado y descifrado. La siguiente prueba realizada fue la generación de claves para AES, esta prueba únicamente se realizó una vez por tamaño de clave. Para el caso de las pruebas de exportación e importación de claves, se utilizaron dos formatos para el archivo de salida, el primero fue guardando la clave en un archivo de texto simple sin ningún tipo de protección, mientras que la otra prueba se realizó utilizando un `KeyStore (KS)` para almacenar la clave, la ventaja de utilizar este tipo de archivos es que cuentan con la protección de una contraseña.

Operación	Tamaño de clave (bits)	Tamaño de entrada (bits)	Modo de operación	Relleno (*padding*)	Tiempo (ms)		
					D-1	D-2	D-3
Cifrado	128	1024	CBC	PKCS7	1.19	1.50	6.42
Descifrado					1.30	1.69	7.05
Cifrado	128	1024	OFB	ISO7816d4	1.18	1.44	5.81
Descifrado					1.04	1.60	6.34
Cifrado	128	1024	CFB	ISO10126d2	1.37	1.63	6.24
Descifrado					1.34	1.73	6.00
Cifrado	128	1024	OPENPGP	X932	1.26	1.39	7.09
Descifrado					1.14	1.59	7.15
Cifrado	128	1024	CBC	ZeroByte	1.17	1.43	5.65
Descifrado					1.09	1.56	6.23
Cifrado	128	3072	CBC	PKCS7	2.62	4.16	13.91
Descifrado					3.22	3.66	15.02
Cifrado	128	3072	OFB	ISO7816d4	2.72	3.57	16.29
Descifrado					3.28	4.33	17.09
Cifrado	128	3072	CFB	ISO10126d2	2.81	3.73	17.06
Descifrado					3.31	4.33	16.65
Cifrado	128	3072	OPENPGP	X932	2.84	4.42	17.69
Descifrado					3.41	3.84	16.91
Cifrado	128	3072	CBC	ZeroByte	2.59	4.10	14.15
Descifrado					3.20	3.60	15.68
Cifrado	192	1024	CBC	PKCS7	1.17	1.71	6.62
Descifrado					1.38	1.77	6.70
Cifrado	192	1024	OFB	ISO7816d4	1.22	1.79	7.48
Descifrado					1.36	1.66	7.54
Cifrado	192	1024	CFB	ISO10126d2	1.40	1.92	8.23
Descifrado					1.59	1.89	7.97
Cifrado	192	1024	OPENPGP	X932	1.35	1.74	7.98
Descifrado					1.39	1.86	8.16
Cifrado	192	1024	CBC	ZeroByte	1.18	1.72	6.52
Descifrado					1.40	1.80	6.60
Cifrado	192	3072	CBC	PKCS7	3.04	4.14	17.39
Descifrado					3.68	4.92	15.71
Cifrado	192	3072	OFB	ISO7816d4	3.06	4.78	20.35
Descifrado					3.64	4.18	17.26
Cifrado	192	3072	CFB	ISO10126d2	3.26	4.98	20.00
Descifrado					3.80	4.30	18.25
Cifrado	192	3072	OPENPGP	X932	3.40	4.44	18.93
Descifrado					3.93	5.11	20.04
Cifrado	192	3072	CBC	ZeroByte	3.86	4.10	17.05
Descifrado					4.68	4.94	17.00
Cifrado	256	1024	CBC	PKCS7	1.44	1.90	7.26

Operación	Tamaño de clave (bits)	Tamaño de entrada (bits)	Modo de operación	Relleno (*padding*)	Tiempo (ms)		
					D-1	D-2	D-3
Descifrado					1.58	2.04	7.85
Cifrado	256	1024	OFB	ISO7816d4	1.44	1.84	8.21
Descifrado					1.58	2.07	8.86
Cifrado	256	1024	CFB	ISO10126d2	1.68	2.14	9.26
Descifrado					1.65	2.18	8.75
Cifrado	256	1024	OPENPGP	X932	1.51	1.95	9.22
Descifrado					1.62	2.10	9.17
Cifrado	256	1024	CBC	ZeroByte	1.45	1.92	7.11
Descifrado					1.57	2.07	8.04
Cifrado	256	3072	CBC	PKCS7	3.51	5.37	20.53
Descifrado					4.14	4.85	20.10
Cifrado	256	3072	OFB	ISO7816d4	3.55	4.72	22.28
Descifrado					4.13	5.43	21.02
Cifrado	256	3072	CFB	ISO10126d2	3.69	5.06	21.53
Descifrado					4.22	5.74	22.20
Cifrado	256	3072	OPENPGP	X932	3.80	5.85	22.84
Descifrado					4.35	5.31	21.48
Cifrado	256	3072	CBC	ZeroByte	3.54	5.40	19.14
Descifrado					4.14	4.90	19.11
Generar clave	128	—	—	—	0.09	0.11	0.35
Generar clave	192	—	—	—	0.09	0.11	0.34
Generar clave	256	—	—	—	0.08	0.1	0.32
Exportar	128	—	—	—	3.21	3.09	3.65
Importar					0.16	0.18	0.29
Exportar	192	—	—	—	3.24	3.16	3.69
Importar					0.16	0.18	0.29
Exportar	256	—	—	—	3.27	3.17	3.5
Importar					0.16	0.18	0.35
Exportar (`KS`)	128	—	—	—	130.75	135.28	364.59
importar (`KS`)					118.29	135.62	341.34
Exportar (`KS`)	192	—	—	—	97.71	150.41	368.38
importar (`KS`)					93.71	144.7	345.88
Exportar (`KS`)	256	—	—	—	96.48	133.14	537.3
importar (`KS`)					92.96	127.03	572.55

Tabla 4.9: Resultados pruebas de rendimiento - AES.

Las siguientes pruebas realizadas corresponden a los módulos de criptografía asimétrica, las cuales fueron divididas en 3 tablas, la primera (tabla 4.10) muestra los resultados de las pruebas de rendimiento realizadas a RSA para las operaciones de cifrado/descifrado, firma/verificación y generación de claves. Al igual que las pruebas de AES, los valores de entrada para las pruebas fueron variados de tal forma que fuera posible apreciar de manera general el comportamiento de este algoritmo. En el caso de cifrado y descifrado, se presentan resultados para cada uno de los modos de operación disponibles con diferentes tamaños de clave y tamaños de entrada, mientras que para la firma y verificación se tomaron únicamente 5 funciones picadillo de todas las disponibles.

Operación	Tamaño de clave (bits)	Tamaño de entrada (bits)	Modo de operación	Picadillo	Tiempo (ms)		
					D-1	D-2	D-3
Cifrado	1024	1024	OAEP	——	2.70	3.69	14.01
Descifrado					7.29	9.90	27.56
Cifrado	1024	1024	PKCS1	——	2.61	3.30	12.12
Descifrado					6.18	8.05	26.86
Cifrado	1024	1024	ISO9796d1	——	2.07	2.76	10.30
Descifrado					6.01	7.98	24.98
Cifrado	1024	3072	OAEP	——	4.42	6.22	23.45
Descifrado					8.71	11.73	33.76
Cifrado	1024	3072	PKCS1	——	4.22	6.15	20.80
Descifrado					8.34	11.88	34.42
Cifrado	1024	3072	ISO9796d1	——	4.30	6.13	20.20
Descifrado					9.30	11.78	32.20
Cifrado	2048	1024	OAEP	——	3.22	4.18	14.09
Descifrado					28.17	39.26	109.68
Cifrado	2048	1024	PKCS1	——	2.76	3.97	12.81
Descifrado					27.72	39.25	107.76
Cifrado	2048	1024	ISO9796d1	——	2.75	3.65	12.95
Descifrado					27.60	38.22	112.39
Cifrado	2048	3072	OAEP	——	5.15	9.78	26.23
Descifrado					30.53	55.36	117.36
Cifrado	2048	3072	PKCS1	——	5.49	7.14	24.58
Descifrado					29.59	42.45	116.86
Cifrado	2048	3072	ISO9796d1	——	4.78	7.17	22.54
Descifrado					30.05	43.16	117.66
Cifrado	4096	1024	OAEP	——	6.08	8.46	27.22
Descifrado					193.52	270.60	758.27
Cifrado	4096	1024	PKCS1	——	5.51	7.81	29.25
Descifrado					192.66	274.83	764.76
Cifrado	4096	1024	ISO9796d1	——	5.13	7.32	28.48
Descifrado					193.26	272.77	759.47
Cifrado	4096	3072	OAEP	——	8.05	11.46	44.39
Descifrado					196.24	274.84	773.69
Cifrado	4096	3072	PKCS1	——	8.88	11.39	42.51
Descifrado					212.81	270.99	769.73
Cifrado	4096	3072	ISO9796d1	——	8.40	10.72	36.42
Descifrado					204.81	271.51	767.12
Firma	1024	1024	——	SHA-1	7.32	9.41	28.04
Verificación					1.19	1.56	3.62
Firma	1024	1024	——	SHA-256	7.61	9.81	28.99
Verificación					1.56	2.15	4.43
Firma	1024	1024	——	SHA-512	7.67	9.98	29.34
Verificación					1.63	2.16	5.45
Firma	1024	1024	——	MD5	7.33	9.45	27.20
Verificación					1.10	1.43	3.47
Firma	1024	1024	——	RIPEMD160	8.33	10.34	29.39
Verificación					2.05	2.83	5.98
Firma	1024	3072	——	SHA-1	7.44	10.13	28.24
Verificación					1.55	1.98	5.12
Firma	1024	3072	——	SHA-256	8.61	11.52	31.09
Verificación					2.63	3.56	7.92

Operación	Tamaño de clave (bits)	Tamaño de entrada (bits)	Modo de operación	Picadillo	Tiempo (ms)		
					D-1	D-2	D-3
Firma	1024	3072	——	SHA-512	8.68	11.47	33.14
Verificación					2.75	3.64	9.83
Firma	1024	3072	——	MD5	7.64	10.01	28.12
Verificación					1.62	1.98	4.87
Firma	1024	3072	——	RIPEMD160	9.83	12.83	33.91
Verificación					3.92	5.20	10.94
Firma	2048	1024	——	SHA-1	33.60	45.60	130.60
Verificación					1.73	2.30	6.74
Firma	2048	1024	——	SHA-256	34.19	45.59	131.63
Verificación					2.09	2.73	7.66
Firma	2048	1024	——	SHA-512	33.78	45.47	132.54
Verificación					2.22	2.81	8.44
Firma	2048	1024	——	MD5	33.32	44.96	139.07
Verificación					1.71	2.21	7.51
Firma	2048	1024	——	RIPEMD160	34.19	45.97	133.31
Verificación					2.57	3.29	9.22
Firma	2048	3072	——	SHA-1	33.57	46.40	132.99
Verificación					2.15	2.90	8.16
Firma	2048	3072	——	SHA-256	34.98	47.96	135.49
Verificación					3.21	4.25	11.34
Firma	2048	3072	——	SHA-512	34.74	48.05	137.39
Verificación					3.33	4.62	13.45
Firma	2048	3072	——	MD5	33.97	45.85	132.64
Verificación					2.28	2.82	8.09
Firma	2048	3072	——	RIPEMD160	36.15	48.70	138.03
Verificación					4.45	5.82	14.21
Firma	4096	1024	——	SHA-1	210.90	281.88	814.34
Verificación					4.49	5.80	20.62
Firma	4096	1024	——	SHA-256	211.42	284.62	816.15
Verificación					4.88	6.38	22.03
Firma	4096	1024	——	SHA-512	205.95	285.09	817.01
Verificación					4.87	6.51	23.52
Firma	4096	1024	——	MD5	203.95	282.21	813.78
Verificación					4.40	5.84	20.00
Firma	4096	1024	——	RIPEMD160	204.81	314.73	841.34
Verificación					5.27	7.40	24.16
Firma	4096	3072	——	SHA-1	204.24	312.75	814.26
Verificación					4.61	6.90	22.04
Firma	4096	3072	——	SHA-256	205.30	302.62	815.91
Verificación					5.62	8.11	25.22
Firma	4096	3072	——	SHA-512	205.37	317.13	868.12
Verificación					6.00	8.58	28.41
Firma	4096	3072	——	MD5	204.25	290.26	815.71
Verificación					4.87	6.66	21.48
Firma	4096	3072	——	RIPEMD160	206.48	289.21	820.59
Verificación					6.95	9.57	27.29
Generar clave	1024	—	—	—	921.36	1477.57	4568.77
Generar clave	2048	—	—	—	6056.62	6414.65	15627.29
Generar clave	4096	—	—	—	35295.28	48580.15	118386.37

Operación	Tamaño de clave (bits)	Tamaño de entrada (bits)	Modo de operación	Picadillo	Tiempo (ms)		
					D-1	D-2	D-3

Tabla 4.10: Resultados pruebas de rendimiento - RSA.

La tabla 4.11 corresponde a los resultados obtenidos de EC para las operaciones de cifrado/descifrado, firma/verificación y generación de claves utilizando un campo primo. Los valores de entrada para estas operaciones fueron variados, primero en la curva utilizada, es importante recalcar que las curvas utilizadas ofrecen un nivel de seguridad equivalente a los tamaños de clave utilizados en las pruebas de RSA, de manera que los resultados obtenidos fueran comparables. Adicionalmente se vario el tamaño de la entrada para los algoritmos de cifrado y firma digital, mientras que para estos últimos también se vario la función picadillo que utilizo.

Operación	Curva	Tamaño de entrada (bits)	Picadillo	Tiempo (ms)		
				D-1	D-2	D-3
Cifrado	P-192	1024	——	144.84	183.60	478.52
Descifrado				145.18	181.21	477.53
Cifrado	P-192	3072	——	149.83	189.95	495.07
Descifrado				150.97	188.62	491.56
Cifrado	P-256	1024	——	211.44	274.83	684.52
Descifrado				212.67	274.77	684.88
Cifrado	P-256	3072	——	207.50	270.19	703.53
Descifrado				212.18	269.26	702.04
Cifrado	P-384	1024	——	362.71	476.27	1295.04
Descifrado				364.70	476.16	1295.65
Cifrado	P-384	3072	——	371.19	479.88	1296.22
Descifrado				369.80	479.88	1294.91
Firma	P-192	1024	SHA-1	145.49	184.43	518.33
Verificación				196.19	254.08	711.28
Firma	P-192	1024	SHA-256	145.43	185.89	468.85
Verificación				196.48	254.33	638.48
Firma	P-192	1024	SHA-512	144.49	185.33	466.06
Verificación				196.77	254.16	642.87
Firma	P-192	1024	MD5	143.18	187.16	473.14
Verificación				196.72	255.38	636.90
Firma	P-192	1024	RIPEMD160	144.33	188.51	467.37
Verificación				197.82	254.79	643.27
Firma	P-192	3072	SHA-1	143.32	185.31	466.40
Verificación				198.16	253.15	638.94
Firma	P-192	3072	SHA-256	144.04	187.35	469.74
Verificación				199.62	254.34	642.67
Firma	P-192	3072	SHA-512	145.16	184.51	473.45
Verificación				197.45	257.41	644.70
Firma	P-192	3072	MD5	144.99	183.84	465.40
Verificación				195.61	253.97	636.70
Firma	P-192	3072	RIPEMD160	146.15	188.61	473.05
Verificación				199.23	257.88	643.50

Operación	Curva	Tamaño de entrada (bits)	Picadillo	Tiempo (ms)		
				D-1	D-2	D-3
Firma	P-256	1024	SHA-1	205.82	278.78	681.30
Verificación				282.82	363.94	935.52
Firma	P-256	1024	SHA-256	204.71	277.08	686.91
Verificación				284.66	364.82	928.66
Firma	P-256	1024	SHA-512	205.11	274.79	679.67
Verificación				284.78	362.97	951.32
Firma	P-256	1024	MD5	204.76	281.54	696.61
Verificación				284.14	355.02	936.03
Firma	P-256	1024	RIPEMD160	202.98	273.57	682.97
Verificación				286.79	371.86	948.73
Firma	P-256	3072	SHA-1	204.63	274.90	699.83
Verificación				286.41	361.66	948.39
Firma	P-256	3072	SHA-256	209.28	267.87	680.30
Verificación				289.05	375.44	925.11
Firma	P-256	3072	SHA-512	206.35	274.99	685.54
Verificación				286.86	365.30	923.15
Firma	P-256	3072	MD5	205.46	277.73	682.15
Verificación				284.93	359.82	917.13
Firma	P-256	3072	RIPEMD160	209.76	269.13	677.11
Verificación				285.15	375.37	931.77
Firma	P-384	1024	SHA-1	362.32	481.91	1295.20
Verificación				516.11	678.27	1821.79
Firma	P-384	1024	SHA-256	360.43	603.03	1295.03
Verificación				518.02	855.31	1810.08
Firma	P-384	1024	SHA-512	365.51	501.13	1284.04
Verificación				514.81	703.32	1806.29
Firma	P-384	1024	MD5	364.04	481.76	1273.03
Verificación				518.86	678.08	1803.26
Firma	P-384	1024	RIPEMD160	364.02	476.98	1277.20
Verificación				515.56	679.59	1812.43
Firma	P-384	3072	SHA-1	364.39	478.12	1287.07
Verificación				514.83	681.11	1814.37
Firma	P-384	3072	SHA-256	365.68	479.95	1283.97
Verificación				517.91	680.06	1810.30
Firma	P-384	3072	SHA-512	364.84	487.12	1287.47
Verificación				518.33	692.94	1814.38
Firma	P-384	3072	MD5	365.19	479.45	1301.56
Verificación				516.09	680.94	1822.31
Firma	P-384	3072	RIPEMD160	364.73	479.87	1287.33
Verificación				517.74	681.75	1818.49
Generar clave	P-192	—	—	154.03	199.70	575.31
Generar clave	P-256	—	—	211.54	276.43	800.93
Generar clave	P-384	—	—	375.29	488.29	1372.47

Tabla 4.11: Resultados pruebas de rendimiento - EC.

Por otra parte la tabla 4.12 corresponde a los resultados obtenidos para ambos algoritmos en las funciones de exportación e importación de claves. Las pruebas de estas

funciones abarcan cada uno de los tipos de clave para cada algoritmo y muestra algunos de los posibles formatos en los que la API es capaz de exportar e importar claves asimétricas.

Algoritmo	Operación	Tamaño de clave (bits)	Tipo de clave	Formato	Tiempo (ms)		
					D-1	D-2	D-3
RSA	Exportar	1024	Pública	DER	3.38	3.42	4.25
	Importar				0.29	0.36	0.79
RSA	Exportar	1024	Privada	PEM	4.46	5.85	15.93
	Importar				4.93	3.56	12.27
RSA	Exportar	1024	Privada	PEM AES-256-CBC	366.54	512.88	1386.99
	Importar				369.80	507.01	1396.60
RSA	Exportar	1024	Privada	PEM PBE-SHA1-3DES	98.06	142.49	359.22
	Importar				94.07	137.81	376.99
RSA	Exportar	1024	Par	PKCS12	133.52	226.67	456.27
	Importar				120.77	214.24	380.17
RSA	Exportar	2048	Pública	DER	3.46	3.37	4.20
	Importar				0.27	0.34	0.81
RSA	Exportar	2048	Privada	PEM	4.66	5.82	20.15
	Importar				5.84	4.88	16.53
RSA	Exportar	2048	Privada	PEM AES-256-CBC	372.95	510.87	1396.40
	Importar				367.91	510.96	1353.76
RSA	Exportar	2048	Privada	PEM PBE-SHA1-3DES	98.01	147.21	391.55
	Importar				97.44	141.88	366.41
RSA	Exportar	2048	Par	PKCS12	126.63	166.75	467.07
	Importar				109.07	143.39	405.77
RSA	Exportar	4096	Pública	DER	3.47	3.50	4.82
	Importar				0.28	0.31	0.59
RSA	Exportar	4096	Privada	PEM	7.55	7.09	26.89
	Importar				5.43	5.26	22.78
RSA	Exportar	4096	Privada	PEM AES-256-CBC	370.98	514.76	1344.05
	Importar				369.15	513.93	1389.32
RSA	Exportar	4096	Privada	PEM PBE-SHA1-3DES	103.76	145.82	393.80
	Importar				96.05	142.96	387.56
RSA	Exportar	4096	Par	PKCS12	124.89	164.80	451.14
	Importar				116.91	154.72	434.44
EC	Exportar	P-192	Pública	DER	4.46	5.20	8.43
	Importar				1.18	1.14	3.53
EC	Exportar	P-192	Privada	PEM	7.59	9.37	17.15
	Importar				4.33	2.98	13.70
EC	Exportar	P-192	Privada	PEM AES-256-CBC	375.40	490.68	1353.12
	Importar				374.95	487.00	1364.64
EC	Exportar	P-192	Privada	PEM PBE-SHA1-3DES	102.25	129.46	374.77
	Importar				96.31	131.56	369.62
EC	Exportar	P-192	Par	PKCS12	127.87	167.67	453.20
	Importar				109.79	148.46	404.44
EC	Exportar	P-256	Pública	DER	4.84	4.94	9.38
	Importar				0.84	1.25	2.25
EC	Exportar	P-256	Privada	PEM	9.00	5.90	17.70
	Importar				2.79	6.11	13.80
EC	Exportar	P-256	Privada	PEM AES-256-CBC	380.89	490.29	1348.41
	Importar				377.88	486.57	1358.80
EC	Exportar	P-256	Privada	PEM	98.61	129.51	360.59

Algoritmo	Operación	Tamaño de clave (bits)	Tipo de clave	Formato	Tiempo (ms)		
					D-1	D-2	D-3
	Importar			PBE-SHA1-3DES	99.82	132.38	355.42
EC	Exportar	P-256	Par	PKCS12	125.42	166.00	460.08
	Importar				113.35	151.18	419.58
EC	Exportar	P-384	Pública	DER	4.60	5.42	9.35
	Importar				1.19	1.04	2.79
EC	Exportar	P-384	Privada	PEM	7.70	9.10	17.77
	Importar				4.12	2.76	13.63
EC	Exportar	P-384	Privada	PEM AES-256-CBC	375.85	494.13	1312.52
	Importar				372.17	486.73	1407.29
EC	Exportar	P-384	Privada	PEM PBE-SHA1-3DES	100.63	133.93	353.15
	Importar				99.92	133.80	374.58
EC	Exportar	P-384	Par	PKCS12	128.31	169.34	531.34
	Importar				111.07	148.46	465.62

Tabla 4.12: Resultados pruebas de rendimiento - Exportar/Importar.

Por último la tabla 4.13 muestra los resultados obtenidos durante las pruebas a las funciones de creación y validación de certificados y CRL. En esta tabla se puede apreciar la capacidad de la API para generar diferentes tipos de certificado dependiendo de los parámetros utilizados, tal es el caso de la creación y validación de certificados mixtos, dependiendo de la clave del emisor será el algoritmo de firma que se podrá utilizar, para la prueba se utilizaron 3 algoritmos para RSA y 3 para EC. Es importante destacar se utilizó una seguridad equivalente a 80 bits en esta prueba.

Operación	Clave Emisor	Clave Titular	Algoritmo de firma	Tipo	Tiempo (ms)		
					D-1	D-2	D-2
Crear Cert.	RSA	RSA	SHA256withRSA	Auto-firmado	49.52	63.61	167.42
Crear Cert.	RSA	RSA	MD5withRSA	Auto-firmado	43.12	58.32	149.64
Crear Cert.	RSA	RSA	RIPEMD160withRSA	Auto-firmado	43.96	58.68	158.39
Crear Cert.	RSA	RSA	SHA256withRSA	Firmado por CA	58.35	75.38	208.01
Crear Cert.	RSA	RSA	MD5withRSA	Firmado por CA	53.86	70.70	203.98
Crear Cert.	RSA	RSA	RIPEMD160withRSA	Firmado por CA	55.45	72.01	183.43
Crear Cert.	RSA	EC	SHA256withRSA	Firmado por CA	57.38	75.56	198.32
Crear Cert.	RSA	EC	MD5withRSA	Firmado por CA	53.02	68.54	274.96
Crear Cert.	RSA	EC	RIPEMD160withRSA	Firmado por CA	56.19	71.95	482.59
Crear Cert.	EC	EC	SHA1withECDSA	Auto-firmado	388.28	510.65	1292.88
Crear Cert.	EC	EC	SHA256withECDSA	Auto-firmado	393.32	511.34	1282.39
Crear Cert.	EC	EC	SHA512withECDSA	Auto-firmado	389.08	511.39	1292.96
Crear Cert.	EC	RSA	SHA1withECDSA	Firmado por CA	402.25	528.81	3099.81
Crear Cert.	EC	RSA	SHA256withECDSA	Firmado por CA	401.96	524.63	1481.47
Crear Cert.	EC	RSA	SHA512withECDSA	Firmado por CA	403.62	526.69	1314.14
Crear Cert.	EC	EC	SHA1withECDSA	Firmado por CA	405.51	539.13	1317.10
Crear Cert.	EC	EC	SHA256withECDSA	Firmado por CA	407.91	540.25	1332.21
Crear Cert.	EC	EC	SHA512withECDSA	Firmado por CA	407.21	533.18	1318.46
Verificar Cert.	RSA	RSA	SHA256withRSA	Auto-firmado	2.92	3.80	5.84

Operación	Clave Emisor	Clave Titular	Algoritmo de firma	Tipo	Tiempo (ms)		
					D-1	D-2	D-2
Verificar Cert.	RSA	RSA	MD5withRSA	Auto-firmado	4.64	3.40	13.06
Verificar Cert.	RSA	RSA	RIPEMD160withRSA	Auto-firmado	4.77	2.94	5.83
Verificar Cert.	RSA	RSA	SHA256withRSA	Firmado por CA	12.44	13.30	22.11
Verificar Cert.	RSA	RSA	MD5withRSA	Firmado por CA	12.87	12.10	20.62
Verificar Cert.	RSA	RSA	RIPEMD160withRSA	Firmado por CA	13.76	13.06	55.52
Verificar Cert.	RSA	EC	SHA256withRSA	Firmado por CA	11.08	12.81	47.91
Verificar Cert.	RSA	EC	MD5withRSA	Firmado por CA	11.35	11.95	68.61
Verificar Cert.	RSA	EC	RIPEMD160withRSA	Firmado por CA	11.37	12.83	112.42
Verificar Cert.	EC	EC	SHA1withECDSA	Auto-firmado	5.18	6.66	9.25
Verificar Cert.	EC	EC	SHA256withECDSA	Auto-firmado	4.87	6.19	9.23
Verificar Cert.	EC	EC	SHA512withECDSA	Auto-firmado	4.10	5.52	9.30
Verificar Cert.	EC	RSA	SHA1withECDSA	Firmado por CA	196.18	267.71	1591.71
Verificar Cert.	EC	RSA	SHA256withECDSA	Firmado por CA	197.76	265.44	766.81
Verificar Cert.	EC	RSA	SHA512withECDSA	Firmado por CA	197.88	268.11	679.41
Verificar Cert.	EC	EC	SHA1withECDSA	Firmado por CA	202.50	268.54	678.01
Verificar Cert.	EC	EC	SHA256withECDSA	Firmado por CA	202.55	269.57	688.30
Verificar Cert.	EC	EC	SHA512withECDSA	Firmado por CA	201.90	267.54	678.32
Crear CRL	RSA	——	SHA256withRSA	——	38.39	48.51	124.11
Crear CRL	RSA	——	MD5withRSA	——	37.72	47.75	122.78
Crear CRL	RSA	——	RIPEMD160withRSA	——	38.33	48.43	125.95
Crear CRL	EC	——	SHA1withECDSA	——	178.74	234.59	586.30
Crear CRL	EC	——	SHA256withECDSA	——	177.78	235.86	612.68
Crear CRL	EC	——	SHA512withECDSA	——	177.86	231.99	585.77
Verificar CRL	RSA	——	SHA256withRSA	——	3.48	4.08	11.38
Verificar CRL	RSA	——	MD5withRSA	——	3.43	4.23	10.45
Verificar CRL	RSA	——	RIPEMD160withRSA	——	3.04	3.48	10.71
Verificar CRL	EC	——	SHA1withECDSA	——	8.07	9.69	17.00
Verificar CRL	EC	——	SHA256withECDSA	——	8.00	8.66	17.49
Verificar CRL	EC	——	SHA512withECDSA	——	8.63	8.90	17.59

Tabla 4.13: Resultados pruebas de rendimiento - X.509.

Capítulo 5

Aplicación de prueba para Android

5.1. Android para dispositivos móviles

Android constituye una pila de software pensada especialmente para dispositivos móviles y que incluye tanto un sistema operativo, como *middleware* y diversas aplicaciones de usuario. Representa la primera incursión de Google en el mercado móvil.

En general, una pila o plataforma de software es un elemento crucial en el desarrollo del mismo, ya que nos proporciona un marco de trabajo que permite crear nuevo software y que éste se pueda ejecutar sobre ella posteriormente. Lo anterior puede ser visto como un modelo de capas en dónde las plataforma de desarrollo funge como intermediario entre el hardware y las aplicaciones que se han desarrollado (ver Figura 5.1). Las plataformas de desarrollo típicas incluyen un sistema operativo (S.O.), lenguajes de programación, sus correspondientes bibliotecas de funciones e interfaces gráficas (*User Interface* o UI).

Figura 5.1: Modelo de capas para desarrollo de software.

Todas las aplicaciones para *Android* se programan en lenguaje *Java* y son ejecutadas en una máquina virtual especialmente diseñada para esta plataforma, que ha sido bautizada con el nombre de *Dalvik*. El núcleo de *Android* está basado en Linux 2.6 y es distribución libre. A los desarrolladores se les proporciona de forma gratuita un SDK y la opción de un *plug-in* para el entorno de desarrollo Eclipse que incluye todas las APIs necesarias para la creación de aplicaciones, así como un emulador integrado para su ejecución. Existe además disponible una amplia documentación de respaldo para este SDK.

El proyecto *Android* está capitaneado por Google y un conjunto de empresas tecnológicas agrupadas bajo el nombre de *Open Handset Alliance* (OHA). El objetivo principal de esta alianza empresarial (que incluye a fabricantes de dispositivos y operadores, con firmas tan relevantes como Samsung, LG, Telefónica, Intel o Texas Instruments, entre muchas otras) es el desarrollo de estándares abiertos para la telefonía móvil como medida para incentivar su desarrollo y para mejorar la experiencia del usuario.

Con *Android* se busca reunir en una misma plataforma todos los elementos necesarios que permitan al desarrollador controlar y aprovechar al máximo cualquier funcionalidad ofrecida por un dispositivo móvil (llamadas, mensajes de texto, cámara, agenda de contactos, conexión Wi-Fi, Bluetooth, aplicaciones ofimáticas, videojuegos, etc.), así como poder crear aplicaciones que sean verdaderamente portables, reutilizables y de rápido desarrollo. En otras palabras, *Android* quiere mejorar y estandarizar el desarrollo de aplicaciones para cualquier dispositivo móvil.

Existen algunas diferencias que hacen de *Android* una opción muy interesante para los fabricantes, y por supuesto, para los usuarios y desarrolladores. A diferencia de sus competidores, *Android* es software libre, lo que permite que los fabricantes puedan usarlo sin necesidad de pagar. Por otra parte, al tener como base Linux, es fácilmente portable y adaptable a casi cualquier hardware. *Android* no es la primera plataforma de desarrollo móvil basado en Linux y que es software libre, Nokia abandera el proyecto *Maemo*, e incluso Ubuntu desarrolla *Ubuntu Mobile*, pero no parecen alcanzar la masa crítica necesaria [Garcia, 2010].

5.1.1. Características

El diseño de *Android* cuenta, entre otras, con las siguientes características:

- Componentes básicos de las aplicaciones, los cuales se pueden sustituir fácilmente por otros.
- Máquina virtual propia, *Dalvik*, que interpreta y ejecuta código escrito en Java.
- Representación de gráficos en 2D y 3D.
- Almacenamiento de datos en SQLite.
- Servicio de localización GSM.
- Soporte para diferentes formatos de contenido multimedia: MPEG-4, H.264, MP3, AAC, OGG, AMR, JPEG, PNG, GIF.
- Conectividad (GSM/EDGE, CDMA, EV-DO, UMTS, Bluetooth y Wi-Fi).
- Soporte para hardware adicional: cámaras de vídeo, pantallas táctiles, GPS, acelerómetros, entre otros.
- Mensajería (SMS y MMS).

- Navegador Web.
- Entorno de desarrollo integral que incluye: un emulador, herramientas de depuración, perfiles de memoria y funcionamiento, un *plug-in* para Eclipse IDE.

5.1.2. Arquitectura

En esta sección se dará una visión global de las capas que integran la arquitectura de *Android*. Cada una de estas capas utiliza servicios ofrecidos por las anteriores, y a su vez ofrece ciertos servicios a las capas superiores [Garcia, 2010, Tudela, 2009] (ver Figura 5.2).

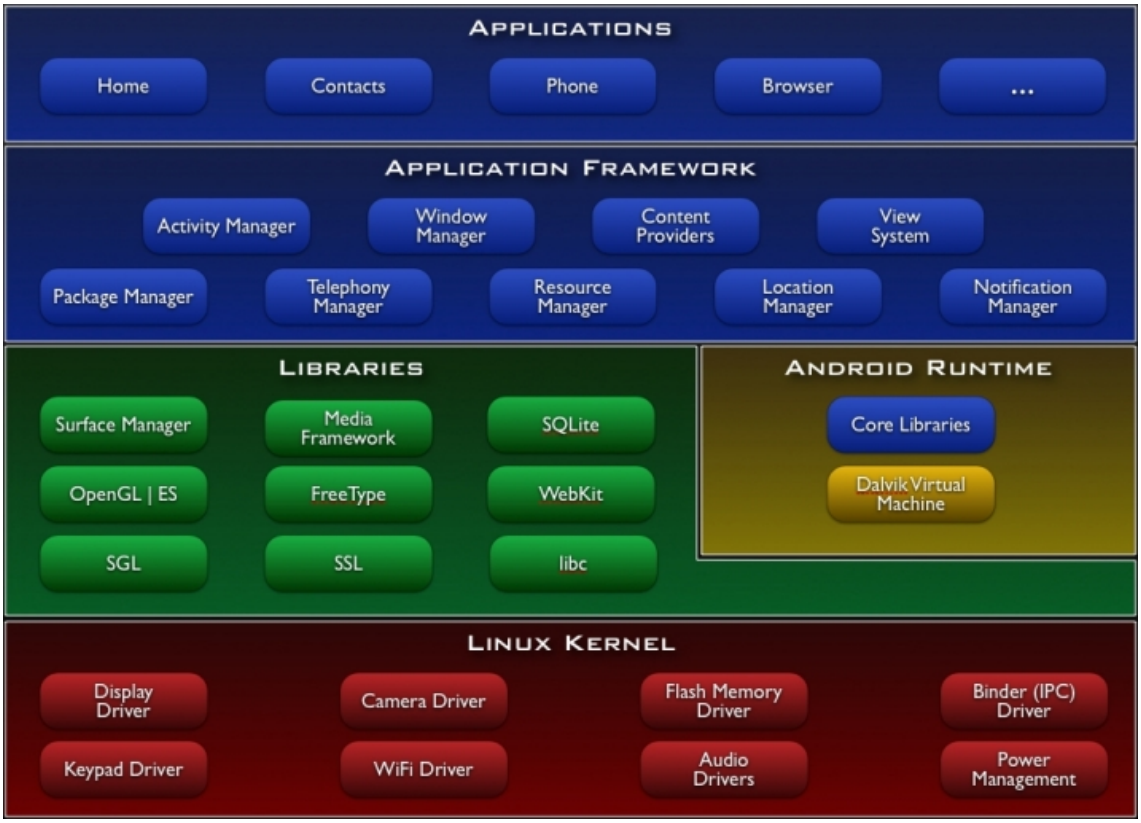


Figura 5.2: Arquitectura de *Android*.

- Aplicaciones: es la capa superior de la pila, éste nivel incluye tanto las aplicaciones incluidas por defecto de *Android* como aquellas que el usuario vaya añadiendo posteriormente, ya sean de terceras empresas o de su propio desarrollo. Todas estas aplicaciones utilizan los servicios, las API y bibliotecas de los niveles que se encuentran debajo.
- *Framework* de aplicaciones: los desarrolladores tienen acceso completo a las APIs del *framework* usado por las aplicaciones base. La arquitectura está diseñada para simplificar el reuso de componentes; cualquier aplicación puede publicar sus capacidades

y cualquier otra puede hacer uso de esas capacidades (sujeto a reglas de seguridad del *framework*). Éste mismo mecanismo permite que los componentes sean reemplazados por el usuario. Este *framework* está formado por varios componentes, entre los cuales podemos encontrar:

- Un extenso conjunto de Vistas tales como listas, cajas de texto, botones, entre otros.
- *Content Providers*, que permiten a las aplicaciones acceder a información de otras aplicaciones o compartir la propia.
- *Resource Manager*, que proporciona acceso a recursos que no son código como pueden ser gráficos, cadenas de texto, etc.
- *Notification Manager*, que permite a las aplicaciones mostrar alarmas personalizadas en la barra de estado.
- *Activity Manager*, que gestiona el ciclo de vida de las aplicaciones.

- Bibliotecas: éstas han sido escritas utilizando C/C++ y proporcionan a *Android* la mayor parte de sus capacidades más características. Junto con el núcleo basado en Linux, estas bibliotecas constituyen el corazón de *Android*. Algunas de ellas son:
  - La biblioteca *libc* incluye todas las cabeceras y funciones según el estándar del lenguaje C. Todas las demás bibliotecas se definen en este lenguaje.
  - La biblioteca *Surface Manager* es la encargada de componer los diferentes elementos de navegación de pantalla. Gestiona también las ventanas pertenecientes a las distintas aplicaciones.
  - *OpenGL/SL y SGL* representan las bibliotecas gráficas y, por tanto, sustentan la capacidad gráfica de *Android*.
  - La biblioteca *Media Libraries* proporciona todos los códecs necesarios para el contenido multimedia soportado en *Android* (vídeo, audio, imágenes estáticas y animadas, etc.)
  - *FreeType*, permite trabajar de forma rápida y sencilla con distintos tipos de fuentes.
  - La biblioteca *SSL* posibilita la utilización de dicho protocolo para establecer comunicaciones seguras.
  - A través de la biblioteca *SQLite*, *Android* ofrece la creación y gestión de bases de datos relacionales, pudiendo transformar estructuras de datos en objetos fáciles de manejar por las aplicaciones.
  - La biblioteca *WebKit* proporciona un motor para las aplicaciones de tipo navegador y forma el núcleo del actual navegador incluido por defecto en la plataforma *Android*

- *Runtime* de *Android*: al mismo nivel que las bibliotecas de *Android* se sitúa el entorno de ejecución. Éste lo constituyen el conjunto de bibliotecas conocidas como '*Core Libraries*', que son bibliotecas con multitud de clases de Java y la máquina virtual *Dalvik*. Dentro de la máquina virtual se ejecutan archivos en el formato *Dalvik Executable* (.dex), el cual está optimizado para dispositivos con pocos recursos en cuanto a memoria se refiere.
- Núcleo de linux: *Android* utiliza el núcleo de Linux 2.6 como una capa de abstracción para el hardware disponible en los dispositivos móviles. Esta capa contiene los drivers necesarios para que cualquier componente de hardware pueda ser utilizado por las aplicaciones de la capa superior.

5.1.3. Componentes de una aplicación

Todas las aplicaciones en *Android* pueden descomponerse en cuatro tipos de bloques o componentes principales: *Activity*, *Broadcast Intent Receiver*, *Service* y *Content Provider*. Cada aplicación de *Android* será una combinación de uno o más de estos componentes.

Los componentes de una aplicación, deberán ser declarados de forma explícita en un archivo con formato XML denominado AndroidManifest.xml [Garcia, 2010, Tudela, 2009], junto a otros datos asociados como nombre de la aplicación, versión, valores globales, clases que implementa, datos que puede manejar, permisos, etc. Este archivo es básico en cualquier aplicación en *Android* y permite al sistema desplegar y ejecutar correctamente la aplicación.

A continuación se detallan los cuatro tipos de componentes en los que puede dividirse una aplicación para *Android* [Tudela, 2009].

Activity

Sin duda es el componente más habitual de las aplicaciones para *Android*. Un componente *Activity* refleja una determinada actividad llevada a cabo por una aplicación, que lleva asociada típicamente una ventana o interfaz de usuario; es importante señalar que no contempla únicamente el aspecto gráfico, sino que éste forma parte del componente *Activity* a través de vistas representadas por clases como *View* y sus derivadas. Este componente se implementa mediante la clase de mismo nombre *Activity*.

Vinculado a este componente se encuentran los *Intents*, una interesante novedad introducida por *Android*. Un *Intent* consiste básicamente en la voluntad de realizar alguna acción, generalmente asociada a unos datos. Lanzando un *Intent*, una aplicación puede delegar el trabajo en otra, de forma que el sistema se encarga de buscar qué aplicación entre las instaladas es la que puede llevar a cabo la acción solicitada. Por ejemplo, abrir una URL en algún navegador web o escribir un correo electrónico desde algún cliente de correo.

Broadcast Intent Receiver

Un componente *Broadcast Intent Receiver* se utiliza para iniciar alguna otra acción dentro de la aplicación actual cuando un determinado evento se produzca (generalmente, abrir un componente *Activity*). Por ejemplo, una llamada entrante o un SMS recibido. No tiene interfaz de usuario asociada, pero puede utilizar el API *Notification Manager*, mencionada anteriormente, para avisar al usuario del evento producido a través de la barra de estado del dispositivo móvil. Este componente se implementa a través de una clase de nombre *BroadcastReceiver*.

Para que *Broadcast Intent Receiver* funcione, no es necesario que la aplicación en cuestión sea la aplicación activa en el momento de producirse el evento. El sistema lanzará la aplicación si es necesario cuando el evento monitorizado tenga lugar.

Service

Un componente *Service* representa una aplicación ejecutada sin interfaz de usuario y que generalmente tiene lugar en segundo plano mientras otras aplicaciones (con interfaz) son las que están activas en la pantalla del dispositivo. Un ejemplo típico de este componente es un reproductor de música. La interfaz del reproductor muestra al usuario las distintas canciones disponibles, así como los típicos botones de reproducción, pausa, volumen, etc. En el momento en el que el usuario reproduce una canción, ésta se escucha mientras se realiza alguna otra acción. Este elemento está implementado por la clase de mismo nombre *Service*.

Content Provider

Con el componente *Content Provider*, cualquier aplicación en *Android* puede almacenar datos en un fichero, en una base de datos SQLite o en cualquier otro elemento. Además, estos datos pueden ser compartidos entre distintas aplicaciones. Una clase que implemente el componente *Content Provider* contendrá una serie de métodos que permiten almacenar, recuperar, actualizar y compartir los datos de una aplicación. Existe una colección de clases para distintos tipos de gestión de datos en el paquete `android.provider`.

5.2. Aplicación de prueba

Para verificar el correcto funcionamiento de la API generada y comprobar la posibilidad de construir una PKI con ayuda de la misma, se implementó una aplicación de prueba para dispositivos móviles. En esta sección se presentan y explican sus principales características, adicionalmente se presenta el diagrama de paquetes, con el cual se pretende ilustrar de manera general el diseño de esta aplicación. Para finalizar en el

apéndice A el lector encontrará el manual de uso con las principales interfaces detallando las características más sobresalientes de esta aplicación.

5.2.1. Características principales

En base al objetivo principal de esta aplicación, que es ilustrar de una mejor manera la gran mayoría de las opciones con las que cuenta la API, la aplicación cuenta con las siguientes características:

- Hace uso de un modelo de red de confianza.
- Contempla el uso de la aplicación por varios usuarios en el mismo dispositivo.
- Permite asignar alias a los diferentes usuarios del sistema.
- Asigna un alias a los propietarios de claves y certificados en la aplicación, de manera que sea sencillo identificar al propietario de estos.
- Permite al usuario administrar las claves, tomando en cuenta los siguientes aspectos:
  - Creación de claves públicas y privadas tanto para RSA como para EC.
  - Mostrar los detalles de las claves almacenadas en la aplicación.
  - Las claves privadas son protegidas utilizando el estándar PKCS#8 y/o PKCS#12.
  - Permite eliminar las claves creadas.
  - Actualizar algunos campos de la clave, como lo son los comentarios y la contraseña.
  - Importar y exportar las claves almacenadas utilizando todos los formatos disponibles en la API.
  - La aplicación permite enviar las claves privadas utilizando diversos medios del dispositivo, como lo son: correo elemento, bluetooth, google drive, entre otros.
  - Para realizar cualquier interacción con claves privadas, la aplicación válida que el usuario tenga permiso para hacerlo pidiendo la contraseña con la cual se almaceno la clave.
- Permite la administración de certificados digitales, contemplando lo siguiente:
  - Creación de certificados digitales auto-firmados a partir de claves existentes o creando un nuevo par.
  - Permite la creación de múltiples certificados para una misma clave.
  - Asocia un certificado a un usuario en la aplicación.
  - Almacena un estatus en la base de datos y una fecha de actualización del mismo, este estatus se actualiza cada que se realiza la verificación de un certificado.

- Los certificados creados siguen el estándar X.509, por lo que cuentan con todos los campos y extensiones obligatorias.
- Al crear un certificado la aplicación permite al usuario ingresar todos los campos del mismo como lo son: periodo de validez, atributos del titular, algoritmo de firma, usos de la clave y el valor para las extensiones propietarias.
- La muestra todos los detalles del certificado, utilizando diferentes vistas para la información de la base de datos, información del titular, emisor y extensiones.
- La aplicación permite firmar certificados existentes utilizando cualquier clave privada almacenada en la misma, siempre y cuando el usuario cuente con el certificado correspondiente a esta clave y la contraseña de la misma.
- Al firmar un certificado, la aplicación permite modificar la información del mismo.
- Verificación de certificados utilizando los métodos explicados en la sección 3.1.2 de este documento.
- Exportar e importar certificados utilizando codificación `DER` o `PEM`.
- Enviar certificados utilizando los medios disponibles en el dispositivo, como lo son: correo electrónico, bluetooth, google drive, entre otros.

- Administración de la red de confianza, mediante el uso de listas de confianza, en las cuales cada usuario asigna un nivel de confianza a otro certificado, este nivel señala que tanto se confía en dicho certificado como emisor de certificados a terceros. la administración de estas listas contempla:
  - Las listas de confianza son asignadas a un usuario en particular del sistema.
  - Al mostrar los detalles de una lista, se muestran todos los detalles de los certificados pertenecientes a la misma y para cada uno de estos señala el nivel de confianza.
  - Permite modificar el nivel de confianza a un certificado.
  - El nivel de confianza puede variar entre -10 y 10.
  - La aplicación permite eliminar un certificado de la lista en cualquier momento.
  - Al agregar un nuevo certificado a una lista, la aplicación valida que este no exista previamente en la lista seleccionada.
- La aplicación cuenta con un modulo sencillo para CRL, el cual permite:
  - Ver el listado de todas las CRL en el dispositivo.
  - Importar y exportar las listas.
  - Creación de una nueva lista, a la cual únicamente se pueden agregar los certificados creados por la CA emisora de la lista.

  - Ver los detalles de los certificados almacenados en la lista.

- Por último la aplicación cuenta con un modulo para cifrar, descifrar, firmar y verificar tanto archivos como mensajes de texto, utilizando las claves y certificados almacenados en la base de datos.

- Como se podrá ver la sección de interfaces de usuario, la aplicación fue creada siguiendo el modelo de diseño *Pure Android*, por lo que utiliza componentes disponibles a partir de la versión 3.0 de *Android*. Sin embargo la aplicación funciona correctamente en dispositivos que tengan como mínimo *Android* 2.1.

5.2.2. Diagrama de Paquetes

La figura 5.3 presenta el diagrama de paquetes de la aplicación de prueba, en este diagrama se puede observar la estructura general de la misma:

- `main`: este es el paquete principal de la aplicación, contiene el *Activity* principal y los *Fragments* que lo conforman.

- `update`: contiene los *Activity* y clases utilizadas para actualizar la información de los diversos elementos en la aplicación como lo son claves y listas de confianza.

- `share`: almacena las clases que implementan los *Activity* utilizados para compartir certificados, claves, listas de confianza y CRL en la aplicación.

- `add`: en este paquete se encuentran las clases que representan los componentes necesarios para agregar un nuevo elemento a la aplicación, estos componentes pueden ser *Activities* o *Fragments*.

- `details`: este es uno de los paquetes más complejos de la aplicación ya que contiene una gran cantidad de clases que representan los *Fragments* utilizados en los diversos paginadores de la aplicación, los cuales muestran los detalles de cada uno de los elementos de la PKI.

- `crypto`: contiene las clases referentes a procesos criptográficos dentro de la aplicación, principalmente contiene los *Fragments* referentes a las operaciones de cifrado/decifrado y firma/verificación tanto de mensajes como archivos.

- `common`: almacena clases que implementan componentes comunes para los demás paquetes, por ejemplo los cuadros de dialogo utilizados en la aplicación y el *Fragment* utilizado para abrir el explorador de archivos.

- `adapter`: este paquete contiene todas las clases que implementan los diversos adaptadores encargados de comunicarse con la capa de datos y crear listas de objetos en la aplicación, estos adaptadores son utilizados para crear diseños personalizados para mostrar información especifica en cada entrada de una lista

- `selection`: finalmente, este paquete contiene una serie de clases que heredan de `ListFragment`, por lo que cada una de estas clases crea una lista para ser mostrada en la aplicación, estas listas implementan algunos comportamientos personalizados como: permitir la selección de un elemento de la lista o mostrar un botón en cada entrada de la lista para mostrar más detalles.

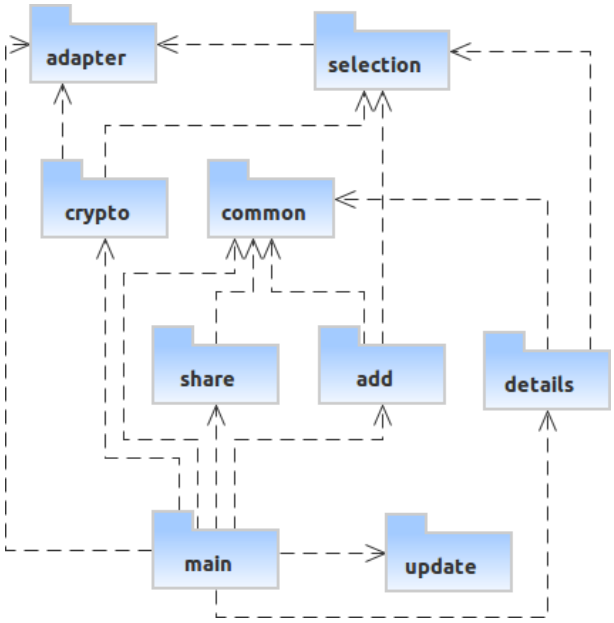


Figura 5.3: Diagrama de paquetes - Aplicación de prueba.

Capítulo 6

Conclusiones y trabajo a futuro

En este trabajo se expuso la importancia que tiene la seguridad en cualquier aspecto de la tecnología, principalmente en el ambiente de dispositivos móviles debido a las capacidades de conectividad que estos dispositivos poseen, se dieron las bases teóricas para tener una mejor comprensión de los servicios de seguridad y los mecanismos existentes para poder brindar estos servicios, particularmente se expuso la PKI como una solución integral en cuanto a servicios de seguridad se refiere.

En base a esto se realizó el diseño de una PKI para dispositivos móviles, la cual toma en consideración los estándares existentes y los mecanismos criptográficas más utilizados actualmente. La implementación de este diseño se realizó en *Android* de manera que el resultado final de este trabajo fuera una API para el desarrollo de aplicaciones seguras. Finalmente, para aterrizar todas la ideas planteadas a lo largo de este trabajo se implementó una aplicación de prueba utilizando la API generada.

En este capítulo se presentan las conclusiones resultantes del trabajo de investigación que se llevo a cabo y para finalizar se mencionan algunas propuestas que se pueden considerar como trabajo a futuro.

6.1. Conclusiones

Al finalizar este trabajo se puede observar una dos de aportes significativos al campo de la seguridad informática, entre los cuales destaca el proceso de diseño propuesto para generar una PKI con todas las características deseables de una infraestructura de este tipo y la generación de una API para la creación de aplicaciones móviles seguras que utilicen una PKI como mecanismo de seguridad para proteger las comunicaciones o archivos utilizados en un sistema.

A pesar de que este trabajo toma como base una biblioteca que contiene la implementación de los algoritmos criptográficos utilizados, la API generada es de gran utilidad

para un desarrollador de aplicaciones móviles, ya que en ella se encapsula gran parte de la dificultad que tiene esta biblioteca para poder utilizar los algoritmos que contiene, ya que esta biblioteca esta pensada para ser utilizada por usuarios con un nivel alto de conocimientos en criptografía mientras que la API está pensada y diseñada para ser utilizada tanto por usuarios con conocimientos avanzados de criptografía como por usuarios con conocimientos muy básicos sobre este tema.

El diseño presentado en este trabajo, corresponde a una PKI hasta cierto punto genérica que puede ser adaptada y configurada según las necesidades del sistema donde se utilice, se plantearon las características de seguridad mínimas que se deben cumplir para que la aplicación resultante tenga un nivel de seguridad confiable y sea compatible con otras aplicaciones gracias al uso de estándares como X.509 y PKCS.

Este trabajo se enfocó principalmente al uso de dos modelos de confianza en una PKI, el modelo jerárquico, el cual se tomó como base para la descripción teórica de una infraestructura que utilice el diseño presentado, como parte del diseño se escribieron las políticas de certificación que deberán ser seguidas para garantizar la seguridad del sistema, uno de los puntos más importantes de estas políticas es la definición del plan de administración de certificados, ya que en una implementación real del modelo jerárquico se requiere de la interacción entre los titulares y la CA para que ésta pueda garantizar que la identidad almacenada en un certificado sea confiable. Sin embargo la implementación de esta infraestructura no se llevó a cabo en este trabajo debido a la necesidad de un repositorio central o servidor que representara a la CA raíz.

Por otra pare se utilizó el diseño creado para la implementación de una red de confianza para dispositivos móviles, este modelo es mucho más adecuado para el ambiente estos dispositivos, ya que si bien estos tienen grandes capacidades de conexión a redes, la mayor parte del tiempo no se están conectados a ninguna, por lo que la necesidad de tener un repositorio central no era la opción más viable.

Adicionalmente en este trabajo se pudo comprobar que la utilización de mecanismos de criptografía asimétrica en dispositivos móviles es cada día más factible, gracias a la rápida evolución de estos dispositivos. Por lo que realizar operaciones con estos mecanismos dentro del dispositivo hoy en día es una buena práctica para garantizar la seguridad de las conexiones entre el dispositivo y un servidor o para proteger el contenido del dispositivo.

Con base a las pruebas realizadas se puede observar que la implementación realizada de los algoritmos de ECC para *Android* puede ser ampliamente mejorada, ya que como se pudo observar en el capítulo 5 los tiempos de ejecución de estos algoritmos sobrepasa considerablemente a los tiempos obtenidos por RSA, a pesar que actualmente se ha demostrado que las EC son un mecanismo más eficiente para criptografía asimétrica. Sin embargo su uso no esta totalmente descartado ya que los tiempos de ejecución son relativamente buenos si se toman en cuenta las restricciones del dispositivo móvil.

Al seguir el proceso de diseño de la PKI se pudo observar la importancia de la integración de diversos estándares en este diseño, de manera que la PKI resultante fuera compatible con otros diseños existentes y pueda interactuar con otras aplicaciones, ya

que de otra manera sería una infraestructura muy cerrada, lo cual en el ambiente de dispositivos móviles no es bueno debido a la gran variedad de fabricantes y aplicaciones existentes actualmente, por lo que mientras más estándares se soporten en el diseño, abra más posibilidad de que la aplicación resultante se utilizable en el ambiente móvil.

En este trabajo se creó una aplicación para *Android* en la cual se utilizó la API generada, comprobando de esta manera la facilidad de uso de la misma, así como su versatilidad para ayudar en la creación de aplicaciones que requieran de servicios de seguridad. Esta aplicación fue creada utilizando las técnicas, componentes y buenas prácticas más actuales para el desarrollo en esta plataforma, lo que ayudó a que la aplicación resultante fuera compatible con un gran número de versiones de *Android* y diversos dispositivos sin la necesidad de realizar ajustes para cada tipo de pantalla o idioma. Además de esto se pudo observar que al seguir las guías de desarrollo se pueden obtener aplicaciones con características muy innovadoras y atractivas para el usuario, mediante la incorporación de diferentes gestos para la interacción del usuario con la interfaz y el uso de diversos mecanismos de navegación dentro de la aplicación. Durante la implementación de esta aplicación se pudo comprobar como ha ido evolucionando *Android* agregando diversos componentes a su sistema como lo son la *Action Bar*, los paginadores y las pestañas para ayudar a los desarrolladores a crear mejores aplicaciones, siempre pensando en el usuario final.

Finalmente uno de los aspectos más importantes a resaltar en este trabajo es que en comparación con las propuestas existentes analizadas en la sección 2.3, la PKI diseñada cubre todos los servicios de seguridad que son deseables, al igual que *OpenPGP Manager* y *APG* sin embargo las demás aplicaciones no ofrecen todos los estos servicios, esto debido a que no hacen uso de estándares como X.509 o PGP en su implementación. Adicionalmente la PKI y la aplicación construida tienen un alto grado de compatibilidad gracias al uso del estándar X.509 para la generación y validación de certificados digitales. En comparación con las demás aplicaciones, la API y la aplicación de prueba generadas tienen una gama más amplia de algoritmos disponibles, ya que se cuentan con algoritmos tanto de criptografía asimétrica como simétrica al contrario de algunas aplicaciones como *DroidCrypt* y *mobilEncrypt* que solo utilizan criptografía simétrica, adicionalmente la aplicación construida es la única que tiene ECC como algoritmo de criptografía asimétrica disponible entre sus opciones, por lo que la API es una opción atractiva para ser utilizada en la construcción de otras aplicaciones. Así mismo la PKI tiene un alto grado de reusabilidad en comparación con las demás aplicaciones, las cuales tienen un grado muy bajo o nulo en algunos casos, esto debido a que como resultado del diseño de la PKI se construyo la API, la cual será distribuida de forma gratuita en la red. Por último los elementos que conforman a una PKI son considerados a lo largo del diseño de la API a pesar que algunos de estos no se encuentran claramente definidos en la aplicación de prueba, como lo es el caso del repositorio ya que por naturaleza una red de confianza no cuenta con este elemento, sin embargo el diseño de la PKI lo considera en forma de una base de datos centralizada, con respecto a las aplicaciones analizadas *DroidCrypt* y *mobilEncrypt* no cuentan con ninguno de los elementos definidos, mientras que aplicaciones como *PKI Webtop*, *OpenPGP Manager* y *APG* consideran algunos de los elementos sin llegar a definir claramente cada uno de ellos dentro de su aplicación.

6.2. Trabajo a futuro

A pesar que los resultados obtenidos fueron bastante satisfactorios, siempre hay detalles que se pueden mejorar o aspectos que se pueden agregar para mejorar el trabajo realizado. Algunos de ellos se listan a continuación:

- Realizar la implementación del modelo jerárquico del diseño presentado, para lo que se requiere de la creación de un servidor centralizado en el que sean almacenados todos los elementos de la PKI. Es deseable que se implementen mecanismos de acceso a la información contenida en el servidor, para lo cual se recomienda le uso de servicios web.
- Implementación del protocolo OSCP para la consulta en linea del estatus de los certificados emitidos dentro de la infraestructura.
- Incorporar estrategias como reto-respuesta para la autenticación de usuarios en el sistema utilizando los certificados y las claves creadas.
- Entrar a detalle en la implementación de la biblioteca de seguridad utilizada para los bloques básicos, principalmente a la implementación de los algoritmos de ECC y modificarlos de tal manera que sigan los algoritmos más eficientes. De esta manera se obtendrían mejores tiempos de ejecución.
- Probar el rendimiento de los algoritmos de curvas elípticas utilizando campos binarios, en este trabajo no se realizó debido a una falla en la biblioteca en las funciones utilizadas para importar las claves desde un archivo.
- Agregar compatibilidad con PGP, de esta manera los archivos y mensajes cifrados en la aplicaciones podrían ser utilizados en otras aplicaciones que soporten este protocolo. Al inicio de este trabajo se comprobó que el soporte ofrecido por la biblioteca para este protocolo es muy básico y tiene algunos problemas, por lo que se optó por no contemplarlo en el desarrollo.
- Creación de estructuras base para el intercambio de mensajes dentro de la PKI, de manera que estas estructuras puedan ser utilizadas durante la comunicación entre el dispositivo y un servidor, en estas estructuras se deberán contemplar campos como: ubicación de emisor, firma de mensajes, contenido, dispositivo del emisor, certificado del emisor entre otros.
- Agregar una sección de configuración general a la aplicación, en la cual se puedan establecer valores predeterminados como: algoritmos, nivel de seguridad , un certificado y una clave privada general; preferencias para personalizar la interfaz como: tema a utilizar y forma de desplegar el los módulos principales(pestañas o menú en forma de lista).
- Mejorar los mecanismos para compartir las listas de confianza de manera que el nivel de confianza sea confidencial y garantizar que solo el propietario de una lista pueda modificar su contenido.

Bibliografía

[Adams and Just, 2004] Carlisle Adams y Mike Just, "PKI: Ten Years Later" In *3rd Annual PKI R&D Workshop*, Gaithersburg MD, USA. , April 2004

[AndroidDeveloper, 2012] Android Developers. (2012, October 10). *Android Developers* [Online]. Available: http://developer.android.com/

[Barker et al., 2012] Elaine Barker, William Barker, William Burr, William Polk, and Miles Smid *Recommendation for Key Management – Part 1: General (Revision 3)*, NIST Special Publication 800-57, July 2012

[Barreira and Compans, 2011] Inigo Barreira and Sonia Compans , *ETSI TS 102 042 - Electronic Signatures and Infrastructures (ESI); Policy requirements for certification authorities issuing public key certificates*, ETSI Technical Specification, France, December 2011

[BE, 2011] Banco de España, *Infraestructura de Clave Pública del Banco de España - Declaración de Prácticas de Certificación*, Spain, 2011

[BFar, 2005] Reza B'Far, *Mobile computing principles*, Cambridge University Press, 2005

[BouncyCastleJava, 2012] The Legion of the Bouncy Castle. (2012). *Bouncy Castle Java cryptography APIs* [Online]. Available: http://www.bouncycastle.org/

[Callas, 2008] Jon Callas, *An Introduction to Cryptography* , PGP Corporation, 2008

[Callas et al., 2007] J. Callas, L. Donnerhacke, H. Finney, D. Shaw, R. Thayer, *OpenPGP Message Format*, RFC 4880, November 2007

[Caronni, 2000] Germano Caronni, *Walking the Web of Trust* In 9th Workshop on Enabling Technologies (WET ICE'2000), IEEE Computer Society Press, Gaithersburg, MD, USA., June 2000

[Cook, 2006] Cook, C.R., *Los cuatro pasos indispensables para administrar*, Panorama editorial, México, 2006

[Cooper et al., 2005] M. Cooper, Y. Dzambasow, P. Hesse, S. Joseph and R. Nicholas *Internet X.509 Public Key Infrastructure: Certification Path Building*, RFC 4158, September 2005

[Cooper et al., 2008] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley and W. Polk *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, RFC 5280, May 2008

[Crook et al., 2011] Stacy K. Crook, Stephen D. Drake, William Stofega and Ramon T. Llamas *Worldwide Smartphone 2011–2015 Forecast and Analysis*, IDC Analyze the Future, FRAMINGHAM, Mass. March 29 2011

[Diffie and Hellman, 1976] W. Diffie and M. Hellman, *New Directions in Cryptography*, IEEE Transactions on Information Theory, Vol. 22, No. 6, pp. 644, November 1976.

[Eastlake and Hansen, 2011] D. Eastlake 3rd, T. Hansen, *US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)*, RFC , May 2011

[Echoworx, 2011] Echoworx Corporation. (2011). *mobilEncrypt Cloud* [Online]. Available: http://www.echoworx.com/mobile/

[Enck et al., 2009] William Enck, Machigar Ongtang and Patrick McDaniel, *Understanding Android Security*, *IEEE Security & Privacy Magazine*, Vol. 7, No. 1, pp. 50–57, January/February 2009.

[Feilner and Graf, 2009] Markus Feilner and Norbert Graf, *Beginning OpenVPN 2.0.9*, Packt Publishing, 2009

[Garcia, 2010] A. García, *Android*, *Linux Magazine*, No. 49, pp. 50-53, February 2010.

[Giessmann and Compans, 2011] Ernst Giessmann and Sonia Compans, *ETSI TS 102 176-1 - Electronic Signatures and Infrastructures (ESI); Algorithms and Parameters for Secure Electronic Signatures; Part 1: Hash functions and asymmetric algorithms*, ETSI Technical Specification, France, July 2011

[Guo et al., 2011] Guibing Guo, Jie Zhang and J. Vassileva, *Improving PGP Web of Trust through the Expansion of Trusted Neighborhood* In 2011 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT), pp. 489 - 494, Lyon, France, August 2011

[Gutmann, 2007] Peter Gutmann, *PKI design for the real world* In Proceedings of the 2006 workshop on New security paradigms, pp. 109-116, Schloss Dagstuhl, Germany, September 2007

[Kohnfelder, 1978] L. Kohnfelder, *Towards a Practical Public-key Cryptosystem*, Dept. of Electrical Engineering and Computer Science, MIT, May 1978.

[Kuhn et al., 2001] D. Richard Kuhn ,Vincent C. Hu, W. Timothy Polk and Shu-Jen Chang, *Introduction to Public Key Technology and the Federal PKI Infrastructure*, NIST Special Publication 800-32, February 2001

[Leontiev and Shefanovski, 2006] S. Leontiev and D. Shefanovski, *Using the GOST R 34.10-94, GOST R 34.10-2001, and GOST R 34.11-94 Algorithms with the Inter-*

net X.509 Public Key Infrastructure Certificate and CRL Profile, RFC 4491, May 2006

[Liping and Lei, 2011] Hou Liping and Shi Lei, *Research on Trust Model of PKI* In 2011 Fourth International Conference on Intelligent Computation Technology and Automation, pp. 232, Guangdong, China, March 2011

[Lucas, 2006] Michael Lucas, *PGP & GPG: email for the practical paranoid*, No Starch Press, April 2006

[Mawloud et al., 2012] Omar Mawloud, Challal Yacine and Bouabdallah Abdelmadjid, *Certification-based trust models in mobile ad hoc networks: A survey and taxonomy*, *Journal of Network and Computer Applications*, Vol. 35, No. 1, pp. 268 - 286, 2012.

[Messerman and Mustafic 2011] Arik Messerman and Tarik Mustafic. (2011). *Droid Crypt* [Online]. Available: https://market.android.com/details?id=de.atm.android.security.encryption.full

[Microsoft, 2004] Microsoft Corporation, *Windows Server 2003 Deployment Kit: Designing and Deploying Directory and Security Services*, Microsoft Corporation, pp. 217 - 330, 2004

[Perlman, 1999] R. Perlman, An overview of PKI trust models, Network, IEEE, Vol. 13, No. 6, pp. 38 -43, November 1999

[Polk et al., 2002] W. Polk, R. Housley and L. Bassham *Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, RFC 3279, April 2002

[Prohic, 2005] Natasa Prohic, *Public Key Infrastructures – PGP vs. X.509* In INFOTECH Seminar Advanced Communication Services (ACS), Stuttgart, Germany , April 2005

[Rodriguez, 2011] R. Rodriguez Martinez. (2011). *OpenPGP Manager* [Online]. Available: https://market.android.com/details?id=com.harpago.pgpmanager

[RSA, 1999] RSA Laboratories, *PKCS 12 v1.0:Personal Information Exchange Syntax*, RSA Laboratories , January 1999

[SANDBOX, 2010] SAFELAYER SANDBOX. (2010). *PKI Webtop for Android* [Online]. Available: http://sandbox.safelayer.com/en/experimental-applications/1-semantic-web-trust-portal/485-pki-webtop-for-android

[Schaad et al., 2005] J. Schaad, B. Kaliski, R. Housley, *Additional Algorithms and Identifiers for RSA Cryptography for use in the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, RFC 4055, June 2005

[Schmeh, 2003] K. Schmeh, *Cryptography and Public Key Infraestruture on the Internet*, John Viley & Sons, 2003.

[Schneier, 1996] Bruce Schneier, *Applied Cryptography, Second Edition: Protocols, Algorthms, and Source Code in C (cloth)* John Wiley & Sons, Inc., 1996

[Singh, 2001] Simon Singh, *The Code Book: How To Make It, Break It, Hack It, Crack It,* Delacorte Press 2001

[Thialfihar, 2011] Thialfihar. (2011). *Android Privacy Guard (APG)* [Online]. Available: https://market.android.com/details?id=org.thialfihar.android.apg

[Tudela, 2009] J.A. Tudela, *Desarrollo de aplicaciones para dispositivos móviles sobre la plataforma de desarrollo Android de Google*, Universidad Carlos III de Madrid Escuela Politécnica Superior, Spain, 2009.

[Tyley, 2012] Roberto Tyley. (May 17, 2012). *Spongy Castle* [Online]. Available: https://github.com/rtyley/spongycastle

Apéndice A

Manual de Uso

En este apéndice se mostraran las principales interfaces de usuario de la aplicación implementada, para cada interfaz se explicaran los detalles más sobresalientes de la misma y se dará una breve reseña de lo que se puede hacer en ella.

A.1. Características de diseño

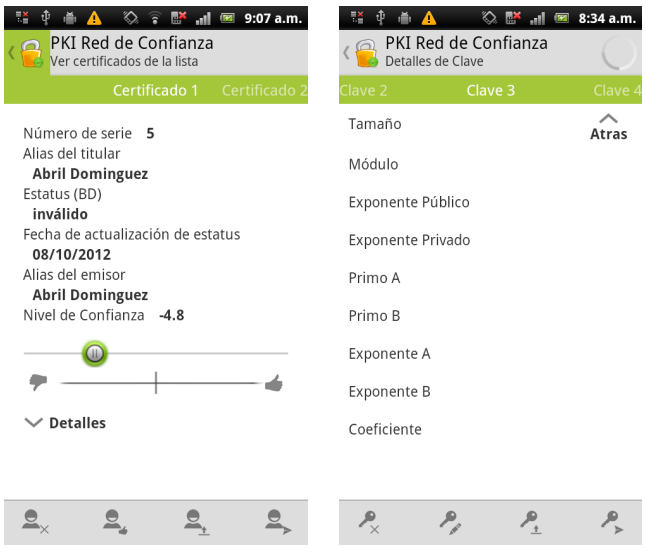
Antes de iniciar con la descripción de las interfaces, se presentarán algunos detalles en el diseño de las mismas, los cuales hacen que la aplicación diseñada sea a pegue al modelo de diseño denominado *Pure Android* explicado detalladamente en [AndroidDeveloper, 2012].

Primero que nada en la figura A.1 se muestran los gestos que pueden ser utilizados en la aplicación para realizar las diferentes operaciones que ofrece.

Figura A.1: Gestos soportados en la aplicación.

El toque, es el gesto más utilizado de los tres y activa la función del elemento sobre el cual se realice esta acción. Por otra parte el toque largo, es utilizado como mecanismo de selección en la aplicación, para realizar este gesto basta con tocar la pantalla y dejar presionado hasta que la acción se realice. Por último el *swipe* es un gesto utilizado para desplazarse entre vistas de la misma jerarquía.

Adicionalmente la navegación de la aplicación se realizo siguiendo las recomendaciones hechas en la guía de *Android Design* [AndroidDeveloper, 2012], por lo que la navegación en la aplicación puede realizarse de dos maneras: interactuando con el botón *home* en la interfaz (figura A.2 (a)) o utilizando el botón de atrás del dispositivo.



(a) Botón *home* (b) Icono de progreso

Figura A.2: Botón *home* e icono de progreso.

Adicionalmente en la figura A.2 (b), se muestra el icono de progreso utilizado, el cual indica que una operación pesada se está llevando a cabo en el dispositivo, sin embargo la aplicación se diseño de tal manera que esta no se congele cuando dichas operaciones se estén ejecutando, al contrario, el usuario puede seguir interactuando de manera normal con la interfaz mientras la operación concluye.

Utilizando los elementos anteriores es posible navegar dentro de la aplicación como se ve en la figura A.3, en la cual se muestra la combinación de varias interacciones para navegar dentro de la aplicación, por ejemplo, estando en la interfaz principal de la aplicación, es posible ver los detalles de las claves tocando el icono de buscar, lo cual mostrará la interfaz correspondiente, donde utilizando el gesto *swipe* se puede navegar entre las diferentes claves de un mismo usuario, adicionalmente es posible ver más detalles de la clave tocando el botón correspondiente.

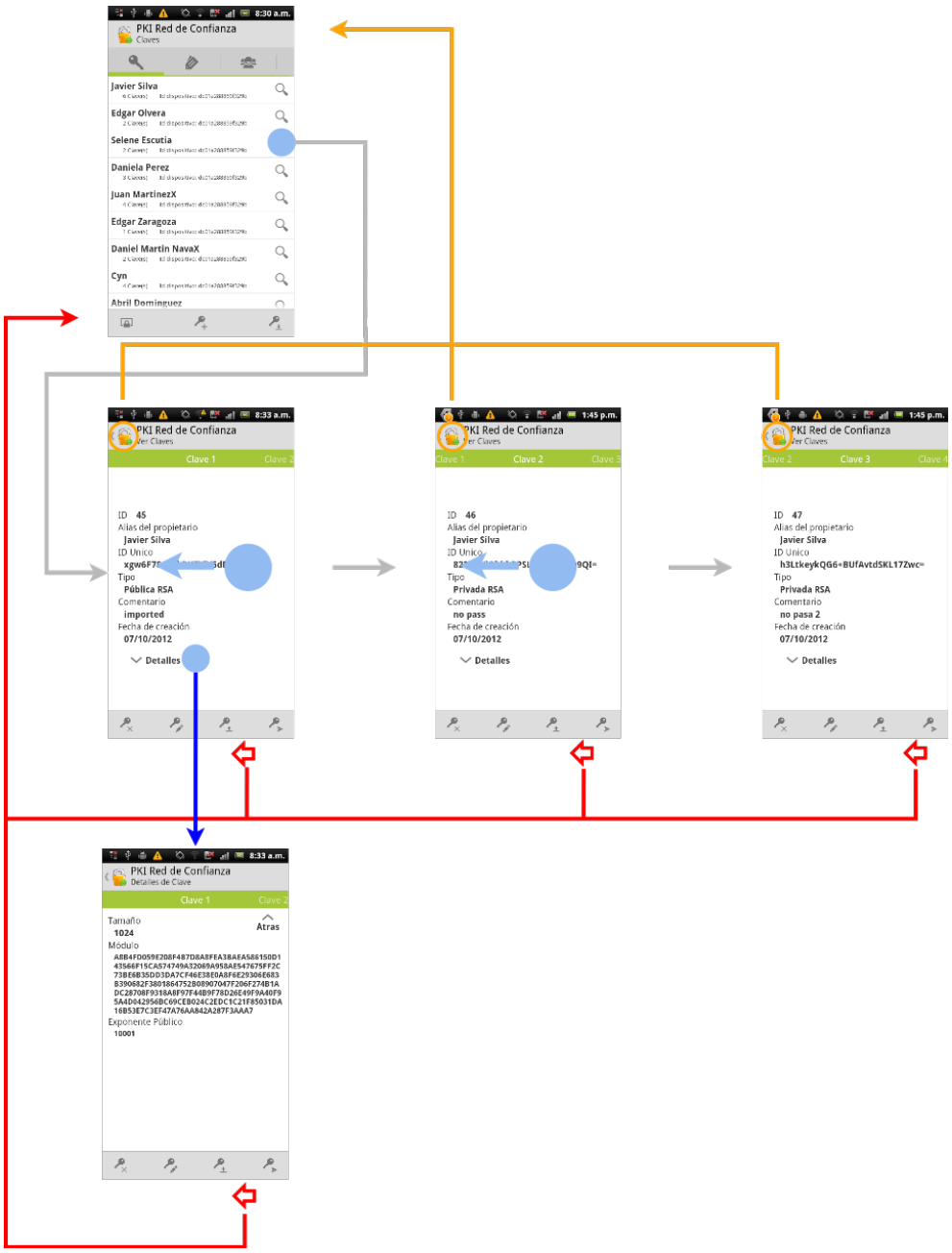


Figura A.3: Navegación en la aplicación - Tipo 1.

En la figura A.3 el comportamiento del botón *home* y el botón de atrás en el dispositivo no tienen una diferencia significativa de comportamiento, ya que ambos hacen que la aplicación regrese a la interfaz principal, sin embargo esto no es siempre así, como se puede ver en la figura A.4, en la que se muestra la navegación de la aplicación mientras se está creando una nueva clave, en este ejemplo se observa como el gesto de toque se puede utilizar tanto para seleccionar un elemento en la lista como para seleccionar una opción en el menú de la aplicación, adicionalmente se utiliza *swipe* para intercambiar entre la

vista de opciones básicas y avanzadas para la generación de la clave. Finalmente como se mencionó, en la figura A.4 se ve que el botón atrás y el botón *home* tienen un comportamiento diferente, ya que mientras el primero regresa a la vista anterior dentro de la tarea, el segundo regresa siempre a la interfaz principal de la aplicación.

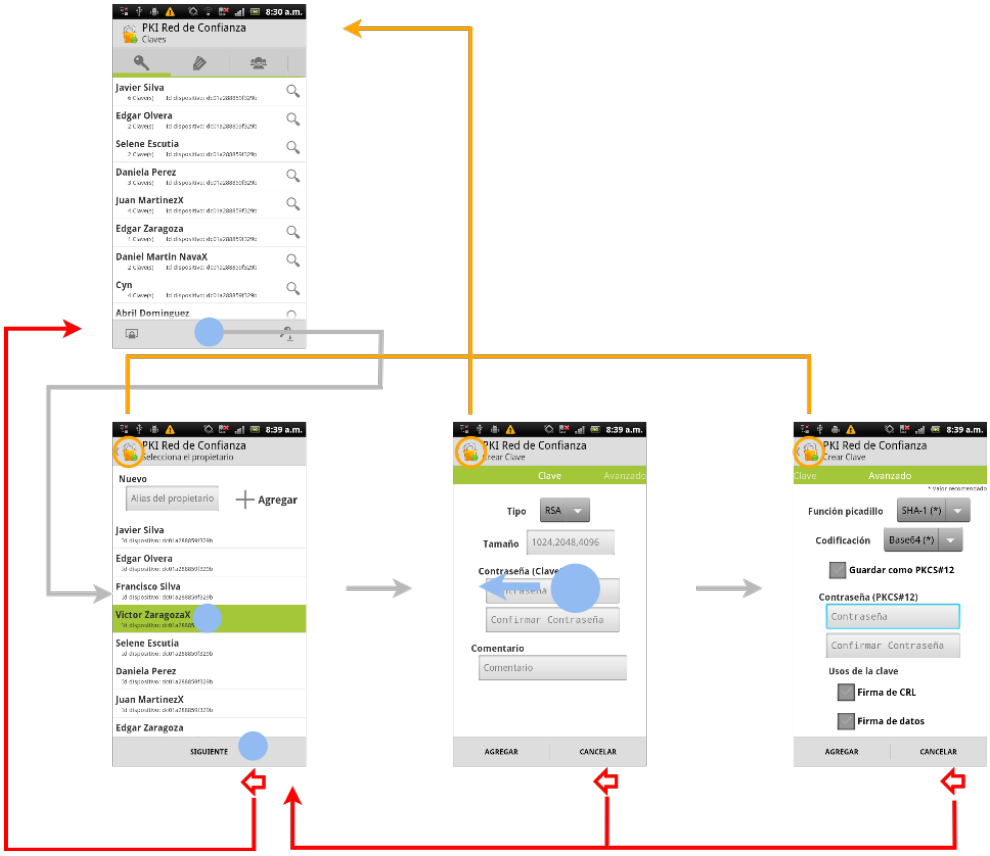


Figura A.4: Navegación en la aplicación - Tipo 2.

Otra características de diseño que se utilizó para la creación de la aplicación fue el uso de la *Action Bar*, que es la barra que se puede observar en parte superior de la interfaz, esta barra esta constituida por el icono, titulo y subtitulo de la aplicación. Una de las ventajas del uso de esta barra, es que ésta es persistente a lo largo de la aplicación, dándole una apariencia más estética a las interfaces. Adicionalmente la *Action Bar* ofrece la posibilidad de agregar botones de acción a la misma (conocidos como *Action Buttons*), estos botones forman parte de la barra y constituyen un menú con las opciones más comunes para la interfaz donde se encuentra. Esta barra contiene 3 elementos principales, los cuales se pueden ver la figura A.5, el primer elemento es la barra de titulo, en la cual pueden ser agregados los *action buttons*, el segundo elemento es la barra superior, que puede ser utilizada para mostrar pestañas dentro de la interfaz, por último está la barra inferior, la cual es utilizada para desplegar los botones de acción en caso de ser requerido.

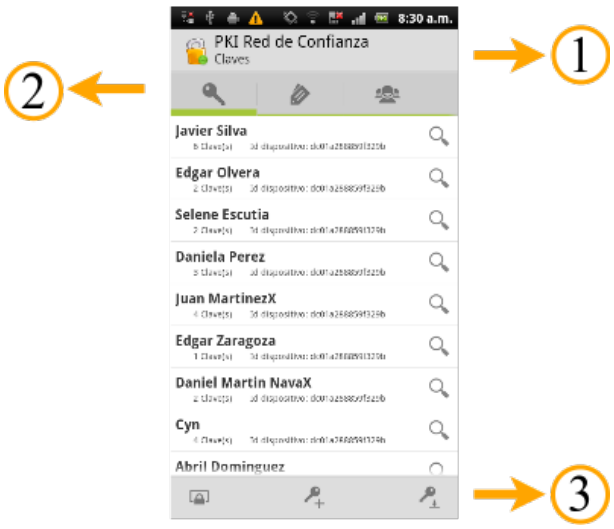


Figura A.5: Elementos *Action Bar*.

Es importante destacar que esta barra es soportada para dispositivos con *Android* 3.0+, sin embargo para el desarrollo de la aplicación se utilizó una biblioteca que permite su uso en dispositivos con versiones inferiores, por lo que es posible utilizarla desde la versión 2.1 de este sistema operativo. En la figura A.5, la interfaz principal de la aplicación cuenta con diferentes pestañas, de manera que el usuario puede navegar entre las principales secciones de la aplicación utilizando *swipe* de una pestaña a otra. Otra de las principales ventajas de utilizar esta barra, es la facilidad que ofrece al reorganizar la interfaz al momento de cambiar la orientación del dispositivo, esto se puede ilustrar mejor en la figura A.6.

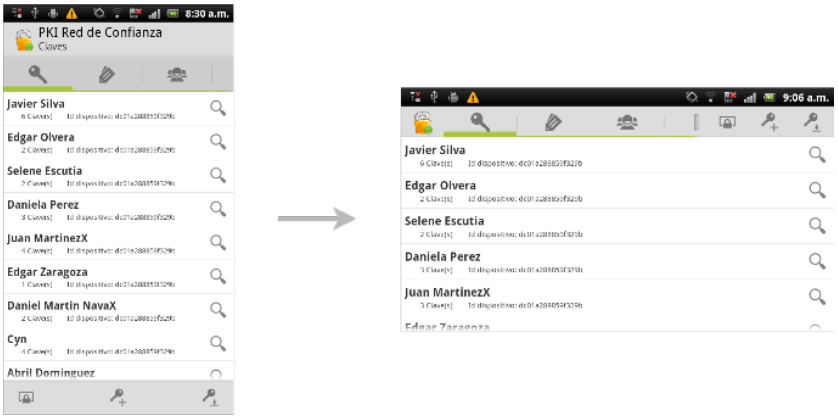


Figura A.6: Cambio de orientación utilizando la *Action Bar*.

Otra de las características de diseño más notables en la aplicación es el soporte para múltiples tamaños de pantalla, lo cual se logro siguiendo las guías de desarrollo de *Android* para la creación de los iconos de toda la aplicación, esta versatilidad se puede comprobar en la figura A.7, en ésta se muestran dos interfaces: la interfaz inicial de claves

y la interfaz de detalles de clave. Como se puede observar el cambio principal es el espacio disponible para mostrar los elementos de estas vistas en cada dispositivo, si que esto afecte la distribución general de la interfaz, por ejemplo, la lista de claves en el dispositivo 3 tiene menos elementos visibles que en los otros dos, mientras que en la pantalla de detalles, el espacio en blanco es mayor para los otros dispositivos.

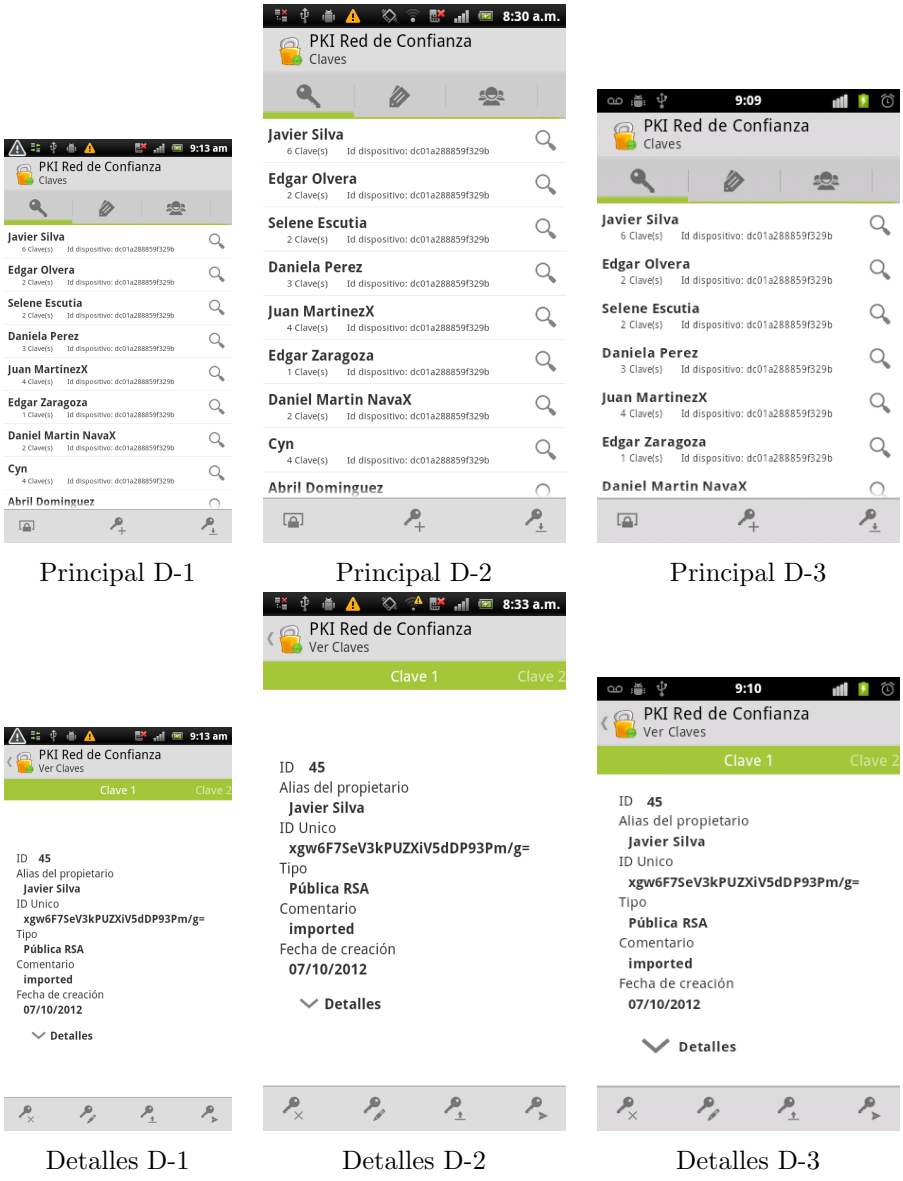


Figura A.7: Soporte de interfaz para múltiples pantallas - Claves.

Por último, la aplicación creada tiene soporte para varios lenguajes, actualmente únicamente español e inglés, pero es posible añadir tantos lenguajes se requieran de forma sencilla, ya que todos los mensajes de la interfaz se encuentran en la carpeta `values` del proyecto principal, por lo que basta con añadir una carpeta nueva con el nombre `values-` y el identificador del idioma soportado, para el caso de español se creó la carpeta

`values-es`, si se quiere agregar francés la carpeta debe llevar por nombre `values-fr` y así sucesivamente. En la figura A.8 se muestran algunas interfaces en español y su equivalente en inglés.

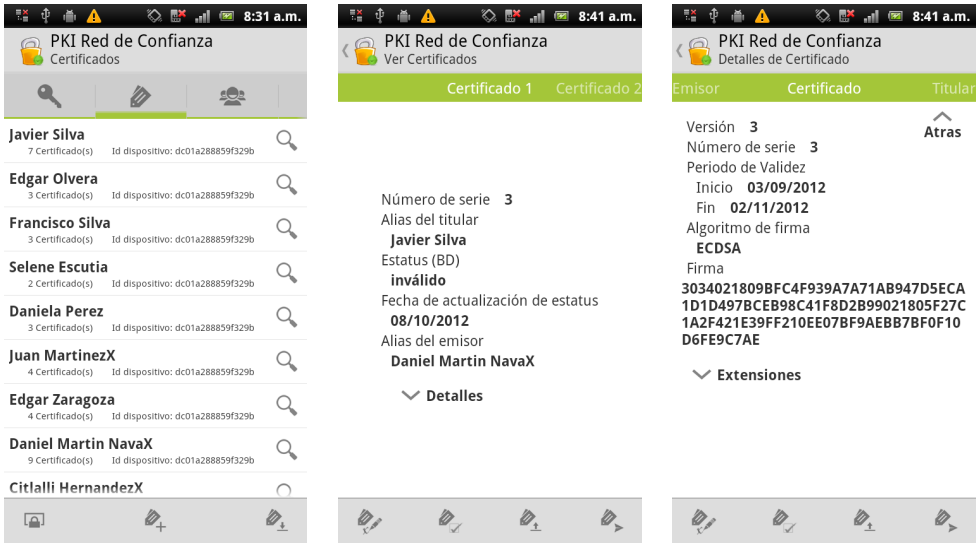


Figura A.8: Soporte de para múltiples idiomas.

A.2. Interfaces de la aplicación

Una vez descritos los características de diseño para las interfaces de la aplicación, en esta sección se explicara el funcionamiento general de la misma, describiendo las interfaces que se crearon para cubrir las características señalas en la sección 5.2.1.

Para iniciar, la interfaz principal de la aplicación está conformada por cuatro pestañas: Claves, Certificados, Red de Confianza y CRL. En la figura A.9 se pueden ver estas interfaces, para cambiar de una pestaña a otra es posible hacer *swipe* entre ellas o se puede seleccionar la pestaña deseada tocando el icono correspondiente.

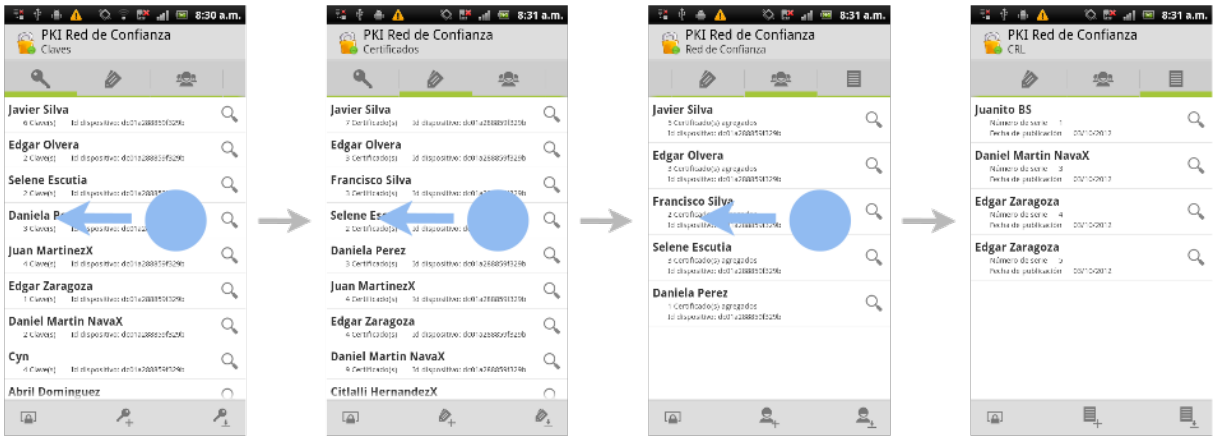


Figura A.9: Interfaz principal.

Otro detalle a resaltar en la figura A.9 son los botones de acción, ubicados en la barra inferior de la interfaz, estos botones son utilizados para iniciar una tarea dependiendo de la pestaña donde se encuentre, de manera general el primer icono abrirá la interfaz para realizar operaciones criptográficas con mensajes o archivos, como ésta es una de las acciones principales de la aplicación este botón está presente en todas las pestañas, por el contrario, los demás botones cambian dependiendo de la pestaña donde el usuario se encuentre. Dichos botones sirven para agregar e importar respectivamente el elemento correspondiente. En cada pestaña se muestran diferentes listas, sin embargo éstas tienen algunos elementos en común, por ejemplo, cada fila incluye un botón para ver los detalles del elemento seleccionado, además las entradas de la lista están agrupadas por el alias del usuario que tiene asociados los elementos y cada usuario muestra el identificado único del dispositivo donde fue creado. De manera particular en la pestaña de claves se muestra el listado de los usuarios que tienen asociada al menos una clave, mientras que en la pestaña de certificados se listarán los usuarios que tengan al menos un certificado en la base de datos. Por otra parte la lista en la pestaña de red de confianza muestra las listas de confianza registradas en el dispositivo, agrupadas mediante el alias del usuario que creo dicha lista, en cada entrada de ésta se muestra el número de certificados agregados a la lista y los elementos comunes con las otras listas. Sin embargo el caso de la pestaña de CRL es un poco diferente, ya que esta lista muestra todas las listas de revocación almacenadas en el dispositivo y cada entrada muestra el alias del emisor de la lista, el número de serie y la fecha de publicación.

Ahora se explicará el modulo de claves, iniciando la interfaz encargada de mostrar los detalles de una clave, para acceder a está interfaz es necesario tocar la pequeña lupa de cualquier entrada en la pestaña de claves. La figura A.10 muestra las vistas de detalles de claves con su respectiva navegabilidad. La interfaz principal esta conformada

por un `ViewPager`, el cual permite desplazarse entre elementos similares en un adaptador utilizando un *swipe*, en este caso se podrá desplazar entre las diferentes claves que tenga asociadas el usuario seleccionado, mostrando los datos básicos de cada clave, si el se desean ver más detalles acerca de la clave, basta con tocar el botón de detalles y la aplicación cambiara la vista según sea el tipo de clave mostrando a detalle la información de la clave seleccionada. A su vez, mientras la aplicación esta en la vista de información detallada, es posible navegar a la lista de claves utilizando el botón correspondiente o simplemente haciendo *swipe* a la siguiente clave de la lista.

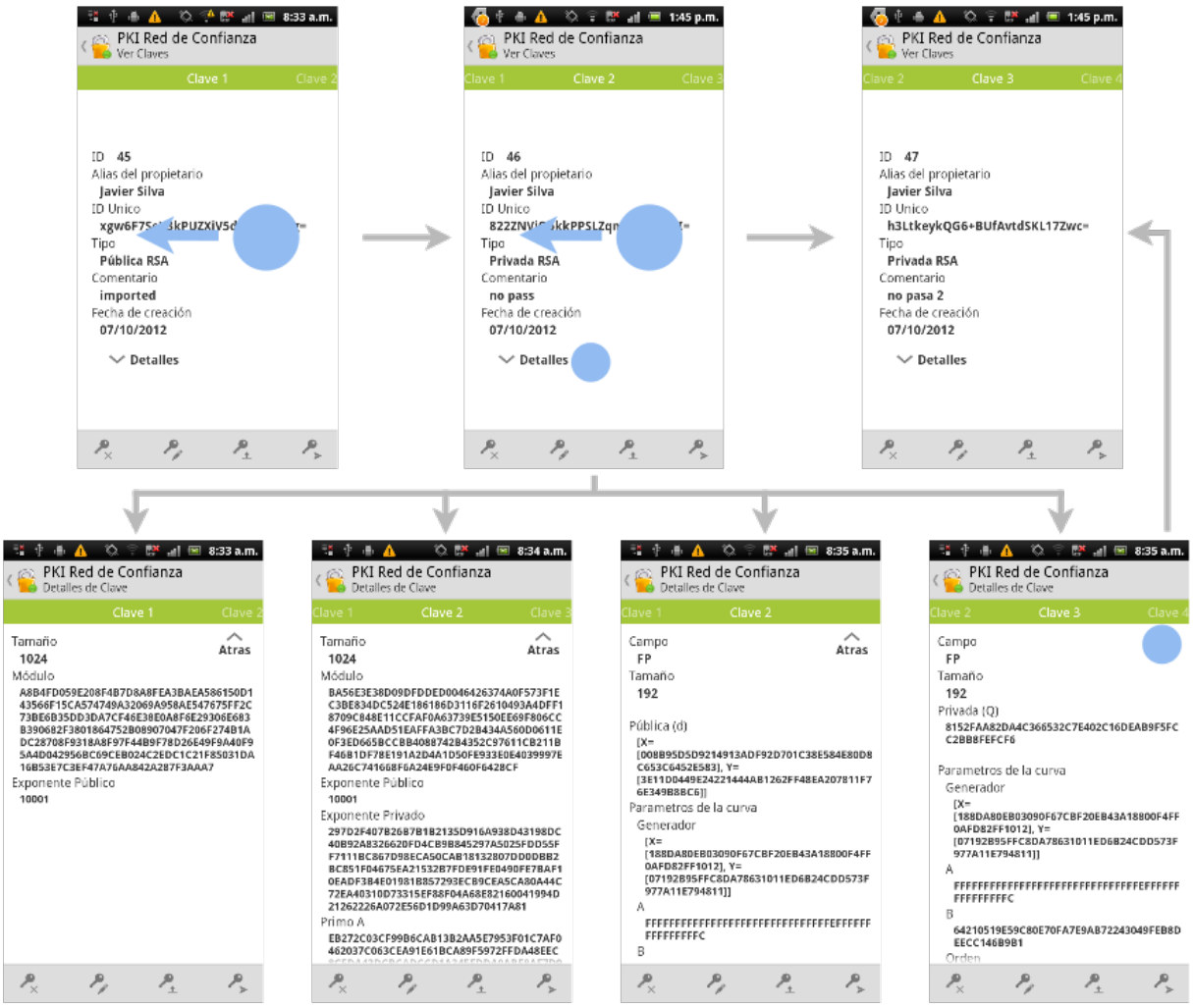
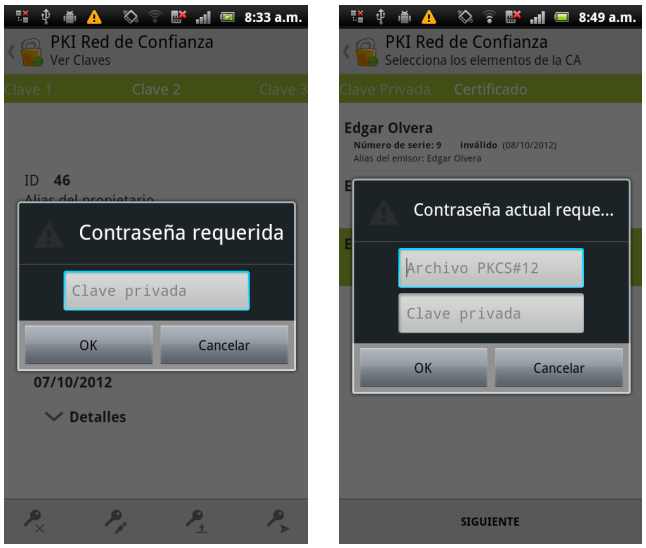


Figura A.10: Interfaz - Detalles de clave.

Es importante mencionar que para el caso de claves privadas, tanto en PKCS#8 como PKCS#12, la aplicación mostrará un cuadro de dialogo según sea el caso para solicitar la contraseña de la clave. En la figura A.11 se muestran los cuadros de dialogo mostrados para el caso de una clave privada (a) y una clave en formato PKCS#12 (b).

Como se puede ver en la barra inferior de la interfaz de detalles mostrada en la figura A.10, esta interfaz cuenta con 4 opciones, las cuales se conservan sin importar en



(a) (b)

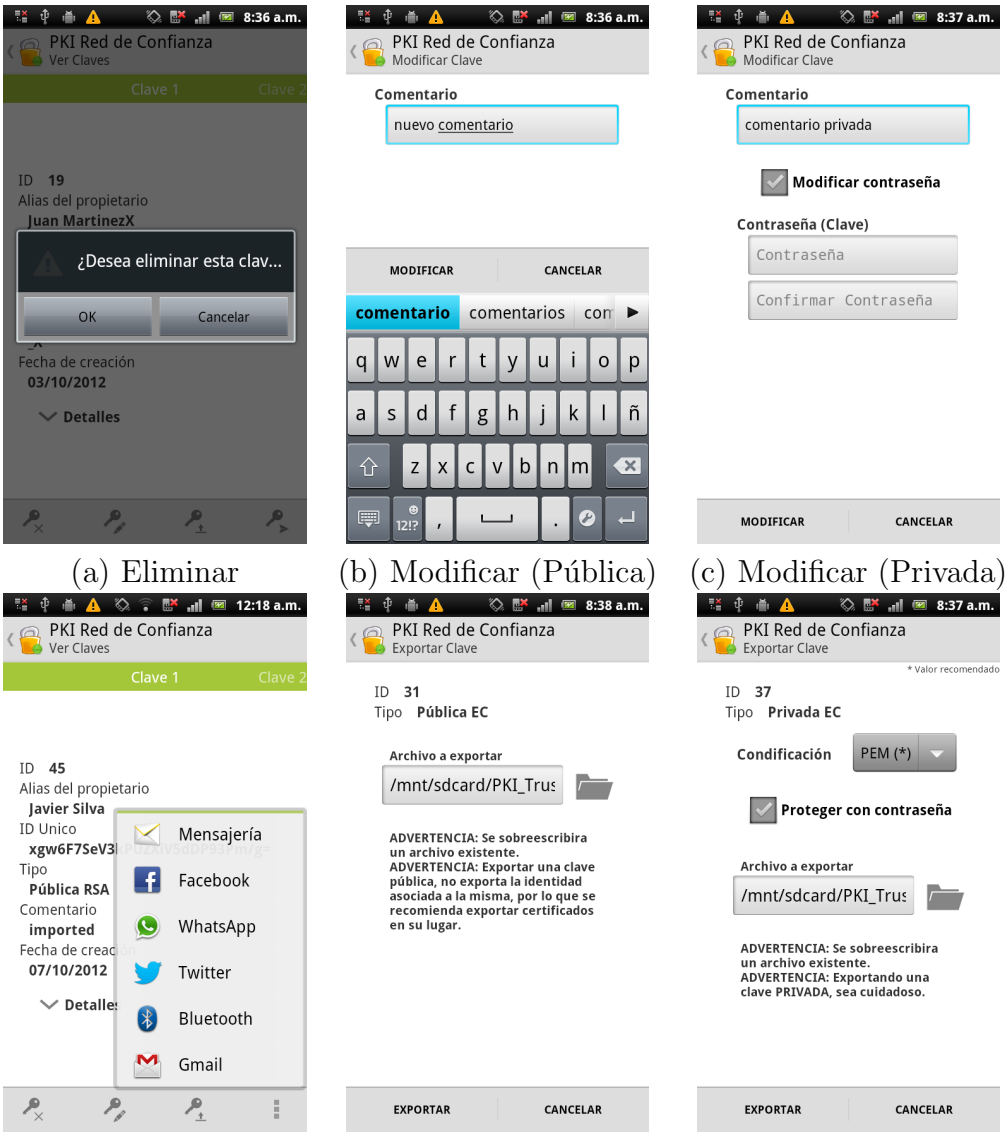
Figura A.11: Cuadro de dialogo para contraseñas.

que nivel de detalle se encuentre la vista, estas opciones son: eliminar, modificar, exportar y compartir respectivamente la clave que se encuentra actualmente en pantalla. Las interfaces correspondientes a estas operaciones se muestran en la figura A.12.

El proceso de creación de una clave, se mostró en la figura A.4, este proceso consiste en seleccionar al propietario de la clave para después abrir una interfaz donde se permite llenar los datos correspondientes según el tipo de clave que se desea crear, estos datos incluyen: tipo de clave, tamaño de clave, comentario y contraseña para proteger la clave privada, como se menciono anteriormente la interfaz de creación tiene dos secciones una para las opciones básicas y otra para opciones avanzadas, por lo que esta interfaz fue creada agregando un `ViewPager` con únicamente dos páginas (una para cada sección), por lo que cambiar entre una y otra se hace con un *swipe*. Dentro de las opciones avanzadas se incluye: selección de picadillo y codificación para la creación del identificado único de la clave y guardar el par de claves en formato PKCS#12.

Finalmente es posible importar claves desde un archivo, para hacer esto es necesario seleccionar el usuario al cual se le asignara esta clave, después la aplicación permite seleccionar el tipo de clave, elegir la clave desde un archivo en el dispositivo (utilizando una aplicación externa para buscar éste) e introducir un comentario para la clave a ser importada. En la figura A.13 se puede ver la interfaz creada para la importación de claves (a) y los posibles tipos de clave que pueden ser importados a la aplicación (b).

Una vez explicado el modulo de claves, se pasará al modulo de certificados, iniciando con la interfaz para crear un nuevo certificado, para acceder a esta interfaz es necesario seleccionar la opción correspondiente en la pestaña de certificados y seleccionar las claves que se utilizarán para crear este certificado, la aplicación contempla el uso de claves existen o de nuevas claves, las cuales serán creadas en el momento. La interfaz de creación de



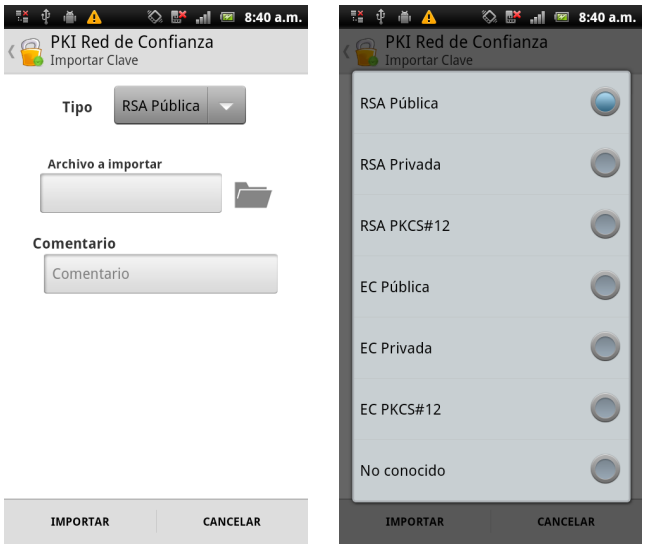
(a) Eliminar (b) Modificar (Pública) (c) Modificar (Privada)

(d) Compartir (e) Exportar (Pública) (f) Privada (Privada)

Figura A.12: Operaciones adicionales - Claves.

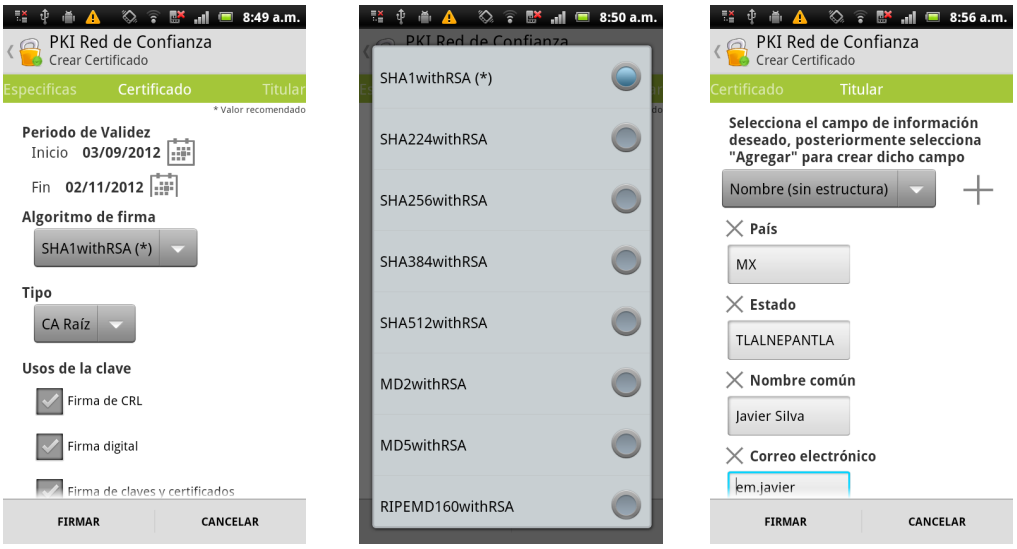
certificados se muestra en la figura A.14 y como se puede ver es consistente con el estilo de las demás interfaces de la aplicación ya que utiliza un paginador para organizar las diferentes secciones de información de un certificado, la página principal (figura A.14 - a) muestra la información referente al certificado como lo son, periodo de validez, uso de la clave, tipo de certificado y el algoritmo de firma (figura A.14 - b); la siguiente página permite agregar información del titular utilizando los atributos de nombre X.500 descritos en el estándar (figura A.14 - c), por último la interfaz cuenta con otra página para introducir los datos correspondientes a las extensiones propietarias al certificado.

Adicionalmente la aplicación tiene la facilidad de importar certificados a partir de archivos existentes en el dispositivo móvil, siempre y cuando estos utilicen codificación `DER`



(a) (b)

Figura A.13: Interfaz para importar claves.

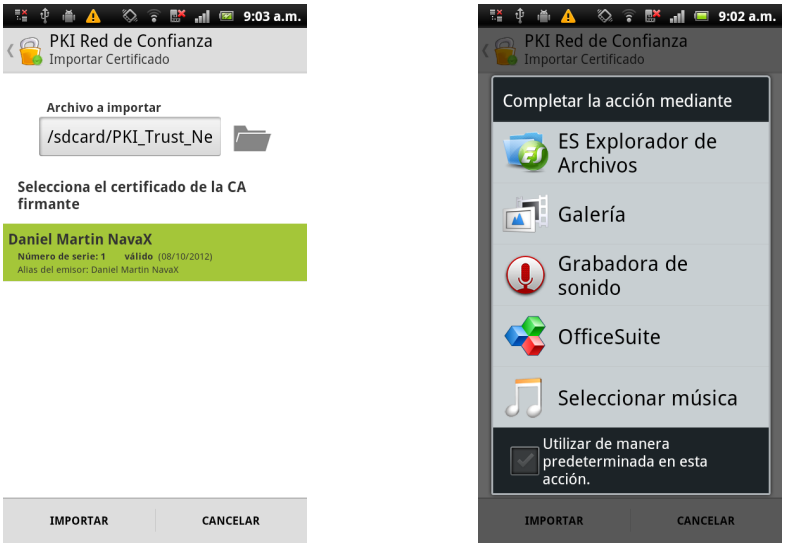


Información Certificado (a) Algoritmos firma - RSA (b) Información del titular (c)

Figura A.14: Interfaz para crear certificados.

o `PEM`, para seleccionar el archivo que se va a importar la aplicación hace uso de las aplicaciones externas instaladas en el dispositivo para poder explorar el árbol de directorios y seleccionar el archivo deseado, una vez seleccionado el certificado a importar, la aplicación intentara buscar automáticamente el certificado de la CA que emitió este certificado en la base de datos y el resultado de esta búsqueda se muestra en forma de lista para que el usuario seleccione el certificado correcto, en la figura A.15 muestra la interfaz de importación con el archivo seleccionado, en ésta se puede ver el listado de posibles certificados

de la CA emisora, adicionalmente se puede ver la lista de las aplicaciones externas que pueden ser utilizadas para seleccionar un archivo dentro del dispositivo, aunque esta lista puede variar de dispositivo en dispositivo dependiendo de las aplicaciones que se tengan instaladas.



Lista posibles CA Seleccionar archivo - Aplicaciones externas

Figura A.15: Interfaz para importar certificados.

Otra de las vistas principales del modulo de certificados, es la interfaz utilizada para ver los detalles de cada certificado (figura A.16) y a partir de ésta es posible navegar en los detalles del certificado, los cuales están divididos en 4 niveles, cada uno de estos contiene un paginador (`ViewPager`) diferente con las páginas necesarias para mostrar la información correspondiente. El primer nivel muestra los detalles generales del certificado, éstos son los datos almacenados en la base de datos del dispositivo, el paginador de este nivel permite navegar entre los todos certificados que estén asociados a un mismo alias, al tocar el botón "Detalles" en cualquier página de este nivel la interfaz pasara al segundo nivel de detalle, en el cual el certificado en sí es decodificado su información se organiza en cuatro páginas: Certificado, Emisor, Titular y Clave Pública, entre las cuales es posible navegar utilizando el gesto *swipe*. El tercer nivel de detalle corresponde a las extensiones almacenadas en un certificado, para acceder a este nivel es necesario tocar el botón "Extensiones" en cualquiera de las páginas del segundo nivel, las extensiones del certificado son mostradas en dos páginas, una para extensiones estándar X.509 y la otra para extensiones propietarias creadas especialmente para esta PKI, al igual que en los niveles anteriores es posible utilizar *swipe* para moverse de una página a otra. Finalmente el cuarto nivel corresponde a un mapa que muestra la posición GPS de donde fue creado el certificado.

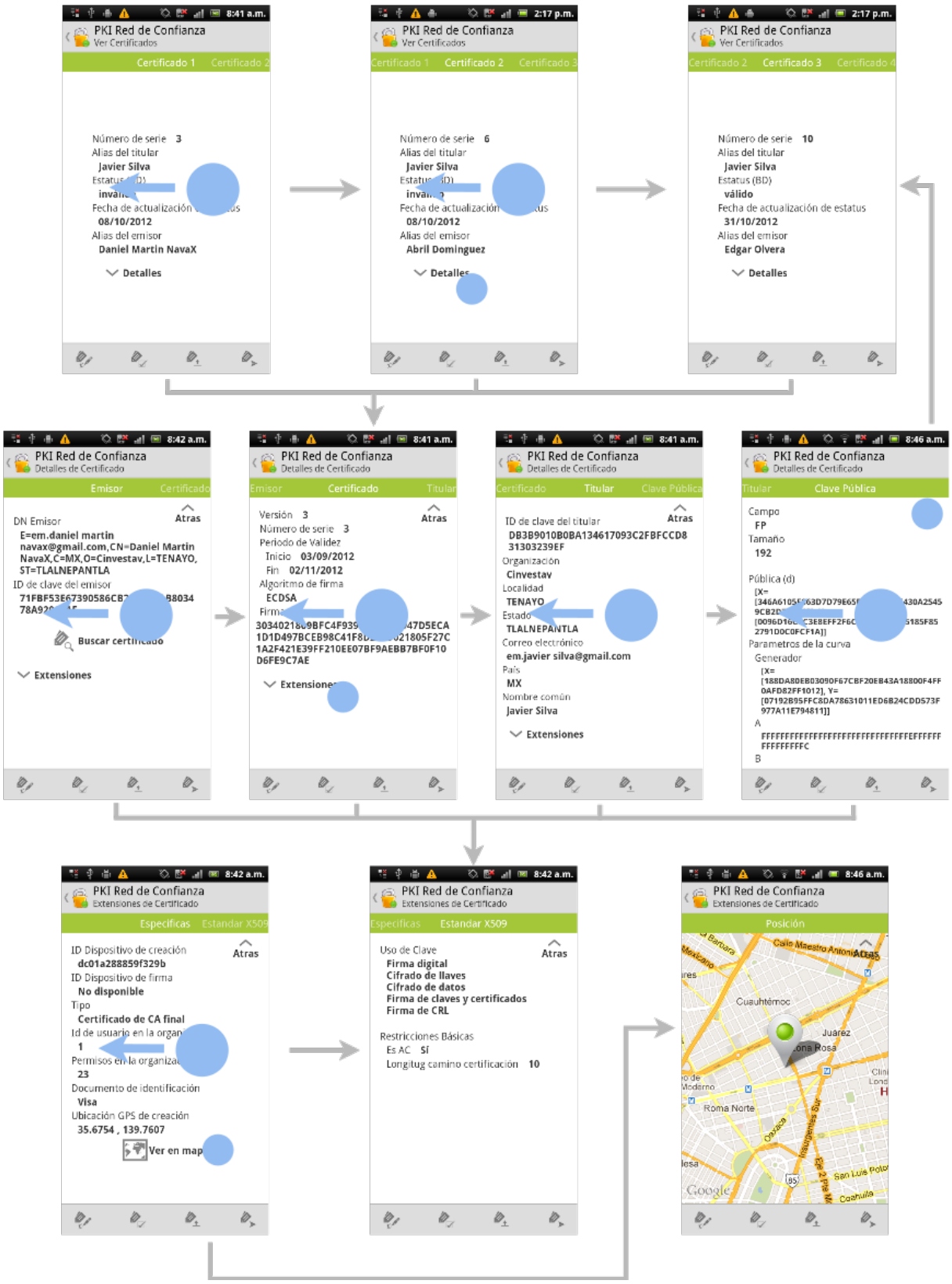


Figura A.16: Interfaz - Detalles de certificado.

Como se puede ver en la figura A.16, todos los niveles de detalle comparten la misma barra inferior, en la cual se muestran las opciones para: firmar, verificar, exportar y compartir el certificado en cuestión. Las interfaces para exportar y compartir un certificado son muy similares a las utilizadas en el modulo de claves, por lo que no se mostrarán en este trabajo para optimizar el espacio disponible. Por otra parte el proceso de firma de un certificado se muestra en la figura A.17, como se puede ver al inicio del proceso se señalan los riesgos que existen en la red de confianza al firmar un certificado para el que no se tenga la certeza de su validez, posteriormente es necesario seleccionar el alias de la entidad que firmará el certificado, después la aplicación abre la interfaz para seleccionar la clave privada y el certificado que se utilizarán para firmar el certificado, es importante señalar que la aplicación comprueba que los elementos seleccionados sean correctos, es decir, que la clave privada corresponda a la clave pública almacenada en el certificado, para lo que se solicita la contraseña de la clave privada, la cual también será utilizada para poder realizar la firma. Una vez seleccionados estos elementos, se abre la interfaz de creación de un nuevo certificado con los datos del mismo precargados, de manera que el emisor pueda modificarlos a su gusto para posteriormente firmar el certificado con la clave seleccionada.

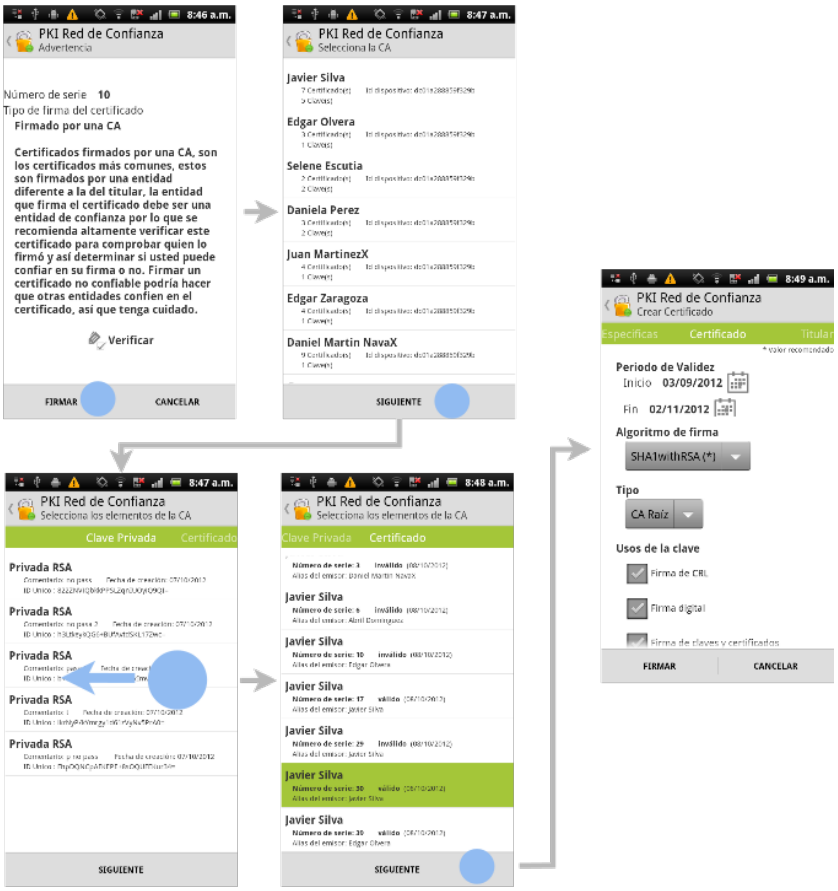


Figura A.17: Interfaz - Proceso de firma de certificado.

El proceso de verificación inicia al seleccionar la opción de verificar tanto en la vista de detalles como en la advertencia inicial al firmar un certificado, la figura A.18 ilustra este

proceso. De manera general existen dos mecanismos para verificar un certificado, los cuales fueron explicados en la sección 3.1.2 y en la interfaz esto se refleja utilizando un paginador para que el usuario pueda seleccionar el método deseado, si se quiere utilizar el PGP+ basta con seleccionar la lista de confianza que se utilizará para la verificación y seleccionar siguiente, esto hará que la aplicación realice la verificación y muestre los resultados, los que serán mostrados en dos páginas, una para mostrar la confianza calculada y la otra para mostrar la lista de los certificados de las entidades emisoras del certificado seleccionado, cada una con el nivel de confianza asignado en la lista seleccionada. De manera similar para utilizar el método de X.509 es necesario seleccionar la CRL que se utilizará para verificar el certificado y tocar el botón de siguiente, al hacer esto la aplicación buscara el camino de certificación utilizando los certificados almacenados en la base de datos y los listará en la segunda página de los resultados, mientras que la primera página mostrara el estatus resultante de la verificación del certificado. En ambos casos la lista mostrada muestra información básica de los certificados, es por esto que se incluye el botón para ir a los detalles del certificado que se desee.

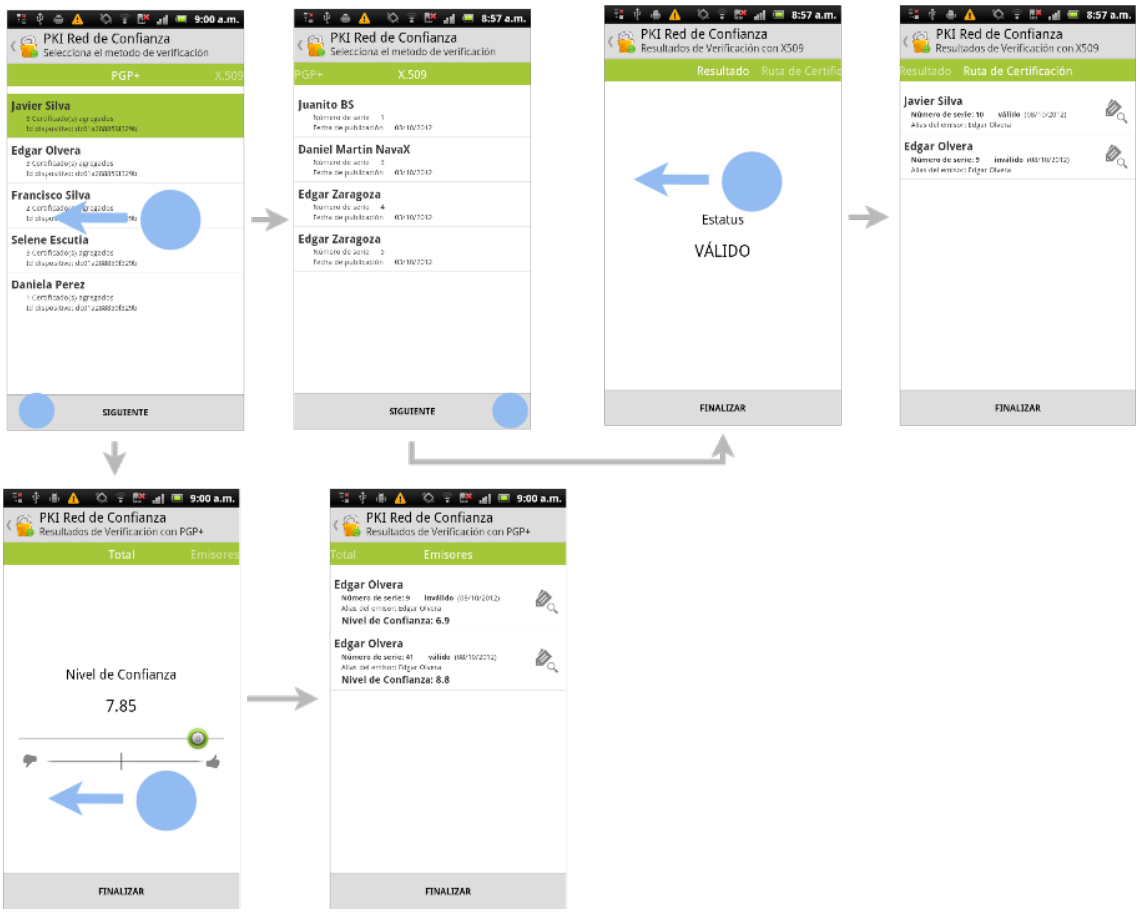
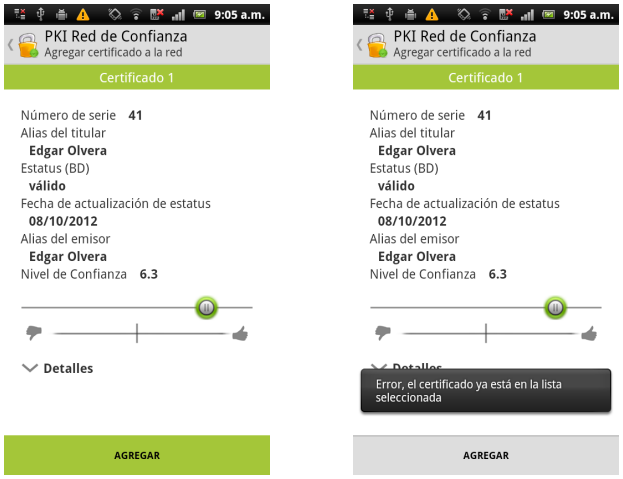


Figura A.18: Interfaz - Proceso de verificación de certificado.

El siguiente modulo que se explicará, es el modulo de red de confianza el cual se encuentra en la tercera pestaña de la interfaz principal. Éste permite al usuario crear listas

de confianza, modificar el nivel de confianza en un certificado, eliminar un certificado de la lista, importar y exportar las listas creadas. Las interfaz principal para agregar nuevo certificado a una lista se ve en la figura A.19, en ésta se pueden ver los detalles del certificado con una barra (a), con la cual el usuario puede asignar el nivel de confianza deslizando el indicador hasta la posición deseada, al igual que las demás interfaces en la aplicación, los datos de ésta se validan para evitar inconsistencias en la base de datos del dispositivo (b).



Agregar certificado (a)

Error al agregar certificado (b)

Figura A.19: Interfaz para agregar certificado a lista de confianza.

Por otra parte la interfaz utilizada para ver los detalles de la lista de confianza es igual a la interfaz para los detalles de certificados, ya que está basada en un paginador con el cual se puede navegar entre los certificados de la lista y es posible acceder a los detalles de estos. Sin embargo como se puede ver en la figura A.20, cada página muestra el nivel de confianza que tiene asignado el certificado en la lista seleccionada. Esta interfaz también tiene la opción de eliminar, modificar, exportar y compartir un certificado.

Por último se presentan las interfaces correspondientes al modulo de operaciones criptográficas que ofrece la aplicación, este modulo permite al usuario cifrar y firmar o descifrar y verificar tanto archivos como mensajes de texto dentro del dispositivo, utilizando los certificados y claves privadas (según corresponda) almacenadas en la base de datos de la aplicación. En la figura A.21 (a) se muestra la interfaz de operaciones sobre archivos, mientras que en la figura A.21 (b) se puede ver la interfaz de operaciones sobre mensajes, para cambiar de una a otra se utiliza el gesto *swipe*. Adicionalmente en la figura A.21 (c) se pueden ver las operaciones disponibles en esta interfaz. Para finalizar es importante mencionar que estas operaciones se pueden hacer solas o en conjunto para cada tipo de entrada, para seleccionar la operación a realizar es necesario activarla en la interfaz como se muestra en la figura A.21 (d), para cada operación es necesario introducir la información necesaria, por ejemplo, el certificado del receptor en caso de cifrado y la clave pública para firmar la entrada seleccionada.

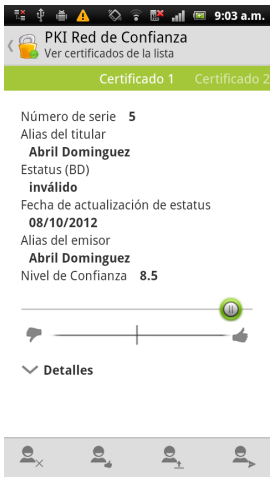
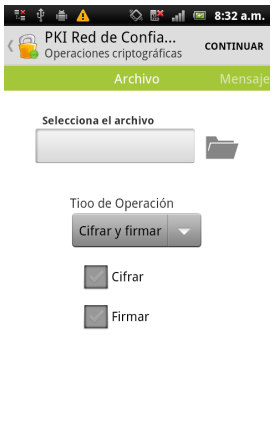
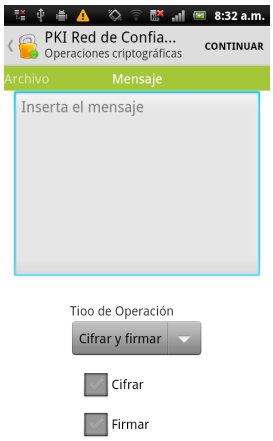


Figura A.20: Interfaz - Certificado en lista de confianza.



(a) Operación sobre archivo



(b) Operación sobre mensaje

(c) Tipo de operación

(d) Opciones de cifrado

Figura A.21: Interfaz - Operaciones criptográficas.

La tesis presentada por Javier Silva Pérez fue aprobada por:

Dr. Guillermo Morales Luna, Director

Dr. Arturo Díaz Pérez

Dr. Dominique Decouchant

México, D.F., 30 de Noviembre de 2012