UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

TRABAJO FIN DE GRADO

Criptografía basada en emparejamientos: motivación, estudio y aplicaciones prácticas

Laura Ramírez Lapeña

Tutor: David Arroyo Guardeño

Junio 2015

Agradecimientos

Me gustaría agradecer a todas aquellas personas que me han ayudado a llegar hasta aquí y a finalizar mi proyecto.

A mi tutor David por haberme guiado durante estos últimos meses.

A todos los profesores que me han ayudado a lo largo de estos 5 años.

A mis amigos de toda la vida por haberme animado y apoyado siempre.

A mis compañeros de la Universidad, especialmente a Javier Sanz, pendiente siempre de ayudarme con cualquier duda que tuviera, y a Rus María Mesas, la mejor amiga y compañera que una podría tener.

A mi familia, porque sin ellos no hubiera sido posible.

Resumen

El gran crecimiento de las Tecnologías de la Información y de las Comunicaciones (TIC) presenta grandes oportunidades para el desarrollo de productos para el tratamiento de la información. Un punto crítico de todas estas tecnologías es la generación y control de identidades digitales. Así, gran cantidad de aplicaciones TIC requieren la identificación de usuarios para garantizar la protección de la información, lo cual incluye la asociación de identidades digitales con identidades físicas. Dicha asociación y su adecuado control constituye uno de los mayores retos en el ámbito actual de las TIC, sobre todo tras la aparición de tecnologías como los servicios *cloud* y el denominado *big data*.

La criptografía constituye el principal medio para la creación y gestión de identidades digitales. La criptografía asimétrica o de clave pública proporciona procedimientos y protocolos para la generación y distribución de claves secretas. En la criptografía pública, cada usuario genera un par de claves pública/privada en base al cual intercambian una clave simétrica. Esto soluciona el problema de la distribución de claves, pero genera uno nuevo, ya que se debe realizar una asociación correcta entre cada usuario (identidad física) y su par de claves (identidad digital). Tradicionalmente, esta asociación se realiza a través de algún tipo de certificado manejado por una autoridad confiable, dando origen a una infraestructura de clave pública (en inglés, Public Key Infrastructure -PKI-). A pesar de que PKI es la elección más popular a la hora de trabajar con identidades digitales, tiene una serie de inconvenientes que van a ser analizados a lo largo del trabajo, y que motivan el estudio de esquemas alternativos.

El objetivo del presente documento es estudiar la criptografía basada en emparejamientos (en inglés, Pairing-Based Cryptography -PBC-) para plantear alternativas a los esquemas PKI tradicionales. Tras el pertinente análisis de las ventajas de la PBC, se utilizará para la generación de identidades digitales con el fin de proteger el correo electrónico mediante IBE (del inglés, Identity-Based Encryption). De esta forma, se muestra una de las posibles aplicaciones prácticas de este tipo de criptografía. A par-

tir del servidor de *gmail*, el sistema permite enviar y recibir correos protegidos con IBE y, alternativamente, con RSA. Con IBE los usuarios no tienen la necesidad de obtener certificados ni de almacenar sus claves, y pueden enviar mensajes simplemente conociendo la identidad del destinatario (i.e., su correo electrónico). Un servidor externo a la aplicación de correo (PKG, del inglés Private Key Generator) se encarga de la generación y gestión de las identidades para todos los usuarios a través de un canal de comunicación seguro.

Para la implementación de IBE se ha utilizado la biblioteca criptográfica *open source* JPBC (*Java Pairing Based Cryptography*). Esta biblioteca está en fase de desarrollo y posee escasa documentación, de modo que su correcto uso y aplicación en el proyecto ha constituido uno de los principales retos del mismo. Así, este TFG contribuye al entendimiento y a la mejora de JPBC aportando un caso de uso que puede ser de ayuda a otros desarrolladores.

Palabras clave

Criptografía, identidad física, identidad digital, distribución de claves, RSA, PBC, IBE, PKG, protección de correo electrónico, JPBC.

Abstract

The recent growth of the Information and Communications Technologies (ICT) has led to the need of generating and controlling digital identities associated to physical identities. Traditionally, this association is performed through certificates managed by a trustworthy authority, giving rise to Public Key Infrastructure (PKI). Although nowadays PKI is the most popular choice, it has several drawbacks that are going to be analysed in this document and that motivates the study of other alternatives.

This paper presents Pairing-Based Cryptography (PBC) as an alternative to the traditional PKI. After studying the advantages of PBC, it will be used to generate digital identities with identity-based encryption (IBE) in order to secure electronic mail. Using a gmail account, the system allows sending and receiving emails secured by IBE and alternatively by RSA. IBE enables sender to encrypt a message when the only information he knows is recipient's identity (e.g. email address). Moreover, users do not need certificates to bind identity with specific public key. The Private Key Generator (PKG), which is an external server, is responsible for generating and managing the identities for all users through a secure channel.

The open source library JPBC (Java Pairing Based Cryptography) has been used to implement IBE. This library is still being developed and does not have much documentation. Therefore, its correct usage in the project has been one of the greatest challenges. This work contributes to improve JPBC library, providing an example that could be helpful for other developers.

Key Words Cryptography, digital identity, physical identity, key distribution, RSA, PBC, IBE, PKG, secure email, JPBC

Índice general

Índice de figuras

Índice de tablas

Glosario

AES: *Advanced Encryption Standard*, algoritmo de cifrado por bloques de cifrado simétrico muy popular. Transforma bloques de texto en claro y los convierten en bloques de texto cifrado. Tiene un tamaño de bloque fijo de 128 bits y tamaños de clave de 128, 192 o 256 bits. Puede operar en varios modos de operación, como CBC (*Cipher-Block Chaining*), que a cada bloque de texto plano le aplica la operación XOR con el bloque cifrado anterior.

API *Application Programming Interface*, conjunto de subrutinas, funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

Arquitectura centralizada: Modelo cliente-servidor en el que las tareas se reparten entre los proveedores de servicios, llamados servidores, y los demandantes, llamados clientes. Un cliente realiza peticiones a otro programa, el servidor, quien le da respuesta.

Arquitectura descentralizada: Red *peer-to-peer* compuesta por una serie de nodos que se comportan como iguales entre sí. Es decir, actúan simultáneamente como clientes y servidores respecto a los demás nodos de la red.

ASN.1: *Abstract Syntax Notation One*, norma para representar datos independientemente de la máquina que se esté usando y sus formas de representación internas.

BDH: *Bilinear Diffie-Hellman*, variante de CDH para aplicaciones bilineales.

Bouncy Castle Colección de APIs utilizados en criptografía.

CA: *Certificate Authority*, autoridad de certificación en la infraestructura de clave pública. Tras una verificación correcta por parte de la RA, genera un certificado que contiene la clave, la identidad del usuario y su propia firma.

CC *Certificateless Cryptography*, variante de la criptografía basada en identidad que trata de evitar el problema *key escrow*.

CDH: *Computational Diffie-Hellman*, suposición que asume que un cierto problema computacional es difícil en un grupo cíclico. Establece que a partir de la tupla (g, g^a, g^b) no se puede calcular computacionalmente g^{ab}

Certificado digital: Fichero generado por una entidad de servicios de certificación (autoridad certificadora) que asocia unos datos de identidad a una persona física, organismo o empresa confirmando de esta manera su identidad digital en Internet. Es necesaria la colaboración de un tercero que sea de confianza para cualquiera de las partes que participe en la comunicación.

Cifrado simétrico: Utiliza la misma clave para cifrar que para descifrar mensajes. Las dos partes de la comunicación deben ponerse de acuerdo de antemano sobre la clave simétrica a usar. Una vez que ambas partes tienen acceso a dicha clave, el emisor cifra el mensaje usándola, lo envía al destinatario y éste lo descifra con la misma clave.

Cifrado asimétrico: Utiliza un par de claves para cifrar y descifrar mensajes. Cada usuario posee su par de claves, una pública que puede obtener cualquier otro usuario, y otra privada que mantiene en secreto. Así, los mensajes son cifrados por un emisor con la clave pública del destinatario, quien podrá descifrarlos usando su clave privada.

Criptosistema: Sistema de cifrado de información.

DDH: *Decisional Diffie-Hellman*, suposición que establece que es difícil distinguir las tuplas de la forma (g, g^a, g^b) de las tuplas aleatorias.

Encapsulamiento: Procedimiento para cifrar la clave simétrica mediante una clave pública.

Gmail: *Google Mail*, servicio de correo electrónico con posibilidades POP3 e IMAP gratuito.

GUI: *Graphical User Interface*, programa que actúa de interfaz de usuario, utilizando un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz.

Hash: Función computable mediante un algoritmo que tiene como entrada un conjunto de elementos, que suelen ser cadenas, y los convierte (mapea) en un rango de salida finito, normalmente cadenas de longitud fija. La idea básica de un valor *hash* es que sirva como una representación compacta de la cadena de entrada.

HTTP: *Hypertext Transfer Protocol*, protocolo usado en cada transacción de la World Wide Web. Define la sintaxis y la semántica que utilizan los elementos de software de la arquitectura web para comunicarse. Es un protocolo orientado a transacciones y sigue el esquema petición-respuesta entre un cliente y un servidor.

IBC: *Identity-Based Cryptography*, criptografía caracterizada por el uso de atributos de identidad de los usuarios (cadenas de caracteres identificativos) a partir de las cuales se puede cifrar y verificar las firmas, sin ser necesario el uso de los certificados digitales de PKI.

IBE: *Identity-Based Encryption*, primitiva importante de IBC y, como tal, es un tipo de cifrado de clave pública en la cual la clave pública de un usuario es alguna cadena única identificativa del usuario (por ejemplo, el *email*).

JCA *Java Cryptographic Architecture*, *framework* para acceder y desarrollar funciones criptográficas en la plataforma Java.

JCE *Java Cryptographic Extension*, proporciona un *framework* e implementación para cifrados, generación de claves, protocolos de establecimiento de claves y Código de Autenticación del Mensaje (MAC). Es un suplemento de la plataforma Java.

JPBC: *Java Pairing-Based Cryptography Library*, versión Java de la PBC.

KEM *Key Encapsulation Mechanisms*, técnicas de cifrado diseñadas para proporcionar seguridad a la transmisión de claves simétricas usando algoritmos de clave asimétrica.

Key escrow: Acuerdo en el que las claves necesarias para descifrar los datos cifrados se mantienen en reserva para que, en determinadas circunstancias, un tercero autorizado puede tener acceso a las claves.

Keytool Herramienta de gestión de claves y certificados. Permite a los usuarios administrar sus propios pares de claves pública/privada y los certificados asociados.

KGC *Key Generator Center*, autoridad parcialmente confiable de la criptografía libre de certificados.

Librería PBC: *Pairing-Based Cryptography Library*, biblioteca desarrollada por Ben Lynn en C para realizar las operaciones matemáticas subyacentes en los sistemas criptográficos basados en emparejamientos.

MVC: *Modelo Vista Controlador*, patrón de arquitectura de software que separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones. Para ello MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador.

OpenPGP: Estándar de cifrado de *emails* más usado en el mundo. Originalmente derivado de PGP.

OpenSSL Paquete de herramientas de administración y bibliotecas relacionadas con la criptografía. Suministra funciones criptográficas que ayudan al sistema a implementar SSL, así como otros protocolos relacionados con la seguridad. También permite crear certificados digitales que pueden aplicarse a un servidor.

PBC: *Pairing-Based Cryptography*, caso particular de IBC especialmente eficiente que está basado en emparejamientos bilineales.

PGP: *Pretty Good Privacy*, programa cuya finalidad es proteger la información distribuida a través de Internet así como facilitar la autenticación de documentos gracias a firmas digitales. Es un criptosistema híbrido que combina técnicas de criptografía simétrica y criptografía asimétrica.

PKG: *Private Key Generator*, generador de claves privadas para los usuarios de un sistema basado en IBE. Para ello publica su clave pública maestra y guarda en secreto su clave privada maestra, a partir de la cual genera las claves privadas de distintas identidades.

PKI: *Public Key Infrastructure*, combinación de hardware y software, políticas y procedimientos de seguridad que permiten la ejecución con garantías de operaciones criptográficas como el cifrado, la firma digital o el no repudio de transacciones electrónicas. La tecnología PKI permite a los usuarios autenticarse frente a otros

usuarios y usar la información de los certificados de identidad para cifrar y descifrar mensajes, firmar digitalmente información, garantizar el no repudio de un envío

RA: *Registration Authority*, autoridad de registro en la infraestructura de clave pública. Recoge peticiones de usuarios para emitir certificados digitales para sus claves públicas. Vertifica las credenciales de cada usuario.

RSA: *Rivest, Shamir y Adleman*, algoritmo criptográfico asimétrico o de clave pública más conocido y utilizado. La seguridad del algoritmo radica en la factorización de números enteros.

SHA: *Secure Hash Algorithm*, familia de funciones *hash* de cifrado. SHA-1 es la segunda versión del sistema y produce una salida resumen de 160 bits (20 bytes) de un mensaje que puede tener un tamaño máximo de 2^{64} bits.

SMTP *Simple Mail Transport Protocol*, protocolo de red utilizado para el intercambio de mensajes de correo electrónico entre computadoras u otros dispositivos.

Sockets: Constituyen el mecanismo para el intercambio entre dos programas de cualquier flujo de datos, generalmente de manera fiable y ordenada.

SSL: *Secure Socket Layer*, protocolo criptográfico que proporciona comunicaciones seguras por la red. Se usan certificados X.509 (criptografía asimétrica) para autenticar a ambas partes de la comunicación así como para intercambiar una clave simétrica.

TLS: *Transport Layer Security*, sucesor de SSL.

X.509: Estándar para infraestructuras de claves públicas. Especifica formatos para certificados de claves públicas y un algoritmo de validación de la ruta de certificación.

Capítulo 1

Introducción

La historia de la criptografía se remonta miles de años atrás, desde el Antiguo Egipto y Roma. Hasta la llegada del ordenador en décadas recientes, la criptografía imperante (ahora llamada **criptografía clásica**) se basaba en métodos de cifrado realizables con papel y lápiz o, como mucho, con una pequeña ayuda mecánica simple, y su único objetivo era hacer ininteligibles los mensajes a receptores no autorizados. Sin embargo, a principios del siglo XX, la creación de máquinas capaces de realizar una gran cantidad de cálculos en poco tiempo propició la invención de métodos de cifrado de mayor complejidad y eficiencia, lo cual tuvo un gran impacto en los conflictos bélicos que se produjeron en todo el mundo.

La aparición de los ordenadores y las comunicaciones digitales supusieron un reto para la seguridad basada en la criptografía clásica, ya que toda la información transmitida a través de la red puede ser interceptada y manipulada si no se toman medidas para impedirlo. Para garantizar la seguridad de las comunicaciones que se transmiten en esta era de información se han desarrollado nuevas técnicas criptográficas como la **criptografía de clave secreta** (AES, DES), **criptografía de clave pública** o asimétrica (RSA, El Gammal), o las **firmas digitales**.

La importancia y necesidad de la criptografía crecen a la par que lo hacen las nuevas tecnologías, es decir, de forma exponencial. Aunque pueda pasarnos desapercibido, la criptografía juega un papel fundamental en nuestra vida diaria y está presente en multitud de procesos: transacciones bancarias, cajeros automáticos, comunicaciones por teléfono móvil, compras por internet, y en general en cualquier tipo de comunicación por red, tales como el correo electrónico o las redes sociales [1]. En efecto, tal y como aparece reflejado en la Agenda Digital de la Unión Europea [2], las TIC actualmente juegan un papel fundamental en el desarrollo económico de los distintos países. Sin embargo, es necesario que la población use de modo habitual estas tecnologías, por lo que deben cumplirse objetivos de seguridad y usabilidad en los sistemas

desarrollados. Es aquí donde la seguridad informática en general, y la criptografía en particular, encuentran uno de sus principales desafíos. En la gestión de identidades se produce una compleja asociación entre la identidad física de una persona y su identidad digital que constituye uno de los puntos de ataques e intervenciones de agentes *maliciosos*.

1.1. Objetivos y motivaciones

La generación de identidades digitales, así como su correcta gestión, constituyen un elemento crítico de la criptografía contemporánea. Problemas como el robo de credenciales o la escalada de privilegios no pueden evitarse sin una política rigurosa para la distribución y asignación de claves y niveles de acceso. De modo estándar, los requerimientos de autenticación y autorización se satisfacen asignando privilegios de acceso a identidades digitales soportadas por infraestructuras de clave pública (en inglés, **Public Key Insfrastructure -PKI-**).

Desde la primera contribución de Diffie y Hellman en 1976 han surgido diversas propuestas orientadas a eliminar la dependencia respecto a una tercera parte encargada de la generación y gestión de dichas identidades. En la última década, la criptografía basada en emparejamientos (en inglés, **Pairing-Based Cryptography -PBC-**) ha surgido como una alternativa a considerar a lo hora de implementar infraestructuras de clave pública [3]. Frente a los tradicionales esquemas verticales y horizontales de autenticación, la PBC permite diseñar protocolos de autenticación basados en identidad (en inglés, **Identity-Based Encryption –IBE-**) o, en caso de que el modelo de atacante lo requiera, en criptografía libre de certificados (en inglés, **Certificateless Cryptography**). Asimismo, la PBC es clave en el contexto de los sistemas criptográficos para el despliegue y control del anonimato.

Así pues, en este trabajo se han analizado los problemas más relevantes de estas infraestructuras de clave pública y se han planteado soluciones mediante criptografía basada en emparejamientos. Para ello, se ha realizado un estudio de las aplicaciones bilineales como base de la criptografía basada en emparejamientos así como un **análisis de la aplicación de la PBC a la gestión de identidades digitales señalando** sus **principales ventajas y desventajas**. Sin embargo, un grueso importante del trabajo reside en el aprendizaje del manejo de la **biblioteca JPBC** o *Java Pairing-Based Cryptography Library* [4] para este tipo de criptografía. Actualmente, la mayoría de las bibliotecas criptográficas (incluida la JPBC) tienen escasa documentación, lo cual dificulta enormemente su entendimiento y su uso.

Finalmente, se ha implementado una arquitectura software con un servidor de

identidades digitales y gestión dinámica de las mismas. A partir de la biblioteca proporcionada JPBC y varias modificaciones realizadas sobre ésta, el resultado final permite **asignar identidades digitales a usuarios mediante su cuenta de correo electrónico**. De esta forma, todos los correos que se envíen desde este sistema serán cifrados utilizando IBE. Alternativamente, se podrán enviar y recibir los correos cifrados mediante RSA.

Es importante destacar que la arquitectura diseñada está relacionada con el **modelo de confianza elegido**: no todos los modelos de confianza son aceptables en todos los contextos. Desplegar un mecanismo de gestión de identidades digitales en una organización privada bajo nuestro control no es lo mismo que desplegar dicho mecanismo en el contexto de plataformas *cloud*. Es más, es posible que en ciertos ámbitos los dispositivos involucrados no tengan suficiente capacidad de cómputo y, consecuentemente, se podrían delegar ciertas tareas criptográficas en un agente sobre el que se tiene confianza.

1.2. Estructura del documento

El presente documento está estructurado en **ocho capítulos**.

El **primer capítulo** es esta introducción, que pone el trabajo en contexto y describe los principales objetivos y las motivaciones del mismo.

En el **segundo capítulo** se describen los problemas de la gestión de identidades así como la criptografía de clave asimétrica y simétrica y los distintos procedimientos de la infraestructura de clave pública. Asimismo, se introduce la criptografía basada en identidad como alternativa a PKI, intentando solventar algunos de los inconvenientes de la infraestructura de clave pública.

El **tercer capítulo** detalla los fundamentos teóricos de la criptografía basada en emparejamientos. A su vez, identifica las distintas entidades que participan en este tipo de esquemas y las funciones que realiza cada una de las partes involucradas en el esquema. Finalmente, describe las ventajas y desventajas de este tipo de criptografía.

El **cuarto capítulo** se centra en el análisis del proyecto software llevado a cabo. Se describen los objetivos y el alcance del mismo y se especifica el catálogo de requisitos completo y los casos de uso.

En el **quinto capítulo** se detalla el diseño del sistema a partir de una serie de diagramas que determinan las interacciones entre los distintos componentes.

El **sexto capítulo** describe los detalles de la implementación del proyecto. Esto incluye la especificación de la estructura del código, las bibliotecas utilizadas y la codificación llevada a cabo.

El **séptimo capítulo** consta de una descripción detallada de las pruebas realizadas al sistema software implementado. En primer lugar, se describen las pruebas unitarias llevadas a cabo en cada componente, y, a continuación, se describen las pruebas de integración para los conjuntos de componentes.

Por último, en el **octavo capítulo** se concluye el trabajo realizado y se resumen las limitaciones del mismo, así como una serie de mejoras que podrían realizarse en un futuro.

Capítulo 2

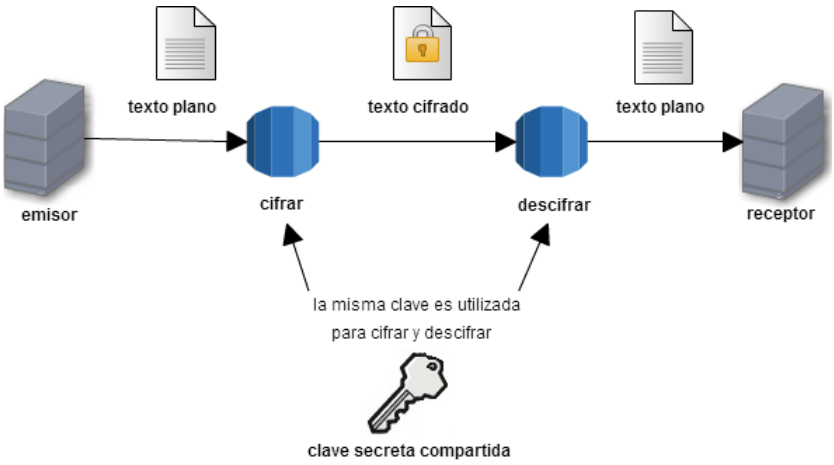
El problema de la gestión de identidades

En este capítulo se introduce los distintos tipos de criptografía (simétrica y asimétrica) así como los procedimientos de la infraestructura de clave pública (centralizados y descentralizados). A su vez, se analiza el problema de la correcta asociación de identidades físicas e identidades digitales y se propone como solución la criptografía basada en identidad.

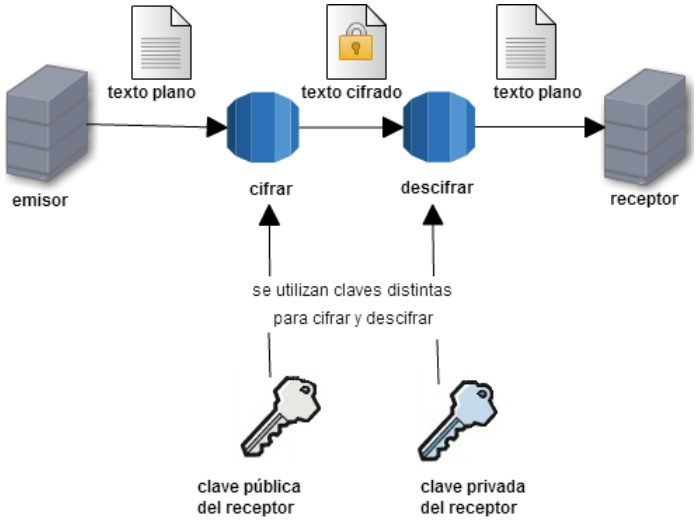
2.1. Análisis del problema

En la **criptografía de clave simétrica o de clave secreta** se utiliza la misma clave para cifrar un texto que para descifrarlo (análogo al funcionamiento de un candado). Requiere por tanto que la clave se intercambie entre los dos extremos de la comunicación por un medio externo seguro y fiable. Por otro lado, la **criptografía de clave asimétrica o de clave pública** parte de la idea de que son necesarias dos claves distintas: una para cifrar (clave pública) y otra para descifrar (clave privada). La clave de cifrado se hace pública y cualquiera puede cifrar textos planos con ella, pero sólo quien posee la clave privada correspondiente a esa clave pública podrá descifrar mensajes. Se plantea de esta forma una solución para poder cifrar las comunicaciones por Internet, dado que este sistema podría servir para intercambiar la clave de un método de cifrado simétrico de forma segura, para posteriormente establecer la transmisión cifrada con dicha clave [1]. Ambos esquemas se pueden observar en la Figura 2.1.

Proporcionar **autenticidad en las claves públicas** es uno de los problemas más difíciles de este último tipo de criptografía. No hay soluciones obvias para ello, puesto que todas ellas tienen sus desventajas y limitaciones, pero existen varias aproxima-



(a)



(b)

Figura 2.1: (a)Esquema del cifrado simétrico; (b) Esquema del cifrado asimétrico.

ciones al problema que se concentran en una idea: **confianza**. Así, la criptografía de clave pública se utiliza cuando las dos partes de una comunicación no pueden intercambiar claves directamente, lo cual ocurre casi siempre en el caso de las redes de ordenadores.

En general, las personas no pueden saber, con sólo mirar una cierta clave, si ésta pertenece a la persona que así lo asegura, por lo que debe haber otras partes que conozcan a ambas partes de la comunicación y que tengan su confianza. Esto conduce a varias implementaciones diferentes de la idea de **tercero de confianza o tercera parte confiable** (en inglés, Third Trusted Partie -TTP-). Los dos enfoques generales pueden describirse como **descentralizado** y **centralizado** (también conocidos como

horizontal y vertical). El primero es como una red *peer-to-peer*, donde cada nodo tiene las mismas capacidades y derechos, mientras que el segundo se asemeja a un modelo cliente-servidor tradicional, donde el servidor juega el papel central y proporciona servicios a los clientes [5].

2.2. Anillo de confianza

Un anillo de confianza (en inglés, web of trust) es un concepto usado en PGP, GnuPG y otros sistemas compatibles con OpenPGP para certificar que un usuario es quién dice ser [6].

Es la implementación más común de la **tercera parte de confianza descentralizada** y su infraestructura es muy simple, ya que no hay nodos adicionales aparte de los usuarios. Se basa en que los usuarios coleccionan las claves públicas de otros y las certifican si están seguros de que una cierta clave pertenece a una determinada persona (por ejemplo, habiéndose encontrado físicamente). Para certificar la correspondencia entre el nombre atribuido a la clave y la clave en sí misma, los participantes firman entre sí sus claves públicas (en inglés, key signing parties). Cuando dos partes no se conocen, deben conocer a alguien en el anillo de confianza que pueda proporcionar un enlace entre ambas.

Un ejemplo de anillo de confianza se puede observar en la Figura 2.2. Cuando María obtenga la clave de Sergio o de Alejandro, puede verificar que realmente pertenece a ellos, pues confía en Marta, que a su vez confía en Sergio y en Alejandro. Sin embargo, si María recibe la clave de Ana, no tiene forma de asociar la clave con Ana, ya que nadie confía en Ana.

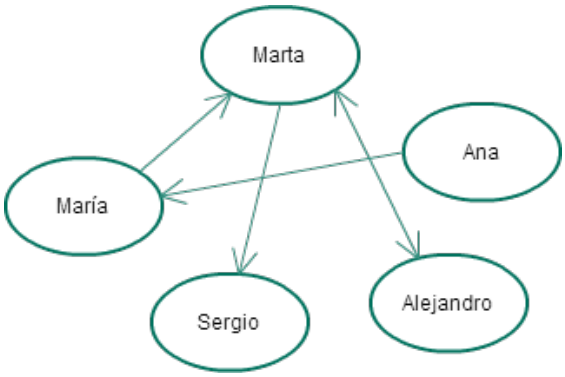


Figura 2.2: Ejemplo anillo de confianza.

Este modelo de confianza es muy **flexible** y ofrece a los usuarios una gran libertad a la hora de la certificación, ya que cada usuario decide por sí mismo si aceptar o

rechazar cierta clave. Además, **no contiene un único punto de fallo** y **no requiere una infraestructura costosa** ya que debido a la falta de una autoridad certificadora, los usuarios recogen las claves públicas y certificados[1] en sus propias máquinas [5].

Sin embargo, esta implementación requiere al mismo tiempo **precaución y supervisión inteligente** por parte de los usuarios y es vulnerable a ciertas manipulaciones. Además, las personas en lugares remotos o con pocos amigos son consideradas menos creíbles y no pueden beneficiarse completamente de la seguridad, a no ser que haya otros usuarios que puedan introducirlo al resto.

Así pues, si la criptografía es para ser usada de modo generalizado, debería ser transparente y fácil de utilizar [7, 8, 9]. En principio, la mayor parte de los problemas mencionados pueden ser resueltos por los esquemas centralizados como la infraestructura X.509 de clave pública.

2.3. Infraestructura de clave pública tradicional

La infraestructura X.509 de clave pública es la solución más popular para demostrar la **autenticidad de claves públicas** [10]. Al igual que en el anillo de confianza, se usan **certificados** para confirmar la relación entre el usuario y su clave pública. Por otro lado, este modelo de PKI es **centralizado y jerárquico**, compuesto de nodos especiales llamados autoridad de registro (en inglés, Registration Authority -RA-) y autoridad de certificación (en inglés, Certificate Authority -CA-). Estas autoridades son las terceras partes de confianza, que no están a cargo de los usuarios normales.

La **autoridad de registro** recoge peticiones de usuarios para emitir certificados digitales para sus claves públicas y debe verificar las credenciales de cada usuario. Tras una verificación correcta, la **autoridad de certificación** genera un certificado que contiene la clave, la identidad del usuario y su propia firma [11]. La pregunta fundamental es: ¿quién concedió a la autoridad de certificación el derecho a la generación de claves de los usuarios? Por lo general, se trata de otra entidad emisora de certificados, que es de aún mayor confianza.

Una de las ventajas más relevantes de la infraestructura PKI es su fácil despliegue e implementación, esto es, su **escalabilidad**. Además, el uso de esta tecnología es **transparente para los usuarios finales**, de modo que sus objetivos de autenticación, integridad y no-repudio se consiguen sin un esfuerzo excesivo por parte de los mismos. Un usuario de PKI tradicional no ha de tener un alto conocimiento técnico sobre esta infraestructura, ya que el establecimiento de una comunicación de confianza

[1] En el contexto de PGP estos certificados contienen la clave pública y el nivel de confianza de los usuarios incluidos en el anillo.

sólo precisa el uso del navegador.

Sin embargo, esta infraestructura es muy **costosa** ya que requiere que las autoridades de confianza obedezcan estrictamente las políticas de seguridad, a lo que hay que añadir los costes asociados a la revocación de certificados [12]. Por otra parte, si comparamos con los anillos de confianza, los diseños de PKI tradicionales son **menos flexibles** y requieren que el usuario siga la cadena de garantías de los certificados generados por la autoridad de certificación. Además, los certificados deben ser verificados por los usuarios (si coinciden o no con la identidad correcta), pero la mayor parte de los usuarios no lo hacen bien, a pesar de que el proceso de comprobación no es muy complicado (por ejemplo, comparando la dirección web con el poseedor especificado en el certificado) [5].

Aprovechando esta debilidad, son muchos los atacantes que registran dominios ya existentes con algún error tipográfico imitando las páginas web originales pero tratando de robar información al usuario. Algunos ejemplos conocidos se muestran en la Tabla 2.1[2].

Dominio falso	**Dominio real**
Wallmatt.com	Walmart.com
Appple.com	Apple.com
BestBuyh.com	Bestbuy.com
Fashiomworld.co.uk	Fashionworld.co.uk
Sportsdierct.com	Sportsdirect.com
Wurbanotfitters.co.uk	Urbanoutfitters.co.uk
Datigdirect.com	Datingdirect.com

Tabla 2.1: Ejemplos de dominios falsos.

Una solución a este último problema podría ser implementar software que realizara las comprobaciones necesarias, pero esto tiene sus limitaciones en el mundo real. Otra solución más radical sería abandonar completamente el uso de certificados, como ocurre en la **criptografía basada en identidad** (en inglés, Identity-Based Cryptography -IBC-).

2.4. Otra aproximación al problema: IBC

Tanto en los anillos de confianza como en PKI tradicional, es necesario establecer certificados que relacionen al usuario con su clave pública. En el caso de la infraestructura de clave pública, el poseedor de un certificado es responsable de la conservación

[2] `http://usatoday30.usatoday.com/tech/news/story/2011-12-26/typosquatting/52229886/1`

y custodia de la clave privada asociada al certificado para evitar el conocimiento de la misma por terceros. Además, las claves públicas que los usuarios, en ambos sistemas, deben almacenar, no tienen una forma muy atractiva ni fácil de recordar. Por último, tanto en PGP como en X.509 la revocación de identidades digitales no es trivial.

Por ello, surge la criptografía basada en identidad o IBC, una de las principales aplicaciones de la PBC, y que constituye el núcleo de este trabajo de fin de grado (TFG). IBC se caracteriza por el **uso de atributos de identidad de los usuarios** (cadenas de caracteres identificativos tales como direcciones de *email* o números de teléfono, **fáciles de memorizar**). A partir de estas cadenas identificativas se puede generar un par de claves pública/privada para intercambiar claves simétricas sin ser necesario el uso de los certificados de PKI. De esta forma, es mucho más fácil proporcionar criptografía a usuarios noveles, ya que los mensajes pueden ser cifrados por los usuarios antes de que éstos interactúen con cualquier entidad [13].

Hay dos **problemas criptográficos** de base que se precisan para, en primer lugar, garantizar la seguridad de las credenciales generadas mediante PBC (problema computacional de Diffie-Hellman) y, en segundo lugar, para hacer factible la asociación entre cadenas identificativas y las identidades criptográficas creadas (problema decisional de Diffie-Hellman).

2.4.1. Problema Diffie-Hellman computacional

La suposición computacional de Diffie-Hellman (en inglés, Computational Diffie-Hellman, -CDH-) asume que un cierto problema computacional es **difícil en un grupo cíclico**[3] [14].

Sea $\mathcal{G}$ un grupo de orden q (ver Definición A.1.10). La suposición CDH establece que, dado (g, g^a, g^b) para un generador g y $a, b \in \{0, .., q-1\}$ aleatorios, es computacionalmente intratable calcular g^{ab}.

La seguridad de muchos criptosistemas se basa en la suposición CDH, como es el caso de IBE. Esta suposición está relacionada con **el problema del logaritmo discreto**, que establece que calcular el logaritmo discreto de un valor en base al generador g es difícil. Si tomar logaritmos discretos en $\mathcal{G}$ fuera fácil, la suposición CDH sería falsa. Es un problema abierto determinar si la suposición del logaritmo discreto es equivalente a la CDH, aunque se ha demostrado en ciertos casos especiales [15].

La suposición CDH está también relacionada con la decisional de Diffie-Hellman (en inglés, Decisional Diffie-Hellman assumption -DDH-).

[3] Ver A.1.8 para una definición de grupo cíclico.

2.4.2. Problema Diffie-Hellman decisional

La suposición decisional de Diffie-Hellman establece que es difícil distinguir las tuplas de la forma (g, g^a, g^b) de las tuplas aleatorias. Si calcular g^{ab} a partir de (g, g^a, g^b) fuera fácil, entonces uno podría detectar tuplas DDH trivialmente. Se cree que CDH es una suposición más débil que DDH: hay grupos para los que detectar tuplas DDH es fácil, pero solucionar el problema CDH es difícil [16].

Capítulo 3

Criptografía basada en emparejamientos

Partiendo de la teoría básica recogida en el Anexo A, en este capítulo se introducen brevemente los fundamentos teóricos de la criptografía basada en emparejamientos. A su vez se describen los diferentes componentes y protocolos que forman una arquitectura basada en este tipo de criptografía. Finalmente, se detallan todas las ventajas y desventajas que tienen los sistemas que se basan en ella.

3.1. Fundamentos teóricos

Los objetivos de PKI e IBE son bastante similares pero la forma en que tratan ciertos problemas es ligeramente diferente. En primer lugar, los usuarios de sistemas basados en IBE pueden establecer una clave pública como una **cadena arbitraria**, aunque única. Además, la clave puede ser algo **fácil de memorizar** como una dirección de *email*. En segundo lugar, **no hay certificados** que asocien a los usuarios con su clave pública; la cadena de identificación única de un usuario garantiza que ningún otro debe ser capaz de descifrar el contenido cifrado con la clave pública de IBE [5].

Los esquemas criptográficos propuestos hasta ahora que usan este tipo de criptografía se basan en la teoría matemática de los **residuos cuadráticos** o en la de los **emparejamientos bilineales** [13]. Sin embargo, la mayoría de los esquemas criptográficos basados en identidad que son eficientes se basan en los **emparejamientos bilineales sobre curvas elípticas** [3, 17]. Es por ello que a este tipo de criptografía se la conoce también como criptografía basada en emparejamientos (PBC).

La idea de esta criptografía es la construcción de un emparejamiento entre dos grupos criptográficos, reduciendo un problema difícil en un grupo a uno normalmente más fácil en otro grupo. Por ejemplo, en grupos con emparejamientos bilineales como

el de Weil [18] o el de Tate [19], el **problema computacional de Diffie-Hellman** se cree que es imposible, mientras que el problema de **decisional de Diffie-Hellman** puede ser resuelto más fácilmente usando la función de emparejamientos.

3.1.1. Emparejamientos bilineales

Consideramos dos grupos $\mathcal{G}_1$ y $\mathcal{G}_2$ de orden primo q. El grupo $\mathcal{G}_1$ es un grupo cíclico aditivo, mientras que el grupo $\mathcal{G}_2$ es multiplicativo. Generalmente, el grupo $\mathcal{G}_1$ **es un grupo construido sobre una curva elíptica**, es decir, está formado por puntos de una curva elíptica, y el grupo $\mathcal{G}_2$ **es un cuerpo finito** [20].

Un emparejamiento es cualquier aplicación bilineal $e : \mathcal{G}_1 \times \mathcal{G}_1 \to \mathcal{G}_2$ que satisfaga las siguientes tres propiedades:

1. **Bilinealidad:**
$$\forall P, Q \in \mathcal{G}_1, \forall a, b \in \mathbb{Z}_q^*$$
$$e(aP, bQ) = e(P, Q)^{ab}$$

2. **No degeneratividad:** e no transforma todos los puntos de $\mathcal{G}_1 \times \mathcal{G}_1$ en uno solo de $\mathcal{G}_2$.

3. **Eficiencia:** el cálculo de e es eficiente para todos los elementos del dominio.

La construcción de emparejamientos bilineales viene con algunas implicaciones:

Teorema 1: El problema del logaritmo discreto en $\mathcal{G}_1$ no es más difícil que en $\mathcal{G}_2$ [18].

Teorema 2: El problema decisional de Diffie-Hellman es fácil en $\mathcal{G}_1$ [18].

3.1.2. Problema Diffie-Hellman Bilineal

Debido a que el problema decisional de Diffie-Hellman es fácil en $\mathcal{G}_1$ (Teorema 2), no podemos usar DDH para construir criptosistemas en el grupo $\mathcal{G}_1$. En su lugar, la **seguridad de los sistemas IBE** se basa en una variante de la suposición CDH llamada suposición bilineal de Diffie-Hellman (en inglés, Bilinear Diffie-Hellman -BDH-).

Sean dos grupos $\mathcal{G}_1$ y $\mathcal{G}_2$ de orden primo q, $e : \mathcal{G}_1 \times \mathcal{G}_1 \to \mathcal{G}_2$ una aplicación bilineal y P un generador de $\mathcal{G}_1$. El problema BDH en $\langle \mathcal{G}_1, \mathcal{G}_2, e \rangle$ consiste en, dado $\langle P, aP, bP, cP \rangle$ para algún $a, b, c \in \mathbb{Z}_q^*$, calcular $W = e(P, P)^{abc} \in \mathcal{G}_2$, que se asume es difícil de resolver.

3.2. Arquitectura

La sistemas basados en identidad permiten a cualquier usuario generar una clave pública a partir de un valor de identidad conocido, como una cadena ASCII. Una tercera parte confiable, llamada **generador de claves privadas** (en inglés, Private Key Generator -**PKG**-), genera las claves privadas correspondientes a cada identidad [13].

Para operar, la PKG publica su clave pública maestra y guarda en secreto su correspondiente clave privada maestra. Cualquier usuario puede calcular la clave pública correspondiente a una cierta identidad ID combinando la clave pública maestra de la PKG con el valor ID. Para obtener la correspondiente clave privada, la parte autorizada para usar la identidad ID contacta con la PKG, quien usa su clave maestra privada para generar la clave privada para dicha identidad. Los pasos involucrados en este tipo de criptografía se muestran en la Figura 3.1.

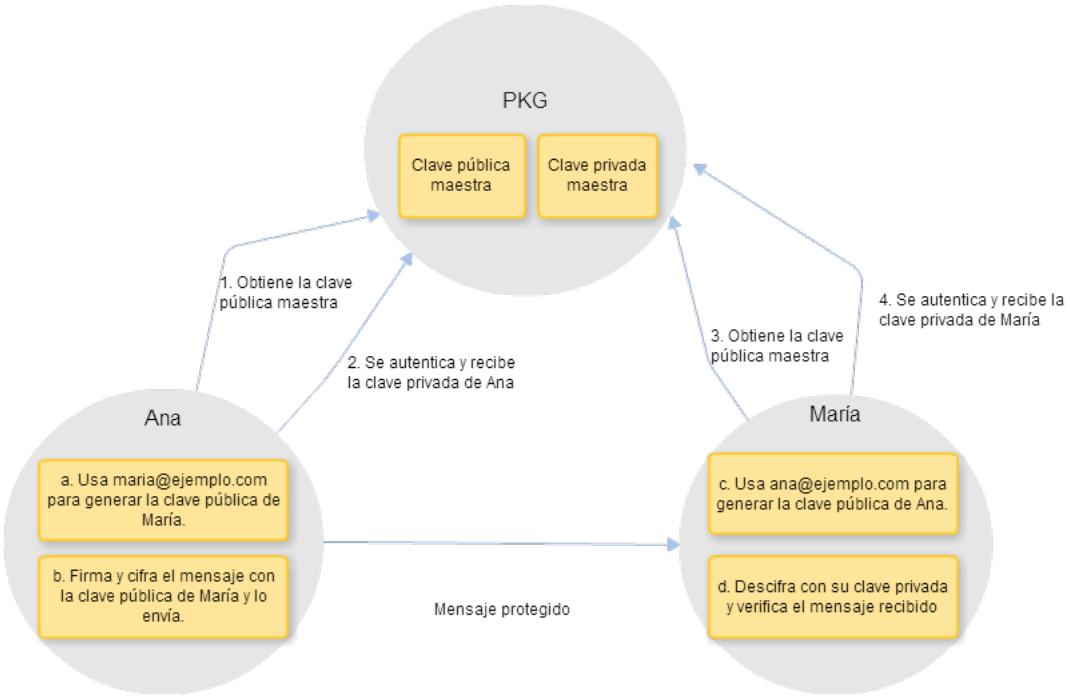


Figura 3.1: Esquema del cifrado basado en identidad.

De esta forma los usuarios pueden cifrar mensajes (o verificar firmas) **sin distribución previa de claves** entre ellos. Esto es extremadamente útil en los casos en los que la pre-distribución de claves autenticadas es imposible de llevar a cabo debido a restricciones tecnológicas. Sin embargo, para poder descifrar o firmar mensajes, el usuario autorizado debe obtener la clave privada apropiada de la PKG [21].

3.2.1. Protocolos IBE

Para realizar las operaciones mencionadas, un sistema completo IBE implementa un conjunto de cuatro algoritmos [18]:

1. **Setup:** Este algoritmo lo ejecuta la PKG una única vez para crear toda la configuración del sistema IBE. La clave privada maestra se guarda en secreto y es usada para derivar las claves privadas de los usuarios, mientras que los parámetros del sistema se hacen públicos. El algoritmo acepta un parámetro de seguridad (la longitud binaria de las claves) y genera el conjunto de parámetros del sistema $\mathcal{P}$, así como el par de claves maestras pública y privada. Publica $\mathcal{P}$, que incluye el espacio del mensaje $\mathcal{M}$ y el espacio de los mensajes cifrados $\mathcal{C}$, así como su clave pública maestra k_m.

2. **Extract:** Este algoritmo lo ejecuta la PKG cuando un usuario pide su clave privada. Recibe los parámetros del sistema $\mathcal{P}$, la clave pública maestra k_m y la identidad $id \in \{0,1\}^*$ del usuario que realiza la petición, y utiliza la clave privada maestra para devolver al usuario su clave privada d.

3. **Encrypt:** El procedimiento es invocado por el emisor para cifrar un mensaje usando una determinada identidad (la del receptor). A partir de los parámetros del sistema públicos $\mathcal{P}$, la identidad del receptor id y el mensaje a cifrar $m \in \mathcal{M}$ genera un mensaje cifrado $c \in \mathcal{C}$.

4. **Decrypt:** Ejecutado por el receptor para descifrar un mensaje usando su correspondiente clave privada. El algoritmo acepta la clave privada del usuario, los parámetros del sistema $\mathcal{P}$, el mensaje a cifrado $c \in \mathcal{C}$ y devuelve el mensaje descifrado $m \in \mathcal{M}$.

Para que todo el sistema tenga sentido y funcione correctamente uno debe postular lo siguiente:
$\forall m \in \mathcal{M}, id \in \{0,1\}^* : Decrypt(Extract(\mathcal{P}, k_m, id), \mathcal{P}, Encrypt(\mathcal{P}, m, id)) = m$

Para una mayor profundización en los algoritmos y en sus fundamentos teóricos consultar [22].

3.2.2. Ventajas

IBE tiene algunas características que no están presentes en la tradicional PKI y que derivan del uso de identidades. En primer lugar, las identidades de los usuarios tienen una forma mucho más atractiva que las claves numéricas. Además, con IBE

es posible cifrar mensajes incluso cuando el receptor no ha generado su par de claves aún, hecho que reduce mucho el ancho de banda necesario y los gastos generales organizativos [5].

Otra ventaja importante es que el **usuario no tiene que custodiar sus credenciales**, pues puede solicitarlas en cualquier momento a la PKG.

En comparación con PKI, la **revocación de claves**, si existe, es mucho más **sencilla**, puesto que cuando la clave privada de algún receptor se vuelve obsoleta o se ha detectado un uso indebido de la misma, el emisor no tiene la necesidad de obtener su nuevo certificado. Incluso si la clave privada cambia, la correspondiente identidad se mantiene, por lo que el emisor puede ser completamente inconsciente de la revocación de la clave privada del receptor.

Como ya se ha mencionado anteriormente, y debido a que las claves públicas de los usuarios son derivadas de identificadores, IBE elimina la necesidad de una infraestructura pública de distribución de claves. Por tanto, **no hay necesidad de distribución de certificados ni de autoridades pesadas**. La autenticidad de las claves públicas está garantizada implícitamente siempre y cuando el transporte de las claves privadas a sus correspondientes usuarios se mantenga seguro [21].

3.2.3. Desventajas

Una de las mayores desventajas de este sistema es que **la PKG debe ser altamente confiable**, ya que es capaz de generar cualquier clave privada de los usuarios y por tanto puede descifrar (o firmar) mensajes sin autorización. Debido a que cualquier clave privada de los usuarios puede ser generada utilizando la clave secreta maestra, el sistema tiene un problema de ***key escrow*** [23], es decir, la PKG actúa como un depósito de claves. Esto no se da en la PKI actual puesto que las claves privadas son normalmente generadas en los ordenadores de los usuarios.

Si la PKG se ve comprometida, todos los mensajes usados por ese servidor durante la vida entera del par de claves pública-privada se ven también comprometidos. Esto convierte a la PKG en un valioso objetivo para atacantes. Para limitar la exposición debido a un servidor PKG comprometido, el par de claves pública-privada maestras puede ser actualizado con un par nuevo de claves independientes. Sin embargo, esto introduce un problema de gestión de claves puesto que todos los usuarios deben tener la clave pública más reciente del servidor.

Por otro lado, se requiere un **canal seguro** entre la PKG y el usuario para transmitir la clave privada de éste. Una conexión SSL es una solución común para un sistema a gran escala. Es importante observar que los usuarios que mantienen co-

nexiones con la PKG deben ser capaces de autenticarse, lo cual podría conseguirse mediante nombre de usuario/contraseña o mediante claves públicas almacenadas en tarjetas inteligentes (*smart cards*).

Finalmente, IBE basa su seguridad en los problemas clásicos de la criptografía asimétrica (el problema del logaritmo discreto), de forma que se vería afectada por ataques llevados a cabo mediante el computador cuántico [24]. Es por ello que, en el contexto determinado por esta tecnología, habría que buscar alternativas basadas en la denominada **criptografía post-cuántica** [25].

3.2.4. Consideraciones adicionales

Los sistemas IBE no pueden ser utilizados para el no repudio puesto que la PKG puede generar todas las claves privadas de los usuarios. Sin embargo, esto hace posible otras propiedades que no son posibles en los sistemas PKI tradicionales, donde el firmante es el único propietario de su clave privada:

* **Esquemas de firmas camaleón**, en el cual las firmas hechas para un cierto destinatario no pueden ser validadas por ningún otro usuario [26].

* Si la PKG gestiona todas las operaciones criptográficas, no se necesita ningún tipo de instalación hardware en el usuario y además se **mejora la usabilidad** del sistema. Por ejemplo, algunas empresas adoptan políticas en las que los mensajes de cierta sensibilidad son automáticamente cifrados o firmados, usando herramientas tales como la búsqueda de palabras clave o expresiones regulares en el contenido de los mensajes [27]. Esta firma automatizada se puede realizar fácilmente mediante IBE.

* Debido a que no son los usuarios los que custodian sus credenciales en sus ordenadores, sino que las solicitan a la PKG, se puede asegurar un **nivel de seguridad más alto** (siempre y cuando la PKG esté debidamente protegida mediante un sistema de bastionado adecuado).

Además de todos los aspectos mencionados, IBE ofrece algunas características interesantes debido a la posibilidad de **codificar información adicional** en el identificador. Por ejemplo, un emisor puede desear especificar una **fecha de expiración** para un cierto mensaje. Así, introduce este *timestamp* en la identidad actual del receptor (posiblemente utilizando algún formato binario como X.509 [28]). Cuando el receptor contacta con la PKG para obtener su clave privada para esa clave pública, la PKG puede evaluar el identificador y declinar la extracción si la fecha de expiración

ha vencido [13]. Generalmente, incrustar datos en el identificador ID corresponde a abrir un canal adicional entre el emisor y la PKG con la autenticidad garantizada por la dependencia de la clave privada con el identificador.

Por otra parte, dependiendo del contexto, el *key escrow* puede ser visto como una característica positiva, por ejemplo dentro de las empresas [21]. Por un lado, permite centralizar el control del ciclo de vida de las claves y esto ayuda a automatizar los procedimientos de rotación de tareas y reasignación de roles en los Sistemas de Gestión de la Seguridad de la Información. Por otro lado, si existen objetivos o restricciones legales/normativas sobre la privacidad de los usuarios de dichos sistemas, entonces *key escrow* es negativo.

Capítulo 4

Análisis

En este capítulo se realiza un análisis del sistema propuesto en el que se detallan los objetivos y el alcance de éste, así como los requisitos funcionales y no funcionales. Además, se incluye el modelo de casos de uso.

4.1. Definición del proyecto

En esta sección se abordan todos los objetivos principales del sistema implementado, el desarrollo de la funcionalidad y el alcance de éste.

4.1.1. Objetivos y funcionalidad

Si bien el principal objetivo de este proyecto es estudiar en profundidad la criptografía basada en emparejamientos, el foco primordial del prototipo desarrollado como resultado de tal estudio es la generación y envío de correo electrónico seguro usando IBE. Así, en lo que sigue se efectuará la creación de un **sistema de correo seguro basado en IBE**, de forma que los correos que puedan ser interceptados por terceros sean ininteligibles. Los usuarios envían y reciben correos de forma segura utilizando el cifrado basado en identidad, para lo cual necesitan contactar con la PKG (servidor externo). Todo ello se realiza de forma transparente al usuario, quien sólo tiene que seleccionar el tipo de cifrado que desea para cada correo que envía.

Así pues, el sistema se divide en **dos aplicaciones independientes** que interactúan la una con la otra y cuya utilización conjunta da sentido al proyecto. Por un lado está la aplicación de correo a la que acceden los usuarios para leer sus correos o enviar nuevos y, por otro lado, está el servidor PKG que atiende las peticiones de los usuarios cuando éstos desean utilizar IBE para cifrar sus mensajes.

La **aplicación de correo** ofrece a un usuario la posibilidad de hacer *login* en el sistema con su correo y su contraseña de *gmail*. En caso de que dicho usuario no

se encuentre registrado deberá crearse una cuenta de *gmail* para poder acceder al sistema.

La **aplicación de la PKG** ofrece a todos los usuarios la posibilidad de utilizar el cifrado basado en identidad para el envío de sus correos. Para ello, es necesario que, durante el envío y la lectura de correos con cifrado IBE, este servidor haya generado su par maestro de claves pública y privada y se encuentre en funcionamiento.

4.1.2. Alcance

Cualquier usuario autenticado en la aplicación de correo podrá acceder a su bandeja de entrada, donde puede visualizar los últimos 20 correos recibidos, así como enviar nuevos correos o responder a los recibidos. A la hora de enviar un correo el usuario podrá elegir el tipo de cifrado con el que desea enviarlo: RSA, IBE o ninguno. De esta forma, la clave de sesión simétrica empleada para el cifrado del cuerpo del mensaje del correo se cifra mediante RSA o IBE de acuerdo a lo seleccionado.

Si un usuario desea utilizar IBE y el servidor externo PKG no está en funcionamiento o no ha generado aún su par de claves pública/privada, no será posible utilizar este tipo de cifrado.

La PKG genera las claves una vez y las debe almacenar, de lo contrario cada vez que recibe una petición sobre una identidad crea un par de claves nuevas que no permiten recuperar claves de sesión previas. Por ello, al generar un nuevo par de claves los mensajes que hayan sido cifrados utilizando otro par ya no podrán descifrarse.

El sistema de correo sólo admite direcciones de *gmail* para autenticarse.

Únicamente los administradores con permiso pueden acceder al servidor PKG para generar su par de claves maestro, iniciarlo o pararlo.

4.2. Catálogo de requisitos

A continuación se describe el catálogo de requisitos del proyecto realizado, dividiéndolos en requisitos funcionales y no funcionales.

4.2.1. Requisitos funcionales

En primer lugar, se muestran los requisitos funcionales de la aplicación de correo, y, en segundo lugar, se detallan los requisitos funcionales del servidor PKG.

Aplicación de correo

RF 1 Para acceder a la aplicación de correo todos los usuarios deberán hacer *login* empleando su usuario y contraseña de *gmail*.

RF 1.1 Para poder acceder es necesario configurar la cuenta de *gmail* desde la página https://www.google.com/settings/security/lesssecureapps para activar la opción "acceso de aplicaciones menos seguras".

RF 1.2 La creación de una cuenta de usuario se hará desde https://mail.google.com/.

RF 2 El usuario podrá salir de la aplicación en cualquier momento.

RF 3 Para cada usuario, el sistema mostrará los 20 últimos correos recibidos, especificando el emisor, el asunto, la fecha y la hora de recepción.

RF 4 Para cada usuario se mostrará el número de mensajes total en la bandeja de entrada.

RF 5 Cada usuario tendrá la opción de visualizar un correo o enviar uno nuevo.

RF 6 Los usuarios podrán enviar nuevos correos escribiendo el correo del destinatario, el asunto (opcional) y el contenido del mensaje y seleccionando el tipo de cifrado (RSA, IBE o ninguno).

RF 6.1 Aunque no es necesario que el destinatario sea una dirección de *gmail* para poder enviar el correo, sí lo es para que éste pueda visualizar correctamente los mensajes, pues para ello deberá acceder al sistema de correo y por tanto tener una dirección de *gmail*.

RF 7 Sólo se podrán visualizar los correos con cifrado IBE si el servidor PKG está iniciado.

RF 8 Al visualizar un correo se mostrarán sus detalles, que incluyen el correo del emisor, el asunto y el contenido. Además, se ofrece la posibilidad de responder directamente al correo que se está visualizando o volver a la bandeja de entrada.

RF 8.1 El usuario podrá responder a un mensaje rellenando únicamente el asunto (opcional), el tipo de cifrado y el contenido del mensaje, pues la dirección del destinatario será la del emisor del correo visualizado.

Aplicación PKG

RF 1 Únicamente los administradores con acceso podrán acceder a la aplicación de la PKG.

RF 1.1 No se puede registrar ningún usuario nuevo.

RF 2 El usuario podrá salir de la aplicación en cualquier momento.

RF 3 Cada administrador tiene la opción de generar un nuevo par de claves o iniciar/parar el servidor.

RF 3.1 Si el par de claves no ha sido generado en ningún momento anterior al acceso a la aplicación, es necesario que sean generadas antes de iniciar el servidor.

RF 3.2 Sólo se podrá parar el servidor una vez éste haya sido iniciado.

RF 3.3 Si se genera un nuevo par de claves se sobreescribe el existente, por lo que no se podrán descifrar los mensajes cifrados con claves privadas generadas con el par maestro anterior.

4.2.2. Requisitos no funcionales

El sistema debe cumplir ciertas medidas de seguridad, eficiencia y usabilidad.

Seguridad

RNF 1 Los datos entre los usuarios y la PKG se envían mediante conexión segura SSL.

RNF 2 Los correos que se envían los usuarios mediante la aplicación de correo se envían mediante conexión segura SSL autenticada.

RNF 3 Se utiliza una función *hash* para almacenar las contraseñas de los administradores que acceden a la PKG.

RNF 4 Todos los usuarios deben identificarse en el sistema con su correo y su contraseña para poder acceder a la aplicación. Asimismo, todos los administradores deben identificarse con su usuario y contraseña en el servidor PKG.

Eficiencia y rendimiento

RNF 5 El tiempo en cifrar y enviar los correos mediante cifrado RSA no será mayor de 4 segundos.

RNF 6 El tiempo en cifrar y enviar los correos mediante cifrado IBE no será mayor de 5 segundos.

RNF 7 El tiempo en cifrar y enviar los correos sin cifrado no será mayor de 3 segundos.

RNF 8 El tiempo en acceder a la bandeja de entrada al iniciar la aplicación de correo no será mayor de 6 segundos.

RNF 9 El tiempo en generar el par de claves la aplicación de la PKG no será mayor de 4 segundos.

RNF 10 El tiempo en descifrar los correos cifrados con RSA no será mayor de 1 segundo.

RNF 11 El tiempo en descifrar los correos cifrados con IBE no será mayor de 1 segundo

Usabilidad e interfaz de usuario

RNF 12 Ambas interfaces siguen el mismo patrón gráfico en toda la aplicación de forma que se garantiza un alto grado de consistencia y mayor comodidad del usuario.

RNF 13 La aplicación de correo mantiene informado al usuario, mostrando un mensaje de confirmación tras el envío correcto de un correo y un mensaje de aviso cuando se produce algún error. Además, el usuario puede, en cualquier momento, volver al menú principal (bandeja de entrada).

RNF 14 La aplicación de PKG mantiene informado al administrador, mostrando un mensaje de confirmación tras la generación de claves y un mensaje de aviso cuando se produce algún error.

Recursos

RNF 15 Cada usuario puede visualizar hasta un máximo de 20 correos en la bandeja de entrada, los últimos recibidos.

Mantenimiento

RNF 16 La aplicación sigue una estructura modular por lo que el código está dividido en distintos paquetes según su funcionalidad, comenzando por *es.uam.eps.tgf_laura*.

RNF 17 Con el fin de facilitar el mantenimiento y la integración con otros sistemas ambas aplicaciones están documentadas detalladamente.

4.3. Casos de uso

Se muestra a continuación en la Figura 4.1 la secuencia de interacciones que se desarrollan entre el sistema y sus actores en respuesta a un evento que inicia un actor principal sobre el propio sistema. De esta forma, se ilustran los requerimientos del sistema ya mencionados.

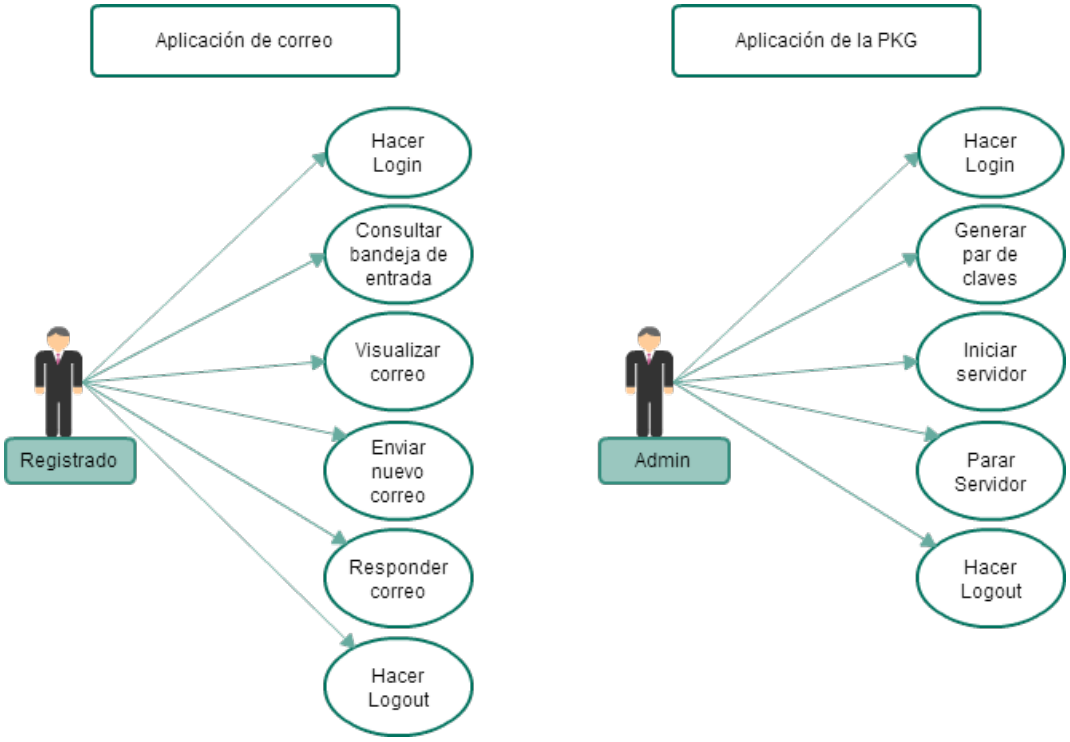


Figura 4.1: Modelo de casos de uso.

En la Tabla 4.1 se muestra el caso de uso de la aplicación de correo más relevante, que es el del **envío de un nuevo correo** (equivalente a responder a un correo que se está visualizando).

Enviar un correo	
Actores	Usuario registrado
Descripción	
El usuario introduce un correo destinatario y/o un asunto, selecciona un tipo de cifrado y escribe el contenido del mensaje que desea enviar.	
Precondiciones	El usuario ha accedido al sistema
Postcondiciones	Se envía el correo al destinatario especificado con los parámetros deseados por el usuario

Tabla 4.1: Caso de uso aplicación de correo.

En la Tabla 4.2 se observa el caso de uso de la aplicación de la PKG más importante, que es el de **iniciar el servidor**.

Iniciar servidor	
Actores	Administrador registrado
Descripción	
El administrador selecciona la opción de inicio del servidor para atender las peticiones de los usuarios.	
Precondiciones	El administrador ha accedido al sistema
Postcondiciones	Los usuarios comienzan a enviar peticiones a la PKG para poder cifrar sus correos con IBE

Tabla 4.2: Caso de uso aplicación de la PKG.

Capítulo 5

Diseño

En este capítulo se describe el diseño del sistema implementado, especificando los componentes del sistema que dan respuesta a las funcionalidades descritas en el capítulo anterior. Se ha realizado en base a diagramas que permiten describir las interacciones entre las distintas entidades.

5.1. Arquitectura del sistema

Por un lado, tenemos la arquitectura **cliente/servidor** entre ambas aplicaciones: la de correo como cliente y la de la PKG como servidor. De esta forma, los clientes o usuarios realizan peticiones a la PKG, que se mantiene en espera hasta que las recibe. Estas peticiones se tratan concretamente del envío de claves públicas y privadas de IBE.

Por otro lado, sin embargo, el control del sistema de correo está determinado por el **Modelo-Vista-Controlador** (MVC) que separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones. Así, se han construido tres componentes distintos que interactúan entre sí: el modelo, la vista y el controlador [29].

El **modelo** es la representación de la información con la que el sistema opera, y envía a la "vista" la información que el usuario solicita para que sea mostrada. Estas peticiones de acceso o manipulación de los datos llegan al "modelo" a través del "controlador".

La **vista** es generalmente la interfaz de usuario, pues presenta la lógica de negocio y los datos (el "modelo") en un formato adecuado para interactuar con el usuario.

El **controlador** hace de intermediario entre la "vista" y el "modelo" pues responde a eventos (acciones del usuario) e invoca peticiones al "modelo" cuando se hace alguna solicitud a la información.

5.2. Entorno tecnológico

Los equipos sobre los que se ejecutan las aplicaciones deben cumplir unos requisitos para el correcto funcionamiento del sistema de correo.

En primer lugar, el equipo en el que se ejecute la aplicación de correo debe tener **acceso a Internet**, pues es necesaria la conexión con el servidor de *gmail*. Además, tanto para la aplicación de correo como para la de la PKG se recomienda la utilización de un equipo con un mínimo de **500MHz de procesador** y **256MB de RAM**[1]. Sin embargo, también se puede trabajar con equipos de menores prestaciones usando versiones optimizadas de la biblioteca PBC original.

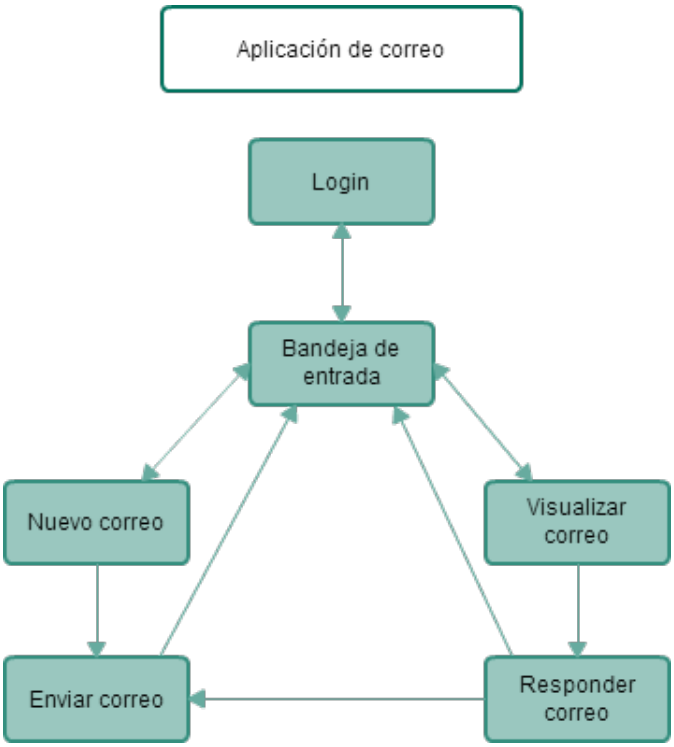


Figura 5.1: Flujos de operación correo.

5.3. Descripción de los flujos de operación e interacción

La aplicación de correo está dividida en 4 actividades principales: el *login*, el acceso a la bandeja de entrada, el envío de un nuevo correo y la lectura de un determinado correo.

[1]Requisitos mínimos propuestos por algunos trabajos como `https://eprint.iacr.org/2011/668.pdf`.

La Figura 5.1 muestra los flujos que se producen entre ellas, comenzando por la actividad de ***login***, que permite al usuario autenticado el acceso a la aplicación. Una vez autenticado el usuario con su correo y contraseña de *gmail*, éste accede a la **bandeja de entrada**, que nos muestra en una lista los 20 últimos correos recibidos. Desde esta actividad tiene la posibilidad de volver a la del *login*, si desea salir de la aplicación, y de **visualizar** uno de los correos recibidos.

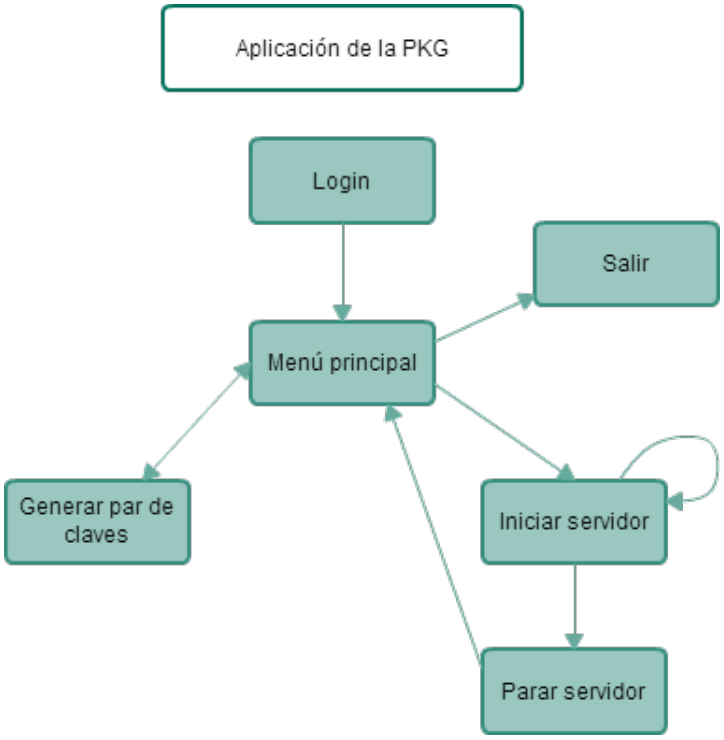


Figura 5.2: Flujos de operación PKG.

Si se selecciona uno de los correos de la lista, se muestra su contenido, incluyendo el emisor, el asunto y el mensaje, así como la posibilidad de **responder** directamente a este mensaje. En este último caso, el correo del destinatario se introduce automáticamente al enviar el correo de respuesta, tras lo cual la aplicación vuelve a mostrar la bandeja de entrada. Desde la bandeja de entrada el usuario puede también redactar y **enviar** un nuevo correo a cualquier destinatario que desee, volviendo después a la bandeja de entrada. Además, en todas las actividades existe la posibilidad de volver hacia atrás.

La aplicación de la PKG tiene a su vez 4 actividades: el *login*, la generación de claves, el inicio del servidor y la parada del servidor. La Figura 5.2 muestra los flujos que se producen entre ellas, comenzando por la actividad de ***login***, que permite al administrador autenticado acceder al menú principal, desde donde podrá realizar las

restantes 3 actividades.

Una vez que el administrador haya accedido al menú principal, éste tiene la posibilidad de **generar un nuevo par de claves** para la PKG, tras lo cual vuelve a mostrarse el menú principal. Sin embargo, el administrador puede **iniciar el servidor** directamente sin necesidad de generar nuevas claves si éstas ya estaban generadas previamente. Una vez iniciado, se puede **parar el servidor** en cualquier momento, volviendo al panel inicial.

Por otro lado, la Figura 5.3 muestra la **interacción** que se desarrolla entre la aplicación de correo (Usuario 1 y Usuario 2) y la aplicación de la PKG. Esta interacción se realiza mediante una conexión segura SSL para proteger los datos que se transmiten entre ellos.

Asimismo, muestra la interacción entre dos usuarios de la aplicación de correo cuando el Usuario 1 envía un mensaje cifrado mediante IBE[2] al Usuario 2, para lo cual también se utiliza una conexión segura (conexión SSL establecida por el servidor de correos *gmail*).

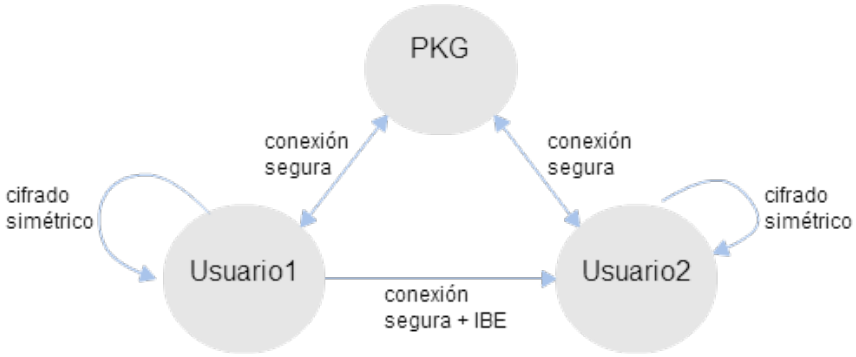


Figura 5.3: Interacción básica entre componentes.

[2]De modo estricto, lo que se envía es el mensaje cifrado con AES y la clave de sesión correspondiente cifrada con la clave IBE pública del destinatario.

Capítulo 6

Implementación

En este capítulo se describen todos los detalles relacionados con la implementación del sistema de correo desarrollado. Esto incluye las descripciones de las características del equipo de desarrollo, de la estructura del código y de la biblioteca JPBC utilizada. La interfaz desarrollada se detalla en el Anexo C

Con objeto de mostrar una aplicación concreta de la criptografía basada en emparejamientos, se ha empleado la biblioteca criptográfica JPBC para crear un cliente de correo que cifra los mensajes mediante cifrado simétrico y lo envía como fichero adjunto en el correo. La clave empleada para esta operación se genera de forma aleatoria y se le envía al destinatario cifrada con su clave pública en ese mismo fichero. El tipo de cifrado asimétrico empleada para tal operación puede ser RSA o IBE y es especificada en el cuerpo del mensaje.

6.1. Equipo de desarrollo

Se detallan a continuación las características hardware y software del equipo sobre el que se ha llevado a cabo el desarrollo.

6.1.1. Hardware

El proyecto completo se ha desarrollado en un equipo con las siguientes características:

Equipo portátil HP Pavilion g series con procesador Intel Core I3 de 2.40GHz, 4GB de memoria RAM, disco duro principal de 451GB y sistema operativo Windows 7 Home Premium.

6.1.2. Software

Se ha hecho uso de una serie de aplicaciones y herramientas para la codificación y el diseño de las dos aplicaciones que componen el sistema de correo implementado.

Software de programación

Las herramientas utilizadas para la codificación han sido las siguientes:

* Eclipse Luna para realizar toda la codificación.
* Control de versiones GitHub.

Software de edición

Para el la creación de diagramas se han utilizado las siguientes herramientas:

* CACOO
* Dia
* Microsoft Office Picture Manager
* Paint

Para redactar el presente documento se ha utilizado LaTeX.

Para la gestión de referencias consultadas se ha utilizado el gestor bibliográfico Mendeley [30].

6.2. Plataformas

Se han utilizado diversas tecnologías y mecanismos para implementar las distintas funcionalidades que conforman el sistema de correo. Las plataformas empleadas en el desarrollo del proyecto son las siguientes:

6.2.1. Java

Java es un lenguaje de programación orientado a objetos independiente de la plataforma. Se utiliza especialmente en las aplicaciones cliente/servidor debido a la facilidad que presenta para la comunicación entre dispositivos.

Todo el proyecto se ha desarrollado en Java, además de hacer uso de la API gráfica ***Swing***, que es la biblioteca para la interfaz gráfica de usuarios avanzada de la plataforma Java *Standard Edition (Java SE)*. También se ha hecho uso del *framework* de

Java para criptografía, el cual se basa en las APIs (*Applicacion Programming Interface*) **JCA** (*Java Cryptography Architecture*) y JCE (*Java Cryptography Extension*). Este par de APIs proporcionan el conjunto de interfaces que un proveedor de servicios criptográficos debe implementar para permitir la generación de claves y el cifrado simétrico y asimétrico. Si bien JCE incluye un proveedor de servicios criptográficos, en determinados contextos es preferible utilizar proveedores alternativos para alcanzar un mayor grado de seguridad. En el caso de la biblioteca JPBC se utiliza el proveedor de servicios *Bouncy Castle*[1].

Para realizar el cifrado asimétrico basado en identidad se ha utilizado la biblioteca criptográfica **JPBC** [4]. En particular, se ha partido de la implementación del criptosistema descrito en [31]. En lo que sigue, se utilizará la misma notación empleada en JPBC para las clases que referencian un cierto critposistema. Así, en el caso de las clases vinculadas a [31] los nombres de las mismas contendrán la cadena "AHIBEDIP10". El acrónimo "AHI" define el esquema general de cifrado (*Anonymous Hierarchical Identity Based Encryption*), mientras que "DIP10" son las siglas de los apellidos de los creadores del criptosistema (De Caro, Iovino, Persiano) y el año de su publicación (es decir, el 2010). Se ha utilizado este criptosistema porque incorpora el encapsulamiento de claves simétricas mediante IBE.

6.2.2. OpenSSL

OpenSSL es un proyecto de software libre que implementa los protocolos de **SSL** y **TLS**. Se puede considerar también como una biblioteca criptográfica de propósito general. En el desarrollo del proyecto se ha utilizado para la **creación de las claves pública y privada de RSA**, que utilizan los usuarios para enviar los correos cifrados mediante RSA. Para ver cómo se han generado dichas claves consultar el Anexo B.

6.2.3. Keytool

De modo alternativo a OpenSSL, se pueden crear credenciales X.509 mediante la herramienta Keytool incluida en el paquete Java SE. Keytool es una herramienta de gestión de claves y certificados que permite a los usuarios administrar su par de claves pública/privada y, asimismo, permite firmar certificados propios dando origen a lo que se suele denominar como **certificados autofirmados**. Sin embargo, ha de tenerse en cuenta que el uso de certificado autofirmados no es una buena práctica de seguridad, pues una infraestructura de clave pública requiere la utilización de autoridades de certificación confiables en el sentido más amplio. No obstante, en dominios restringidos

[1] `http://bouncycastle.org/`

de confianza, como es el caso del entorno de pruebas, se puede hacer uso de este tipo de certificados. En la medida que el desarrollo de este TFG corresponde a esta situación, en este proyecto se ha utilizado Keytool para el **manejo de certificados** (i.e., creación del certificado del servidor y exportación/importación de su clave pública) para habilitar la comunicación **SSL** (*Secure Socket Layer*, comunicación segura por https) **entre el servidor PKG y sus clientes**.

SSL permite, una vez esté habilitado, que todas las comunicaciones entre el servidor y el cliente estén cifradas, de forma que terceros no puedan entender los datos que viajan del cliente al servidor y viceversa. En este caso, el cliente son los usuarios que desean realizar peticiones de claves al servidor PKG. Para ver cómo se han generado los certificados necesarios consultar el Anexo B.

6.3. Estructura y documentación del código

Tanto en la aplicación de correo como en la aplicación de la PKG se han dividido las clases en distintos paquetes según la funcionalidad de cada una de ellas. Para la aplicación de correo, todos estos paquetes comienzan por *es.uam.eps.tfg_laura* mientras que para la aplicación de la PKG los paquetes comienzan por *uam.eps.pkg_tfg*. Es decir, se ha seguido una **notación de dominio inverso**.

6.3.1. Aplicación de correo

A continuación se describen los paquetes en los que se ha dividido la aplicación de correo. Debajo de cada uno de los paquetes se describen las clases que contiene cada uno de ellos como sigue:

es.uam.eps.tfg_laura.crypto. Este paquete contiene todas las clases relacionadas con criptografía, es decir, todos las implementaciones de cifrados y descifrados, tanto de IBE como de RSA.

AHIBEDIP10KeyPairGeneratorPBC.java: Contiene los métodos necesarios para generar la clave pública de la PKG a partir de unos ciertos parámetros públicos que ésta envía al usuario. Esta clase modifica la clase AHIBEDIP10KeyPairGenerator.java original de la biblioteca JPBC, en la que se encuentra implementado el método de generación del par público/privado maestro de la PKG. El procedimiento de generación de claves corresponde al criptosistema definido en [31].

KEMCipherAHIBEDIP10KEM.java: Contiene los métodos para que el usuario pueda realizar el *setup* de la clave pública que le envía la PKG, utilizando AHIBEDIP10KeyPairGeneratorPBC. Esta clase modifica la clase KEMCipherAHIBEDIP10.java de la JPBC para poder incorporar funcionalidad de *setup* mencionada. También contiene los métodos necesarios para realizar el cifrado mediante IBE de cualquier mensaje.

Symmetric.java: Contiene los métodos necesarios para manejar el cifrado de clave simétrico, que se utiliza para generar una clave AES con la que se cifrarán todos los correos que se envíen entre usuarios. Así, contiene exclusivamente los métodos necesarios para generar una clave AES y cifrar y descifrar mensajes con ella.

Asymmetric.java: Esta clase contiene los métodos para el cifrado asimétrico, tanto para RSA como para IBE. Por un lado, implementa métodos para cifrar ficheros (que contienen la clave AES generada con Symmetric) con la clave pública de RSA y para descifrarlos con la clave privada de RSA. Por otro lado, contiene los métodos que el usuario utiliza para obtener la clave pública de la PKG y su propia clave privada. Además, llama a los métodos necesarios de KEMCipherAHIBEDIP10KEM para realizar el cifrado y el descifrado de la clave AES mediante IBE.

es.uam.eps.tfg_laura.files. Este paquete contiene la creación de los ficheros adjuntos que se envían los usuarios cuando envían un correo desde la aplicación. Contiene también la interpretación de dichos ficheros en el receptor del correo.

FileParser.java: Contiene un método que, a partir del mensaje redactado en el correo, crea el fichero que se va a enviar como adjunto en el correo. El fichero tiene una determinada estructura que es interpretada adecuadamente por el usuario receptor. En primer lugar, se escribe un *byte* indicando si el mensaje está cifrado o no. Si el correo no se envía cifrado, tras el indicador se escribe el mensaje. Si el correo se envía cifrado, tras el indicador se escribe la longitud del mensaje cifrado con AES, seguido del propio mensaje cifrado. Finalmente se escribe la clave AES cifrada mediante IBE o RSA. Para ello utiliza los métodos del paquete *es.uam.eps.tfg_laura.crypto*.

es.uam.eps.tfg_laura.gui. Este paquete contiene todas las clases relacionadas con la interfaz de usuario de la aplicación de correo. Estas clases se identifican con el diagrama de flujos de operación de la Figura 5.1.

Gui.java: Establece las propiedades generales de toda la interfaz y crea el primer panel (InitialPanel).

InitialPanel.java: Panel de inicio de la aplicación que contiene el *login* del usuario.

MailPanel.java: Panel que muestra en una tabla los 20 últimos mensajes recibidos de la bandeja de entrada de un usuario. Permite seleccionar uno cualquiera para poder visualizarlo, así como enviar un nuevo correo.

SendMailPanel.java: Panel para el envío de un nuevo correo (o respuesta a uno que se esté visualizando). Permite introducir el destinatario, asunto (opcional), el tipo de cifrado con el que se va a enviar y el contenido del mensaje. El controlador llama a los métodos de la clase Mail para enviar el correo con un adjunto, que ha sido creado previamente usando la clase FileParser.

ReceiveMailPanel.java: Panel que muestra un correo recibido que ha sido seleccionado en la bandeja de entrada. Indica el emisor del correo, el asunto y el mensaje en sí, y permite contestar directamente a dicho emisor. El controlador llama a los métodos de la clase Mail para recibir el correo con un adjunto, que ha sido interpretado previamente usando la clase FileParser.

es.uam.eps.tfg_laura.mail. Este paquete contiene todas las clases relacionadas con el envío y recepción de correos mediante el servidor de *gmail* así como el main de la aplicación.

Main.java: Inicia la interfaz de usuario y con ello toda la aplicación.

Mail.java: Contiene los dos métodos principales para el envío y la recepción de correos. Hace uso de FileParser para crear/interpretar el fichero con los datos cifrados así como de SSLEmail para enviar/recibir ese fichero como adjunto en un correo.

SSLEmail.java: Con la ayuda de EmailUtil, envía los correos con los adjuntos generados previamente. Utiliza el protocolo SMTP para iniciar una sesión segura entre el usuario que desea enviar el mensaje y el servidor de *gmail*.

GmailInbox.java: Se conecta mediante SMTP y de forma segura al servidor de *gmail* para mostrar un determinado correo seleccionado en la bandeja de entrada. Con la ayuda de FileParser, procesa el adjunto del correo, que es donde se encuentran los datos cifrados, para que el receptor visualice correctamente el correo.

EmailUtil.java: Contiene los métodos de envío de correos simples, con adjunto o con imágenes. Establece todos los parámetros necesarios en el *email* y lo transporta utilizando la sesión creada en SSLEmail.

es.uam.eps.tfg_laura.sockets. Este paquete contiene la creación y el manejo de un *socket* SSL por parte del usuario para poder comunicarse con la PKG.

UserSocket.java: Crea un *socket* SSL que escucha en el mismo puerto que el de la PKG y contiene los métodos necesarios para el envío y la recepción de datos a través de ese *socket*.

es.uam.eps.tfg_laura.test. Este paquete contiene clases que han servido para realizar pruebas sobre ciertos componentes a lo largo del proyecto. Además, contiene las clases que han generado las estadísticas de medición de tiempos.

6.3.2. Aplicación de la PKG

A continuación se describen los paquetes en los que se ha dividido la aplicación de la PKG:

es.uam.eps.pkg_tfg.crypto. Este paquete contiene todas las clases relacionadas con criptografía, es decir, toda la funcionalidad para el cifrado y descifrado mediante IBE.

AHIBEDIP10KeyPairGeneratorPKG.java: Esta clase modifica la clase AHIBEDIP10KeyPairGenerator.java original de la librería JPBC, en la que se encuentra implementado el método de generación del par público/privado maestro de la PKG, para poder almacenar el par de claves en un fichero.

KEMCipherAHIBEDIP10KEM.java: Contiene los métodos para que la PKG pueda realizar el *setup* de su par de claves, utilizando para ello AHIBEDIP10KeyPairGeneratorPKG. Esta clase modifica la clase KEMCipherAHIBEDIP10.java de la JPBC para poder incorporar la funcionalidad de *setup* a partir de parámetros almacenados en un fichero. De esta forma, no es necesario generar un nuevo par de claves cada vez que se inicie la PKG, pues podrá leer los parámetros necesarios para recuperar el par de claves. Contiene también los métodos necesarios para generar la clave privada de cada usuario que la solicite.

PKG.java: Permite generar un nuevo par de claves (*setup*), cargar los parámetros necesarios para recuperar el par de clave pública/privada de la última vez que se generó y obtener la clave privada de un determinado usuario. Para ello hace uso de la clase KEMCipherAHIBEDIP10KEM.

HashText.java: Genera el fichero con todos los administradores y sus contraseñas utilizando la función *hash* SHA-1.

es.uam.eps.pkg_tfg.gui. Este paquete contiene todas las clases relacionadas con la interfaz de usuario de la aplicación de la PKG. Estas clases se identifican con el diagrama de flujos de operación de la Figura 5.2.

Gui.java: Establece las propiedades generales de toda la interfaz y crea el primer panel (Login).

Login.java: Panel de inicio de la aplicación que contiene el *login* del administrador.

MainPanel.java: Menú principal de la aplicación desde la cual se puede generar un nuevo par de claves para la PKG, poner el servidor en funcionamiento o pararlo.

es.uam.eps.pkg_tfg.main. Este paquete contiene únicamente la clase Main.java que inicia la interfaz de usuario y con ello toda la aplicación.

es.uam.eps.pkg_tfg.sockets. Este paquete contiene la creación y el manejo de un *socket* SSL por parte de la PKG, que se mantiene en espera de aceptar nuevas peticiones de los usuarios.

PKGSocket.java: Crea un *socket* SSL que escucha en el puerto 1240 a la espera de recibir peticiones de los usuarios que desean enviar/recibir correos cifrados mediante IBE. Así, esta clase contiene los métodos necesarios para el envío y la recepción de datos a través del *socket*.

es.uam.eps.pkg_tfg.test. Este paquete contiene clases que han servido para realizar pruebas sobre ciertos componentes a lo largo del proyecto. Además, contiene la clase Statistics.java que se ha utilizado para medir el tiempo de generación del par de claves por parte de la PKG.

En cuanto a la **documentación del código**, todas las clases y métodos se encuentran detalladamente comentadas para poder facilitar el mantenimiento y la reutilización del código.

6.4. Biblioteca JPBC

La biblioteca Java Pairing-Based Cryptography (JPBC) [4] es una versión en Java de la biblioteca Pairing-Based Cryptography (PBC) desarrollada por Ben Lynn

[32] en C para realizar las operaciones matemáticas subyacentes en los criptosistemas basados en emparejamientos (en inglés, pairings). De esta forma, facilita rutinas como la generación de curvas elípticas, aritmética en curvas elípticas y computación de emparejamientos.

Cualquier aplicación que desee utilizar la biblioteca PBC debe primero inicializar un objeto *pairing*. Esto lleva a que se deben configurar curvas elípticas, grupos y otros objetos matemáticos. Después, los elementos pueden ser inicializados y manipulados para operaciones criptográficas. Para la creación de los *pairings* se necesita una serie de parámetros de distintos tipos (A, A1, D, E, F, G). Cada uno de ellos indica el tipo de curva y cuerpo sobre el que se ha construido el *pairing*. La biblioteca PBC incluye parámetros para ciertos emparejamientos, de los cuales algunos son apropiados para uso criptográfico.

La biblioteca JPBC proporciona además la implementación para algunos criptosistemas en el contexto del *framework* Bouncy Castle [33]. Ninguno de ellos posee la funcionalidad completa de este proyecto, pero han ayudado a la comprensión y manejo de la biblioteca. El **criptosistema DIP10**, llamado así por las iniciales de los autores y basado en *Fully Secure Anonymous HIBE and Secret-Key Anonymous IBE with Short Ciphertexts* [31], se ha utilizado como base de partida para la implementación.

JPBC implementa generadores de parámetros para *pairings* tanto para aplicaciones bilineales (basadas en curvas elípticas) como para aplicaciones multilineales (aplicaciones lineales de más de dos variables). Algunos parámetros pre-generados se pueden encontrar en una de las carpetas del proyecto, y son utilizados por los distintos criptosistemas implementados.

La biblioteca JPBC está dividida según la funcionalidad en una serie de paquetes:

JPBC_api Este paquete implementa las interfaces, los objetos y las operaciones más básicas de este tipo de criptografía: *Element*, *ElementPow*, *Pairing*, *PairingParameters*, *Field*, *Vector*... Es el único paquete de toda la biblioteca que tiene documentación *javadoc* [4].

JPBC_crypto En este paquete se encuentran todas las operaciones criptográficas para los distintos criptosistemas proporcionados, así como pruebas y algunos ejemplos de uso de cada uno de ellos.

JPBC_mm Este paquete contiene la implementación necesaria para la generación de parámetros para aplicaciones multilineales.

JPBC_plaf Este paquete contiene la implementación de distintos tipos de *pairings* de acuerdo a cada uno de los tipos de parámetros (A, A1, D, E, F, G). Estos

métodos son la base de la generación de parámetros que se utilizan para generar el par de claves público/privada de la PKG, así como las claves privadas de los usuarios.

JPBC_test Este paquete contiene pruebas unitarias para las operaciones entre elementos y la generación de todo tipo de *pairings*, con distintos tipos de parámetros.

JPBC suministra una clase de soporte llamada **KEMCipher**, que simplifica la integración de un KEM con un Cipher. **KEM** (*Key Encapsulation Mechanisms*) consiste en una serie de técnicas de cifrado diseñadas para proporcionar seguridad a la transmisión de claves simétricas usando algoritmos de clave asimétrica (clave pública). En la práctica, los sistemas de clave pública son muy pesados para usarlos en la transmisión de mensajes largos. En su lugar, se utilizan para intercambiar las claves simétricas, que son relativamente cortas. Así, la clave simétrica es la que se usa para cifrar el mensaje largo.

Además, la JPBC proporciona también un ejemplo de cómo usar KEMCipher, a partir del cual se ha implementado **KEMCipherAHIBEDIP10KEM.java**. En la **aplicación de correo** esta clase contiene los métodos para generar una clave simétrica (AES 256) y cifrarla utilizando la criptografía basada en emparejamientos. Asimismo, es posible cifrar y descifrar una cadena con dicha clave simétrica y generar la clave pública de la PKG a partir de ciertos parámetros públicos. Para la **aplicación de la PKG** esta clase contiene los métodos necesarios para generar el par de claves maestro pública/privada.

Para poder permitir que el usuario, desde la aplicación de correo, construya la clave pública de la PKG a partir de los parámetros que ésta le envía, ha sido necesario modificar el generador de claves del criptosistema DIP10. La clase **AHIBEDIP10KeyPairGenerator** ha sido modificada en la **aplicación de la PKG** (AHIBEDIP10KeyPairGeneratorPKG) para que ésta pueda recuperar su par maestro a partir de un fichero guardado durante la última generación de claves. Por otro lado, esta misma clase ha sido modificada en la **aplicación de correo** (AHIBEDIP10KeyPairGeneratorPBC) para que los usuarios puedan construir exclusivamente la clave pública de la PKG (y no su clave privada) a partir de una serie de parámetros públicos.

6.5. Codificación

En esta sección se van a detallar los **flujos de datos que se transmiten entre** los distintos **componentes** de la aplicación de correo, y entre la aplicación de correo y la PKG. El grueso del proyecto consiste en la implementación del cifrado IBE para el envío de correos entre usuarios, con la intervención de la PKG. Sin embargo, adicionalmente se ha implementado el sistema de forma que también es posible enviar correos utilizando el cifrado RSA.

Como ya se ha mencionado anteriormente, se ha utilizado OpenSSL (ver Anexo B) para generar los certificados para la clave pública y privada de RSA que utilizarán todos los usuarios que deseen enviar **correos cifrados con RSA**. Así pues, cada usuario posee su propia clave pública y su clave privada para el cifrado y descifrado de mensajes. Cuando un usuario quiere enviar un mensaje a otro, en primer lugar habrá de obtener la clave pública de dicho destinatario. Para ello, a petición del emisor, el destinatario envía por correo electrónico su certificado con la clave pública. En el entorno de pruebas sólo se han generados dos certificados de usuario, y cada uno de ellos es autofirmado por el propietario de las claves. En un sistema real todos los usuarios dispondrán de certificados digitales distintos y firmados por autoridad certificadora confiable (p. ej., la Fábrica de Moneda y Timbre de España). Los pasos que sigue cada uno de los componentes (siempre transparente al usuario final que utiliza el sistema de correo) se pueden observar en la Figura 6.1.

En la figura, una vez generados los pares de claves RSA, si Ana desea enviar un mensaje a María debe obtener, en primer lugar, su clave pública. Asumiendo que Ana tiene dicha clave, cuando Ana desea enviar un mensaje cifrado genera una clave AES y la cifra utilizando la clave pública RSA de Maria. Para ello, utiliza las clases Symmetric y Asymmetric. El mensaje que se desea enviar es cifrado con la clave simétrica generada, y se envía todo junto en un fichero como adjunto del correo, siguiendo el formato explicado en la Figura 6.5. Para crear tal fichero utiliza la clase FileParser y para enviarlo como adjunto las clases del paquete *es.uam.eps.tfg_laura.mail*. En el cuerpo del correo se escribe el tipo de cifrado que se ha realizado sobre el mensaje (IBE, RSA o None). Una vez que el usuario receptor (María) selecciona dicho correo de su bandeja de entrada, la aplicación lee el cuerpo del correo ("RSA") para interpretar el fichero adjunto correctamente. Así, se utiliza la clave privada RSA de María para descifrar la clave AES contenida en el fichero recibido. Con esta clave María ya puede descifrar el contenido del mensaje y la aplicación se encarga de mostrarle correctamente el correo. De nuevo, se han utilizado las clases Symmetric, Asymmetric, FileParser y las del paquete *mail*.

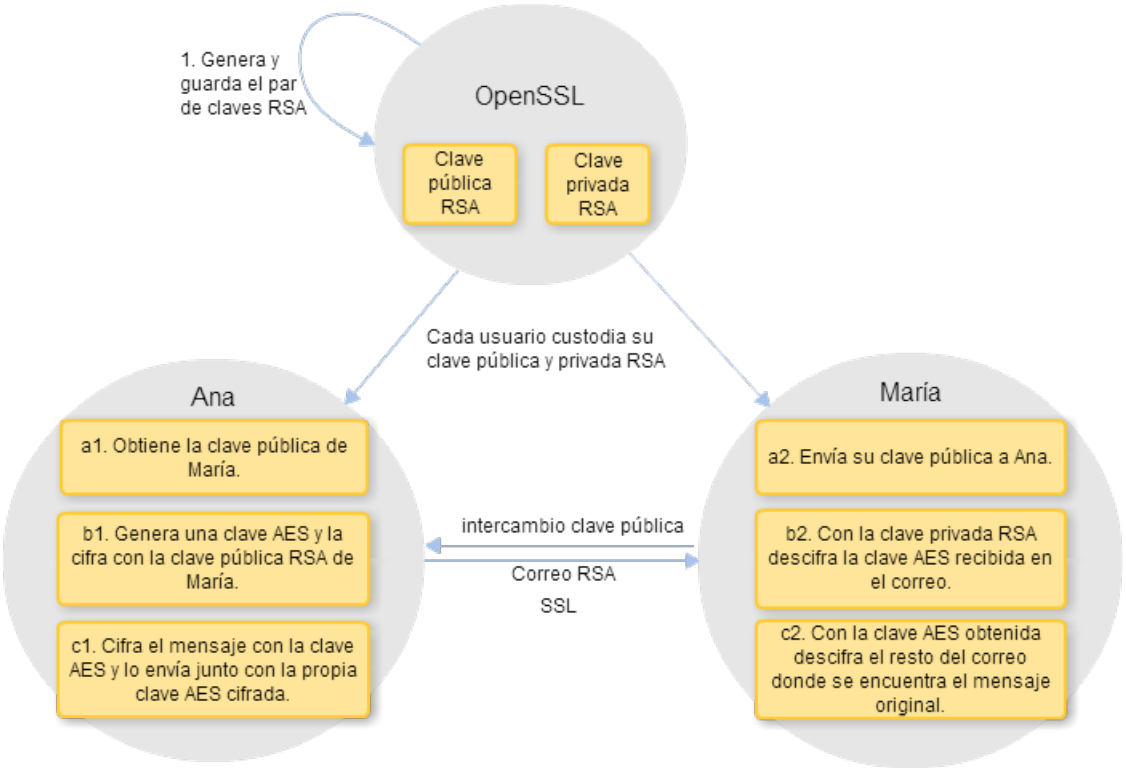


Figura 6.1: Diagrama de interacción RSA.

Al contrario que para el cifrado mediante RSA, el **cifrado basado en identidad no** necesita importar el **certificado con la clave pública del destinatario**. En su lugar el emisor hace una petición de a la PKG para obtener su clave IBE pública, es decir, se requiere la **transmisión de datos** entre la **aplicación de la PKG** y la **aplicación de correo**. La Figura 6.2 nos muestra los pasos que se realizan para ello.

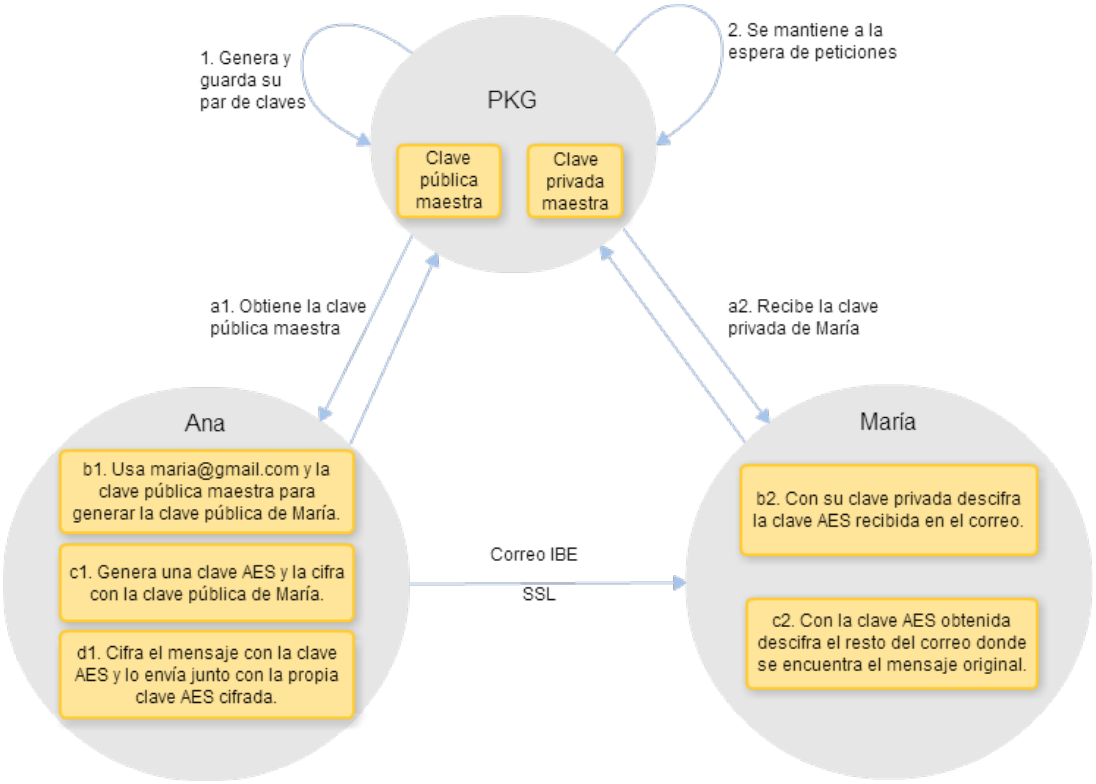


Figura 6.2: Diagrama de interacción IBE.

Para que el envío o la lectura de correos con este tipo de cifrado tenga lugar es necesario que la aplicación de la PKG esté iniciada y que, además, se haya generado en algún momento previo su par de claves pública/privada. Así, un administrador con acceso a la PKG se encarga de generar su par de claves (clase PKG), que son almacenadas para su uso posterior, e inicia el servidor a la espera de peticiones (clase PKGSocket).

Cuando Ana desea enviar un correo a María necesita la identidad (correo electrónico) de María y la clave pública de la PKG para poder generar la clave pública de María. Para obtener la clave pública de la PKG se requiere la petición por parte del usuario de los parámetros necesarios para construirla. Una vez tenga la clave pública de María, Ana la utiliza para cifrar la clave AES que ha generado para a su vez cifrar el mensaje que desea enviar. Así, envía un fichero como adjunto en el correo que contiene tanto el mensaje cifrado con la clave simétrica, como la clave AES cifrada con la clave pública de María. Además, escribe en el cuerpo del mensaje el tipo de cifrado utilizado ("IBE"). Una vez que María recibe el correo, identifica el tipo de cifrado en el cuerpo y solicita a la PKG los parámetros necesarios para generar su clave privada. De esta forma, María puede descifrar la clave AES y con ella el mensaje original. Todo esto ocurre de forma transparente a ambos usuarios finales.

Los mensajes y datos que se intercambia el emisor con la PKG pueden verse en la Figura 6.3, mientras que los mensajes intercambiados entre receptor y PKG se pueden ver en la Figura 6.4. Además, la primera detalla las acciones ya mencionadas que realiza el emisor (Usuario 1) tras recibir los parámetros necesarios de la PKG y la segunda los pasos que realiza el receptor (Usuario 2) para poder descifrar un correo recibido.

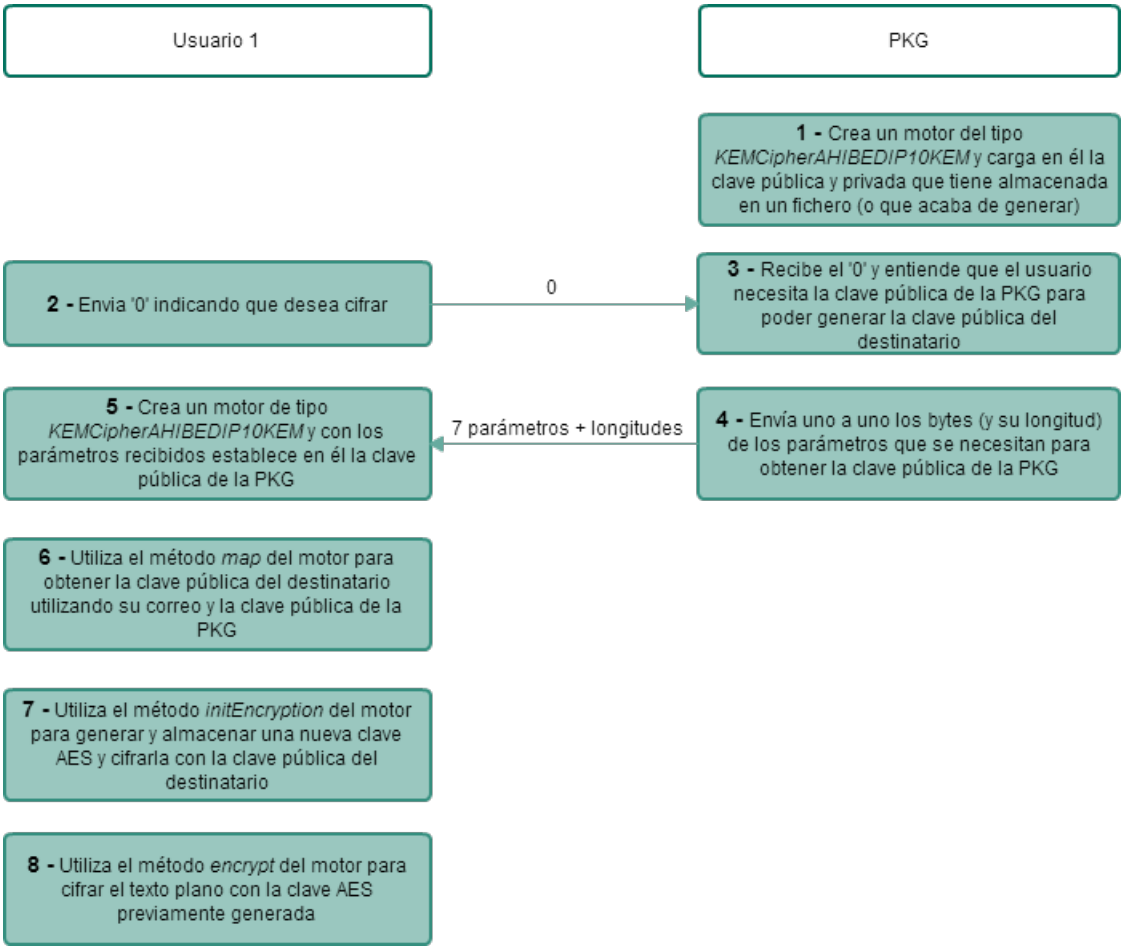


Figura 6.3: Detalle de los mensajes entre el emisor y la PKG.

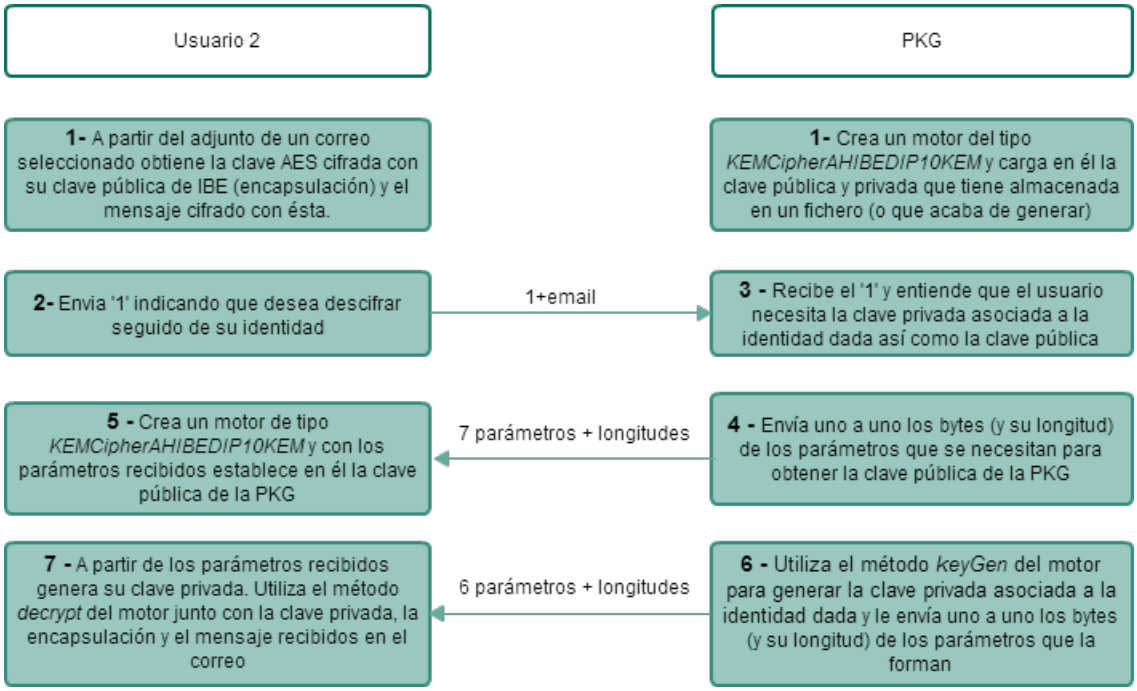


Figura 6.4: Detalle de los mensajes entre el receptor y la PKG.

El formato del fichero que se envía como **adjunto en el correo** es el que muestra la Figura 6.5. En primer lugar, se escribe un *byte* indicando si el mensaje ha sido cifrado o no. A continuación, se escribe la longitud del mensaje cifrado con la clave AES, seguido del propio mensaje cifrado. Finalmente y sin ningún espacio, se escribe la clave AES cifrada mediante clave asimétrica (ya sea IBE o RSA).

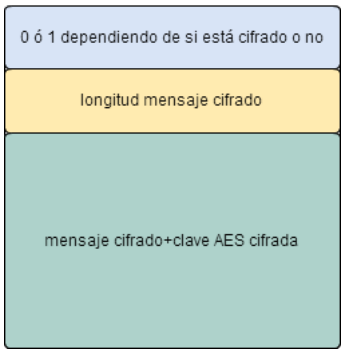
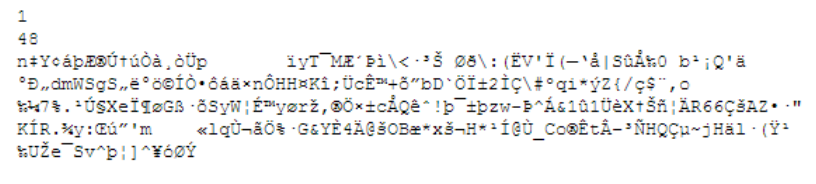
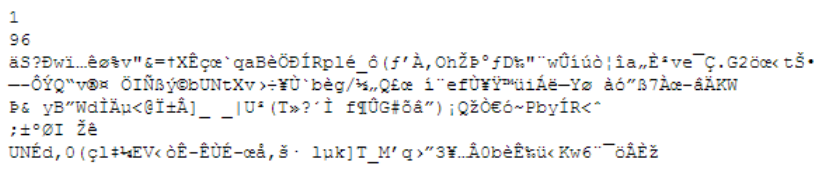


Figura 6.5: Estructura fichero adjunto.

Un ejemplo de este fichero para el cifrado RSA se puede observar arriba de la Figura 6.6, mientras que un ejemplo para el cifrado IBE se puede observar debajo.



(a)



(b)

Figura 6.6: (a)Fichero con cifrado RSA; (b) Fichero con cifrado IBE.

6.5.1. Implementación de las funcionalidades criptográficas

La aplicación de correo requiere el uso de cifrado simétrico (AES) y cifrado asimétrico (RSA, IBE) mientras que la aplicación de la PKG hace uso sólo del cifrado asimétrico (IBE). Así, para la implementación de los cifrados se han utilizado los ya mencionados *frameworks* JCA y JCE y la ya descrita biblioteca JPBC, quien utiliza el proveedor de servicios *Bouncy Castle*.

Para la cifra simétrica de los mensajes se ha empleado el algoritmo estándar internacional de cifrado en bloque de acuerdo con el NIST (*National Institute of Standards and Technology*), esto es, el algoritmo **AES 256**. El modo elegido para este algoritmo ha sido *Cipher Block Chaining* (CBC) [6, p. 78] con mecanismo de *padding* PKCS#7 [6, p. 238].

Para el almacenamiento de contraseñas de los administradores que tienen acceso a la PKG se ha empleado la **función** *hash* **SHA-1** [6, p .88]. Al ser una función de un único sentido, no se puede calcular la cadena original a partir del *hash*. De esta forma, para que los usuarios puedan acceder a la aplicación de la PKG deben escribir su nombre de usuario y contraseña de forma que el *hash* de ambos coincida con los almacenados por la aplicación.

Capítulo 7

Plan de Pruebas

En este capítulo se describen las pruebas llevadas a cabo durante el desarrollo y después de la codificación del proyecto. Las pruebas pueden ejecutarse en cualquier punto del proceso de desarrollo de software y su objetivo es encontrar la máxima cantidad de errores posibles antes de la finalización del proyecto. Se han realizado dos tipos de pruebas: pruebas unitarias y pruebas de integración.

Las **pruebas unitarias** son las que ejecuta normalmente el equipo de desarrollo en su entorno de trabajo. Consisten en la ejecución de actividades que permiten al desarrollador verificar que los componentes unitarios están codificados a prueba de casos extremos. Es decir, se verifica que todos los componentes unitarios (módulos) soportan el ingreso de datos erróneos o inesperados, demostrando así el control en el tratamiento de errores.

Las **pruebas de integración** son ejecutadas por el equipo de desarrollo tras las pruebas unitarias. Su objetivo es comprobar que los elementos del software que interactúan entre sí funcionan de manera correcta.

La planificación y ejecución de pruebas son una parte fundamental del desarrollo software, pues permiten comprobar funcionalidad y descartar problemas ante flujos de información anormales. Ejemplos de los problemas derivados de una evaluación no suficientemente rigurosa los podemos encontrar en los recientes casos de vulnerabilidades en SSL [34, 35, 36], el fallo de implementación de la biblioteca glibc [37] y el grave error en la comprobación de argumentos de entrada en la *shell* de Linux [38].

7.1. Pruebas unitarias

Tanto para la aplicación de correo como para la aplicación de la PKG se han realizado pruebas unitarias de cada clase, comprobando su correcto funcionamiento individual.

Para las dos clases modificadas de la JPBC (KEMCipherAHIBEDIP10KEM y AHIBEDIP10KeyPairGenerator) se ha utilizado una clase proporcionada por la propia biblioteca: **KEMCipherAHIBEDIP10KEMTest**. En ella se prueba el correcto funcionamiento del cifrado y descifrado de mensajes utilizando identidades. Para ello se comprueba que la clave privada generada para una cierta identidad puede descifrar la encapsulación (clave AES) cifrada previamente con su clave pública y que con esa clave simétrica puede obtener el mensaje original.

7.1.1. Aplicación de correo

A continuación se listan las clases *test* creadas para realizar las pruebas unitarias en la aplicación de correo. Todas ellas comprueban los casos en los que los parámetros introducidos son incorrectos, mostrando los mensajes informativos o lanzando las excepciones correspondientes.

SymmetricTest: comprueba el correcto funcionamiento de los métodos en la clase Symmetric. Para ello, genera una nueva clave simétrica (AES) con la que cifra el contenido de un fichero en otro fichero. A continuación, se descifra dicho fichero y se comprueba que la clave obtenida coincide con la original.

AsymmetricRSATest: comprueba el correcto funcionamiento de los métodos en la clase Asymmetric para el cifrado RSA. Para ello, cifra una clave simétrica con una clave pública RSA y escribe el resultado en un fichero. A continuación, utiliza la clave privada para descifrarlo y comprueba que se obtiene la misma clave simétrica que al inicio.

AHIBEDIP10KeyPairGeneratorPBCTest: se han realizado pruebas incorporando la funcionalidad añadida a la JPBC en el lado del usuario. Es decir, se comprueba que se puede obtener la clave pública de la PKG a partir de ciertos parámetros públicos, y que se puede obtener la clave privada de un usuario a partir de una serie de parámetros recibidos.

FileParserTest: comprueba que los ficheros que se envían como adjuntos se han formado bien, y que se pueden interpretar correctamente en el receptor para el tipo de cifrado dado. El *test* comprueba que se realiza la operación de cifrado correcta si se reciben los parámetros correctos. Esto es, dos "true" si se desea cifrar con RSA, un "true" y un "false" si se desea cifrar con IBE y dos "false" si no se desea cifrar. A partir del fichero que contiene la clave simétrica cifrada con una clave pública de RSA o con la clave pública de IBE, obtiene la longitud

y escribe en el fichero la estructura mostrada en la Figura 6.5. Por otro lado, el *test* obtiene el tipo de cifrado utilizado en el fichero y utiliza la longitud escrita en él para recuperar la clave simétrica cifrada. Con ella descifra el resto del fichero y comprueba que el mensaje coincide con el original.

SSLEmailTest: comprueba el envío de correos con texto en el cuerpo y con archivos o imágenes adjuntos mediante SSL y sobre el servidor de *gmail*.

GmailInboxTest: un usuario se autentica frente al servidor de *gmail* y se muestran todos los correos de su bandeja de entrada en orden de recepción (del más antiguo al más actual).

LoginTest: comprueba que no se puede acceder a la bandeja de entrada sin autenticarse correctamente. Si la autenticación es incorrecta se muestra un mensaje de aviso.

7.1.2. Aplicación de la PKG

A continuación se listan las clases *test* creadas para realizar las pruebas unitarias en la aplicación de la PKG:

HashTextTest: se genera un archivo para el acceso de la PKG que contiene los nombres de usuarios y el *hash* SHA-1 de sus contraseñas. Dado un nombre y una contraseña se comprueba si se encuentra entre los usuarios con acceso o no.

AHIBEDIP10KeyPairGeneratorPKGTest: se han realizado pruebas incorporando la funcionalidad añadida a la JPBC. En este test se comprueba que la recuperación del par maestro de la PKG a partir de los parámetros almacenados en un fichero es correcta.

LoginTest: comprueba que no se puede acceder al menú principal de la PKG sin autenticarse correctamente. Si la autenticación es incorrecta se muestra un mensaje de aviso.

7.2. Pruebas de integración

En las pruebas de integración se va a probar la correcta interacción entre los distintos componentes que ya han sido probados individualmente en las pruebas unitarias. Gran parte de estas pruebas de integración se han realizado mediante clases *test* dentro de cada una de las dos aplicaciones del sistema, es decir, entre componentes de la

misma aplicación. Sin embargo, otras se han realizado para comprobar la interacción entre ambas aplicaciones.

7.2.1. Aplicación de correo

Las siguientes clases *test* comprueban la integración entre clases dentro de la aplicación de correo:

FileEncryptionTest: comprueba la correcta integración de las clases Symmetric y Asymmetric. Para ello, crea una clave simétrica y la cifra utilizando la clave pública de RSA. A continuación, cifra el contenido de un fichero usando la clave simétrica y lo escribe en otro fichero. Descifra la clave simétrica usando la clave privada de RSA y comprueba que coincide con la clave generada. Finalmente, descifra con ella el contenido del fichero cifrado y comprueba que se obtiene el original.

AsymmetricIBETest: comprueba el correcto funcionamiento de los métodos en la clase Asymmetric para el cifrado IBE. Para ello es necesaria una correcta integración entre las clases Asymmetric, AHIBEDIP10KeyPairGeneratorPBC y KEMCipherAHIBEDIP10KEM. Además, se debe generar un par de claves pública/privada de la PKG para poder obtener la clave pública y privada de cualquier usuario y eso no es posible en ninguno de los módulos de la aplicación de correo. Puesto que no se desea probar aún la integración completa con la PKG, para este *test* se va a establecer un par maestro aleatorio en el motor del usuario. De esta forma, se comprueba la generación de la clave pública de un usuario a partir de su identidad (correo electrónico de *gmail*) para cifrar una clave simétrica. A continuación, se genera la correspondiente clave privada del usuario y se descifra la clave simétrica, comprobando que coincide con la de partida.

Envío de correos mediante RSA: se integran todos los módulos de envío de *emails* con los de criptografía RSA, así como el módulo para la creación de adjuntos (FileParser). Para ver el resultado de esta prueba consultar el Anexo D (ver sección D.2).

Acceso a un correo cifrado con RSA: se integran todos los módulos de recepción de *emails* con los de criptografía RSA, así como el módulo para la interpretación de adjuntos (FileParser). Para ver el resultado de esta prueba consultar el Anexo D (ver sección D.2).

7.2.2. Aplicación de la PKG

Las siguientes clases *test* comprueban la integración entre clases dentro de la aplicación de la PKG:

PKGTest: se comprueba la generación del par de claves pública/privada de la PKG, para lo cual es necesaria la correcta integración con las clases AHIBEDIP10KeyPairGeneratorPKG y KEMCipherAHIBEDIP10KEM. Este par de claves se almacena en un fichero para poder recuperarlas en el momento en que un usuario realice una petición a la PKG. Este test comprueba también que a partir de este fichero se recupera el par maestro original.

7.2.3. Integración entre ambas aplicaciones

SocketsTest: comprueba la interacción entre el *socket* de un usuario y el *socket* de la PKG. Para ello crea un *socket* de cada tipo y comprueba que el envío y la recepción de todo tipo de mensajes funciona en ambos extremos.

Envío de correos mediante IBE: cuando un usuario desea enviar un correo cifrado mediante IBE, es necesario solicitar la clave pública maestra a la aplicación de la PKG mediante los *sockets* abiertos en ambos extremos. Así, la PKG envía (clase PKGSocket) todos los parámetros públicos (clase PKG) que el usuario necesita para generar la clave pública (clases UserSocket y Asymmetric). Una vez obtiene esta clave pública puede generar, utilizando la identidad del destinatario (correo electrónico), la clave pública de éste y con ella cifrar una clave AES generada (clase Symmetric). A su vez, cifra con la clave AES el mensaje original y genera el archivo que se envía como adjunto con todos los datos necesarios (clase FileParser). Este archivo se envía como adjunto utilizando las clases de envío de *emails*. Así, esta prueba integra prácticamente todas las clases, tanto de la aplicación de correo como de la PKG. Para ver el resultado de esta prueba consultar el Anexo D (ver sección D.2).

Acceso a un correo cifrado con IBE: para poder leer correctamente un correo con un adjunto cifrado con IBE (clases SSLEmail y GmailInbox) el usuario necesita solicitar a la PKG (clases PKGSocket y UserSocket) los parámetros necesarios para construir la clave pública maestra y su propia clave privada. Así, la PKG carga su par de claves pública/privada (clases PKG y AHIBEDIP10KeyPairGeneratorPKG) y genera la clave privada del usuario a partir de su identidad y la clave privada maestra (clases PKG y AHIBEDIP10KeyPairGeneratorPKG) y la envía parámetro a parámetro por la conexión abierta entre

ambos. Una vez que el usuario obtiene la clave pública de la PKG puede crear el motor (clase KEMCipherAHIBEDIP10KEM) necesario para realizar la operación de descifrado con su clave privada. De esta forma el usuario descifra la clave simétrica contenida en el fichero adjunto y con ella puede descifrar el mensaje original. La clase FileParser se encarga la correcta interpretación del mensaje cifrado recibido y de mostrarlo de una forma comprensible. Para ver el resultado de esta prueba consultar el Anexo D (ver sección D.2).

7.2.4. Medición de tiempos

Tras la medición de tiempos realizada en el Anexo D (ver sección D.1), se puede concluir que se han cumplido todos los requisitos no funcionales de eficiencia y rendimiento, ya que todos los tiempos obtenidos son menores que los establecidos en el catálogo de requisitos (ver sección 4.2.2).

Capítulo 8

Conclusiones y Trabajos Futuros

El objetivo principal del trabajo realizado era estudiar el problema existente en la generación y gestión de identidades digitales en la criptografía contemporánea, proponiendo la criptografía basada en emparejamientos como solución. Esta necesidad surge debido a problemas actuales como el robo de credenciales o la escalada de privilegios en sistemas basados en las tradicionales infraestructuras de clave pública (PKI). La criptografía basada en emparejamientos ha surgido como alternativa a dichas infraestructuras, con el fin de solventar algunas de sus principales desventajas. Este tipo de criptografía se fundamenta también en la idea de una tercera parte de confianza llamada PKG, que se encarga de generar el par de claves pública/privada para cada usuario del sistema en base a sus identidades.

La criptografía basada en emparejamientos permite diseñar protocolos IBE para prescindir del uso de certificados y facilitar un manejo más flexible de las identidades digitales. A su vez, las claves públicas de los usuarios poseen una forma mucho más atractiva y éstos no tienen que custodiar sus credenciales, pues pueden solicitarlas a la PKG en cualquier momento. Por tanto, no hay necesidad de distribución de claves ni de autoridades *pesadas*. Además, con el cifrado basado en identidad es posible cifrar mensajes incluso cuando el receptor aún no ha generado su par de claves, y la revocación de claves (si existe) es mucho más sencilla.

La principal desventaja de los sistemas que se basan en este tipo de criptografía es que la PKG debe ser totalmente confiable, ya que es capaz de generar cualquier clave privada de los usuarios y así descifrar y firmar mensajes sin autorización. De esta forma, se introduce un problema de *key escrow*. Con un sistema basado en IBE, la autenticidad de las claves públicas está garantizada siempre y cuando el transporte de las claves privadas a sus correspondientes usuarios se mantenga seguro.

Un gran número de sistemas han sido propuestos para eliminar el *key escrow*, siendo el ejemplo más significativo la criptografía libre de certificados (en inglés, Cer-

tificateless Cryptography). En este tipo de criptografía, una autoridad parcialmente confiable llamada KGC (*Key Generation Center*) genera una clave privada parcial para cada usuario. Mediante criptografía basada en la identidad, esta clave se genera a partir de la cadena de identidad del usuario y de la clave privada maestra de la KGC. En este caso la KGC no tiene la clave privada completa y, por tanto, no puede cifrar ni firmar por el usuario. De esta forma, no se necesita ningún certificado adicional para la clave pública de los usuarios. La clave ya no es tan fácil de memorizar como lo es en IBE original, pero el nivel de confianza en la tercera parte es mucho menor. La flexibilidad es uno de los atributos más significativos de esta criptografía. De hecho, puede ser fácilmente transformada en PKI tradicional o en IBE.

El estudio realizado acerca de la criptografía basada en emparejamientos ha sido plasmado a lo largo de todo este documento, siendo el principal objetivo la aplicación de PBC para la generación de protocolos IBE. En primer lugar, se analizó el problema existente en la gestión de identidades digitales, proponiéndose la criptografía basada en emparejamientos como soporte de posibles soluciones. A continuación, se describieron los fundamentos teóricos de la PBC, los principales protocolos IBE y sus ventajas y desventajas. Posteriormente, se presentó el sistema de correo implementado basado en IBE como ejemplo de aplicación práctica de este tipo de criptografía. También se detalló al completo toda la funcionalidad y la seguridad que ofrece la aplicación desarrollada. Así, el sistema de correo permite el acceso a usuarios con cuenta en *gmail*, y éstos pueden enviar y recibir correos utilizando el cifrado RSA o el cifrado IBE. Finalmente, se especificó el plan de pruebas realizado para verificar el correcto funcionamiento de todas las características del sistema. Así pues, se puede concluir que el proyecto desarrollado cumple con los requisitos inicialmente planteados y que se han resuelto satisfactoriamente algunos de los problemas de los esquemas tradicionales, como la distribución de claves y el manejo de certificados.

Sin embargo, esta implementación puede ser mejorada en varios aspectos, que podrían ser desarrollados en un trabajo futuro. En primer lugar, el cuerpo del mensaje indica el tipo de cifrado utilizado al enviar un correo, lo que puede causar problemas si el gestor de correos modificara dicho cuerpo. Así, podría introducirse el indicador del tipo de cifrado dentro del fichero adjunto con el resto del mensaje y firmar los usuarios cada uno de sus correos. En segundo lugar, la generación de un nuevo par de claves por parte de la PKG provoca que los usuarios no puedan volver a leer los correos antiguos, pues esto cambia la generación de las claves pública y privada para cada uno. En un trabajo futuro este inconveniente podría solucionarse manteniendo una base de datos con un historial de claves para poder abrir todos los correos. Sin embargo, esto supondría perder una de las ventajas de la criptografía basada en emparejamientos,

ya que los usuarios tendrían que almacenar de alguna forma sus claves. En tercer lugar, en esta implementación no se puede enviar correos a múltiples destinatarios. Ahora bien, esta funcionalidad se podría incluir fácilmente sin más que incorporar la clave de sesión cifrada con las claves IBE públicas de cada uno de los destinatarios. Por último, la exportación e importación de credenciales IBE debería implementarse siguiendo el estándar ASN.1, de modo similar al resto de credenciales de clave pública (RSA, ElGammal...).

Como parte fundamental del trabajo realizado, es preciso destacar la dificultad que entraña el manejo de bibliotecas criptográficas que se encuentran en fase de desarrollo, como es el caso de la biblioteca JPBC empleada en este TFG. La falta de documentación y de ejemplos y casos de uso ha supuesto un obstáculo para llevar a cabo el desarrollo del proyecto. No obstante, dichos obstáculos fueron superados satisfactoriamente, por lo que este TFG constituye un claro ejemplo de caso de uso que puede ser de gran utilidad para otros desarrolladores de sistemas para la gestión segura de información. Así, las funcionalidades desarrolladas tienen aplicación más allá del caso de uso recogido en este trabajo, por ejemplo en aplicaciones para el almacenamiento de información en la nube.

Anexo A

Álgebra

Este anexo es una breve introducción a las estructuras algebraicas mencionadas a lo largo del trabajo. Esto es, grupos, cuerpos y sus propiedades así como aplicaciones bilineales.

A.1. Grupo

Definición A.1.1. Un grupo es una estructura algebraica G sobre el que se define una operación interna $\cdot$ que cumple las siguientes propiedades:

a. **Propiedad asociativa**
$\forall a, b, c \in G,\ a \cdot (b \cdot c) = (a \cdot b) \cdot c.$

b. **Elemento neutro**
$\exists e \in G$ tal que $\forall a \in G$ se cumple $a \cdot e = a = e \cdot a$. Se denota $e = 1$.

c. **Elemento inverso**
Para cada $a \in G$, existe $a^{-1} \in G$ de modo que $a \cdot a^{-1} = e = a^{-1} \cdot a$.

Definición A.1.2. G es un grupo abeliano si, además de las tres propiedades mencionadas, cumple la **propiedad conmutativa**, es decir, $a \cdot b = b \cdot a$ para cualesquiera $a, b \in G$.

Observación A.1.3. Si no da lugar a confusión, el signo $\cdot$ se omite. También es habitual denotar a la operación interna como $+$ cuando el grupo es abeliano, tal y como ocurre en los anillos. En tal caso, el elemento neutro se representa por 0 y el inverso de a con $-a$.

Observación A.1.4. El cardinal de un grupo se llama *orden del grupo* y se denota por $|G|$.

Ejemplo A.1.5. *El conjunto de los números enteros* $\mathbb{Z}$ *y la operación de suma definen un grupo abeliano. El elemento identidad es 0, mientras que* $-a$ *es el inverso de* $a \in \mathbb{Z}$.

Definición A.1.6. Un *subgrupo* de un grupo G es un subconjunto $H \subset G$ tal que, para cualesquiera $a, b \in H$ se tiene que $ab \in H$ y $h^{-1} \in H, \forall h \in H$. Es decir, H tiene estructura de grupo con la misma operación que G. Para indicar que un subconjunto $H \subset G$ es un subgrupo se suele escribir $H<G$.

Definición A.1.7. Dado un grupo G cuya operación es notada multiplicativamente (resp. aditivamente) se define la *potencia* (resp. el *múltiplo*) de un elemento $a \in G$ para cada $n \in \mathbb{Z}$.

$$\begin{cases} a^n = \begin{cases} 1 \ si \ n = 0 \\ a^m \cdot a \ si \ n = m + 1 \end{cases} \\ a^{-n} = (a^{-1})^n \end{cases}$$

(resp.

$$\begin{cases} n \cdot a = \begin{cases} 0 \ si \ n = 0 \\ (m \cdot a) + a \ si \ n = m + 1 \end{cases} \\ (-n) \cdot a = n \cdot (-a) \end{cases}$$

)

Nótese que en la definición anterior $(a^n)^m = a^{nm}$, $a^n \cdot a^m = a^{n+m}$.

Definición A.1.8. Sea G un grupo. Se dice que G es un *grupo cíclico* si existe $a \in G$ tal que $G = \langle a \rangle \overset{not}{=} \langle \{a\} \rangle = \{a^n : n \in \mathbb{Z}\}$. En este caso, a se denomina *generador* del grupo G.

Proposición A.1.9. *Si* G *es un grupo cíclico, entonces* G *es abeliano.*

Definición A.1.10. Sea G un grupo y $a \in G$. Se dice que

1. a es de orden infinito si las distintas potencias de a (resp. múltiplos) son distintas entre sí.

2. a tiene orden finito si existen n, m / $0 \leq n < m$ de modo que $a^n = a^m$.

Definición A.1.11. Sea G un grupo y $a \in G$ de orden finito. El orden de a es el menor entero estrictamente positivo m para el cual $a^m = 1$.

A.2. Anillo

Definición A.2.1. Un *anillo* es un conjunto A con dos operaciones internas, $+$ y $\cdot$, tales que $(A, +)$ es un grupo abeliano y se verifican las siguientes propiedades:

a. **Propiedad distributiva del producto respecto a la suma**
$(a+b)\cdot c = a\cdot c + b\cdot c$ y $a\cdot(b+c) = a\cdot b + a\cdot c$ para cualesquiera $a, b, c \in A$.

b. **Propiedad asociativa del producto**
$(a\cdot b)\cdot c = a\cdot(b\cdot c)$ para cualesquiera $a, b, c \in A$.

c. **Propiedad conmutativa del producto**
$a\cdot b = b\cdot a$ para cualesquiera $a, b \in A$. Se dice que el anillo es *conmutativo*.

d. **Elemento neutro para el producto**
Existe $1 \in A$ tal que $a\cdot 1 = 1\cdot a = a$ para cualquier $a \in A$. Se dice que el anillo tiene *unidad*.

Definición A.2.2. Una *unidad* en un anillo A es un elemento $a \in A$ que tiene inverso para el producto, es decir, $\exists a^{-1} \in A$ tal que $aa^{-1} = a^{-1}a = 1$.

A.3. Cuerpo

Definición A.3.1. Un *cuerpo* es un anillo conmutativo en el que todos los elementos no nulos son unidades. Por tanto, un cuerpo es un conjunto K en el que se han definido dos operaciones, $+$ y $\cdot$, llamadas adición y multiplicación respectivamente, que cumplen las siguientes propiedades:

a. K **es cerrado para la adición y la multiplicación**
$a\cdot b \in K$ y $a+b \in K$ para cualesquiera $a, b \in A$.

b. **Propiedad asociativa del producto y la suma**
$(a\cdot b)\cdot c = a\cdot(b\cdot c)$ y $a+(b+c) = (a+b)+c$ para cualesquiera $a, b, c \in K$.

c. **Propiedad conmutativa de la multiplicación y de la adición**
$a\cdot b = b\cdot a$ y $a+b = b+a$ para cualesquiera $a, b \in K$.

d. **Propiedad distributiva de la multiplicación con respecto a la adición**
$(a+b)\cdot c = a\cdot c + b\cdot c$ y $a\cdot(b+c) = a\cdot b + a\cdot c$ para cualesquiera $a, b, c \in K$.

e. **Elemento neutro para la multiplicación y la adición**
Existe $0 \in K$ tal que $a+0 = a$ para cualquier $a \in K$.
Existe $1 \neq 0 \in K$ tal que $a\cdot 1 = 1\cdot a = a$ para cualquier $a \in K$.

f. **Elemento opuesto e inverso multiplicativo**
   Para cada $a \in K$, existe un elemento $-a \in K$, tal que $a + (-a) = 0$.
   Para cada $a \neq 0 \in K$, existe un elemento $a^{-1} \in K$, tal que $a \cdot a^{-1} = 1$.

Ejemplo A.3.2. *Los números racionales* $\mathbb{Q} = \{\frac{a}{b} \mid a, b \in \mathbb{Z}, b \neq 0\}$*, donde está incluido el conjunto* $\mathbb{Z}$ *de los números enteros, forman un cuerpo.*

Definición A.3.3. Un *cuerpo finito* es un cuerpo definido sobre un conjunto finito de elementos. Todos los cuerpos finitos tienen un número de elementos $q = p^n$, para algún número primo p y algún entero positivo n.

A.4. Aplicaciones bilineales

Consideramos dos grupos $\mathcal{G}_1$ y $\mathcal{G}_2$ de orden primo q. El grupo $\mathcal{G}_1$ es un grupo cíclico aditivo, mientras que el grupo $\mathcal{G}_2$ es multiplicativo. Generalmente, el grupo $\mathcal{G}_1$ **es un grupo de curva elíptica**, es decir, está formado por puntos de una curva elíptica, y el grupo $\mathcal{G}_2$ es un cuerpo finito.

Definición A.4.1. Un emparejamiento es cualquier aplicación bilineal $e : \mathcal{G}_1 \times \mathcal{G}_1 \rightarrow \mathcal{G}_2$ que satisfaga las siguientes tres propiedades:

a. **Bilinealidad**

$$\forall P, Q \in \mathcal{G}_1, \forall a, b \in \mathbb{Z}_q^*$$
$$e(aP, bQ) = e(P, Q)^{ab}$$

b. **No degeneratividad**
   El mapa e no transforma todos los puntos de $\mathcal{G}_1 \times \mathcal{G}_1$ en uno solo de $\mathcal{G}_2$.

c. **Eficiencia**
   El cálculo de e es eficiente para todos los elementos del dominio.

A.5. Notación

En esta sección se describe la notación seguida para las definiciones matemáticas a lo largo del trabajo:

El conjunto de todas las clases de equivalencia se denota con $\mathbb{Z}/q\mathbb{Z} = \mathbb{Z}_q = \{0, 1, 2, \ldots, q-1\}$. Sin embargo, $\mathbb{Z}_q^*$ hace referencia a todos los números en $\mathbb{Z}_q$ que tienen inverso multiplicativo. Por tanto, si q es primo, $\mathbb{Z}_q = \{1, 2, 3, \ldots, q-1\}$, pues todos los números a excepción del 0 tienen inverso multiplicativo.

Cuando nos referimos a las identidades teóricas de los usuarios en los protocolos IBE decimos que $id \in \{0,1\}^*$. Esto quiere decir que id define el conjunto de cadenas binarias de longitud finita.

Anexo B

Creación de certificados

B.1. Creación de certificados con OpenSSL

En el presente proyecto la generación de las claves RSA de cada uno de los usuarios han sido generadas mediante la herramienta OpenSSL. En concreto, se han generado claves de 2048 bits que se almacenan en ficheros con el formato estándar PEM. Para ello se empleó el siguiente comando:

```
openssl genrsa -out private.pem 2048
```

Una vez se ha generado el fichero PEM con el par de claves (pública/privada) RSA, se procedió a volcar ambas en distintos ficheros de formato estándar DER:

```
openssl pkcs8 -topk8 -in private.pem -outform DER -out
    private.der -nocrypt
openssl rsa -in private.pem -pubout -outform DER -out public.
    der
```

Si bien se ha empleado OpenSSL para la generación de credenciales, podría haberse realizado esta operación de modo directo en el código haciendo uso de la biblioteca criptográfica de Java.

B.2. Manejo de certificados con Keytool

Tal y como se ha comentado en la Sección 4.2.2, la comunicación entre el cliente de correo y el servidor PKG se debe realizar de modo seguro. Para ello ha de establecerse un canal SSL entre la PKG y el cliente de correo, lo que requiere la creación de un certificado con la clave pública del servidor y que debe ser importado por el cliente. Para realizar esta operación se ha hecho uso de la herramienta Keytool tal y como sigue:

```
keytool -genkey -keyalg RSA -alias certificado_pkg -keystore
    certificados.jks
```

Con esta instrucción se crea el certificado y se incluye en el almacén de credenciales *certificados.jks*.

Para que el cliente tenga el certificado con la clave pública se debe exportar la clave pública contenida en el certificado de la PKG que está en el anterior almacén de credenciales:

```
keytool -export -keystore certificados.jks -alias
    certificado_pkg -file pkg_clavePublica.cer
```

Con esto se tendría un certificado que el cliente puede descargarse de cualquier sitio e instalarlo en su sistema. En este caso instalarlo significa introducirlo en el almacén de credenciales del cliente:

```
keytool -importcert -file pkg_clavePublica.cer -keystore
    certificados_cliente.jks -alias certificado_pkg
```

Por último, se debe ejecutar la aplicación correspondiente con las siguientes opciones:

```
java -Djavax.net.ssl.keyStore=certificados.jks -Djavax.net.
    ssl.keyStorePassword=123456 es.uam.eps.pkg_tfg.main.
    MainPKG
java -Djavax.net.ssl.keyStore=certificados_cliente.jks -
    Djavax.net.ssl.keyStorePassword=456789 es.uam.eps.
    tfg_laura.main.MainCorreo
```

Anexo C

Interfaz

La interfaz proporciona al usuario final un entorno visual sencillo que permite la comunicación con el sistema de correo. Puesto que tenemos dos aplicaciones distintas que se comunican entre sí, también hay dos interfaces de usuario distintas. Para la implementación de la interfaz se ha hecho uso de la biblioteca gráfica de Java *Swing*.

C.1. Interfaz de la aplicación de correo

La primera vista de la aplicación de correo se trata de la pantalla de ***login*** (Figura C.1).

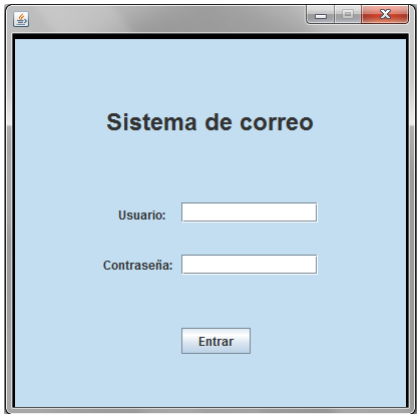


Figura C.1: Login aplicación de correo.

En ella el usuario debe introducir su correo de *gmail* y su contraseña para poder acceder a la **bandeja de entrada** (Figura C.2), que es la siguiente vista de la aplicación.

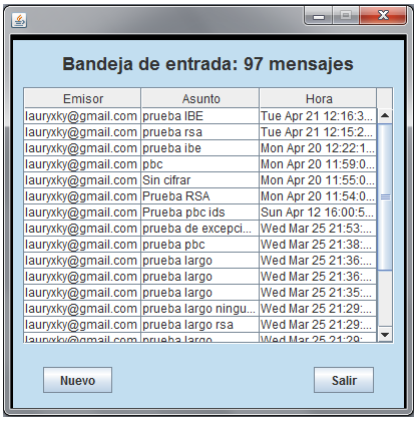


Figura C.2: Bandeja de entrada.

En la bandeja de entrada el usuario puede seleccionar alguno de los correos recibidos o enviar un nuevo correo.

Al **enviar un nuevo correo** (Figura C.3) se debe rellenar el destinatario y el contenido del mensaje. Además, es necesario seleccionar un tipo de cifrado para éste, y opcionalmente se puede establecer un asunto del correo.

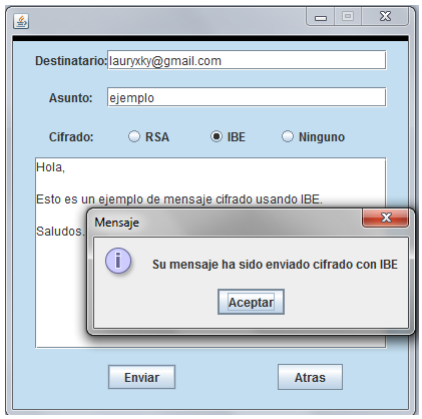


Figura C.3: Bandeja de entrada.

Para **visualizar** uno de los correos recibidos basta con hacer doble *click* sobre uno de los de la lista. Esta vista muestra el correo del emisor, el asunto y el contenido del mensaje. Ofrece la posibilidad de volver a la bandeja de entrada o responder directamente al correo (Figura C.4).

(a) (b)

Figura C.4: (a)Visualización de un correo; (b) Responder a un correo.

Finalmente, la Figura C.5 muestra el **diagrama de navegabilidad** entre todas las vistas ya explicadas.

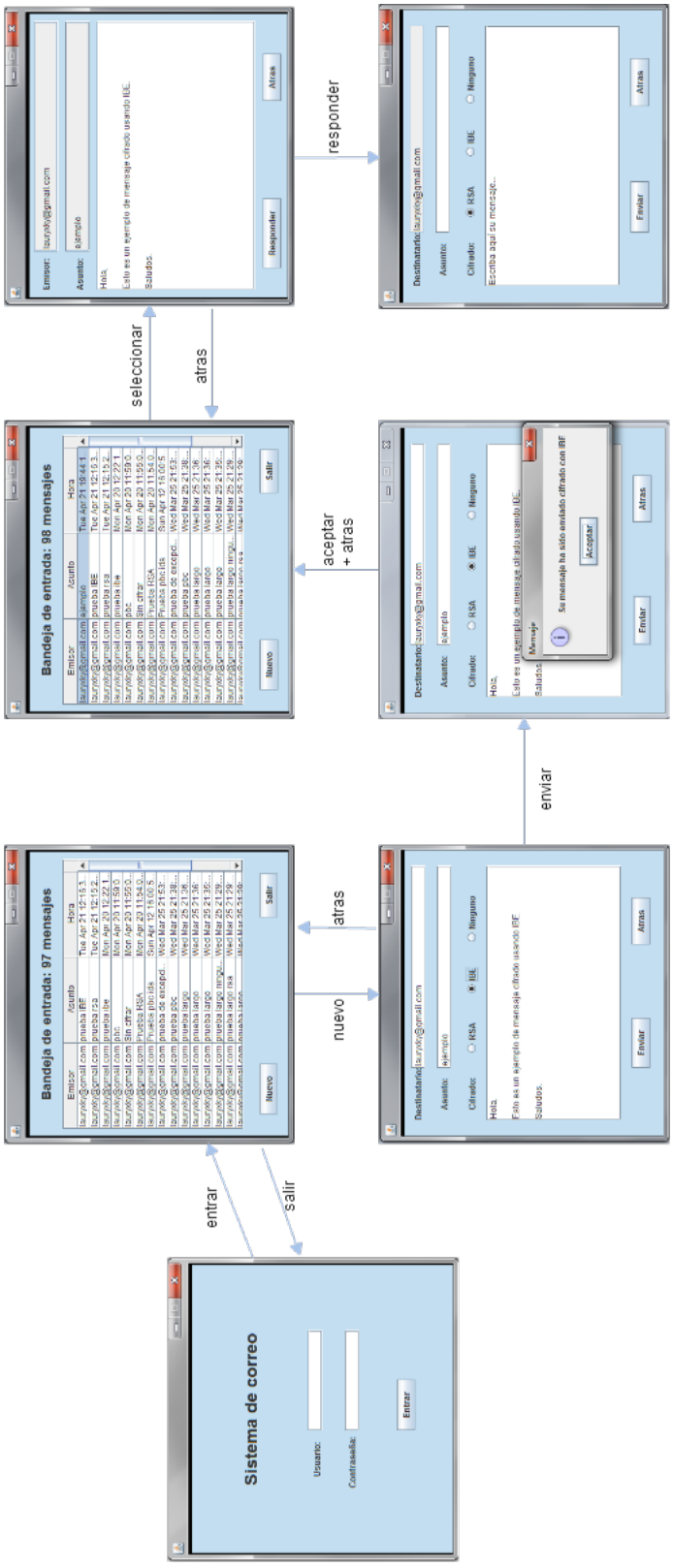


Figura C.5: Interfaz gráfica para el correo.

C.2. Interfaz de la aplicación PKG

La primera vista de la aplicación de la PKG se trata también de la pantalla de ***login*** (Figura C.6).

Figura C.6: Login aplicación de la PKG.

En ella el administrador debe introducir su nombre de usuario y su contraseña para poder acceder al **menú principal de la PKG** (Figura C.7), que es la siguiente vista de la aplicación.

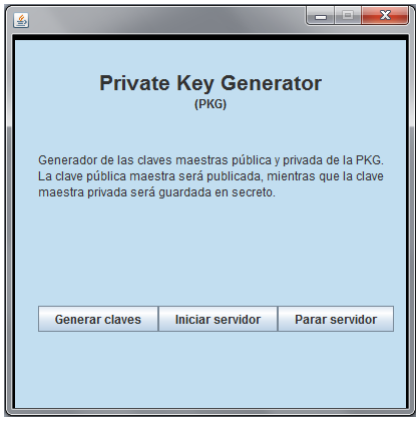


Figura C.7: Menú principal PKG.

En el menú principal el usuario puede **generar un nuevo par de claves**, **iniciar el servidor** si las claves ya están creadas o **parar el servidor** si se encuentra en funcionamiento. La Figura C.8 muestra el diagrama de navegabilidad de la aplicación de la PKG.

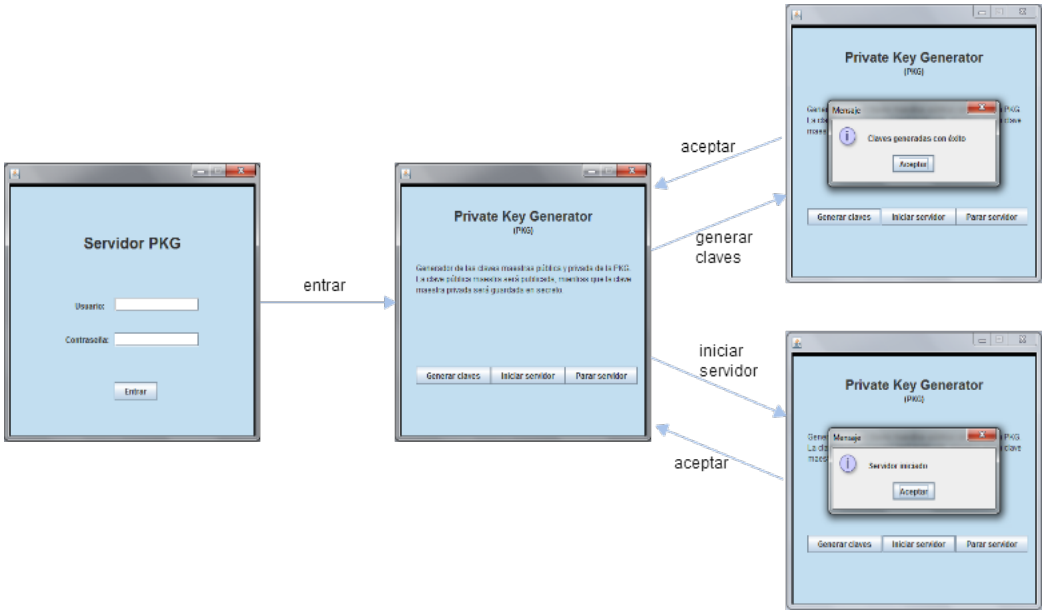


Figura C.8: Interfaz gráfica para la PKG.

Anexo D

Pruebas

Este anexo contiene una medición de tiempos de realización de ciertas actividades importantes, así como algunas de las pruebas de integración realizadas sobre la aplicación de correo, *gmail* y el servidor PKG.

D.1. Medición de tiempos

Para la medición de los tiempos para la eficiencia y rendimientos de los requisitos no funcionales se ha utilizado *System.nanoTime()* de Java antes y después de lo que deseamos medir. Los resultados obtenidos se han escrito en un fichero en nanosegundos, y posteriormente se ha hecho la media de los datos, la varianza y la desviación típica siguiendo las siguientes fórmulas:

$$\overline{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

$$s^2 = \frac{1}{N-1}\sum_{i=1}^{N} x_i - \overline{x}$$

donde $\overline{x}$ es la media muestra, N el número de datos totales, x_i cada uno de los datos tomados y s^2 es la varianza de los datos.

D.1.1. Cifrado y envío de correos mediante RSA

Se ha medido conjuntamente el tiempo que tarda la aplicación de correo en cifrar un correo mediante RSA y en enviar dicho correo. Este tiempo incluye la creación del archivo adjunto que se envía en el correo, que contiene la clave simétrica cifrada mediante RSA y el mensaje original cifrado con dicha clave simétrica. Además, incluye el tiempo de envío del correo mediante el servidor de *gmail*. Los resultados obtenidos

se muestran en la Figura D.1, donde se puede observar una media muestral de 2,074 segundos a partir de 20 datos y una desviación típica de 0,13687 segundos.

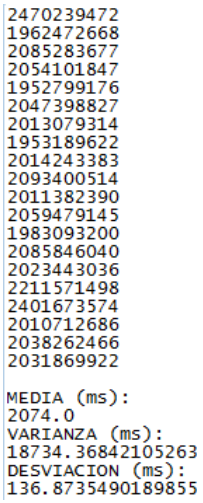


Figura D.1: Tiempos de envío de correos cifrados mediante RSA.

D.1.2. Cifrado y envío de correos mediante IBE

Se ha medido conjuntamente el tiempo que tarda la aplicación de correo en cifrar un correo mediante IBE y en enviar dicho correo. Este tiempo incluye la creación del archivo adjunto que se envía en el correo, que contiene la clave simétrica cifrada mediante IBE y el mensaje original cifrado con dicha clave simétrica. Además, incluye el tiempo de envío del correo mediante el servidor de *gmail*. Los resultados obtenidos se muestran en la Figura D.2, donde podemos observar que la media muestral obtenida a partir de 20 datos es de 2,52 segundos y una desviación típica de 0,3611577 segundos.

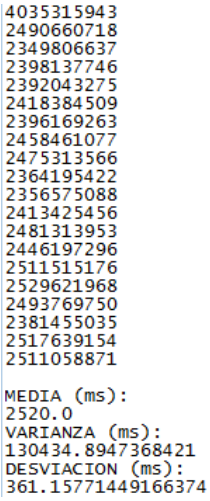


Figura D.2: Tiempos de envío de correos cifrados mediante IBE.

D.1.3. Envío de correos sin cifrar

Para el envío de correos sin cifrado de ningún tipo se han obtenido los resultados que muestra la Figura D.3, en la que se puede ver una media muestral de 2,062 segundos y una desviación típica de 0,15391 segundos, obtenidos a partir de una muestra de 20 datos. Este tiempo incluye la creación del fichero que se envía como adjunto en el correo, aunque en este caso se encuentra el mensaje en claro y no hay ninguna clave simétrica en él.

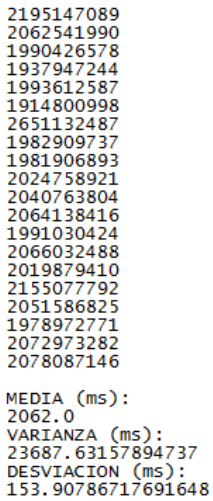


Figura D.3: Tiempos de envío de correos sin cifrado.

A partir de estas estadísticas obtenidas para el cifrado y envío de correos mediante RSA, IBE o ninguno, se puede concluir que la mayor parte del tiempo se emplea en el envío en sí del correo por el servidor de *gmail*. Esto se observa en el hecho de que el tiempo de media de los correos sin cifrar es solamente un poco menor que el de los otros dos. Además, el cifrado de correos mediante IBE emplea ligeramente más tiempo que el cifrado mediante RSA.

D.1.4. Acceso a la bandeja de entrada

Se han tomado tiempos de acceso por primera vez a la bandeja de entrada, no se han considerado los tiempos de acceso a la bandeja de entrada al salir de visualizar un correo. Por tanto, este tiempo incluye la conexión al servidor de *gmail* por SMTP y la lectura de los 20 últimos correos recibidos. Los resultados obtenidos se muestran en la Figura D.4.

```
6915652881
5353725641
5822968684
4864109250
5134075795
4922498310
4947298286
4890098527
4992271470
5044631490
4854754786
4972243237
5181872766
5083555535
5055835976
5087134127
5178324108
5051232728
5087569905
5543741332

MEDIA (ms):
5198.0
VARIANZA (ms):
218230.84210526315
DESVIACION (ms):
467.1518405243237
```

Figura D.4: Tiempos de acceso a la bandeja de entrada.

Se observa que la media obtenida a partir de 20 muestras es de 5,198 segundos y la desviación típica de los datos es de 0,46715 segundos.

D.1.5. Generación de claves PKG

Para medir el tiempo que tarda la aplicación de la PKG en generar un nuevo par de claves y almacenarlas para su posterior uso, se ha utilizado *System.nanoTime()* antes y después de la generación. A los resultados obtenidos (en nanosegundos) se le ha hecho la media, la varianza y la desviación típica y convertido a milisegundos, de forma que obtenemos un fichero de tiempos como muestra la Figura D.5.

```
2627565432
2115838141
2412115129
2278598276
2428570329
2499171424
2777226581
2025550078
2076339331
2171119025
2091394826
3056167835
2550186478
2128790869
2086985304
2053753311
2730120269
2310248813
2670905840
2112167602

MEDIA (ms):
2359.0
VARIANZA (ms):
88432.42105263157
DESVIACION (ms):
297.37589184840044
```

Figura D.5: Tiempos de generación de claves de la PKG.

Así, la media obtenida a partir de 20 muestras es de 2,359 segundos y la desviación

típica de los datos es de 0,297376 segundos.

D.1.6. Descifrado de correos mediante RSA

Se ha tomado una muestra de 20 datos para medir aproximadamente cuánto tiempo emplea el descifrado de correos que han utilizado RSA. Este tiempo es el que tarda en descifrar el mensaje cifrado contenido en el archivo adjunto de un correo, es decir, incluye el tiempo de descifrar la clave simétrica con RSA y de descifrar con ella el mensaje original. Los resultados se muestran en la Figura D.6, donde se puede ver que la media del descifrado RSA es de 77 milisegundos con una desviación típica de 11,4281 milisegundos.

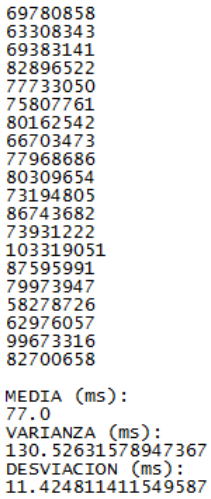


Figura D.6: Tiempos de descifrado de correos mediante RSA.

D.1.7. Descifrado de correos mediante IBE

Los resultados de tomar 20 muestras para medir el tiempo que se emplea en el descifrado de correos cifrados utilizando IBE se muestran en la Figura D.7. Este tiempo es el que tarda en descifrar el mensaje cifrado contenido en el archivo adjunto de un correo, es decir, incluye el tiempo de descifrar la clave simétrica con IBE y de descifrar con ella el mensaje original. Se puede observar que la media del descifrado IBE es de 645 milisegundos con una desviación típica de 65,21624 milisegundos, considerablemente mayor que para RSA.

```
645340367
622639735
607096289
904436848
622203957
607257514
637311625
640464277
619629491
615897798
616956665
660386024
678423538
655604018
625920682
606012191
616361800
619894207
621554353
688404083

MEDIA (ms):
645.0
VARIANZA (ms):
4253.1578947368425
DESVIACION (ms):
65.21623950165205
```

Figura D.7: Tiempos de descifrado de correos mediante IBE.

D.2. Pruebas integración

La Figura D.8 muestra el envío correcto de un *email* con cifrado RSA en la aplicación de correo, y la Figura D.9 muestra la carpeta enviados de *gmail* con dicho correo.

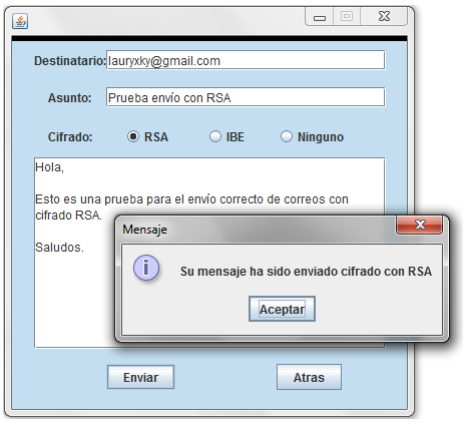


Figura D.8: Envío correcto de un correo con RSA en la aplicación de correo.

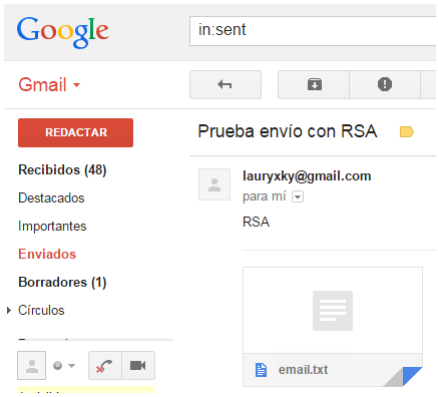


Figura D.9: Envío correcto de un correo con RSA en *gmail*.

La Figura D.10 muestra la recepción correcta de un *email* con cifrado RSA en la aplicación de correo, y la Figura D.11 muestra la bandeja de entrada de *gmail* con dicho correo.

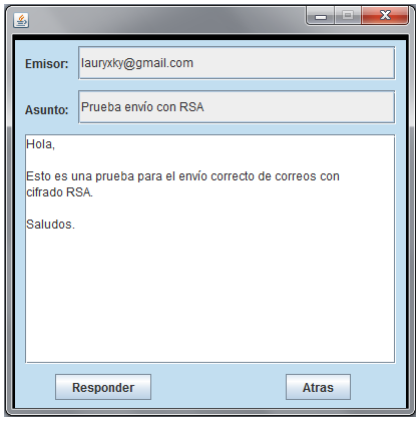


Figura D.10: Recepción correcta de un correo con RSA en la aplicación de correo.

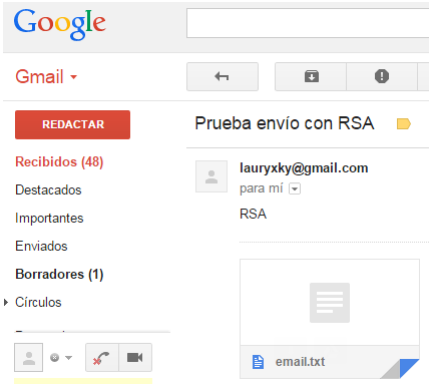
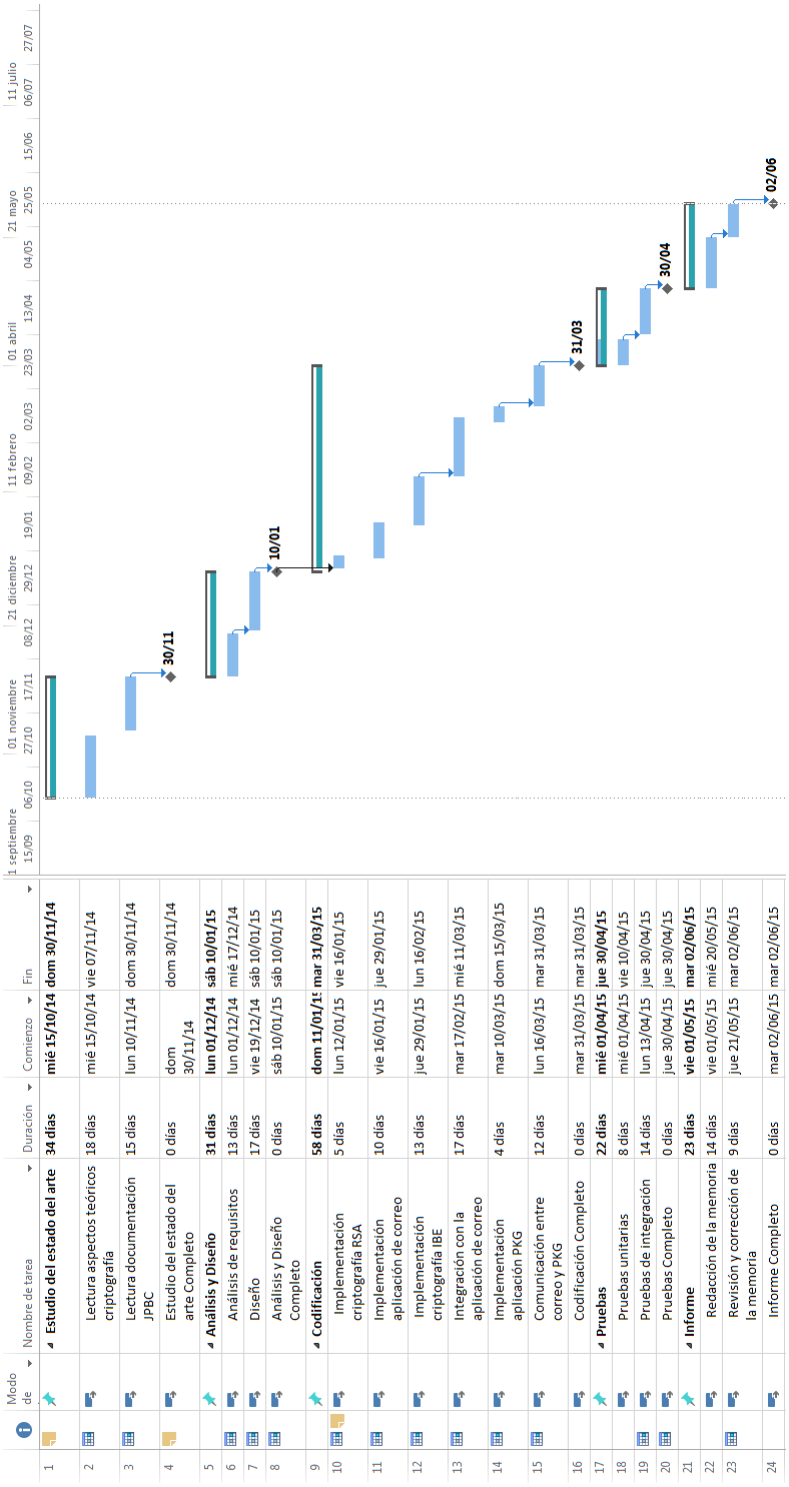


Figura D.11: Recepción correcta de un correo con RSA en *gmail*.

Anexo E

Diagrama de planificación

La Figura E.1 muestra el diagrama de Gantt, que expone el tiempo de dedicación previsto para las distintas tareas a lo largo del desarrollo del trabajo de fin de grado. La Figura E.2 muestra la barra temporal con las actividades principales del diagrama de Gantt.



	Modo de	Nombre de tarea	Duración	Comienzo	Fin
1		**Estudio del estado del arte**	**34 días**	**mié 15/10/14**	**dom 30/11/14**
2		Lectura aspectos teóricos criptografía	18 días	mié 15/10/14	vie 07/11/14
3		Lectura documentación JPBC	15 días	lun 10/11/14	dom 30/11/14
4		Estudio del estado del arte Completo	0 días	dom 30/11/14	dom 30/11/14
5		**Análisis y Diseño**	**31 días**	**lun 01/12/14**	**sáb 10/01/15**
6		Análisis de requisitos	13 días	lun 01/12/14	mié 17/12/14
7		Diseño	17 días	vie 19/12/14	sáb 10/01/15
8		Análisis y Diseño Completo	0 días	sáb 10/01/15	sáb 10/01/15
9		**Codificación**	**58 días**	**dom 11/01/15**	**mar 31/03/15**
10		Implementación criptografía RSA	5 días	lun 12/01/15	vie 16/01/15
11		Implementación aplicación de correo	10 días	vie 16/01/15	jue 29/01/15
12		Implementación criptografía IBE	13 días	jue 29/01/15	lun 16/02/15
13		Integración con la aplicación de correo	17 días	mar 17/02/15	mié 11/03/15
14		Implementación aplicación PKG	4 días	mar 10/03/15	dom 15/03/15
15		Comunicación entre correo y PKG	12 días	lun 16/03/15	mar 31/03/15
16		Codificación Completo	0 días	mar 31/03/15	mar 31/03/15
17		**Pruebas**	**22 días**	**mié 01/04/15**	**jue 30/04/15**
18		Pruebas unitarias	8 días	mié 01/04/15	vie 10/04/15
19		Pruebas de integración	14 días	lun 13/04/15	jue 30/04/15
20		Pruebas Completo	0 días	jue 30/04/15	jue 30/04/15
21		**Informe**	**23 días**	**vie 01/05/15**	**mar 02/06/15**
22		Redacción de la memoria	14 días	vie 01/05/15	mié 20/05/15
23		Revisión y corrección de la memoria	9 días	jue 21/05/15	mar 02/06/15
24		Informe Completo	0 días	mar 02/06/15	mar 02/06/15

Figura E.1: Diagrama de Gantt.

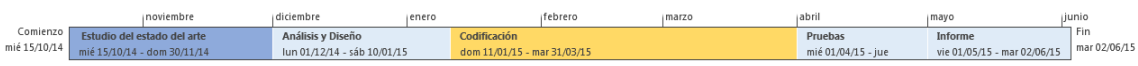


Figura E.2: Escala de tiempos.

Bibliografía

[1] L. R. Lapeña and Juan Sidrach, "Historia de la criptografía," Tech. Rep., 2014.

[2] E. Commission, "EU Cybersecurity plan to protect open internet and online freedom and opportunity - Cyber Security strategy and Proposal for a Directive," http://ec.europa.eu/digital-agenda/en/news/eu-cybersecurity-plan-protect-open-internet-and-online-freedom-and-opportunity-cyber-security, Digital Agenda for Europe, Tech. Rep., February 2013.

[3] D. Moody, R. Peralta, R. Perlner, A. Regenscheid, A. Roginsky, and L. Chen, "Report on pairing-based cryptography," *Journal of Research of the National Institute of Standards and Technology*, vol. 120, p. 11, 2015.

[4] A. De Caro and V. Iovino, "jpbc: Java pairing based cryptography," in *Proceedings of the 16th IEEE Symposium on Computers and Communications, ISCC 2011*, Kerkyra, Corfu, Greece, June 28 - July 1, 2011, pp. 850–855.

[5] K. Bak, "Certificateless Cryptography," Tech. Rep., 2009.

[6] N. Ferguson and B. Schneier, *Practical cryptography*. Wiley New York, 2003, vol. 141.

[7] A. Whitten and J. D. Tygar, "Why johnny can't encrypt: A usability evaluation of pgp 5.0." in *Usenix Security*, vol. 1999, 1999.

[8] S. Sheng, L. Broderick, C. A. Koranda, and J. J. Hyland, "Why johnny still can't encrypt: evaluating the usability of email encryption software," in *Symposium On Usable Privacy and Security*, 2006.

[9] S. Clark, T. Goodspeed, P. Metzger, Z. Wasserman, K. Xu, and M. Blaze, "Why (special agent) johnny (still) can't encrypt: A security analysis of the apco project 25 two-way radio system." in *USENIX Security Symposium*. Citeseer, 2011.

[10] I.-T. R. X.509, "Information technology – open systems interconnection – the directory: Public-key and attribute certificate frameworks," Tech. Rep., 11 2008.

[11] Comodo, "What is PKI? - A Complete overview." [Online]. Available: https://www.comodo.com/resources/small-business/digital-certificates1.php

[12] C. Ellison and B. Schneier, "Ten risks of PKI: What you're not being told about public key infrastructure," *Comput Secur J*, vol. 16, no. 1, pp. 1–7, 2000.

[13] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Advances in cryptology*. Springer, 1985, pp. 47–53.

[14] F. Bao, R. H. Deng, and H. Zhu, "Variations of Diffie-Hellman problem," in *Information and Communications Security*. Springer, 2003, pp. 301–312.

[15] A. Joux and K. Nguyen, "Separating decision diffie-hellman from diffie-hellman in cryptographic groups, 2001," *Manuscript. Available from eprint. iacr. org.*

[16] D. Boneh, "The decision diffie-hellman problem," in *Algorithmic number theory*. Springer, 1998, pp. 48–63.

[17] S. Lang, *Elliptic functions*. Springer, 1987.

[18] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," in *Advances in Cryptology—CRYPTO 2001*. Springer, 2001, pp. 213–229.

[19] G. Frey, M. Muller, and H.-G. Ruck, "The tate pairing and the discrete logarithm applied to elliptic curve cryptosystems," *Information Theory, IEEE Transactions on*, vol. 45, no. 5, pp. 1717–1719, 1999.

[20] M. F. M. Vilicich, "Diseño e Implementación de un Esquema de Encriptación y Firmas Basado en Identidad para Dispositivos BUG," 2010. [Online]. Available: http://www.captura.uchile.cl/jspui/handle/2250/13565

[21] Wikipedia, "ID-Based Encryption." [Online]. Available: http://en.wikipedia.org/wiki/ID-based_encryption

[22] R. Rivest, "Rivest Lecture 25 6.89: Pairing-Based Cryptography," pp. 1–5, 2004.

[23] B. Schneier, "The risks of key recovery, key escrow, and trusted third party encryption," 2013.

[24] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM journal on computing*, vol. 26, no. 5, pp. 1484–1509, 1997.

[25] D. J. Bernstein, "Introduction to post-quantum cryptography," in *Post-quantum cryptography*. Springer, 2009, pp. 1–14.

[26] G. Ateniese and B. de Medeiros, "Identity-based chameleon hash and applications," in *Financial Cryptography*. Springer, 2004, pp. 164–180.

[27] Proofpoint, "Encryption Made Easy: The Advantages of Identity Based Encryption," Tech. Rep.

[28] IETF, "Internet X.509 Public Key Infrastructure Certificate and CRL Profile." [Online]. Available: https://www.ietf.org/rfc/rfc2459

[29] Wikipedia, "Modelo-Vista-Controlador." [Online]. Available: http://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93controlador

[30] "Mendeley." [Online]. Available: https://www.mendeley.com/

[31] A. De Caro, V. Iovino, and G. Persiano, "Fully secure anonymous HIBE and secret-key anonymous IBE with short ciphertexts," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6487 LNCS, pp. 347–366, 2010.

[32] B. Lynn, "The pairing-based cryptography (PBC) library." [Online]. Available: http://crypto.stanford.edu/pbc

[33] "Bouncy Castle," Inc, Legion of the Bouncy Castle. [Online]. Available: https://www.bouncycastle.org/

[34] M. Green, "Attack of the week: FREAK (or 'factoring the NSA for fun and profit')," A Few Thoughts on Cryptographic Engineering. [Online]. Available: http://blog.cryptographyengineering.com/2015/03/attack-of-week-freak-or-factoring-nsa.html

[35] Google Inc., "This POODLE bites: exploiting the SSL 3.0 fallback," Google Online Security Blog. [Online]. Available: http://googleonlinesecurity.blogspot.com.es/2014/10/this-poodle-bites-exploiting-ssl-30.html

[36] Logjam, "The LogJam Attack." [Online]. Available: https://weakdh.org/

[37] Ontinet, "GHOST: la nueva (y grave) vulnerabilidad para Linux. ¡Actualiza sin falta tu sistema!" [Online]. Available: http://blogs.protegerse.com/laboratorio/2015/01/28/ghost-la-nueva-y-grave-vulnerabilidad-para-linux-actualiza-sin-falta-tu-sistema/

[38] David A. Wheeler, "Shellshock." [Online]. Available: http://www.dwheeler.com/essays/shellshock.html

Índice