



UNIVERSIDAD DE LA SERENA
FACULTAD DE CIENCIAS
DEPARTAMENTO DE MATEMÁTICAS
ESCUELA DE INGENIERÍA EN COMPUTACIÓN



SICFA

“SISTEMA DE CIFRADO Y FIRMADO DIGITAL DE ARCHIVOS”

Por

Sergio Mauricio Pastén Rivera

Andrés Armando Vega Vega

Profesor Guía

Dr. Eric R. Jeltsch Figueroa.

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO EN COMPUTACIÓN

JULIO 2010

Dedicatoria

Primero a mis padres, por darme la posibilidad de estudiar y estar a mi lado en todo momento. Segundo, a mis abuelos, por haber creído en mí y apoyarme siempre. Por último, a todas aquellas personas que de una manera u otra hicieron esto posible.

Andrés Vega Vega.

A mi familia, por su apoyo de manera incondicional.

Sergio Pastén Rivera.

Resumen

Este documento, presenta parte de la información obtenida del proceso de desarrollo de una herramienta de software que usa una combinación de criptografía fuerte de clave pública y criptografía simétrica para proveer servicios de seguridad para comunicaciones electrónicas y almacenamiento de datos. Estos servicios incluyen confidencialidad, integridad, autenticación de origen de datos y no-repudio de origen.

Abstract

This document, presents some of the information from the process of developing software tool that uses a combination of strong public – key and symmetric cryptography to provide security services for electronic communications and data storage. These services include confidentiality, data integrity, data origin authentication and non-repudiation of origin.

Índice General

Introducción.....	1
Capítulo 1 Definición del problema.....	3
1.1 Marco Teórico.....	4
1.1.1 Amenaza	4
1.1.2 Ataques.....	5
1.1.3 Servicios de seguridad	6
1.1.4 Mecanismos o técnicas de seguridad.....	7
1.1.5 Criptografía	8
1.1.6 Aplicaciones Criptográficas	18
1.2 Contexto	21
1.3 Definición y delimitación del problema	22
1.4 Objetivos Generales y Específicos.....	24
1.4.1 Objetivos Generales	24
1.4.2 Objetivos específicos.....	24
Capítulo 2 SICFA-DTX235: Análisis del sistema	27
2.1 Ciclo de vida	28
2.2 Generalidades del proyecto	28
2.2.1 Descripción del proyecto	28
2.2.2 Planificación del proyecto	31
2.3 Requerimientos del proyecto	33
2.3.1 Requerimientos para Administrador de claves.....	34
2.3.2 Requerimientos para Llavero Personal o Gestor de claves seguras.....	38
2.3.3 Requerimientos para Complementos Simétricos	39
2.3.4 Requerimientos para Editor SICFA-DTX235	40
2.3.5 Requerimientos para Mecanismos de seguridad	40

Capítulo 3 SICFA-DTX235: Diseño del sistema.....	43
3.1 Descripción conceptual de la solución	43
3.1.1 Diagrama de flujo de Iniciar sistema.....	45
3.1.2 Diagrama de flujo de Administrador de claves (AC)	45
3.1.3 Diagrama de flujo de Llavero personal o Gestor de claves seguras (GC).....	46
3.1.4 Diagrama de flujo de Complementos simétricos (CS).....	46
3.1.5 Diagrama de flujo de Editor SICFA (ESIC)	46
3.2 Descripción de los subsistemas	52
3.2.1 ADMINISTRADOR DE CLAVES (AC)	52
3.2.2 MECANISMOS DE SEGURIDAD (MS)	56
3.3 Diagramas de secuencias.....	65
3.3.1 Diagrama de secuencias para: “Cargar claves propias”	65
3.3.2 Diagrama de secuencias para: “Exportar clave(s) pública(s)”	68
3.3.3 Diagrama de secuencias para: “Generar pareja de claves subordinadas”	70
3.3.4 Diagrama de secuencias para: “Cifrar archivo con algoritmo simétrico”	72
3.3.5 Diagrama de secuencias para: “Cifrar contenido del editor”	74
3.3.6 Diagrama de secuencias para: “Copiar datos al portapapeles”.....	76
Capítulo 4 SICFA: Implementación del Sistema	78
4.1 Herramientas y técnicas utilizadas	78
4.1.1 Lenguaje de programación Java	78
4.1.2 Entorno de desarrollo NetBeans IDE 6.0	78
4.1.3 Herramientas de Diagramas UML: Visual Paradigm.....	78
4.1.4 Base64	79
4.2 Librerías utilizadas	79
4.2.1 JCA (Java Cryptography Architecture)	79
4.2.2 JCE (Java Cryptography Extension).....	79
4.2.3 Compresión en Java: Paquete java.util.zip	81
4.3 Módulos Principales.....	82

4.3.1 ADMINISTRADOR DE CLAVES (AC)	82
4.3.2 LLAVERO PERSONAL O GESTOR DE CLAVES SEGURAS (GC)	96
4.3.3 COMPLEMENTOS SIMÉTRICOS (CS).....	99
4.3.4 EDITOR SICFA (ESIC).....	101
4.3.5 MECANISMOS DE SEGURIDAD (MS)	103
Capítulo 5 SICFA-DTX235: Pruebas	108
5.1 Pruebas de validación.....	108
5.2 Pruebas de rendimiento.....	114
Capítulo 6 Conclusiones.....	120
Con respecto a la investigación	120
Con respecto al desarrollo del software	122
Mejoras a SICFA-DTX235 y trabajos futuros	122
Glosario.....	124
Bibliografía	126
Anexos	130

Índice de tablas

Tabla 1.1: Mecanismos y servicios de seguridad	8
Tabla 1.2: Uso de las claves públicas/privadas para confidencialidad y firma digital.	19
Tabla 3.1: Tabla ASCII codificación Base64.....	64
Tabla 4.1: Descripción de las clases de JCA 1.6.....	80
Tabla 4.2: Paquete java.util.zip.....	81
Tabla 5.1: Plantilla de casos de prueba para pruebas de validación.	108
Tabla 5.2: Prueba de validación 03	109
Tabla 5.3: Prueba de validación 07	110
Tabla 5.4: Prueba de validación 09	110
Tabla 5.5: Prueba de validación 25	111
Tabla 5.6: Prueba de validación 32	112
Tabla 5.7: Prueba de validación 39	112
Tabla 5.8: Prueba de validación 45.	113
Tabla 5.9: Características de la máquina donde se realizaron las pruebas de rendimiento.	114
Tabla 5.10: Descripción del tamaño de los archivos.....	114
Tabla 5.11: Resultados de las mediciones (mseg) de las operaciones de cifrado SICFA.	114
Tabla 5.12: Estimaciones de tiempo (mseg) para el cifrado de 1 MB en SICFA-DTX235.	115
Tabla 5.13: Resultados de las mediciones (mseg) de las operaciones de descifrado SICFA.	116
Tabla 5.14: Estimaciones de tiempo (mseg) para el descifrado de 1 MB en SICFA-DTX235.....	116
Tabla 5.15: Resultados de mediciones de la generación de resúmenes MDC SICFA-DTX235.....	117
Tabla 5.16: Resultados de mediciones de la carga de registros en el llavero personal. .	118

Índice de figuras

Figura 1.1: Tipos de ataques.....	6
Figura 1.2: Esquema de funcionamiento de un criptosistema.....	11
Figura 1.3: Esquema de funcionamiento de un sistema simétrico.....	12
Figura 1.4: Transmisión de información empleando criptosistema asimétrico.	13
Figura 1.5: Combinación de criptosistemas asimétricos y simétricos.	15
Figura 1.6: Estructura iterativa de una función resumen.....	16
Figura 1.7: Códigos de autentificación de mensajes.....	17
Figura 1.8: Esquema de una firma digital basada en funciones resumen y algoritmos de cifrado asimétricos.	20
Figura 1.9: Combinación de criptosistemas asimétricos y simétricos con firma digital.....	21
Figura 3.1: Modelo conceptual de SICFA-DTX235.....	44
Figura 3.2: Símbolos de representación.....	45
Figura 3.3: Diagrama de flujo de Iniciar sistema.....	47
Figura 3.4: Diagrama de flujo de Administrador de claves (AC).....	48
Figura 3.5: Diagrama de flujo de Llavero personal o Gestor de claves seguras (GC).....	49
Figura 3.6: Diagrama de flujo de Complementos simétricos (CS).....	50
Figura 3.7: Diagrama de flujo de Editor SICFA (ESIC).	51
Figura 3.8: Iconos representativos de paquetes.....	52
Figura 3.9: Estructura de archivos: A: Anillo de claves públicas, B: anillo de claves privadas, C: anillo de confianzas.	54
Figura 3.10: Listas doblemente enlazadas con paquetes del sistema.	55
Figura 3.11: Estructura árbol para la presentación de datos por pantalla.	55
Figura 3.12: Datos contenidos en cada nodo de la estructura árbol.	56
Figura 3.13: Esquema de procedimiento de cifrado híbrido de archivo.....	57
Figura 3.14: Esquema de procedimiento de descifrado híbrido de archivo.....	58
Figura 3.15: Esquema de procedimiento de generación de firma digital de archivo.....	59
Figura 3.16: Esquema de procedimiento de verificación de firma digital de archivo.	60
Figura 3.17: Esquema de procedimiento de cifrado (A) y descifrado simétrico PBE (B) de archivo.	61

Figura 3.18: Esquema de procedimiento de generación (A) y verificación (B) de valor hash MAC de archivo.....	62
Figura 3.19: Esquema de procedimiento de generación de firma digital, y cifrado, de documento de editor SICFA.	63
Figura 3.20: Proceso de conversión a Base64 de la palabra Sol.....	64
Figura 3.21: Diagrama de secuencias para: “Cargar claves propias”.....	67
Figura 3.22: Diagrama de secuencias para: “Exportar clave(s) pública(s)”.....	69
Figura 3.23: Diagrama de secuencias para: “Generar pareja de claves subordinadas”....	71
Figura 3.24: Diagrama de secuencias para: “Cifrar archivo con algoritmo simétrico”.....	73
Figura 3.25: Diagrama de secuencias para: “Cifrar contenido del editor” ..	75
Figura 3.26: Diagrama de secuencias para: “Copiar datos al portapapeles”.....	77
Figura 4.1: Interfaz gráfica para iniciar sistema.	88
Figura 4.2: Interfaz gráfica para ingreso de datos de pareja de claves principal.....	89
Figura 4.3: Interfaz gráfica del Administrador de claves.	90
Figura 4.4: Interfaz gráfica para el ingreso de datos de pareja de claves subordinadas. ..	90
Figura 4.5: Interfaz gráfica para la edición de datos de una pareja de claves principal....	92
Figura 4.6: Interfaz gráfica para la exportación de claves públicas.....	94
Figura 4.7: Interfaz gráfica para validación de identidad de usuario.	95
Figura 4.8: Interfaz gráfica de opciones de configuración del sistema.	96
Figura 4.9: Interfaz gráfica de Llavero personal o Gestor de claves seguras.....	98
Figura 4.10: Interfaz gráfica para el ingreso de datos de un nuevo registro.....	99
Figura 4.11: Interfaz gráfica de Complementos Simétricos.....	100
Figura 4.12: Interfaz gráfica para el ingreso de datos de generación de un archivo auto-descifrable.....	100
Figura 4.13: Interfaz gráfica para el ingreso de datos de borrado seguro de archivos.	101
Figura 4.14: Interfaz gráfica del Editor SICFA.	102
Figura 4.15: Interfaz gráfica para la selección de clave pública para una operación de generación de firma digital.	103
Figura 4.16: Mensaje digitado en editor SICFA y su salida en formato ASCII.....	105
Figura 4.17: Firma digital de archivo digital, representada en formato ASCII.....	106
Figura 5.1: Gráfico de mediciones de las operaciones de cifrado SICFA-DTX235.	115
Figura 5.2: Gráfico de mediciones de las operaciones de descifrado SICFA-DTX235...	116

Figura 5.3: Gráfico de mediciones de la generación de resúmenes MDC SICFA-DTX235.....	117
Figura 5.4: Gráfico de mediciones de la carga de registros en el llavero personal.....	118
Figura 6.1: Importancia de la concienciación de los usuarios en seguridad informática. Encuesta por Eset a 947 profesionales de seguridad informática.....	121

Introducción

La criptografía como medio de proteger la información personal es un arte tan antiguo como la propia escritura. Como tal, permaneció durante siglos vinculada muy estrechamente a los círculos militares y diplomáticos, puesto que eran los únicos que en principio tenían auténtica necesidad de ella.

En la actualidad la situación ha cambiado drásticamente: el desarrollo de las comunicaciones electrónicas, unido al uso masivo y generalizado de los computadores, hace posible la transmisión y almacenamiento de grandes flujos de información confidencial que es necesario proteger. Es entonces cuando la Criptografía pasa de ser una exigencia de minorías a convertirse en una necesidad real del hombre de la calle, que ve en esta falta de protección de sus datos privados una amenaza para su propia intimidad.

El cifrado de los datos nos va a permitir desde proteger nuestro correo personal para que ningún curioso lo pueda leer, hasta controlar el acceso a nuestros documentos o archivos de forma que sólo personas autorizadas puedan examinar (o lo que quizás es más importante, modificar) su contenido.

En este trabajo se presenta el proceso de desarrollo e implementación de una herramienta de software, la cual utilizando sistemas y técnicas criptográficas, así como una de sus aplicaciones: la firma digital, permite proporcionar al usuario final una serie de servicios de seguridad necesarios para la protección de los datos e información que éste almacena en algún medio de almacenamiento y/o transmite a través de una red de comunicación. Los servicios de seguridad proporcionados son: confidencialidad, integridad de los datos, autenticación de origen de datos y no repudio de origen.

La necesidad de realizar este trabajo se debe a los siguientes motivos:

Régimen de estudio: La carrera de Ingeniería en Computación de la Universidad de La Serena en su malla curricular no tiene presente ningún curso que trate propiamente tal sobre el tema de seguridad informática y criptografía.

Robo de identidad: Hoy en día Internet y el robo de datos privados e información confidencial, permiten que otra persona se haga pasar por nosotros y así robarnos dinero, comprar con nuestras tarjetas y enviar correos electrónicos muchas veces dañando nuestra imagen. Lo anterior nos hace pensar y reflexionar en lo importante que resulta el proteger nuestros datos e información, que usamos principalmente para ingresar a sistemas informáticos, cuentas de correo, realizar transacciones comerciales y movimientos de dinero. Es por eso que es necesaria una herramienta que nos brinde la protección de claves y envío de texto protegido y firmado en correos electrónicos.

Creación de una aplicación: Crear una herramienta alternativa a las existentes actualmente como PGP(Pretty Good Privacy) y GPG(GNU Privacy Guard) por nombrar algunas. Esto con el objetivo de brindar una base y dar apoyo para que futuros estudiantes se animen a implementar una herramienta de este tipo adaptándola a sus necesidades propias.

Investigación: Abordar temas importantes como seguridad informática y criptografía e investigar sobre ellos, saber por ejemplo por qué es tan importante la seguridad en nuestros días, a qué amenazas nos enfrentamos nosotros y nuestros equipos informáticos, qué es la criptografía, qué técnicas criptográficas existen y conocer el funcionamiento de las aplicaciones criptográficas como firmas digitales, certificados y protocolos de comunicación segura.

El desarrollo de este proyecto está organizado en los siguientes capítulos:

El capítulo 1, presenta el marco teórico en el que se sustenta todo el desarrollo del proyecto; se define el problema, sus limitaciones y beneficios. Además, se detallan los objetivos generales y específicos del trabajo.

El capítulo 2, trata el análisis del problema a solucionar; se define la meta y los objetivos a alcanzar; se presenta el ciclo de vida utilizado. Además, se detallan los requerimientos principales de la herramienta de software.

El capítulo 3, contempla la etapa de diseño de software; se presenta un marco conceptual de la solución; se detallan los diagramas de flujo de las funcionalidades principales. Además, se muestran los diagramas de secuencias con su respectiva descripción.

El capítulo 4, cubre la etapa de implementación del desarrollo del software; se destacan las librerías y herramientas utilizadas en la codificación. Además, se presenta el detalle de las clases y métodos relevantes para el funcionamiento del software.

El capítulo 5, muestra las pruebas funcionales y de rendimiento realizadas al software con sus respectivas conclusiones.

El capítulo 6, presenta las conclusiones obtenidas a partir de todo el proceso de desarrollo del proyecto y se proponen algunas mejoras para futuras versiones de la aplicación.

Capítulo 1 | Definición del problema

“Ante el miedo e inseguridad de un enemigo poderoso, sólo aparentemente, podemos sobreponernos y superar nuestro propio miedo e inseguridad”

Wallraff, Günter

Muchas de las actividades que se realizan en los países desarrollados dependen en mayor o en menor medida de sistemas y redes informáticas. El crecimiento de Internet (véase [WEB-15]) y de los servicios telemáticos (comercio electrónico, servicios multimedia de banda ancha, administración electrónica, herramientas de comunicación como el correo electrónico o la videoconferencia) ha contribuido a popularizar aún más el uso de la informática y de las redes de computadores, hasta el punto de que en el presente no se circunscriben al ámbito laboral y profesional, sino que incluso se han convertido en un elemento cotidiano en muchos hogares, con un creciente impacto en las propias actividades de comunicación y de ocio de los ciudadanos.

La información constituye un recurso que en muchos casos no se valora adecuadamente por su intangibilidad (situación que no se produce con los equipos informáticos, la documentación impresa o las aplicaciones) y, además, las medidas de seguridad no contribuyen a mejorar la productividad de los sistemas y redes informáticas, sino más bien, todo lo contrario, ya que pueden reducir el rendimiento de los equipos y las aplicaciones, por lo que las organizaciones y las personas son reticentes a dedicar recursos a esta tarea. Lo anterior nos hace reflexionar y pensar en lo importante que es el tema de la seguridad de la información para concienciar y estimular a las personas a que utilicen herramientas de seguridad, necesarias para la protección de los datos e información que los sistemas informáticos almacenan, procesan y transmiten.

Es importante proteger los datos e información, que almacenamos en nuestro computador, que enviamos a través de un canal inseguro (red) y que usamos para ingresar a sistemas informáticos, ingresar a cuentas de correo, realizar transacciones comerciales y movimientos de dinero. Lo anterior conlleva a la creciente necesidad, por parte de los usuarios, de una herramienta de software que permita: el cifrado de información, contenida en formato digital, para asegurar la confidencialidad de los datos; firmado digital de documentos o archivos, para asegurar la integridad de los datos; autenticación de origen de datos y no repudio de origen; envío de información firmada digitalmente y/o cifrada a través del correo electrónico; protección de claves o contraseñas para el acceso a entornos informáticos.

Existen herramientas que se acercan a las necesidades descritas anteriormente como por ejemplo: PGP, GPG y CryptoForge. Si bien PGP es la herramienta más conocida a nivel mundial, ésta carece de un gestor de claves seguras y contraseñas para ingresar a sistemas informáticos y, un editor de texto que evite el almacenamiento de mensajes en claro en el disco. Actualmente, PGP sólo incorpora las funcionalidades básicas (cifrado asimétrico, cifrado simétrico, firma digital) en su versión gratuita, dejando para su versión pagada las funcionalidades más avanzadas (ejemplo, archivos auto-descifrables que se descifran de forma automática haciendo doble clic sobre él, sin necesidad de que el usuario de destino deba tener PGP instalado en su PC).

Con respecto a CryptoForge, que si bien ha ganado muchos premios y reconocimientos a nivel mundial, sólo implementa cifrado simétrico. Un sistema simétrico utiliza una única clave tanto en el proceso de cifrado como en el proceso de descifrado, lo que constituye un problema al momento de intercambiar la clave de cifrado en una comunicación.

Dentro de las alternativas disponibles, la que más se acerca a una protección efectiva de datos e información en formato digital y que además es gratuita, es GPG. Este proyecto implementa una herramienta de seguridad basada en las funcionalidades básicas de GPG (cifrado asimétrico, cifrado simétrico, firma digital) e incorpora algunas adicionales de las cuales se puede destacar: la administración de claves seguras y contraseñas, para acceder a otros sistemas informáticos; creación de archivo auto-descifrable, el cual al ser ejecutado descifra el contenido cifrado en su interior y restaura el archivo original; editor de texto, para el cifrado de mensajes sin que tengan que ser almacenados en claro en el disco duro.

En lo que resta del capítulo, se abordan los siguientes temas:

1. Marco teórico; se tratan las bases teóricas en las que se sustenta el desarrollo del proyecto y que permiten una mejor comprensión del documento.
2. Contexto; breve reseña histórica de la criptografía y el conjunto de algoritmos y técnicas que ésta ofrece para el desarrollo de herramientas de seguridad que proporcionen servicios de seguridad al usuario, necesarios para la protección de datos e información en formato digital.
3. Definición del problema y limitaciones; se dan a conocer los objetivos generales y específicos a lograr con este proyecto. Además, se detallan los beneficios y limitaciones de la herramienta.

1.1 Marco Teórico

1.1.1 Amenaza

Una **amenaza** es un peligro potencial, que puede afectar a un sistema en un determinado momento o dadas ciertas circunstancias. El peligro podría ser una persona (un cracker o un espía), una cosa (una pieza defectuosa del equipo), o un evento (un incendio o una inundación) que podrían explotar una vulnerabilidad del sistema.

Las amenazas se pueden clasificar en 3 categorías: naturales, accidentales o intencionales.

01. **Amenazas naturales.** Son aquellas que tienen una probabilidad de ocurrencia menor. Por ejemplo, terremotos, inundaciones, incendios, sequías, tsunamis, etcétera. Sin embargo, el hecho que sean amenazas poco probables no implica que contra ellas no se tomen unas medidas básicas, ya que si se produjeran generarían los mayores daños.
02. **Amenazas accidentales:** Son aquellas que aparecen de forma no premeditada. Estas van desde un operador que derrama una taza de café sobre un computador hasta un usuario que tropieza con un cable de alimentación de un servidor y lo desconecta de la línea eléctrica, pasando por temas como el borrado accidental de datos, hardware defectuoso, errores de programación, permisos inadecuados, falta de formación por parte del personal, etcétera.

03. **Amenazas intencionales:** Presuponen la participación maliciosa de un sujeto o entidad que pretende hacer uso indebido de los componentes de un sistema informático y de los datos e información que éstos almacenan, procesan y transmiten. Como hurto de información, sabotaje, espionaje, fraude, vandalismo, etcétera.

1.1.2 Ataques

Un **ataque** es un asalto a la seguridad del sistema que deriva desde una amenaza intencional, es decir, un acto inteligente que es un atentado deliberado (especialmente en el sentido de un método o técnica) para evadir los servicios de seguridad y violar la política de seguridad de un sistema.

01. **Activo vs. Pasivo:** Un ataque activo intenta alterar los recursos del sistema o afectar su operación. Un ataque pasivo intenta aprender o hacer uso de información del sistema, pero no afecta los recursos del sistema.
02. **Interno vs. Externo:** Un ataque interno es un ataque iniciado por una entidad dentro del perímetro de seguridad, es decir, una entidad que está autorizada a acceder a los recursos del sistema, pero los usa de forma no autorizada por quienes están cargo de dar autorización para acceder a ellos. Un ataque externo es iniciado desde afuera del perímetro, por un usuario, ilegítimo o no autorizado, del sistema. En Internet, los atacantes externos potenciales van desde bromistas amateurs hasta criminales organizados, terroristas internacionales y gobiernos hostiles [RFC 2828].

Tipos de ataques

Los ataques se pueden clasificar en 4 categorías:

01. **Interrupción:** Un recurso del sistema se destruye o no llega a estar disponible o se inutiliza; esta es una agresión de disponibilidad. Ejemplos: destrucción del hardware; borrado de programas, datos; fallos en el sistema operativo; etcétera. Su detección es inmediata.
02. **Intercepción:** Un elemento no autorizado consigue acceder a un recurso. Esta es una agresión a la confidencialidad. El elemento no autorizado puede ser una persona, una entidad, un programa o un ordenador. Ejemplos: intervención de las líneas, copia ilícita de archivos o programas, etcétera. Su detección es difícil a veces no deja huellas.
03. **Modificación:** Un elemento no autorizado no sólo gana acceso, sino que modifica el recurso; esta es una agresión a la integridad. Ejemplos: cambios de valores en un base de datos, alteración de un programa para que funcione de una forma diferente, modificación de elementos de hardware, etcétera. Su detección es difícil según las circunstancias.
04. **Generación:** Una parte no autorizada inserta objetos falsos en el sistema. Esta es una agresión a la autenticidad. Ejemplos: añadir transacciones en red o incorporación de registros a una base de datos. Su detección es difícil.

En la figura 1.1 aparecen representados las cuatro clases en las que pueden clasificarse los ataques contra la seguridad.

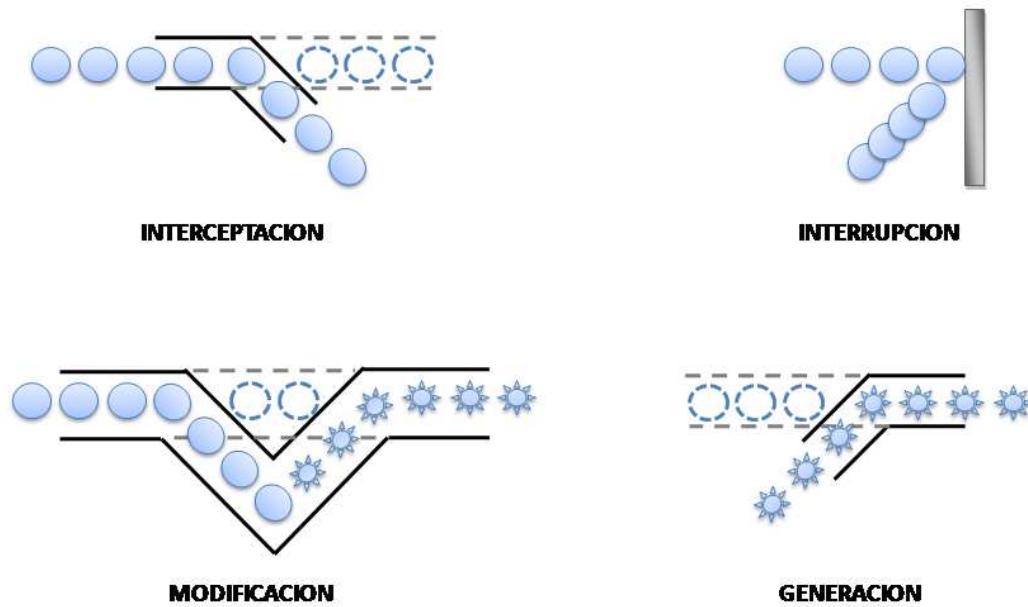


Figura 1.1: Tipos de ataques.

El ataque por interceptación se considera “pasivo”, mientras que los otros tres tipos se clasifican como ataques “activos”. Para poder detectar y prevenir estos ataques, los sistemas informáticos deben incorporar mecanismos que proporcionen servicios de seguridad.

1.1.3 Servicios de seguridad

Los servicios de seguridad son la respuesta a las amenazas, y responden al qué se debe hacer para satisfacer los requerimientos de seguridad de una persona u organización y hacer frente a las amenazas. Un servicio hace frente a ataques contra la seguridad empleando uno varios mecanismos.

Los servicios de seguridad pueden resumirse en:

Confidencialidad

Mediante este servicio o función de seguridad se garantiza que cada mensaje transmitido o almacenado en un sistema informático sólo podrá ser leído por su legítimo destinatario. Si dicho mensaje cae en manos de terceras personas, éstas no podrán acceder al contenido del mensaje original. Por lo tanto, este servicio pretende garantizar la confidencialidad de los datos almacenados en un equipo, de los datos guardados en dispositivos de backup y/o de los datos transmitidos a través de redes de comunicaciones.

Integridad

La función de integridad se encarga de garantizar que un mensaje o archivo no ha sido modificado desde su creación o durante su transmisión a través de una red informática. De este modo, es posible detectar si se ha añadido o eliminado algún dato en un mensaje o archivo almacenado, procesado o transmitido por un sistema o red informática.

No repudio

El objeto de este servicio de seguridad, consiste en implementar un mecanismo probatorio que permita demostrar la autoría y envío de un determinado mensaje, de tal modo que el usuario que lo ha creado y enviado a través del sistema, no pueda posteriormente negar esta circunstancia, situación que también se aplica al destinatario del envío. Este es un aspecto de especial importancia en las transacciones comerciales y que permite proporcionar a los compradores y vendedores una seguridad jurídica que va a estar soportada por este servicio.

En un sistema informático, por lo tanto, se puede distinguir entre la no repudiación de origen y la no repudiación de destino.

Autenticación

La autenticación garantiza que la identidad del creador de un mensaje o documento es legítima, es decir, gracias a esta función, el destinatario de un mensaje podrá estar seguro de que su creador es la persona que figura como remitente de dicho mensaje.

Asimismo, también se puede hablar de la autenticidad de un equipo que se conecta a una red o intenta acceder a un determinado servicio. En este caso, la autenticación puede ser unilateral, cuando sólo se garantiza la identidad del equipo (usuario o terminal que se intenta conectar a la red) o mutua, en el caso de que la red o el servidor también se autentica de cara al equipo, usuario o terminal que establece la conexión. Además de verificar la identidad de nuestro interlocutor también se verifica la integridad de los mensajes que de él recibimos.

Disponibilidad

La disponibilidad de un sistema informático también es una cuestión de especial importancia para garantizar el cumplimiento de sus objetivos, ya que se debe diseñar un sistema lo suficientemente robusto frente a ataques e interferencias como para garantizar su correcto funcionamiento, de manera que pueda estar permanentemente a disposición de los usuarios que deseen acceder a sus servicios.

Dentro de la disponibilidad también se debe considerar la recuperación del sistema frente a posibles incidentes de seguridad, así como frente a desastres naturales o intencionados (incendios, inundaciones, sabotajes, etcétera).

Se debe tener en cuenta que de nada sirven los demás servicios de seguridad si el sistema informático no se encuentra disponible para que pueda ser utilizado por sus legítimos usuarios y propietarios.

1.1.4 Mecanismos o técnicas de seguridad

Un mecanismo o técnica de seguridad es un procedimiento diseñado para detectar, prevenir o recuperarse de un ataque contra la seguridad.

Existe una gran variedad de mecanismos de seguridad diseñados para contrarrestar las amenazas enumeradas, si bien ninguno de ellos por sí solo es capaz de hacer frente a todas las clases de ataque. El presente trabajo hace uso de un cierto tipo de técnicas de seguridad: los algoritmos y mecanismos criptográficos.

La tabla 1.1 muestra la relación entre servicios de seguridad y mecanismos criptográficos.

Mecanismo Servicio	Cifrado	Firma digital	Funciones de integridad	Funciones de autenticación
Autenticación de origen de datos	✓	✓		✓
Confidencialidad	✓			
Integridad de datos	✓	✓	✓	✓
No repudio con prueba de origen		✓		✓

Tabla 1.1: Mecanismos y servicios de seguridad.

Tanto los sistemas simétricos de cifrado como los más novedosos, basados en algoritmos de clave pública, ofrecen soluciones para los distintos servicios de seguridad.

1.1.5 Criptografía

La **Criptografía** es la ciencia que se encarga de estudiar las distintas técnicas empleadas para transformar (cifrar) la información y hacerla irreconocible a todos aquellos usuarios no autorizados de un sistema informático, de modo que sólo los legítimos propietarios puedan recuperar (descifrar) la información original.

La finalidad de la criptografía se ha ido modificando con el paso de los años. En un principio esta ciencia sólo era utilizada para preservar la confidencialidad de la información, es decir para garantizar su accesibilidad sólo para los usuarios autorizados. Sin embargo, actualmente esta faceta de la criptografía comparte protagonismo con otras dedicadas a proteger la autenticidad e integridad de la información.

El **Criptoanálisis** es la ciencia que se ocupa de estudiar herramientas y técnicas que permitan descifrar los códigos y vencer los sistemas de protección definidos por la Criptografía.

Por último, a la ciencia de inventar sistemas de cifrado de la información (Criptografía) y de romper su seguridad (Criptoanálisis) se le conoce colectivamente con el término de **Criptología**.

1.1.5.1 Terminología y conceptos básicos

Lo que sigue es una lista de términos y conceptos básicos que irán apareciendo a lo largo del documento.

Dominios y codominios del cifrado

A denota un conjunto finito llamado **alfabeto**. Por ejemplo A = {0, 1}, el alfabeto binario, es un alfabeto frecuentemente usado. Nótese que todo alfabeto puede ser codificado en términos del alfabeto binario.

M denota un conjunto llamado el **espacio de mensajes**. M consiste de cadenas de símbolos desde un alfabeto. Un elemento de M es llamado texto claro o texto original.

C denota un conjunto llamado el **espacio de texto cifrado**. C consiste de cadenas de símbolos desde un alfabeto, el cual puede diferir del alfabeto para M. Un elemento de C es llamado criptograma.

Transformaciones de cifrado y descifrado

K denota un conjunto llamado el **espacio de claves**. Un elemento de K es llamado una clave.

Cada elemento $k \in K$ unívocamente determina una biyección $M \rightarrow C$, denotándose mediante E_k a la transformación de cifrado.

Para cada $k' \in K$, $D_{k'}$ denota un biyección $C \rightarrow M$ ($D_{k'}: C \rightarrow M$). $D_{k'}$ es llamada una transformación de descifrado.

El proceso de aplicar la transformación E_k al mensaje $m \in M$ es usualmente llamado cifrado de m.

El proceso de aplicar $D_{k'}$ al criptograma $c \in C$ es usualmente llamado descifrado de c.

Un criptosistema consiste de un conjunto $\{E_k: k \in K\}$ de transformaciones de cifrado y un conjunto correspondiente $\{D_{k'}: k' \in K\}$ de transformaciones de descifrado con la propiedad que para cada $k \in K$ hay una única clave $k' \in K$ tal que $D_{k'}(E_k(m)) = m$ para todo $m \in M$.

Las claves k y k' en la definición anterior son referidas como un par de claves y a veces denotadas por (k, k') . Nótese que k y k' podrían ser la misma.

Para construir un criptosistema se requiere seleccionar un espacio de mensajes M, un espacio de texto cifrado o criptograma C, un espacio de claves K, un conjunto de transformaciones de cifrado $\{E_k: k \in K\}$, y su correspondiente conjunto de transformaciones de descifrado $\{D_{k'}: k' \in K\}$.

Participantes en una comunicación

Una entidad o parte es alguien o algo que desea enviar, recibir, o manipular información; una entidad puede ser una persona, un computador, etcétera.

Un emisor es una entidad en una comunicación de 2 partes, el cual es el legítimo transmisor de la información.

Un receptor es una entidad en una comunicación de 2 partes, el cual es el recipiente deseado de la información.

Un intruso es una entidad en una comunicación de 2 partes, el cual no es el emisor ni el receptor, y que trata de derrotar los servicios de seguridad que están siendo proveídos para la protección de la información enviada entre emisor y receptor. Un intruso a menudo intentará jugar el rol del legítimo emisor o el legítimo receptor.

Canales

Un canal es un medio de transporte de información desde una entidad a otra.

Un canal inseguro es un medio de transporte de información en el cual un intruso puede reordenar, borrar, insertar o leer datos.

Un canal seguro es un medio de transporte de información en el cual un intruso no tiene la habilidad para reordenar, borrar, leer o insertar datos.

Tipos de claves

Clave secreta de sesión. La clave secreta de sesión es una clave simétrica generada de forma aleatoria, que se utiliza para cifrar datos que se transmiten entre dos partes.

Clave privada. Una clave privada es la mitad de un secreto de un par de claves que se utiliza en un algoritmo de clave pública. Las claves privadas suelen utilizarse para descifrar una clave de sesión, para firmar digitalmente un mensaje o para descifrar un mensaje que haya sido cifrado con la clave pública correspondiente.

Clave pública. Una clave pública es la mitad pública de un par de claves; pública y privada. Suele utilizarse para cifrar una clave de sesión simétrica o para verificar una firma digital. La clave pública se puede utilizar para cifrar un mensaje, garantizando de este modo que sólo pueda descifrar el mensaje la persona con la correspondiente clave privada.

Clave basada en una frase de paso (passphrase). Se utiliza para inicializar un algoritmo de cifrado simétrico. En concreto, se pasa la frase de paso por una función hash para obtener una clave binaria, que es la que se acaba dando al algoritmo de cifrado.

Clave segura (del sistema). Información generada por una máquina, en un formato no legible por un humano ya que se trata de una secuencia de bits o de símbolos de una determinada longitud y que controla las operaciones de cifrado y descifrado.

Contraseña (password). Reservado para la secuencia de información establecida por una persona mediante una determinada combinación de caracteres alfanuméricos que debe memorizar para poder utilizarla posteriormente.

Longitud de clave. Se mide típicamente en los bits que ocupa la clave. Así, una clave de un número de 1024 bits sería un número cualquiera desde el 0 hasta el $1,8 * 10^{308}$ (2^{1024}). Al representar las longitudes de clave como potencias de 2, es importante darse cuenta de la relación existente entre las longitudes de clave. Una clave de 1025 bits es el doble de largo que una de 1024.

Protocolo

En criptografía llamamos protocolo al orden en que tenemos que ejecutar los algoritmos criptográficos para resolver un problema criptográfico de forma segura.

Los protocolos existen porque, a veces, si estos pasos se ejecutan de forma incorrecta, se puede comprometer la seguridad del sistema criptográfico, aunque todos los algoritmos utilizados sean algoritmos totalmente seguros.

Bruce Schneier [SCH-96], propuso la utilización de nombres reales para los personajes involucrados en un protocolo de seguridad, el nombre a utilizar depende del rol que este desempeñando el personaje. Los nombres propuestos por Schneier y que se utilizan en este trabajo son los siguientes:

- **Alice** Participa en todos los protocolos.
- **Bob** Participa en protocolos que involucren a dos o más personajes.
- **Eve** Participa en un protocolo como atacante pasivo y/o activo.

La utilización de letras, en reemplazo de nombres, también es considerado en este trabajo y su relación es: Emisor → A; Receptor → B y Atacante → Intruso.

1.1.5.2 Esquema de un criptosistema

Un criptosistema puede ser usado como muestra la figura 1.2 para el propósito de lograr confidencialidad. Dos entidades emisor y receptor primero eligen o intercambian de forma secreta un par de claves (k, k'). Si el emisor desea enviar un mensaje $m \in M$ al receptor, él calcula $c = E_k(m)$ y transmite el criptograma c al receptor. Tras la recepción de c , el receptor calcula $D_{k'}(c) = m$ y por lo tanto recupera el mensaje original m .

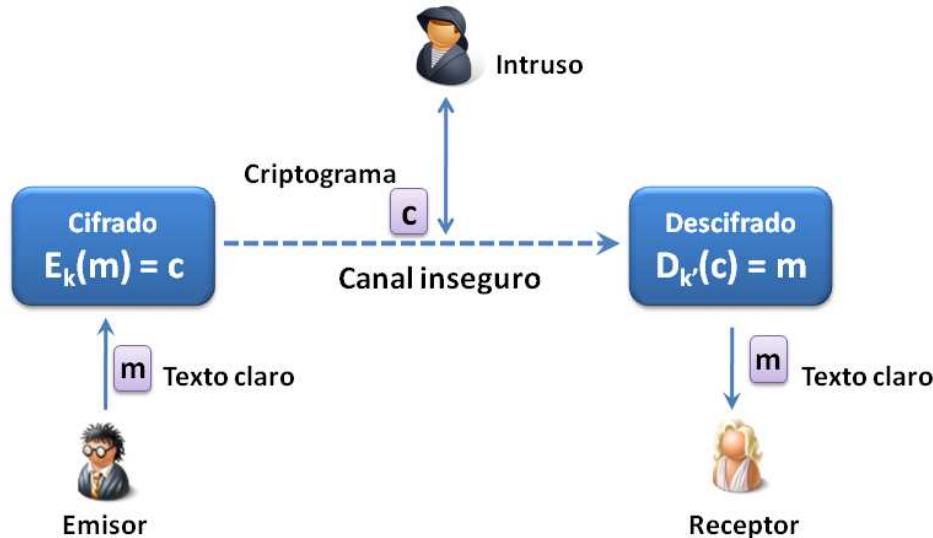


Figura 1.2: Esquema de funcionamiento de un criptosistema.

Los criptosistemas se clasifican en 2 grandes grupos en base a las claves utilizadas; criptosistemas simétricos o de clave secreta y criptosistemas asimétricos o de clave pública.

1.1.5.3 Criptosistemas simétricos o de clave secreta

Se denomina criptosistema de clave secreta (de clave privada, de clave única o simétrico), a aquel criptosistema en el que la clave de cifrado, k , puede ser calculada a partir de la de descifrado, k' , y viceversa. En la mayoría de estos sistemas, ambas claves coinciden, y por supuesto han de mantenerse como un secreto entre emisor y receptor: si un atacante descubre la clave utilizada en la comunicación, ha roto el criptosistema. En la figura 1.3 se ilustra como el emisor A emplea una clave para cifrar información que desea transmitir a otro usuario B; este último deberá utilizar la misma clave para recuperar la información original.

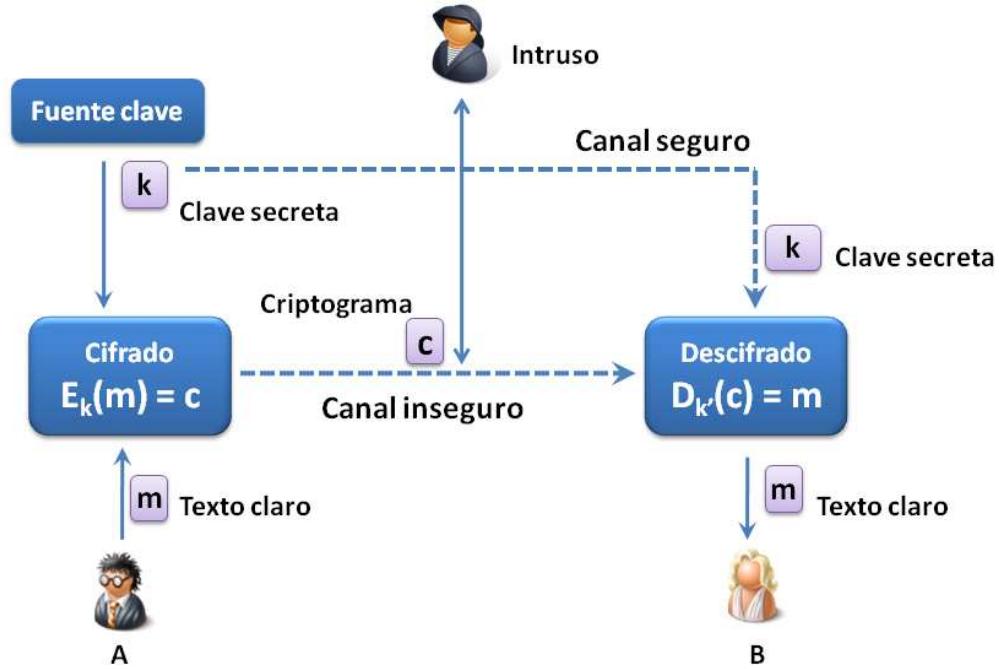


Figura 1.3: Esquema de funcionamiento de un sistema simétrico.

Este tipo de algoritmos se caracterizan por ser muy rápidos y eficientes desde el punto de vista computacional, ya que se basan en operaciones matemáticas sencillas realizadas sobre los símbolos del mensaje original. Por ello, requieren de un reducido tiempo de cálculo para realizar el cifrado y descifrado de los mensajes.

Sin embargo, presentan un importante problema: ¿cómo intercambiar la clave utilizada para el cifrado/descifrado a través de un canal seguro? Sin duda, se trata de una cuestión de especial relevancia ya que toda la seguridad del sistema depende de la confidencialidad de la clave (ésta sólo puede ser conocida por los usuarios A y B).

Por otra parte, también se debe tener en cuenta el problema de la gestión de claves, ya que se requiere una clave distinta para cada posible interacción entre dos usuarios del sistema, por lo que el número de claves secretas necesarias crece en un orden igual a N^2 , siendo N el número de usuarios distintos del sistema. Se requieren $\frac{N(N-1)}{2}$ claves distintas, con N = N° de usuarios.

Este tipo de criptosistema se pueden clasificar, según su funcionamiento de cifrado, en cifrado en flujo o cifrado en bloque.

Cifrado en flujo (bit a bit o byte a byte)

El algoritmo de cifrado se aplica a un elemento de información (carácter, bit) mediante un flujo que constituye la clave y que en teoría es aleatorio y de un tamaño superior al del mensaje [GOM-07].

Cifrado en bloque

El texto claro se divide en bloques de tamaño fijo de x bits (64, 128, etcétera). El algoritmo de cifrado se aplica a cada uno de estos bloques, con la utilización de una clave secreta compartida entre emisor y receptor, obteniendo su respectivo bloque cifrado [GOM-07].

El mecanismo conocido como “Padding” (rellenado) puede ser necesario para completar alguno de los bloques de un determinado mensaje con bits adicionales hasta alcanzar el tamaño de bloque con el que trabaja el algoritmo. De las técnicas de padding, la más utilizada en los algoritmos de clave secreta es PKCS #5 (Public Key Cryptography Standard 5). La técnica consiste en llenar los bytes restantes del último bloque con un número, que es el número de bytes que han quedado sin llenar en el último bloque.

1.1.5.4 Criptosistemas asimétricos o de clave pública

Los sistemas criptográficos asimétricos surgen a principios de los años 70 para dar respuesta al problema de intercambio de la clave de los sistemas simétricos. Se basan en problemas numéricos muy complejos (como la factorización de números primos o el cálculo de logaritmos discretos).

Estos sistemas emplean una doble clave (K_v , K_p). K_v se conoce como clave privada y K_p se conoce como clave pública. Una de ellas sirve para la transformación E de cifrado y la otra para la transformación D de descifrado. En muchos casos son intercambiables, esto es, si se emplea una para cifrar la otra sirve para descifrar y viceversa. Estos criptosistemas deben cumplir además que a partir del conocimiento de la clave pública K_p no permita calcular la clave privada K_v . Ofrecen un abanico superior de posibilidades, pudiendo emplearse para establecer comunicaciones seguras por canales inseguros – puesto que únicamente viaja por el canal la clave pública-, o para llevar a cabo autenticaciones.

En general, los sistemas asimétricos vienen dados por el esquema representado en la figura 1.4.

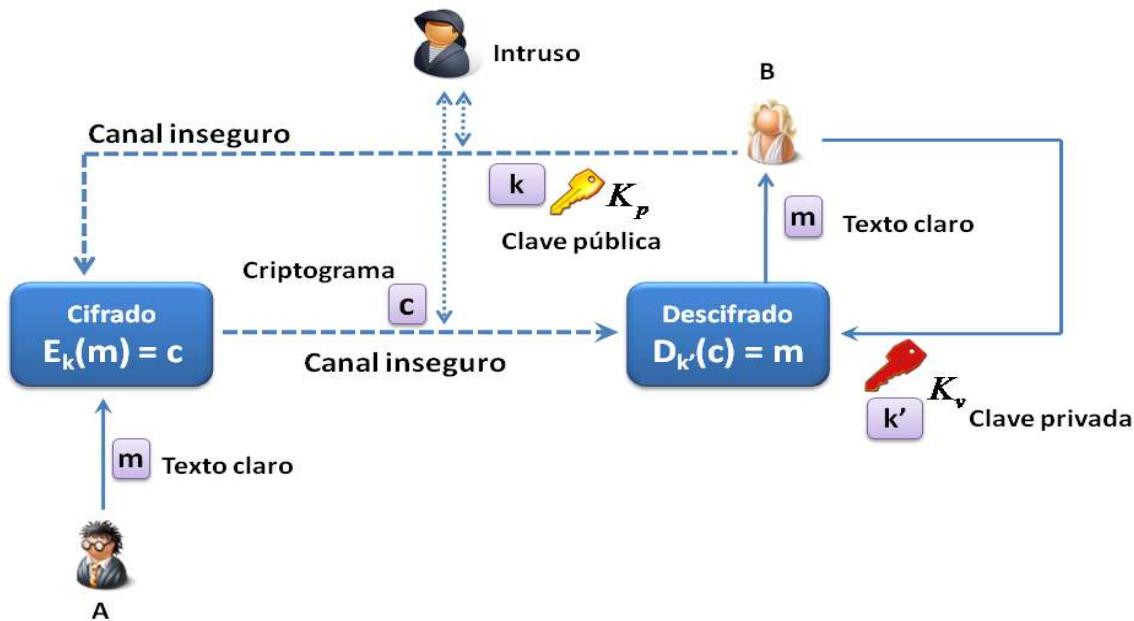


Figura 1.4: Transmisión de información empleando criptosistema asimétrico.

El emisor A quiere enviar un mensaje a B. Para ello solicita a B su clave pública K_p . A genera entonces el mensaje cifrado $E_k(m)$. Una vez hecho esto, únicamente quien posea la clave K_v – en el ejemplo, B – podrá recuperar el mensaje original m.

Nótese que para este tipo de aplicación, la clave que se hace pública es aquella que permite cifrar los mensajes, mientras que la clave privada es aquella que permite descifrarlos.

La seguridad en la criptografía asimétrica es un problema mucho más delicado que en la simétrica, ya que el único dato no público es la clave que sirve para el descifrado. Además, no es fácil reunir seguridad y efectividad en un cifrado de clave pública; y solamente unos pocos de los numerosos criptosistemas que han sido propuestos lo han logrado. Por lo general, estos algoritmos están basados en las llamadas funciones unidireccionales (véase anexo digital “Criptografía moderna”).

1.1.5.5 Criptosistemas Híbridos

En el mundo real, los criptosistemas asimétricos o de clave pública no son sustitutos de los criptosistemas simétricos o de clave secreta, pero sin duda la más difícil de las decisiones está en cuál de los 2 criptosistemas se debe utilizar. A grandes rasgos y en general se puede mencionar algunas ventajas de ambos criptosistemas.

De este modo, la gestión de claves (key management) es mucho más sencilla en los criptosistemas asimétricos. La gestión de claves se refiere a los procesos y mecanismos utilizados para la generación y mantenimiento de las claves que facilitan las comunicaciones seguras entre los usuarios de un sistema. Con estos criptosistemas asimétricos, cada usuario sólo debe memorizar su clave privada, ya que las claves públicas son conocidas por todos. Además, la clave pública facilita una aplicación criptográfica fundamental hoy en día como es la firma digital.

En los criptosistemas simétricos es posible conseguir mayores velocidades a la vez que las claves que se utilizan son más cortas. Los criptosistemas asimétricos deben utilizar claves mucho más largas para ofrecer un nivel de protección equivalente a la de los criptosistemas simétricos: 512, 1024 o 2048 bits, trabajando sobre bloques de bits del mensaje a cifrar. Por este motivo, se dice que los criptosistemas simétricos son entre 100 y 1000 veces más rápidos que los asimétricos, ya que estos últimos requieren de mayores recursos computacionales, por lo que algunos autores se han referido al algoritmo RSA como “Really Slow Algorithm” (algoritmo realmente lento).

La solución criptográfica óptima se alcanza combinando clave secreta y clave pública.

En implementaciones más prácticas la criptografía de clave pública es usada para asegurar y distribuir claves de sesión; aquellas claves de sesión son usadas con algoritmos simétricos para asegurar el tráfico del mensaje. Esto es a veces llamado un criptosistema híbrido.

Ejemplo:

Dos usuarios A y B mediante un criptosistema asimétrico intercambian una clave de sesión de forma segura. Esta clave de sesión les va a permitir cifrar y descifrar los datos en un criptosistema simétrico, tal y como muestra la figura 1.5.

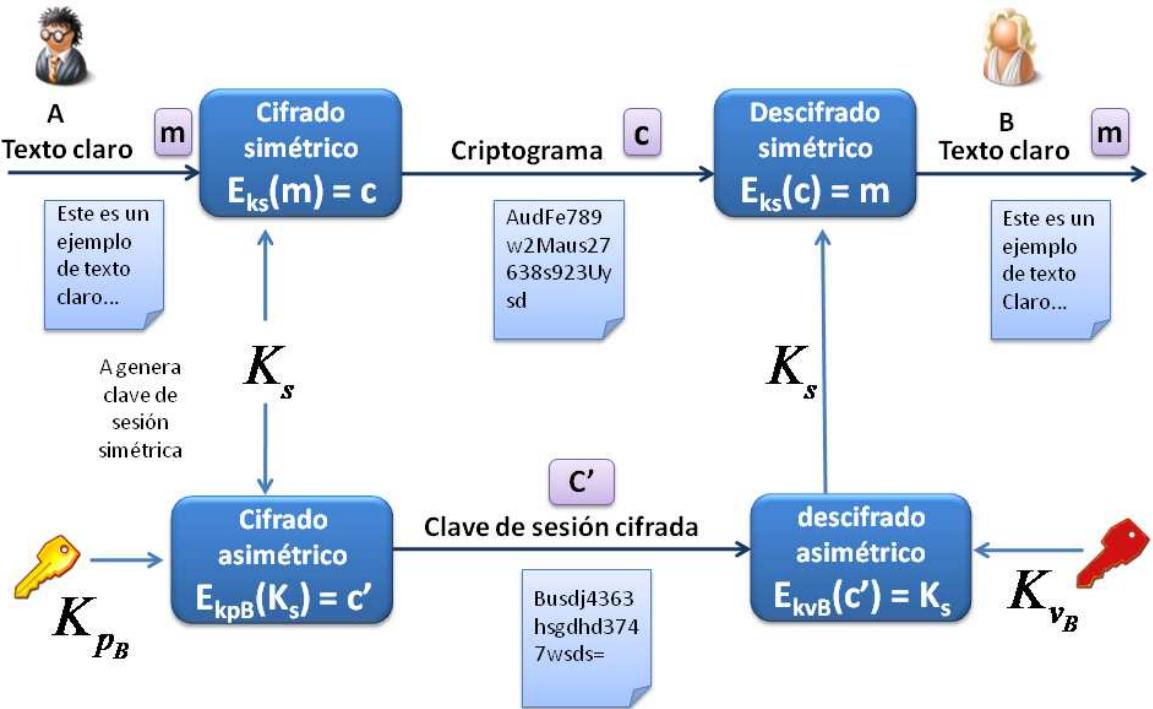


Figura 1.5: Combinación de criptosistemas asimétricos y simétricos.

En el ejemplo planteado, el usuario A utiliza una determinada clave de sesión K_s , para cifrar el mensaje original y, a su vez, procede a cifrar esta misma clave con la clave pública K_{p_B} del usuario B, de modo que sólo B pueda recuperar la clave de sesión necesaria para descifrar el mensaje original (porque, para obtener esta clave, es necesario emplear la clave privada K_{v_B} de B).

La técnica anteriormente descrita para proteger la confidencialidad de una clave simétrica mediante un algoritmo de cifrado asimétrico se conoce con el nombre de “**sobre digital**”.

Con la combinación de los criptosistemas simétricos y asimétricos se consigue garantizar totalmente la confidencialidad de la comunicación, y se mejora en la rapidez de los procesos de cifrado y descifrado.

1.1.5.6 Funciones Hash

Las funciones Hash son usadas para probar que los datos transmitidos no han sido alterados. Una función hash (función resumen) es una función computable que se aplica a un mensaje m de tamaño variable, para obtener una representación de tamaño fijo (de 128, 160, 256, 384 ó 512 bits) del propio mensaje: $H(m)$, que es llamado su valor hash. Así pues, las funciones hash se definen como sigue:

$$H: M \rightarrow M, H(m) = m'$$

El valor del hash es también llamado un message digest (compendio de mensaje) o fingerprint (huella digital).

Las propiedades deseables para toda función Hash criptográfica son:

01. **Compresión:** Dado un mensaje m de longitud arbitraria, su resumen $H(m)$ tiene una longitud fija n , normalmente menor que la longitud de m .
02. **Facilidad de cálculo:** Dado m , es fácil calcular $H(m)$.
03. **Unidireccionalidad:** Dado $H(m)$, es difícil calcular m .
04. **Difusión:** el resumen $H(m)$ debe ser una función compleja de todos los bits del mensaje m : si se modifica un solo bit del mensaje m , el hash $H(m)$ debería cambiar la mitad de sus bits aproximadamente.
05. **Resistencia débil a las colisiones:** dado m , es difícil encontrar otro mensaje m'' tal que $H(m) = H(m'')$ (colisión).
06. **Resistencia fuerte a las colisiones:** es difícil encontrar un par (m, m'') tal que $H(m) = H(m'')$.

Existen funciones resumen que emplean en sus cálculos una clave adicional – los denominados MAC (Message Authentication Codes), que se verán más adelante – y otras que no la usan, denominada genéricamente MDC (Modification Detection Codes).

1.1.5.7 Funciones de integridad de mensajes

Los MDC permiten la creación de un código o secuencia de bits que permite detectar si el contenido de un mensaje ha sido modificado. En general, los MDC se basan en la idea de funciones de compresión, estas funciones se encadenan de forma iterativa, haciendo que la entrada en el paso i sea función del i -ésimo bloque del mensaje (m_i) y de la salida del paso $i - 1$ (ver figura 1.6), en general, se suele incluir en alguno de los bloques del mensaje m – al principio o al final –, información sobre la longitud total del mensaje. De esta forma se reduce las probabilidades que 2 mensajes con diferentes longitudes den el mismo valor en su resumen.

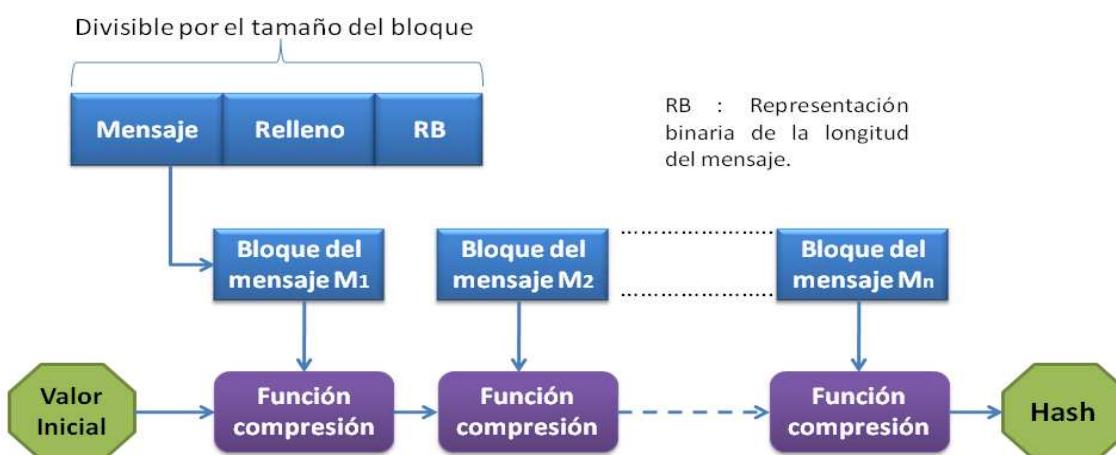


Figura 1.6: Estructura iterativa de una función resumen.

1.1.5.8 Funciones de autenticación de mensajes

Los MAC permiten la obtención de un código o secuencia de bits que permite probar la integridad del contenido y la autenticación del origen de un mensaje, al generar una clave que depende tanto del usuario como del propio mensaje¹. Esta aplicación ha propiciado el desarrollo de la firma digital, así como el desarrollo de mecanismos para el control de la integridad y autenticidad del software. Se pueden distinguir varios tipos:

Basados en cifrado por bloques: Son los más comunes, y consisten en cifrar el mensaje empleando un algoritmo por bloques en modo de operación CBC. El valor del MAC será entonces el resultado de cifrar el último bloque del mensaje (véase figura 1.7).

HMAC: Se basa en el uso de cualquier función MDC existente, aplicada sobre una versión del mensaje a la que se ha añadido un conjunto de bits, calculados a partir de la clave que se quiere emplear. Por ejemplo, la función HMAC a la que da lugar el algoritmo MD5 tiene la siguiente estructura.

$$MD5(k \oplus opad, MD5(k \oplus ipad, m))$$

Donde k es la clave – alargada con ceros por la derecha hasta tener 64 bytes de longitud, opad es el byte con valor hexadecimal 5C repetido 64 veces, ipad es el valor hexadecimal 36 repetido 64 veces, m es el mensaje, y la coma representa la concatenación.

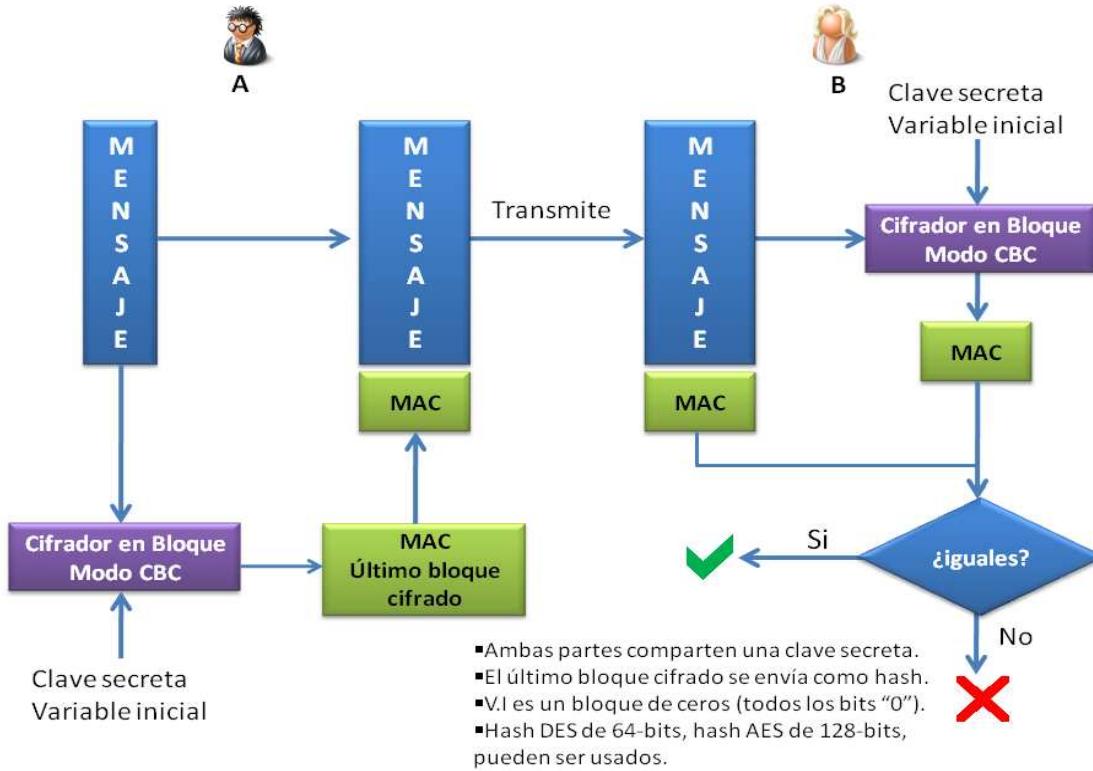


Figura 1.7: Códigos de autenticación de mensajes.

¹ Se aplica la función hash a los datos del mensaje y a una clave conocida por el usuario.

1.1.6 Aplicaciones Criptográficas

1.1.6.1 Firma digital

El desarrollo de las telecomunicaciones en estos últimos años ha creado toda una variedad de nuevas necesidades. Una de las principales hoy en día es la de firmar los documentos o mensajes transmitidos a través de redes de computadores. Así, se requiere un nuevo planteamiento, donde una firma digital sustituye a la firma manual y cumple las mismas propiedades que ésta.

Una firma digital es una secuencia de bits que se añade a una pieza de información cualquiera, y que permite garantizar su autenticidad de forma independiente del proceso de transmisión, tantas veces como se desee [LUC-09].

Otra definición de firma digital o electrónica es propuesta por el organismo internacional ISO (documento ISO 7498-2):

"La firma electrónica son los datos añadidos a un conjunto de datos que permiten al receptor probar el origen y la integridad de los datos, así como protegerlos contra falsificaciones".

La firma digital puede verse como una versión computarizada de la firma manual.

1.1.6.2 Características de una firma digital

La firma digital debe ser:

- Barata y fácil de generar.
- No falsificable, el intento de falsificación debe llevar asociada la resolución de un problema numérico intratable.
- Fácil de autenticar, pudiendo cualquier receptor establecer su autenticidad aun después de mucho tiempo.
- Irrevocable, el autor de una firma no puede negar su autoría.
- Va ligada indisolublemente al mensaje. Una firma digital válida para un documento no puede ser válida para otro distinto.
- Deben depender tanto del mensaje como del autor. Esto debe ser así porque en otro caso el receptor podría modificar el mensaje y mantener la firma, produciendo así un fraude.

Aunque se intenta que las firmas digitales y firmas manuales tengan el mismo funcionamiento, hay 2 características distintivas:

- La firma digital supera a la firma manuscrita en cuanto que permite la integridad de los datos de una forma más efectiva. Con la firma digital se puede verificar que el documento no ha sido alterado desde que fue firmado, hecho que no lo permite la firma manuscrita.
- Otra diferencia de la firma digital sobre la manuscrita, es que si 2 documentos son diferentes entonces la firma digital es diferente. En otras palabras la firma digital cambia de documento a documento, si un sujeto firma 2 documentos diferentes producirá 2 documentos firmados diferentes. Si 2 sujetos firman un mismo documento, también se producen 2 diferentes documentos firmados.

El emisor A envía un mensaje firmado digitalmente al receptor B, este último no sólo debe convencerse de que el mensaje fue firmado por el primero, sino que, además, debe ser capaz de demostrar a un juez que A realmente firmó ese mensaje.

El proceso de firma debe transformar el mensaje en un mensaje firmado de forma que la firma sólo pueda ser calculada por el emisor. Para ello, la transformación de firma usará alguna información que posea únicamente el emisor. Esta información secreta se acepta como prueba de la identidad del firmante, es decir, establece el origen del mensaje.

1.1.6.3 Firmas digitales asimétricas

Para firmar digitalmente un documento con técnicas de criptografía asimétrica, tal como pretende mostrar la tabla 1.2, se utiliza la clave privada para firmar, y la clave pública para comprobar la firma. Recuérdese que cuando se trató el tema de los criptosistemas asimétricos se vio que para lograr la confidencialidad de los datos se usaba la clave pública para cifrar y la clave privada para descifrar.

	Clave privada	Clave pública
Confidencialidad	Descifrar	Cifrar
Firma digital	Firmar	Verificar

Tabla 1.2: Uso de las claves públicas/privadas para confidencialidad y firma digital.

Realmente firmar es equivalente a cifrar con la clave privada, y verificar es equivalente a descifrar con la clave pública, con lo que a veces el proceso de firmar se le llama “cifrar con la clave privada” y al de verificar “descifrar con la clave pública”. El criptosistema RSA es utilizado comúnmente en la actualidad para firmar digitalmente un documento o archivo, utilizando este concepto.

La forma más extendida de calcular firmas digitales consiste en emplear una combinación de cifrado asimétrico y funciones resumen. La opción de utilizar funciones resumen se debe a que los sistemas de cifra asimétricos son muy lentos y el mensaje podría tener miles o millones de bytes, lo que haría que el proceso de cifrado con la clave privada se convierta en algo tedioso.

El esquema de funcionamiento para la obtención de una firma digital con criptosistema asimétrico y función hash queda ilustrado en la figura 1.8.

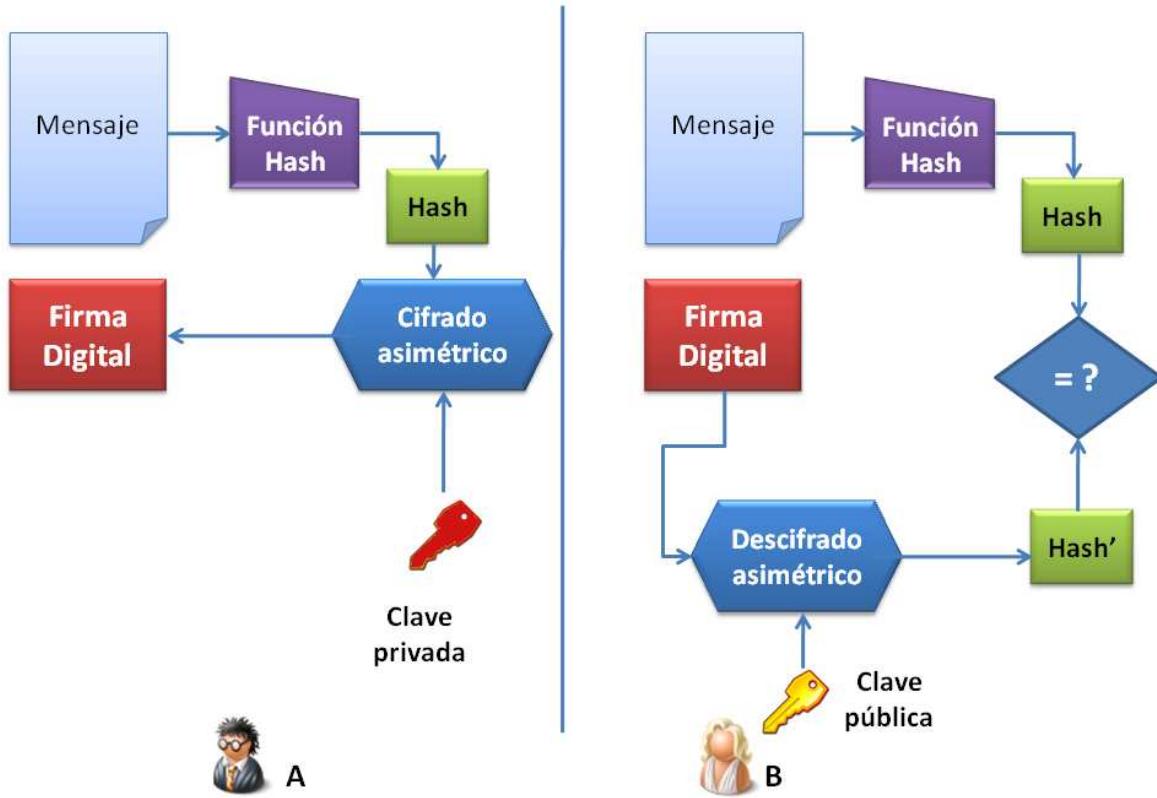


Figura 1.8: Esquema de una firma digital basada en funciones resumen y algoritmos de cifrado asimétricos.

El usuario A genera un resumen del mensaje y lo cifra empleando la clave de cifrado, que en este caso será la clave privada. La clave de descifrado se habrá hecho pública previamente, y debe estar en poder B. A envía entonces a B el criptograma correspondiente al hash del mensaje (firma digital). B puede ahora generar su propio hash y compararlo con el valor hash' obtenido del criptograma enviado por A. Si coinciden, el mensaje será auténtico, puesto que el único que posee la clave para cifrar es precisamente A.

Si además de capacidad de firma digital se desea privacidad, entonces la firma digital puede ser utilizada conjuntamente con un cifrado de clave pública o bien con un cifrado de clave secreta.

A pesar del elegante diseño de los sistemas de autenticación y cifrado basados únicamente en clave pública, dadas las actuales velocidades de computo, es más habitual combinar clave secreta y pública para los objetivos de la confidencialidad y la firma digital respectivamente. Así, dado un mensaje primero se obtiene su firma con clave pública y luego se cifra con clave secreta. Además lo normal es que la clave secreta se transmita además cifrándola con clave pública, de manera que se logra una perfecta compenetración entre ambos tipos de criptografía. La figura 1.9 muestra el esquema de funcionamiento al que se hace referencia.

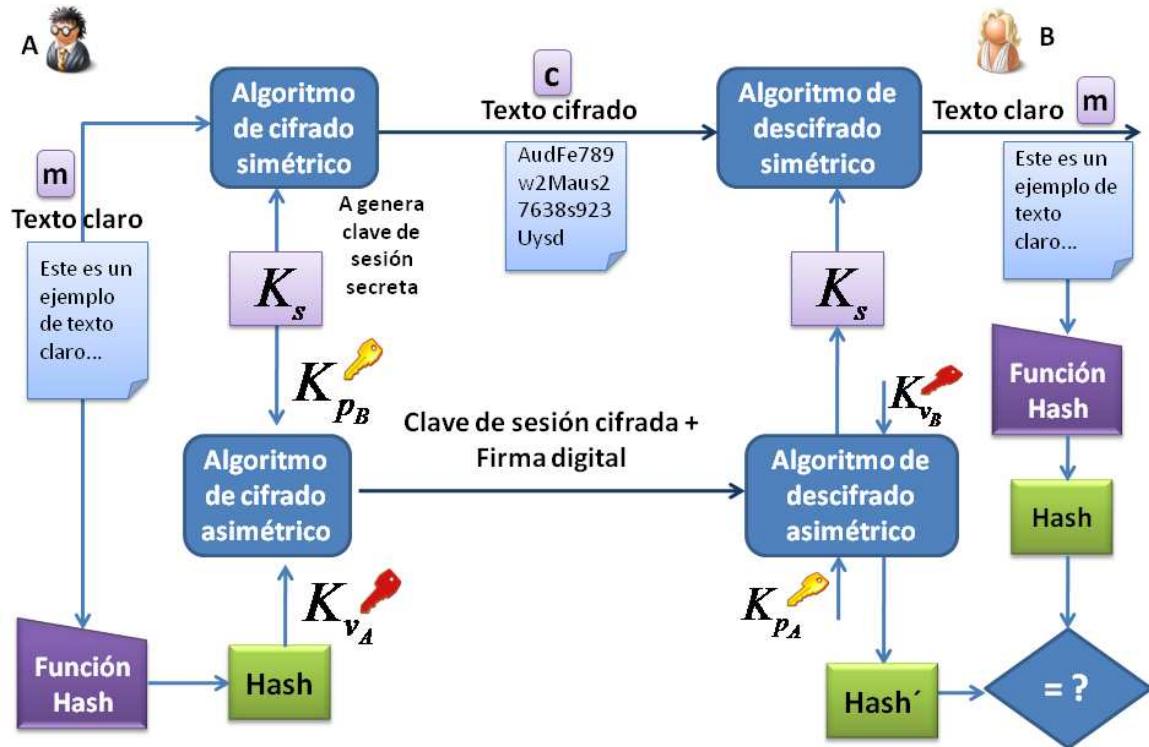


Figura 1.9: Combinación de criptosistemas asimétricos y simétricos con firma digital.

A modo de resumen, la firma electrónica o digital de un mensaje o transacción permite garantizar la integridad, la autenticación y la no repudiación en un sistema informático. Además, como se puede apreciar en la figura 1.9 se puede lograr también la confidencialidad del mensaje, logrando una transmisión a través de un canal inseguro con un nivel de seguridad muy deseado en la actualidad.

1.2 Contexto

Históricamente, el estudio y la aplicación de la criptografía han estado casi exclusivamente en manos de militares y diplomáticos. Sin embargo, en la sociedad actual ha surgido la necesidad de la criptografía civil, debido a la utilización de gran cantidad de información (personal, financiera, comercial y tecnológica) que se almacena en bancos de datos y se transmite a través de redes de computadores.

El primer uso de la escritura secreta de la que se tiene constancia data del siglo V a.C., durante la guerra entre Atenas y Esparta. En aquel caso, el cifrado conocido como escítila se basaba únicamente en la alteración del mensaje mediante la escritura de los símbolos de forma vertical sobre una cinta enrollada en un rodillo, de manera que al desenrollarla los símbolos del mensaje quedaban desordenados o traspuestos, por lo que sólo se podía leer el mensaje tras enrollar la cinta en un rodillo de igual grosor.

Otra de las primeras noticias sobre criptografía proviene de la época de los romanos. El cifrado en ese caso consistía en una sustitución de determinados símbolos por otros según una regla fija. Este método, denominado método César (siglo I a.C.), consistía en reemplazar cada letra del texto claro por otra letra que se encuentra un número fijo de posiciones más adelante en el alfabeto, por ejemplo; la letra A por la D, la B por la E, y así sucesivamente.

Como en muchas otras áreas científicas, el mayor desarrollo de la criptología, tuvo lugar durante las 2 guerras mundiales. En este caso se debió a la necesidad de establecer comunicaciones secretas militares y diplomáticas utilizando nuevas tecnologías, como la telegrafía y la radiotécnica. En la segunda guerra mundial, la máquina de cifrado alemana Enigma fue rota por la oficina criptoanalítica británica capitaneada por el matemático Alan Turing, mediante la máquina Colossus, precursora de los computadores modernos [CAB-03].

Ahora bien, en la segunda mitad del siglo XX, con el desarrollo de la cultura informática e Internet, han surgido nuevas aplicaciones de la criptología, debido fundamentalmente al manejo de gran cantidad de información, que está a disposición de muchos usuarios, lo cual plantea la necesidad de que los datos estén protegidos durante su transmisión y durante su almacenamiento.

Al mismo tiempo, este desarrollo de la informática produjo un cambio radical en el concepto de seguridad de los sistemas criptográficos, pues aquellos que eran supuestamente seguros frente a procedimientos manuales sucumbieron ante la eficacia de los computadores. De esta forma, la supuesta seguridad de los sistemas antiguos o clásicos ha tenido que ser sustituida por una seguridad matemática y computacionalmente demostrable en los sistemas modernos.

Existen 3 trabajos fundamentales sobre los que se apoya prácticamente toda la teoría criptográfica actual. Dos de ellos, desarrollados por Claude Shannon en sus artículos [SHA-48] y [SHA-49], en los que sienta las bases de la teoría de la información y de la criptografía moderna. El tercero, publicado por Whitfield Diffie y Martin Hellman en 1976 [DIH-76], que introduce el concepto de Criptografía Asimétrica, abriendo enormemente el abanico de aplicación de esta disciplina.

La confidencialidad de la información, es decir, el establecimiento de comunicaciones secretas sobre canales inseguros, no es el único propósito de la criptografía, aunque históricamente sí fue lo que produjo su nacimiento, y durante muchos años ha sido el único objetivo. La criptografía moderna actualmente, busca además, garantizar la integridad de un mensaje, autenticación del origen del mensaje y el no repudio del mensaje.

En este contexto se propone el uso de algoritmos criptográficos para ser usados en forma individual y/o combinados para satisfacer las necesidades de los usuarios en torno a la protección de los datos e información, almacenados en formato digital. Lo anterior se refiere al desarrollo de una herramienta que proporcione al usuario final los servicios de seguridad (confidencialidad, integridad, autenticación, no-repudio), necesarios para contrarrestar las amenazas existentes, empleando uno o varios mecanismos criptográficos.

1.3 Definición y delimitación del problema

Un producto de seguridad es aquel que permite la implementación real de los mecanismos o técnicas de seguridad capaces de proporcionar al usuario final los servicios de seguridad necesarios para contrarrestar las amenazas que pueden afectar la confidencialidad, integridad y disponibilidad de la información que un sistema informático procesa, almacena y transmite.

El objetivo principal de este proyecto se enfoca en el desarrollo de un producto de seguridad basado principalmente en el uso de técnicas criptográficas como son el cifrado simétrico, cifrado asimétrico, firma digital y funciones hash; capaz de proporcionar al usuario final los siguientes servicios de seguridad: confidencialidad, autenticación de origen de datos (el emisor del mensaje es quien dice ser, y no otro), integridad de datos (el mensaje que leemos es el mismo que nos enviaron) y no repudio (el emisor no puede negar el haber enviado el mensaje). Esto con el objetivo de satisfacer necesidades de los usuarios que quieren proteger información o datos para ellos sensibles. También, con esto se quiere dar a conocer, para así comprender, el uso de técnicas criptográficas en la protección de datos e información confidencial.

Entre las funciones que se destacan y de las que dispondrá la herramienta están la gestión de claves (públicas y privadas), el cifrado y descifrado de archivos digitales, el firmado digital de archivos y la posibilidad de exportar e importar claves públicas para el intercambio de información entre 2 o más usuarios de manera segura y confidencial. Otras de las funciones destacadas son: un destructor de archivos, que impida la recuperación del archivo una vez que haya sido borrado desde el medio de almacenamiento y un generador de archivo auto-descifrable, este tipo de archivo se descifra de forma automática haciendo doble clic sobre él, sin necesidad de que el usuario de destino deba tener la herramienta instalada en su PC. Además, contará con un gestor de claves seguras y contraseñas para el acceso a sistemas informáticos.

Al contar con esta herramienta, los beneficios esperados son:

- Almacenamiento de contenido cifrado en formato digital (binario o ASCII), permitiendo el acceso a su contenido en claro, sólo a la persona autorizada.
- Envío seguro de mensajes a través de un canal inseguro. Por ejemplo, en el envío de un correo electrónico a través de Internet.
- Envío seguro (contenido cifrado) de documentos y/o archivos a través de un canal inseguro.
- Identificación del emisor de un mensaje, documento o archivo. Proporcionando autenticación y por lo tanto, el emisor no puede negar la autoría de un mensaje; esto se logra gracias a la firma digital.
- Detectar si un intruso ha agregado, borrado y/o modificado información de un archivo o datos de un mensaje. Esto se logra mediante el uso de funciones de integridad de mensajes (MDC: Modification Detection Codes) y funciones de autenticación (MAC: Message Authentication Codes).
- Poner a disposición del usuario claves seguras para su uso en otros sistemas, como por ejemplo: páginas de registros en Internet (redes sociales, correo electrónico, blogs, etcétera).
- El usuario puede almacenar y gestionar las claves seguras que haya generado o sus propias contraseñas, para evitar el olvido de alguna de ellas y no tener que almacenarlas en medios inseguros como algún papel o archivos de edición de texto.
- Borrado seguro de archivo, destruyendo el contenido del mismo sobrescribiendo un número determinado de veces, según el algoritmo utilizado, con datos aleatorios.

También se deja en claro los límites de esta herramienta:

- La herramienta sólo cifrará documentos o archivos individuales, descartándose la posibilidad de cifrar directorios o sectores del disco.
- El tamaño de los archivos inciden en el tiempo de ejecución de los algoritmos de cifrado y descifrado.
- La herramienta no permite el cifrado de archivos con algoritmos de clave pública debido a que éstos realizan complejas funciones matemáticas, lo que se traduce en una velocidad de cifrado demasiado baja.
- La herramienta no tendrá la capacidad de generar resultados esperados si el formato de los archivos con los que trabaja no es respetado o sufren alteración por entidades externas.
- Debido a las implementaciones de los algoritmos criptográficos simétricos con los que trabaja la herramienta, sólo se podrán usar claves de sesión no superiores a los 128 bits de longitud.

1.4 Objetivos Generales y Específicos

1.4.1 Objetivos Generales

1. Estudio sobre criptografía y el uso de técnicas criptográficas en la protección de datos privados e información confidencial.
2. Desarrollar una herramienta de software basada en el uso de técnicas criptográficas para el cifrado, descifrado y firmado digital de documentos o archivos digitales que contienen datos privados y/o información confidencial del usuario final. Garantizando la confidencialidad e integridad del contenido del documento o archivo y que la identidad del creador de un mensaje o documento es legítima en caso de ser intercambiado con otros usuarios. Además, se debe garantizar la no repudiación de origen (un usuario que haya creado y enviado un mensaje a través del sistema no pueda posteriormente negar esta circunstancia).

La herramienta se implementará sobre una arquitectura criptográfica, que proveerá la implementación de los algoritmos criptográficos, garantizando la incorporación de otras implementaciones en caso de que se requieran.

1.4.2 Objetivos específicos

Con respecto al objetivo general N°1

- 1.1 Estudiar conceptos básicos de seguridad informática, obteniendo una visión global del tema que sirva de introducción al tema de criptografía.
 - 1.1.1. Búsqueda de libros, cursos, documentos, estudios que traten sobre el tema de seguridad informática.
 - 1.1.2. Selección de la información que resulte más relevante.
 - 1.1.3. Filtrado de la información seleccionada identificando y eliminando aquella que sea errónea.
 - 1.1.4. Organizar la información de tal manera que facilite su comprensión.

1.2 Estudiar los orígenes y evolución de la criptografía, las técnicas criptográficas existentes y su aplicación en protocolos de comunicación segura, certificados y firmas digitales.

1.2.1 Búsqueda de libros, cursos, documentos, estudios que traten sobre criptografía (orígenes y evolución).

1.2.2 Búsqueda de libros, cursos, documentos, estudios que traten sobre técnicas criptográficas y aplicaciones criptográficas.

1.2.3 Selección de la información más relevante con respecto a los resultados de las búsquedas anteriores.

1.2.4 Filtrado de la información seleccionada.

1.3 Analizar las técnicas criptográficas más importantes partiendo de las clásicas hasta llegar a las modernas y sus principales aplicaciones.

1.3.1 Identificar algoritmos criptográficos más conocidos y que hayan sido importantes para la evolución de la criptografía. Además de los que serán usados en el desarrollo de la aplicación.

1.3.2 Comprender el funcionamiento de estos algoritmos y las técnicas que ocupan, ayudando con esto al desarrollo de la aplicación.

1.3.3 Conocer el uso de aplicaciones criptográficas como los protocolos de comunicación segura, certificados y firmas digitales.

1.4 Redactar la información recopilada de forma de obtener un resumen detallado e incorporarlo como anexos digitales que acompañen al informe final.

Con respecto al objetivo general N°2

2.1 Estudiar la arquitectura criptográfica de Java.

2.1.1. Estudiar JCA (Java Cryptography Architecture).

2.1.2. Estudiar JCE (Java Cryptography Extensions).

2.2 Instalar la herramienta GPG (GNU Privacy Guard) para su uso y análisis, de tal manera que permita orientar a los alumnos sobre cuáles son las funcionalidades principales de una herramienta de su tipo.

2.3 Desarrollar la herramienta en la plataforma Java SE (Standard Edition).

2.4 Implementar el administrador de claves.

2.4.1. Firmar claves públicas.

2.4.2. Gestionar claves públicas y privadas.

2.4.3. Exportar claves públicas para otros usuarios.

2.4.4. Importar claves públicas de otros usuarios.

2.5 Implementar el administrador de usuarios

2.5.1. Identificar y validar al usuario al inicio de su sesión personal.

2.5.2. Gestionar las sesiones de cada usuario y sus claves.

- 2.6 Implementar el cifrador y descifrador de documentos o archivos digitales
 - 2.6.1. Elección de los algoritmos que permiten cifrar y descifrar y que están disponibles en la arquitectura de seguridad de Java.
- 2.7 Implementar el generador y verificador de firma digital
 - 2.7.1. Elección de los algoritmos que permiten firmar digitalmente y que están disponibles en la arquitectura de seguridad de Java.
 - 2.7.2. Elección de funciones Hash que permiten recibir un texto de longitud variable y devolver otro de longitud fija (generalmente más pequeño) y que están disponibles en la arquitectura de seguridad de Java.
- 2.8 Implementar administrador de claves seguras y contraseñas para el uso que el usuario estime conveniente.
 - 2.8.1. Elección de un algoritmo que permite generar claves seguras y con un nivel seguridad considerado alto por los expertos en criptografía.
 - 2.8.2. Almacenar estas claves debido a su longitud y para evitar olvido de estas por parte del usuario.

Capítulo 2 | SICFA-DTX235: Análisis del sistema

“Fiarse de todo el mundo y no fiarse de nadie son dos vicios. Pero en el uno se encuentra más virtud, y en el otro más seguridad”

Séneca, Lucio Anneo.

Este capítulo presenta la información que se ha obtenido en esta fase, sobre el sistema que se quiere implementar.

Las primeras actividades realizadas por parte del grupo de trabajo fueron; una serie de reuniones con el profesor guía, en las cuales se definió el ámbito y la naturaleza del problema que debe resolverse para poder entender los alcances, definir prioridades y establecer las restricciones del proyecto; se negociaron las ordenes de entrega, los límites de tiempo y otros asuntos relacionados con el proyecto, para posteriormente realizar una planificación del trabajo. Como consecuencia de estas actividades se estableció que el ciclo de vida del software más apropiado para este proyecto es el modelo de ciclo de vida *incremental* dadas las características presentadas en el apartado 2.1.

Las actividades realizadas por el grupo de trabajo y que se desarrollaron en paralelo al proceso de desarrollo del sistema, tienen relación con el estudio sobre criptografía y el uso de técnicas criptográficas en la protección de datos privados e información confidencial. Entre estas actividades se destacan; la búsqueda, selección y filtrado de la información, dejando sólo aquella actualizada y recomendada por fuentes confiables.

Otras de las actividades, realizadas por el equipo de desarrollo, están la instalación y estudio de la herramienta de seguridad GPG, con el objetivo de tener indicios de las funcionalidades que debe tener una herramienta de este tipo. De GPG se puede decir que es una aplicación con licencia Open Source desarrollada en el lenguaje de programación C [WEB-17]. GPG no posee una interfaz gráfica de usuario (GUI) sino que cuenta con una interfaz de usuario de texto simple (Shell), en donde el usuario debe teclear los comandos necesarios para ejecutar la funcionalidad deseada, como por ejemplo la generación de una pareja de claves subordinadas.

Una actividad que se ejecutó en forma paralela al estudio de GPG, fue el estudio de la arquitectura de seguridad de Java. Esta actividad fue de vital importancia ya que permitió verificar la factibilidad de implementar las funcionalidades de GPG en el lenguaje de programación Java.

En lo que resta del capítulo se presentan; el ciclo de vida del software utilizado, la meta y los objetivos de la aplicación, incluyendo el propósito, los límites, los beneficios y planificación del software, para concluir con un resumen de los requerimientos de software agrupados por subsistema.

2.1 Ciclo de vida

El modelo de ciclo de vida seleccionado por el grupo de trabajo, para el desarrollo del proyecto, fue el modelo de Ciclo de vida incremental.

Este modelo es una repetición del ciclo de vida en cascada, aplicándose este ciclo en cada funcionalidad del software a construir. Al final de cada ciclo se entrega una versión al cliente que contiene una nueva funcionalidad. Por lo tanto, este ciclo de vida, permite realizar una o más entregas al cliente antes de terminar el proyecto.

Este modelo de ciclo de vida fue elegido por las siguientes razones:

- Permite que el cliente pueda ir siguiendo los avances que el software está teniendo y sugerir nuevas ideas, sin afectar con ello a lo ya logrado.
- Se realiza construyendo por módulos que cumplen las diferentes funciones del sistema. Esto permite ir aumentando gradualmente las capacidades del software.
- Este ciclo de vida facilita la tarea del desarrollo permitiendo a cada miembro del equipo desarrollar un módulo particular en el caso de que el proyecto sea realizado por un equipo de programadores.
- Si se detecta un error grave, sólo se desecha la última iteración.
- No es necesario disponer de todos los requerimientos de todas las funcionalidades en el comienzo del proyecto y además facilita la labor del desarrollo con la conocida filosofía de divide y vencerás.

2.2 Generalidades del proyecto

2.2.1 Descripción del proyecto

Propósito

Los propósitos de la herramienta de software a construir son los siguientes:

- Ser una herramienta de seguridad en comunicaciones electrónicas.
- Proporcionar los servicios de confidencialidad, integridad de datos, autenticación del origen de datos y no repudio con prueba de origen.
- Lograr la implementación de las técnicas y mecanismos de seguridad necesarios para proporcionar al usuario final los servicios de seguridad antes mencionados.
- Agregar seguridad al almacenamiento y transmisión de documentos o archivos digitales de un usuario.

Alcance

Meta

Desarrollar una herramienta de software basada en el uso de técnicas criptográficas para el cifrado, descifrado y firmado digital de archivos digitales que contienen datos privados y/o información confidencial del usuario final. Garantizando la confidencialidad e integridad del contenido del archivo y la autenticidad del origen de los datos, en caso de ser intercambiado con otros usuarios. Además, se debe garantizar la no repudiación de origen.

Objetivos

1. Estudiar y analizar las técnicas criptográficas más importantes, partiendo de las clásicas hasta llegar a las modernas y sus principales aplicaciones.
 - 1.1. Identificar algoritmos criptográficos más conocidos y que hayan sido importantes para la evolución de la criptografía. Además de los que serán usados en el desarrollo de la aplicación.
 - 1.2. Comprender el funcionamiento de estos algoritmos y las técnicas que ocupan, ayudando con esto al desarrollo de la aplicación.
 - 1.3. Conocer el uso de aplicaciones criptográficas como los protocolos de comunicación segura, certificados y firmas digitales.
2. Instalar la herramienta GPG (GNU Privacy Guard) para uso y análisis, de tal manera que permita orientar a los alumnos sobre cuáles son las funcionalidades principales de una herramienta de su tipo.
3. Estudiar y analizar la arquitectura criptográfica de Java.
 - 3.1. Estudiar JCA (Java Cryptography Architecture).
 - 3.2. Estudiar JCE (Java Cryptography Extension).
4. Implementar la herramienta de software en la plataforma Java SE (Standard Edition), utilizando implementaciones de algoritmos criptográficos, proveídos por terceros (proveedor, SUN), a través de la arquitectura criptográfica de Java. La herramienta estará organizada en 5 subsistemas.
 - 1.1 Mecanismos de seguridad (MS): Permitirá la generación de parejas de claves (pública y privada) principal y subordinadas, cifrado/descifrado simétrico y asimétrico, generación/verificación de valor hash, generación/verificación de firma digital, generación de números pseudoaleatorios.
 - 1.2 Administrador de claves (AC): Permitirá el almacenamiento de pareja de claves (pública y privada) principal y subordinadas, añadir y eliminar componentes a las claves, revocar claves, importar y exportar claves públicas.
 - 1.3 Complementos Simétricos (CS): Permitirá el ingreso de los datos necesarios para llevar a cabo alguna operación de cifrado/descifrado simétrico, cifrado/descifrado simétrico PBE, generación/verificación valores hash MDC, generación/verificación valores hash MAC, borrado seguro de archivo y generación de archivo auto-descifrable (no requiere del software para descifrar el contenido del archivo cifrado).
 - 1.4 Llavero personal o gestor de claves seguras (GC): Permitirá la creación, edición, almacenamiento y eliminación de registros que contienen claves seguras o contraseñas y los datos asociados con ellas.
 - 1.5 Editor SICFA (ESIC): Permitirá al usuario la creación de mensajes digitales, que contienen texto y/o imágenes, los cuales pueden ser firmados digitalmente y/o cifrados.

Beneficios

- Almacenamiento de contenido cifrado en formato digital (binario o ASCII), permitiendo el acceso a su contenido en claro sólo a la persona autorizada.
- Envío seguro de mensajes a través de un canal inseguro. Por ejemplo, en el envío de un correo electrónico a través de Internet.
- Envío seguro (contenido cifrado) de documentos y/o archivos a través de un canal inseguro.
- Identificación del emisor de un mensaje, documento o archivo. Proporcionando autenticación y por lo tanto, el emisor no puede negar la autoría de un mensaje; esto se logra gracias a la firma digital.
- Detectar si un intruso ha agregado, borrado y/o modificado información de un archivo o datos de un mensaje. Esto se logra mediante el uso de funciones de integridad de mensajes (MDC: Modification Detection Codes) y funciones de autenticación (MAC: Message Authentication Codes).
- Poner a disposición del usuario claves seguras para su uso en otros sistemas, como por ejemplos: páginas de registros en Internet (redes sociales, correo electrónico, blogs, etcétera).
- El usuario puede almacenar y gestionar las claves seguras que haya generado o sus propias contraseñas, para evitar el olvido de alguna de ellas y no tener que almacenarlas en medios inseguros como algún papel o archivos de edición de texto.
- Borrado seguro de archivo, destruyendo el contenido del mismo sobrescribiendo un número determinado de veces, según el algoritmo utilizado, con datos aleatorios.
- Se elimina la necesidad de contar con la herramienta para descifrar el contenido de un archivo cifrado con un algoritmo simétrico PBE.
- Permitir al usuario generar un archivo ejecutable que se descifra de forma automática, sin necesidad de que el usuario de destino deba tener instalado en su computador el sistema que creó este archivo.

Nombre del software

El nombre del software es: **Sistema de Cifrado y Firmado digital de Archivos Digitales**, independientes de su extensión; la importancia de los números primos, en el tema de seguridad de la información y criptografía, lleva a que el nombre incluya los tres primeros de ellos: 2, 3 y 5. (SICFA-DTX235).

También resulta válida su abreviación: **Sistema de Cifrado y Firmado digital de Archivos**. (SICFA).

Límites del proyecto

- La herramienta sólo cifrará documentos o archivos individuales, descartándose la posibilidad de cifrar directorios o sectores del disco.
- El tamaño de los archivos inciden en el tiempo de ejecución de los algoritmos de cifrado y descifrado.

- La herramienta no permite el cifrado de archivos con algoritmos de clave pública debido a que éstos realizan complejas funciones matemáticas lo que se traduce en una velocidad de cifrado demasiado baja (según [GOM-07], entre 100 y 1.000 veces más lentos que los simétricos).
- La herramienta no tendrá la capacidad de generar resultados esperados si el formato de los archivos con los que trabaja no es respetado o sufren alteración por entidades externas.
- Debido a las implementaciones de los algoritmos criptográficos simétricos con los que trabaja la herramienta, sólo se podrán usar claves de sesión no superiores a los 128 bits de longitud.
- La generación de claves, cifrado simétrico, cifrado asimétrico, funciones hash y firma digital, quedan sujetas a la implementación proporcionada por el proveedor del servicio criptográfico.

2.2.2 Planificación del proyecto

Dada la magnitud del problema a desarrollar y considerando la gran cantidad de funciones con las que contará la herramienta de software SICFA-DTX235, el equipo de desarrollo decidió abordarlo en 3 iteraciones, para el ciclo de vida planteado.

PRIMERA ITERACIÓN

Actividades

- Búsqueda, selección, filtrado y organización de la información relacionada con los temas a tratar (seguridad informática y criptografía).
- Estudio de los conceptos básicos de seguridad informática y criptografía.
- Estudio de sistemas criptográficos asimétricos (algoritmos RSA y DSA).
- Estudio de la aplicación criptográfica: firma digital.
- Selección de las herramientas y librerías a utilizar para el desarrollo e implementación del sistema.
- Estudio de la arquitectura criptográfica de java (JCA y JCE).
- Instalación y utilización de GPG (Gnu Privacy Guard).
- Se inició la etapa de ingeniería de software, con la recolección y análisis de requerimientos, logrando un documento con la especificación del 70% de los requisitos del sistema.
- Se especificaron pruebas de validación.
- Se diseñó e implementó el 100% de los requerimientos propuestos.
- Se realizaron pruebas de unidad (caja blanca y caja negra).

Logros

- Implementación del administrador de claves.
- Implementación del administrador de usuarios.
- Implementación del cifrador y descifrador de archivos (cifrado híbrido).
- Implementación del generador y verificador de firmas digitales.

Semanas a trabajar: 19 semanas.

Problemas acontecidos

- Enfermedad de uno de los integrantes del grupo de trabajo.
- Problemas con el diseño de interfaces gráficas.

Semanas adicionales: 3 semanas.

Semanas totales: 22 semanas (Febrero 2009 --> Julio 2009).

SEGUNDA ITERACIÓN

Actividades

- Estudio de sistemas criptográficos simétricos (algoritmos Blowfish, DES, TripleDES y AES).
- Estudio de funciones hash.
- Se continúa con el análisis de requisitos del software, modificándose e incluso eliminándose algunos requisitos ya propuestos en la iteración anterior, logrando un documento con la especificación del 90% de los requisitos del sistema. La meta a lograr en ese instante era 90% y se logró.
- Se especificaron pruebas de validación.
- Se diseñó e implementó el 100% de los requerimientos propuestos en esta etapa.
- Se realizaron pruebas de integración.
- Se realizaron pruebas de unidad (caja blanca y caja negra).

Logros

- Implementación del gestor de claves seguras y contraseñas.
- Implementación del cifrador y descifrador de archivos (cifrado simétrico y simétrico PBE).
- Implementación del generador y verificador de valores hash.

Semanas a trabajar: 15 semanas.

Problemas acontecidos

- Problemas de integración de algunos módulos.
- Problemas con restricciones del sistema operativo Windows.
- Se decide seguir estándar de codificación de SUN [WEB-18].

Semanas adicionales: 2 semanas.

Semanas totales: 17 semanas (Julio 2009 --> Octubre 2009).

TERCERA ITERACIÓN

Actividades

- Se termina con el análisis de requisitos del software, modificándose algunos requisitos ya propuestos en la iteración anterior, logrando un documento con la especificación del 100% de los requisitos del sistema.
- Se consideraron nuevas funcionalidades a agregar al sistema.
- Se especificaron nuevas pruebas de validación.
- Se diseñó e implementó el 100% de los requerimientos propuestos en esta etapa.
- Se terminó con la etapa de documentación de la fase de diseño de software.
- Se realizaron pruebas de integración.

- Se realizaron pruebas de unidad (caja blanca y caja negra)
- Se realizaron pruebas de validación y rendimiento.

Logros

- Implementación del editor SICFA-DTX235.
- Implementación de la funcionalidad de borrado seguro de archivos.
- Implementación del generador de archivo auto-descifrable.

Semanas a trabajar: 12 semanas.

Problemas acontecidos

- Problemas de integración de algunos módulos.
- Problemas con cambio de diseño de algunas interfaces gráficas.
- Problemas en el grupo de desarrollo.

Semanas adicionales: 1 semana.

Semanas no trabajadas: 9 semanas.

Semanas totales: 22 semanas (Noviembre 2009 --> Abril 2010).

2.3 Requerimientos del proyecto

La ingeniería de requisitos proporciona el mecanismo apropiado para entender lo que el cliente quiere, analizar las necesidades, evaluar la factibilidad, negociar una solución razonable, especificar la solución sin ambigüedades, validar la especificación, y administrar los requisitos conforme éstos se transforman en un sistema operacional [PRE-05].

En este apartado se presenta una visión general de los requerimientos principales obtenidos a lo largo de este proyecto, los cuales se dividen en 5 partes:

- Requerimientos para Administrador de claves.
- Requerimientos para Llavero Personal o Gestor de Claves.
- Requerimientos para Complementos Simétricos.
- Requerimientos para Editor SICFA-DTX235.
- Requerimientos para Mecanismos de Seguridad.

Para una revisión más detallada del análisis de los requisitos del software, véase el anexo digital: “Especificación de Requerimientos”, que se entrega junto con este documento de memoria de título.

Terminología y conceptos previos

Lo que sigue es una lista de términos y conceptos, necesarios para una mejor comprensión de la fase de análisis de requerimientos y diseño del software.

Tag: Valor único de ocho bits que identifica a un paquete y registro, distinguiéndolo del resto.

Paquete: Un paquete es un fragmento de datos que tiene una etiqueta (Tag) que especifica su contenido. Un anillo de claves se compone de un número de paquetes.

Registro: Un registro es un fragmento de datos que tiene una etiqueta (Tag) que especifica su contenido. Un archivo del llavero personal se compone de registros.

Anillo de claves: Un anillo de claves es un único archivo en el que se pueden efectuar operaciones de extracción e inserción de claves.

Password Based Encryption (PBE): Se llama PBE a las técnicas criptográficas que obtienen la clave binaria a partir de una contraseña o frase de paso. En concreto, se pasa la contraseña por una función hash para obtener una clave binaria, que es la que se acaba dando al algoritmo de cifrado.

2.3.1 Requerimientos para Administrador de claves

El subsistema administrador de claves debe permitir: gestionar claves públicas y privadas, propias; gestionar claves públicas de otros usuarios; gestionar identidades de usuario, que asocian una clave pública con una persona real; ingreso de datos para la ejecución de operaciones de cifrado/descifrado de archivo (cifrado híbrido); ingreso de datos para la generación/verificación de firma digital de archivo; ingreso de datos para la generación/verificación de firma digital de claves públicas subordinadas propias e identidades de usuario propias.

En los procedimientos de gestión de las claves se deben tener en cuenta las siguientes actividades: generación de las claves, distribución de las claves, activación y utilización de las claves, almacenamiento y recuperación de las claves, destrucción y revocación de las claves.

Los requerimientos, organizados por funcionalidad, asociados con este subsistema son:

Generación de una pareja de claves principal (pública y privada)

Para utilizar SICFA-DTX235 se debe generar un nuevo par de claves (pública y privada) principal. Esta pareja de claves sólo será utilizada para la generación/verificación de firma digital de claves públicas subordinadas, identidades de usuario y revocación de claves. Los requerimientos asociados son:

- R02. El sistema debe permitir el ingreso de los datos relacionados con una nueva pareja de claves principal y que le permitirá al usuario comenzar a utilizar el sistema.
- R04. El sistema debe permitir obtener una huella digital (20 bytes) para identificar en forma única a una clave pública principal y así distinguirla de la que poseen otros usuarios.
- R06. El sistema debe permitir la creación de un paquete de clave pública principal.
- R07. El sistema debe permitir obtener un identificador único (8 bytes) para identificar en forma única un paquete del sistema; paquete de clave pública principal, paquete de clave privada principal, paquete de identidad de usuario, paquete de clave pública subordinada, paquete de clave privada subordinada, paquete de confianza, paquete de revocación, paquete de firma.
- R08. El sistema debe permitir la creación de un paquete de una nueva identidad de usuario primario, con lo cual se asocia una persona real con una clave pública.
- R10. El sistema debe permitir la creación de un paquete de firma digital de identidad de usuario primario.
- R11. El sistema debe permitir la creación de un paquete de clave privada principal.

- R12. El sistema debe permitir la creación de un paquete de confianza, para indicar la confianza que tiene el usuario en una clave pública principal.

Gestión de paquetes en memoria

- R13. El sistema debe permitir gestionar en memoria los paquetes creados por el usuario para evitar el continuo acceso a los archivos digitales donde se almacenan los datos.
- R14. El sistema debe permitir almacenar un paquete en memoria.
- R15. El sistema debe permitir localizar un paquete almacenado en memoria.
- R16. El sistema debe permitir eliminar un paquete almacenado en memoria.

Recuperación de paquetes desde archivos digitales

Los paquetes que crea y utiliza el sistema pueden ser obtenidos desde archivos digitales para su posterior gestión en memoria. Los requerimientos asociados son:

- R17. El sistema debe permitir obtener desde un archivo digital los paquetes de clave pública principal; el paquete de revocación, si existe, con su paquete de firma; los paquetes de clave pública subordinada y su paquete de firma respectivo; paquete de identidad de usuario y su paquete de firma respectivo. Este archivo debe conservar todos estos paquetes tanto propios como de otros usuarios.
- R18. El sistema debe permitir obtener desde un archivo digital los paquetes de clave privada principal y los paquetes de clave privada subordinada. Los paquetes contenidos en este archivo son sólo los que pertenecen al usuario.
- R19. El sistema debe permitir obtener desde un archivo digital los paquetes de confianza, que indican la confianza que el usuario tiene en las claves públicas de otro usuario.

Escritura de paquetes en archivos digitales

Los paquetes que crea y utiliza el sistema y que están en memoria pueden ser respaldados en archivos digitales para su posterior recuperación. Los requerimientos asociados son:

- R20. El sistema debe permitir escribir desde memoria a un archivo digital los paquetes de clave pública principal; el paquete de revocación, si existe, con su paquete de firma; los paquetes de clave pública subordinada y sus paquetes de firma respectivos; paquetes de identidad de usuario y sus paquetes de firma respectivos. Este archivo debe conservar todos estos paquetes tanto propios como de otros usuarios.
- R21. El sistema debe permitir escribir desde memoria a un archivo digital los paquetes de clave privada principal y los paquetes de clave privada subordinada. Los paquetes contenidos en este archivo son sólo los que pertenecen al usuario, no debe permitir el almacenamiento de paquetes ajenos de claves privadas.
- R22. El sistema debe permitir escribir desde memoria a un archivo digital los paquetes de confianza que indican la confianza que el usuario tiene en las claves públicas de otro usuario.

Gestión de par de claves principal

Un par de claves principal se compone de una clave pública y otra privada. Una clave pública se compone de la parte pública de la clave maestra de firmado, las partes públicas de las claves subordinadas (sub-claves) de firmado y cifrado, y de un juego de identificadores de usuario que se usa para asociar la clave pública con una persona real; cada una de estas partes contiene datos sobre sí misma. Los requerimientos asociados son:

- R23. El usuario deberá introducir una clave de sesión para poder ingresar al administrador de claves. Esta clave de sesión es la que se ingresa en el requerimiento R02.
- R24. El usuario deberá introducir la clave de sesión para poder validar operaciones de edición y eliminación de datos. Esta clave de sesión es la que se ingresa en el requerimiento R02.
- R25. El usuario deberá introducir una frase de paso para poder firmar digitalmente los datos de paquetes del sistema; paquete de revocación, paquete de identidad de usuario, paquete clave pública subordinada. Esta frase de paso es la que se ingresa en el requerimiento R02.
- R26. El sistema debe permitir mostrar al usuario los datos de clave pública principal (propia y ajenas), los datos de identidad de usuario primario, los datos de confianza, los datos de claves públicas subordinadas (propias y ajenas), los datos de identidad de usuario asociado con cada clave pública subordinada y los datos de firma digital asociada con cada clave pública subordinada.
- R27. El sistema debe permitir ingresar los datos relacionados con una nueva pareja de claves subordinadas y que le permitirá al usuario cifrar y firmar archivos.
- R28. El sistema debe permitir la creación de un paquete de clave pública subordinada.
- R29. El sistema debe permitir la creación de un paquete de firma digital de clave pública subordinada.
- R30. El sistema debe permitir la creación de un paquete de clave privada subordinada.
- R31. El sistema debe permitir ingresar los datos relacionados con una nueva identidad de usuario y que le permitirá al usuario asociar esta identidad con cero o varias claves públicas subordinadas.
- R32. El sistema debe permitir la creación de un paquete de identidad de usuario.
- R33. El sistema debe permitir la creación de un paquete de firma digital de identidad de usuario.
- R34. El sistema debe permitir la edición de datos de clave pública principal ajena. Los datos que se pueden editar corresponden a los de paquete de confianza (fotografía, nivel de confianza).
- R35. El sistema debe permitir la edición de datos de clave pública subordinada propia. Los datos que se pueden editar corresponden a fecha de caducidad e identificador único de identidad de usuario asociado con la clave pública subordinada.
- R36. El sistema debe permitir modificar el usuario primario, seleccionando como primario otro usuario.

- R37. El sistema debe permitir ingresar los datos relacionados con una revocación de la pareja de claves principal (propia) y sus parejas de claves subordinadas.
- R38. El sistema debe permitir la creación de un paquete de revocación de la pareja de claves principal (propia) y sus parejas de claves subordinadas.
- R39. El sistema debe permitir la creación de un paquete de firma digital de revocación.
- R40. El sistema debe permitir eliminar un paquete de revocación, el cual contiene los datos asociados con una revocación de la pareja de claves principal (propia) y sus parejas de claves subordinadas. Además de su paquete de firma digital asociado. El paquete de revocación sólo puede ser eliminado si es una revocación temporal.
- R41. El sistema debe permitir eliminar un paquete de identidad de usuario y su paquete de firma digital. Paquete de identidad de usuario sólo puede ser eliminado si es una identidad de usuario no primario o si no está asociado a ninguna clave pública subordinada.
- R42. El sistema debe permitir eliminar un paquete de clave pública subordinada ajena. Además de su paquete de firma digital asociado.
- R43. El sistema debe permitir eliminar un paquete de clave pública subordinada propia y su paquete de clave privada, asociado. Además de su paquete de firma digital asociado.
- R44. El sistema debe permitir eliminar un paquete de clave pública principal ajena y todos los paquetes asociados con ella (paquete de revocación, si existe, con su paquete de firma digital asociado; paquetes de identidad de usuario con su paquete de firma digital asociado; paquetes de clave pública con su paquete de firma digital asociado).

Cifrado, descifrado, firmado digital y verificación de firma digital de datos

- R45. El sistema debe permitir ingresar los datos necesarios para inicializar un cifrador para cifrar el contenido de un archivo (cifrado híbrido).
- R46. El sistema debe permitir ingresar los datos necesarios para inicializar un descifrador para restaurar un archivo cuyo contenido fue cifrado (cifrado híbrido).
- R47. El sistema debe permitir ingresar los datos necesarios para inicializar un generador de firma digital para firmar el contenido de un archivo.
- R48. El sistema debe permitir ingresar los datos necesarios para inicializar un verificador de firma digital para verificar la firma digital de un archivo.

Intercambio de claves

Para que el usuario pueda distribuir sus claves públicas y que la gente pueda comunicarse con él de forma segura, es necesario que pueda exportar sus claves públicas. Así, como comparte sus claves públicas también es necesario que importe claves públicas de otros usuarios para que pueda comunicarse con ellos. Los requerimientos asociados son:

- R49. El sistema debe permitir al usuario compartir sus claves públicas (principal y subordinadas) con otro(s) usuario(s).

- R50. El sistema debe permitir al usuario obtener claves públicas (principal y subordinadas) de otro(s) usuario(s).

Requerimientos generales del Administrador de Claves

- R51. El sistema debe permitir cambiar las opciones de configuración para: las operaciones de cifrado y descifrado de archivos (cifrado en ASCII blindado, permitir cifrar con claves que no sean de confianza, mostrar detalle de claves al descifrar y cifrar archivo con clave predeterminada); firmado digital de archivos y verificación de firma digital (firmado en ASCII blindado, mostrar detalle de clave al verificar firma y firmar archivo con clave predeterminada); opciones de configuración de preferencias (pedir clave de sesión en cada operación).
- R52. El sistema debe proporcionar ayuda sobre el uso de las funcionalidades del sistema.

2.3.2 Requerimientos para Llavero Personal o Gestor de claves seguras

El subsistema Llavero Personal debe permitir: creación de registros; edición de registros; eliminación de registros; almacenamiento de registros en archivo digital; recuperación de registros desde archivo digital para su gestión en memoria; almacenamiento de clave segura o contraseña, cifrada; copia de clave segura o contraseña al portapapeles para un traslado seguro. Cada registro contiene en su interior datos asociados con una clave segura generada por el sistema o contraseña ingresada por el usuario.

Los requerimientos, organizados por funcionalidad, asociados con este subsistema son:

Gestión de registros en memoria

- R53. El sistema debe permitir gestionar en memoria los registros creados por el usuario en el llavero personal de contraseñas y claves seguras del sistema, esto para evitar el continuo acceso al archivo digital donde se almacenan los registros.
- R54. El sistema debe permitir almacenar un registro en memoria.
- R55. El sistema debe permitir localizar un registro en memoria.
- R56. El sistema debe permitir eliminar un registro en memoria.

Creación de registro para almacenamiento de datos asociados con una contraseña o clave segura.

- R57. El sistema debe permitir ingresar los datos relacionados con un nuevo registro para almacenarlo en el llavero personal.

Edición y eliminación de registros

- R58. El sistema debe permitir modificar los datos de nivel de confianza en el dominio y comentario sobre dominio, de un registro.
- R59. El sistema debe permitir eliminar un registro desde el llavero personal.

Recuperación de registros desde archivo digital

- R60. El sistema debe permitir obtener desde un archivo digital los registros que pertenecen a un llavero personal propio de un usuario.

Escritura de registros en archivo digital

- R61. El sistema debe permitir escribir desde memoria a un archivo digital los registros.

Copia de datos al portapapeles

- R62. El sistema debe permitir copiar el usuario, la contraseña o clave segura de un registro al portapapeles. El usuario es dueño de pegarlo donde estime conveniente.

2.3.3 Requerimientos para Complementos Simétricos

El subsistema Complementos Simétricos debe permitir: borrado seguro de archivo; generación de archivo auto-descifrable (se descifra de forma automática haciendo doble clic sobre él, sin necesidad de que el usuario de destino deba tener el software instalado en su PC); ingreso de datos para la ejecución de operaciones de cifrado/descifrado de archivo (cifrado simétrico); ingreso de datos para la ejecución de operaciones de cifrado/descifrado de archivo (cifrado simétrico PBE); ingreso de datos para la ejecución de operaciones de generación/verificación valor hash MDC; ingreso de datos para la ejecución de operaciones de generación/verificación valor hash MAC.

Los requerimientos, organizados por funcionalidad, asociados con este subsistema son:

Cifrado y descifrado simétrico de datos

- R63. El sistema debe permitir ingresar los datos necesarios para inicializar un cifrador simétrico para cifrar el contenido de un archivo.
- R64. El sistema debe permitir ingresar los datos necesarios para inicializar un descifrador simétrico para recuperar el contenido de un archivo.

Cifrado y descifrado simétrico PBE de datos

- R65. El sistema debe permitir ingresar los datos necesarios para inicializar un cifrador simétrico PBE para cifrar el contenido de un archivo.
- R66. El sistema debe permitir ingresar los datos necesarios para inicializar un descifrador simétrico PBE para recuperar el contenido de un archivo.

Generación y verificación de un valor hash MDC

- R67. El sistema debe permitir ingresar los datos necesarios para inicializar una función de integridad MDC para obtener un valor hash de un archivo.
- R68. El sistema debe permitir ingresar los datos necesarios para inicializar una función de integridad MDC para verificar un valor hash de un archivo.

Generación y verificación de un valor hash MAC

- R69. El sistema debe permitir ingresar los datos necesarios para inicializar una función de autenticación MAC para obtener un valor hash de un archivo.
- R70. El sistema debe permitir ingresar los datos necesarios para inicializar una función de autenticación MAC para verificar un valor hash de un archivo.

Requerimientos generales de Complementos Simétricos

- R71. El sistema debe permitir borrado seguro de un archivo.
- R72. El sistema debe permitir la generación de un archivo auto-descifrable (descifrador + archivo cifrado).

2.3.4 Requerimientos para Editor SICFA-DTX235

El subsistema Editor SICFA-DTX235 debe permitir: crear mensajes, en los que además de texto se puede insertar imágenes; ingreso de datos para la ejecución de operaciones de cifrado/descifrado de archivo (cifrado híbrido); ingreso de datos para la ejecución de operaciones de generación de firma digital y cifrado de mensaje/descifrado y verificación de firma digital de mensaje.

Los requerimientos, organizados por funcionalidad, asociados con este subsistema son:

Requerimientos generales de Editor SICFA-DTX235

- R73. El sistema debe permitir aplicar formato al texto; negrita, cursiva, color, subrayado.
- R74. El sistema debe permitir aplicar funciones básicas sobre el texto del editor; cortar, copiar y pegar en el documento.
- R75. El sistema debe permitir insertar una imagen prediseñada en el documento.
- R76. El sistema debe permitir deshacer y rehacer acciones realizadas en el documento del editor.

Cifrado, descifrado, firmado digital/cifrado y descifrado/verificación de firma digital

- R77. El sistema debe permitir ingresar los datos necesarios para inicializar un cifrador para cifrar el contenido del documento.
- R78. El sistema debe permitir ingresar los datos necesarios para inicializar un descifrador para recuperar el contenido de un mensaje creado en el editor.
- R79. El sistema debe permitir ingresar los datos necesarios para inicializar un generador de firma digital para firmar el contenido del documento y un cifrador para cifrar el contenido del documento.
- R80. El sistema debe permitir ingresar los datos necesarios para inicializar un descifrador para recuperar el contenido de un mensaje creado en el editor y un verificador de firma digital para verificar la firma digital del documento recuperado.

2.3.5 Requerimientos para Mecanismos de seguridad

El subsistema Mecanismos de seguridad debe permitir: la generación de parejas de claves (pública y privada) principal y subordinadas; cifrado/descifrado simétrico y asimétrico; generación/verificación valor hash MDC y MAC; generación/verificación firma digital; generación de números pseudoaleatorios.

Los requerimientos, organizados por funcionalidad, asociados con este subsistema son:

Generación de claves

- R03. El sistema debe permitir generar una pareja de claves (pública y privada).

Generación de firma digital

- R09. El sistema debe permitir obtener una firma digital para garantizar la integridad de un paquete del sistema.

Cifrado y descifrado de datos

- R45. El sistema debe permitir la combinación de cifrado simétrico y cifrado asimétrico para cifrar un archivo independiente de su extensión, el contenido cifrado se debe guardar en un archivo digital.
- R46. El sistema debe permitir descifrado de un archivo cifrado con la combinación de cifrado simétrico y asimétrico; recuperando el contenido original y restaurando el archivo que fue cifrado.
- R11b. El sistema debe permitir cifrar datos con un cifrador simétrico PBE.
- R63b. El sistema debe permitir el cifrado simétrico de un archivo independiente de su extensión, el contenido cifrado se guarda en un archivo digital.
- R64b. El sistema debe permitir el descifrado simétrico de un archivo; recuperando el contenido original y restaurando el archivo que fue cifrado.
- R65b. El sistema debe permitir el cifrado simétrico PBE de un archivo independiente de su extensión, el contenido cifrado se guarda en un archivo digital.
- R66b. El sistema debe permitir el descifrado simétrico PBE de un archivo; recuperando el contenido original y restaurando el archivo que fue cifrado.
- R65b. El sistema debe permitir la combinación de cifrado simétrico y asimétrico para cifrar el contenido que se encuentra en el documento del editor SICFA-DTX235.
- R66b. El sistema debe permitir descifrar un archivo, cifrado con la combinación de cifrado simétrico y asimétrico, con contenido cifrado; recuperando el contenido original y lo debe mostrar en un nuevo documento del editor.
- R80b. El sistema debe permitir el descifrado de un archivo, cifrado con la combinación de cifrado simétrico y asimétrico, con contenido cifrado; recuperando el contenido original y debe mostrarlo en un nuevo documento del editor. Además, debe verificar si el contenido original fue realmente firmado por un usuario en particular.

Firmado digital y verificación de firma digital de datos

- R47. El sistema debe permitir firmar digitalmente un archivo independiente de su extensión. El contenido de la firma digital se debe guardar en un archivo digital.
- R48. El sistema debe permitir verificar si un archivo independiente de su extensión fue firmado realmente por un usuario en particular.
- R79b. El sistema debe permitir el firmado digital junto con la combinación de cifrado simétrico y asimétrico para firmar y posteriormente cifrar el contenido que se encuentra en el documento del editor.

Generación y verificación de valor hash MDC de datos

- R04b. El sistema debe permitir aplicar una función de integridad a un conjunto de datos para obtener un valor hash de ellos.
- R67b. El sistema debe permitir generar un valor hash, con la ayuda de una función de integridad MDC, de un archivo independiente de su extensión. El contenido del resumen o valor hash se debe guardar en un archivo digital.
- R68b. El sistema debe permitir verificar si un valor hash generado por una función de integridad MDC corresponde a un archivo independiente de su extensión.

Generación y verificación de valor hash MAC de datos

- R69b. El sistema debe permitir generar un valor hash, con la ayuda de una función de autenticación MAC, de un archivo independiente de su extensión. El contenido del resumen o valor hash se debe guardar en un archivo digital.
- R70b. El sistema debe permitir verificar si un valor hash generado por una función de autenticación MAC corresponde a un archivo independiente de su extensión.

Capítulo 3 | SICFA-DTX235: Diseño del sistema

“La desconfianza es madre de la seguridad”

Aristófanes.

Para el diseño del sistema SICFA-DTX235 se siguió una metodología de desarrollo de software orientada a objetos basada en UML. La selección de esta metodología se debe principalmente a que el equipo de desarrollo está familiarizado con ella y ha sido utilizada en proyectos anteriores. Además, el lenguaje de programación en el cual se desarrollará el sistema es orientado a objeto facilitando aún más el trabajo.

Las ventajas más importantes de la programación orientada a objetos son las siguientes:

- Mantenibilidad (facilidad de mantenimiento). Los programas que se diseñan utilizando el concepto de orientación a objetos son más fáciles de leer y comprender y el control de la complejidad del programa se consigue gracias a la ocultación de la información que permite dejar visibles sólo los detalles más relevantes.
- Modificabilidad (facilidad para modificar los programas). Se pueden realizar añadidos o supresiones a programas simplemente añadiendo, suprimiendo o modificando objetos.
- Reusabilidad. Los objetos, si han sido correctamente diseñados, se pueden usar numerosas veces en distintos proyectos.
- Fiabilidad. Los programas orientados a objetos suelen ser más fiables ya que se basan en el uso de objetos ya definidos que están ampliamente testeados.

El capítulo está organizado de la siguiente manera: se presenta una descripción conceptual del sistema, donde se establecen cinco subsistemas; se muestra el diagrama de flujo asociado a cada subsistema; se realiza una descripción de los subsistemas; además, se muestran y especifican los diagramas de secuencias de algunas de las funcionalidades que ofrecerá cada subsistema.

3.1 Descripción conceptual de la solución

El objetivo de SICFA-DTX235 es proporcionar una serie de servicios de seguridad a un usuario final. Para ello se debe dividir el sistema en subsistemas tratando de aislar las funcionalidades afines. De esta forma el sistema quedó dividido en 5 componentes:

- Mecanismos de seguridad (MS). Este subsistema colabora con todos los demás subsistemas y permite la ejecución de operaciones criptográficas sobre un flujo de bits. Las operaciones que se pueden destacar son: generación de parejas de claves (pública y privada), cifrado/descifrado simétrico y asimétrico, generación/verificación de valor hash MDC y MAC, generación/verificación de firma digital y generación de números pseudoaleatorios. Dada la complejidad y requerimientos de seguridad que conlleva la implementación de algoritmos criptográficos, éstos son proveídos por una biblioteca externa.

- Administrador de claves (AC). Este subsistema permite: gestionar claves públicas y privadas, propias; gestionar claves públicas de otros usuarios; gestionar identidades de usuario, que asocian una clave pública con una persona real; ingreso de datos para la ejecución de operaciones de cifrado/descifrado de archivo (cifrado híbrido); ingreso de datos para la generación/verificación de firma digital de archivo; ingreso de datos para la generación/verificación de firma digital de claves públicas subordinadas propias e identidades de usuario propias.
- Complementos simétricos (CS). Este subsistema permite: borrado seguro de archivo; generación de archivo auto-descifrable; ingreso de datos para la ejecución de operaciones de cifrado/descifrado de archivo (cifrado simétrico); ingreso de datos para la ejecución de operaciones de cifrado/descifrado de archivo (cifrado simétrico PBE); ingreso de datos para la ejecución de operaciones de generación/verificación valor hash MDC; ingreso de datos para la ejecución de operaciones de generación/verificación valor hash MAC.
- Llavero personal o Gestor de claves seguras (GC). Este subsistema permite: creación de registros; edición de registros; eliminación de registros; almacenamiento de registros en archivo digital; recuperación de registros desde archivo digital para su gestión en memoria; almacenamiento de clave segura o contraseña, cifrada; copia de clave segura o contraseña al portapapeles para un traslado seguro. Cada registro contiene en su interior datos asociados con una clave segura generada por el sistema o contraseña ingresada por el usuario.
- Editor SICFA (ESIC). Este subsistema permite: crear mensajes, en los que además de texto se puede insertar imágenes; ingreso de datos para la ejecución de operaciones de cifrado/descifrado de mensaje (cifrado híbrido); ingreso de datos para la ejecución de operaciones de generación de firma digital y cifrado de mensaje/descifrado y verificación de firma digital de mensaje.

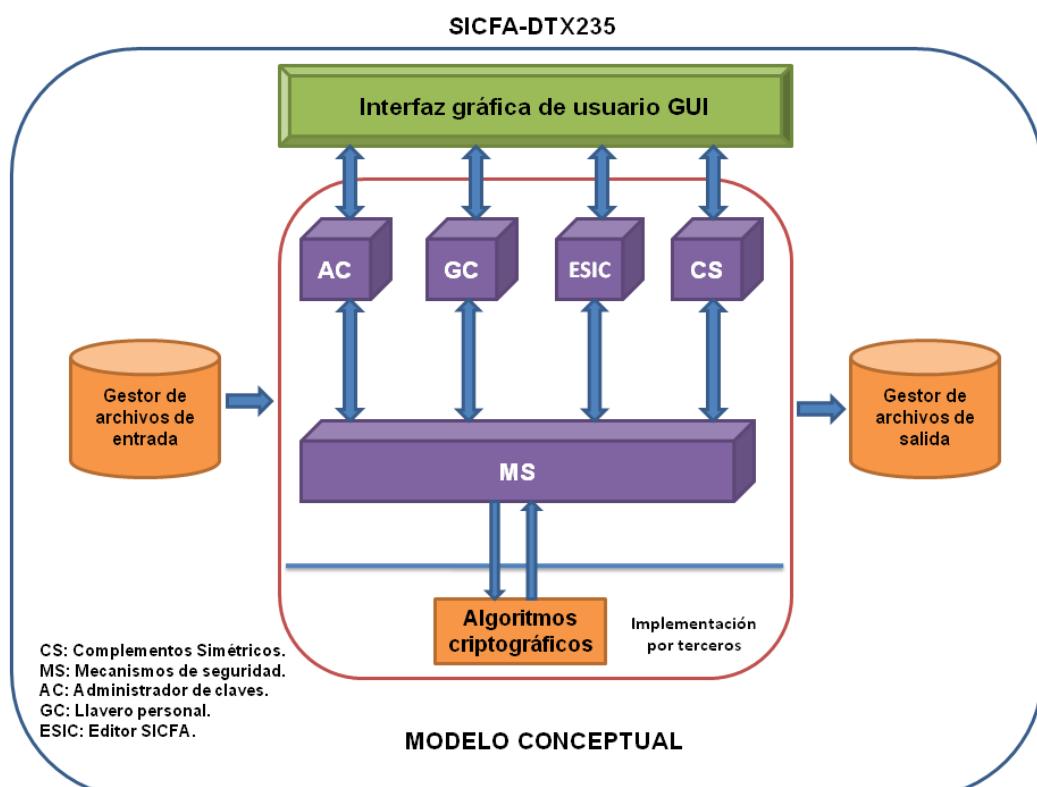


Figura 3.1: Modelo conceptual de SICFA-DTX235.

En la figura 3.1 se muestra el modelo conceptual de SICFA-DTX235, en el cual es posible apreciar la interacción de los distintos subsistemas que componen la herramienta de seguridad que será implementada. Se puede destacar que el subsistema MS se relaciona directamente con una librería que provee la implementación de algoritmos criptográficos, los cuales son aplicados en forma independiente o combinada para satisfacer las necesidades de los otros subsistemas, los que a través de sus interfaces gráficas ofrecen al usuario la posibilidad de ejecutar operaciones criptográficas, permitiéndole agregar seguridad al almacenamiento y transmisión de archivos digitales. El sistema trabajará principalmente con archivos, por lo que resulta importante contar con gestores de archivos que faciliten esta tarea.

A continuación se muestran los diagramas de flujo necesarios para describir las funciones más relevantes que ofrece cada subsistema. Los diagramas de flujo constituyen un importante medio gráfico para mostrar y conocer el conjunto de operaciones que permiten la ejecución de una función ofrecida por el sistema.

Símbolos de representación

Los símbolos de representación que se utilizaron en el desarrollo de los diagramas de flujo están representados en la figura 3.2.

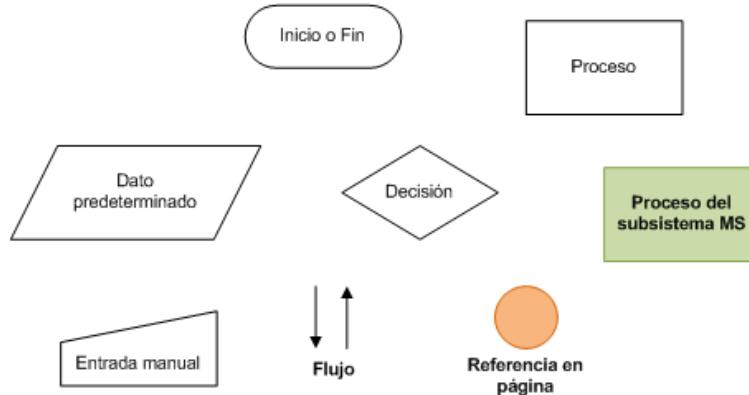


Figura 3.2: Símbolos de representación.

3.1.1 Diagrama de flujo de Iniciar sistema

Este diagrama de flujo describe los detalles algorítmicos de las funciones que permiten al usuario comenzar a utilizar el sistema. Estas funciones son: Generar pareja de claves y cargar claves propias. La figura 3.3 muestra el diagrama de flujo de iniciar sistema.

3.1.2 Diagrama de flujo de Administrador de claves (AC)

Este diagrama de flujo describe los detalles algorítmicos de algunas de las funciones ofrecidas por el subsistema AC. Las funciones consideradas en el diagrama son: Generar pareja de claves subordinadas, agregar identidad de usuario, editar identidad de usuario, editar clave pública principal, editar pareja de claves subordinadas, eliminar pareja de claves subordinadas, eliminar claves públicas ajenas y eliminar identidad de usuario. La figura 3.4 muestra el diagrama de flujo del subsistema Administrador de claves.

3.1.3 Diagrama de flujo de Llavero personal o Gestor de claves seguras (GC)

Este diagrama de flujo describe los detalles algorítmicos de algunas de las funciones ofrecidas por el subsistema GC. Las funciones consideradas en el diagrama son: Agregar nuevo registro, cargar claves propias, editar registro, eliminar registro y copiar dato al portapapeles. La figura 3.5 muestra el diagrama de flujo del subsistema Llavero personal.

3.1.4 Diagrama de flujo de Complementos simétricos (CS)

Este diagrama de flujo describe los detalles algorítmicos de algunas de las funciones ofrecidas por el subsistema CS. Las funciones consideradas en el diagrama son: Cifrado simétrico PBE, descifrado simétrico PBE, generar valor Hash MDC, generar valor Hash MAC, verificar valor Hash MDC y verificar valor Hash MAC. La figura 3.6 muestra el diagrama de flujo del subsistema Complementos simétricos.

3.1.5 Diagrama de flujo de Editor SICFA (ESIC)

Este diagrama de flujo describe los detalles algorítmicos de algunas de las funciones ofrecidas por el subsistema ESIC. Las funciones consideradas en el diagrama son: Firmar/cifrar contenido del editor, descifrar/verificar archivo de editor, cifrar contenido del editor y descifrar archivo de editor. La figura 3.7 muestra el diagrama de flujo del subsistema Editor SICFA.

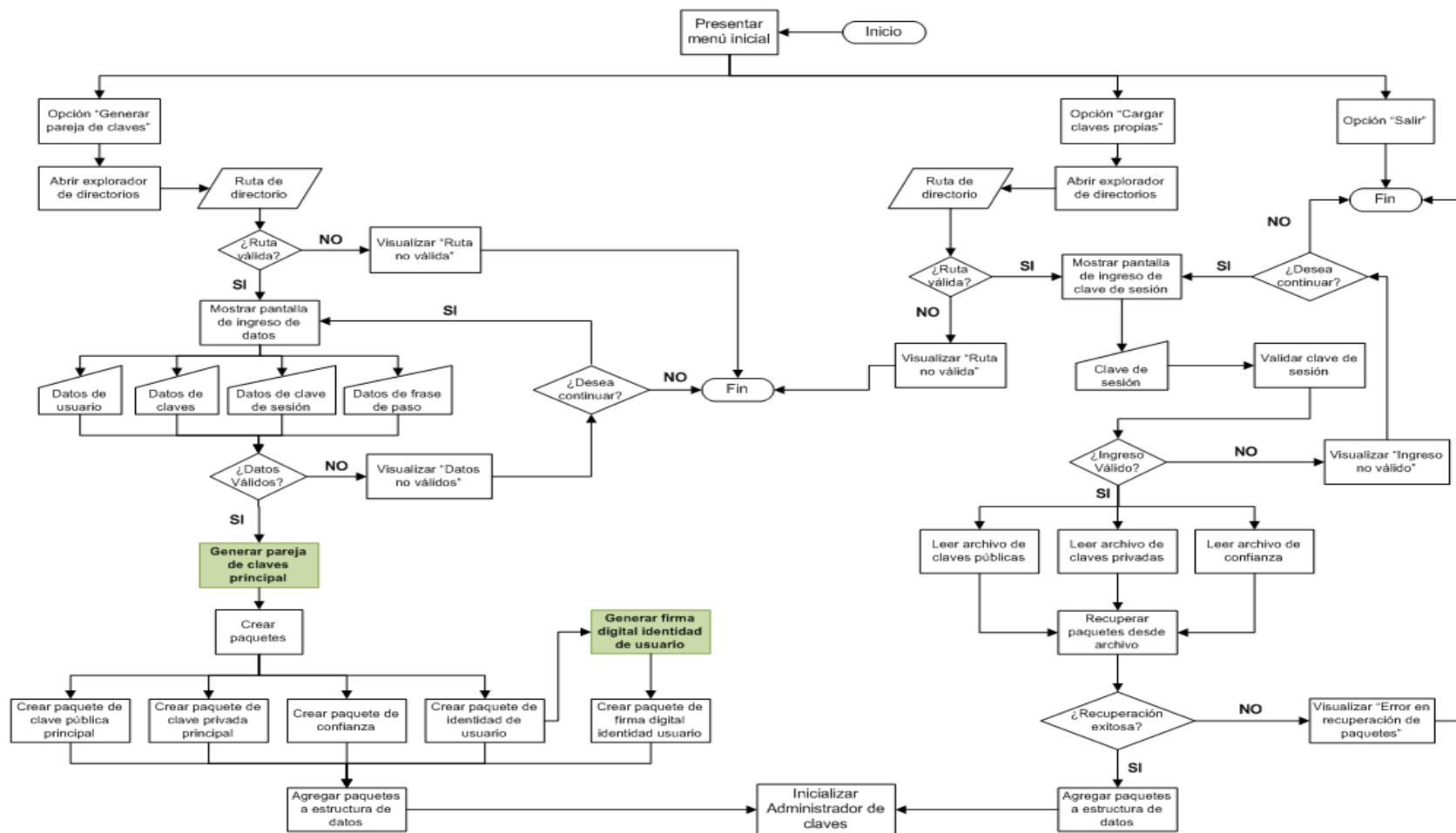


Figura 3.3: Diagrama de flujo de Iniciar sistema.

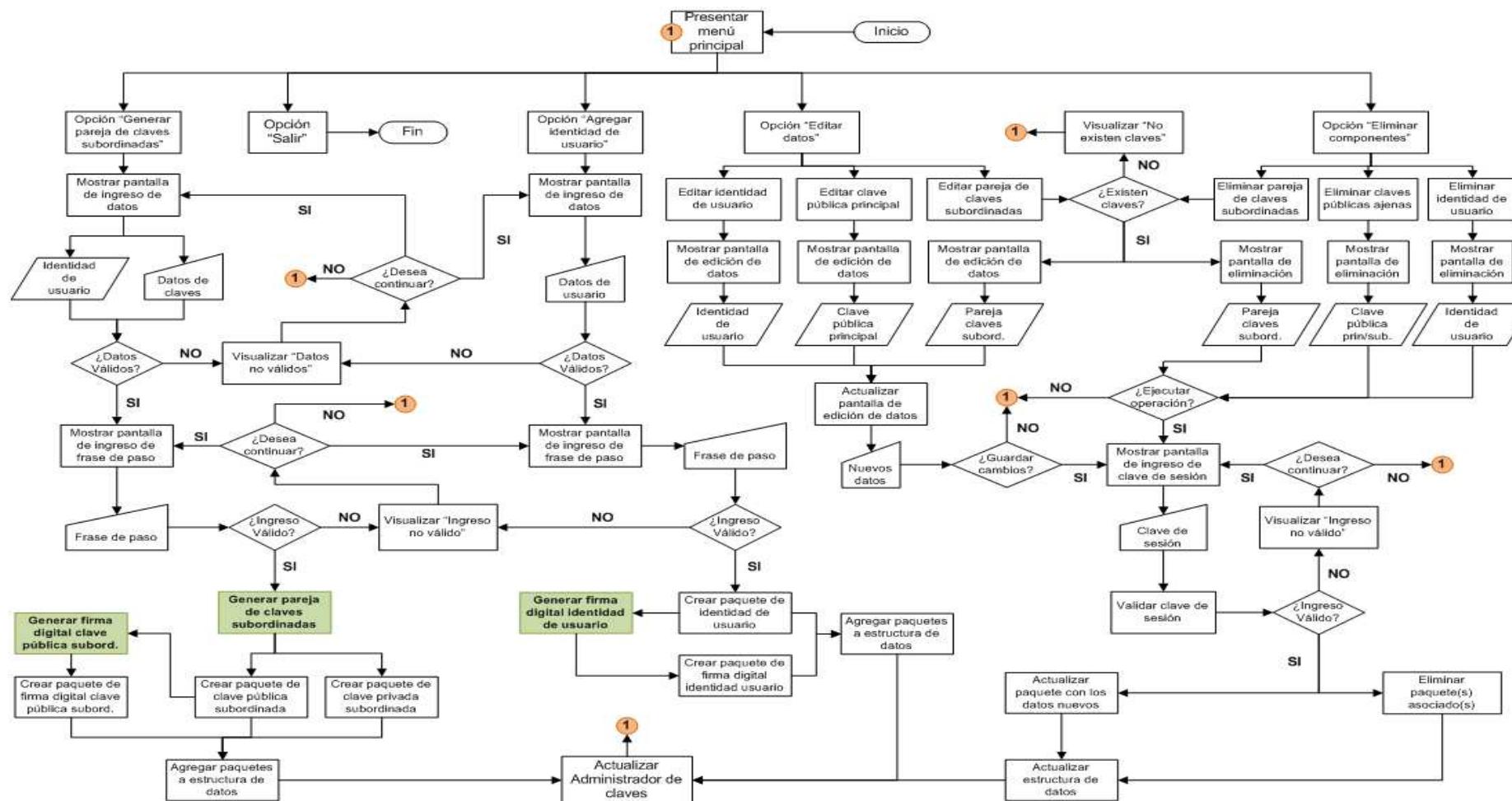


Figura 3.4: Diagrama de flujo de Administrador de claves (AC).

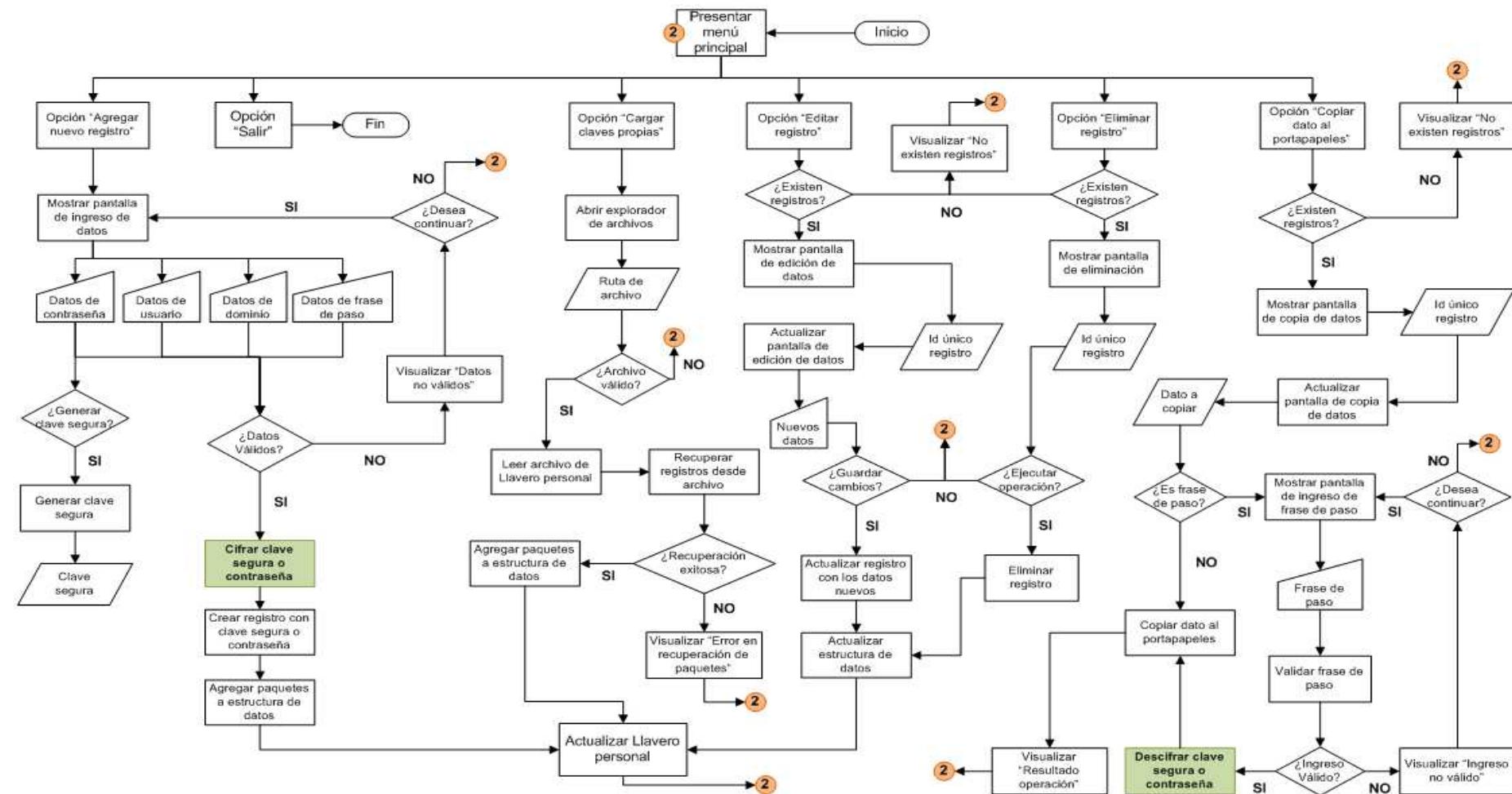


Figura 3.5: Diagrama de flujo de Llavero personal o Gestor de claves seguras (GC).

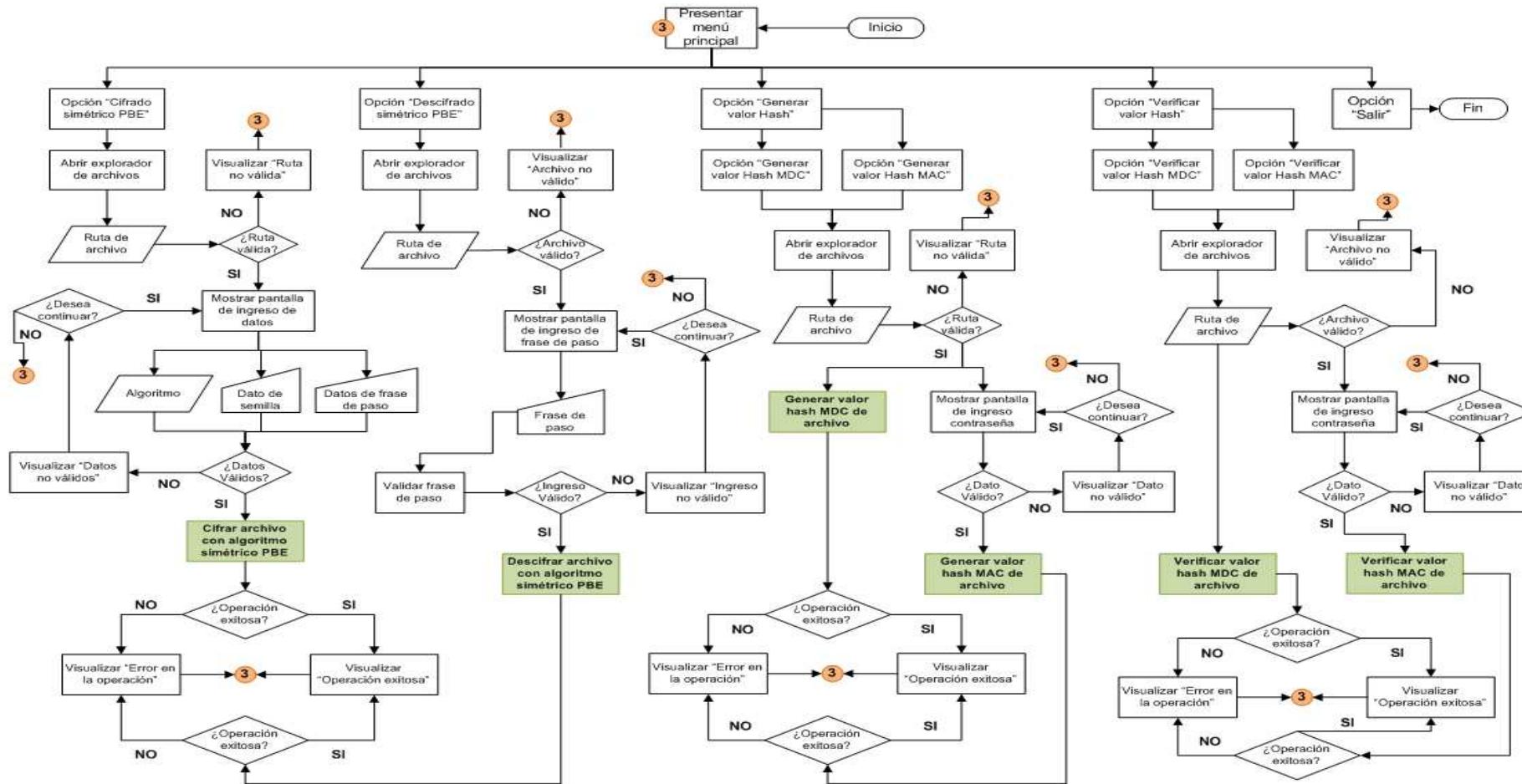


Figura 3.6: Diagrama de flujo de Complementos simétricos (CS).

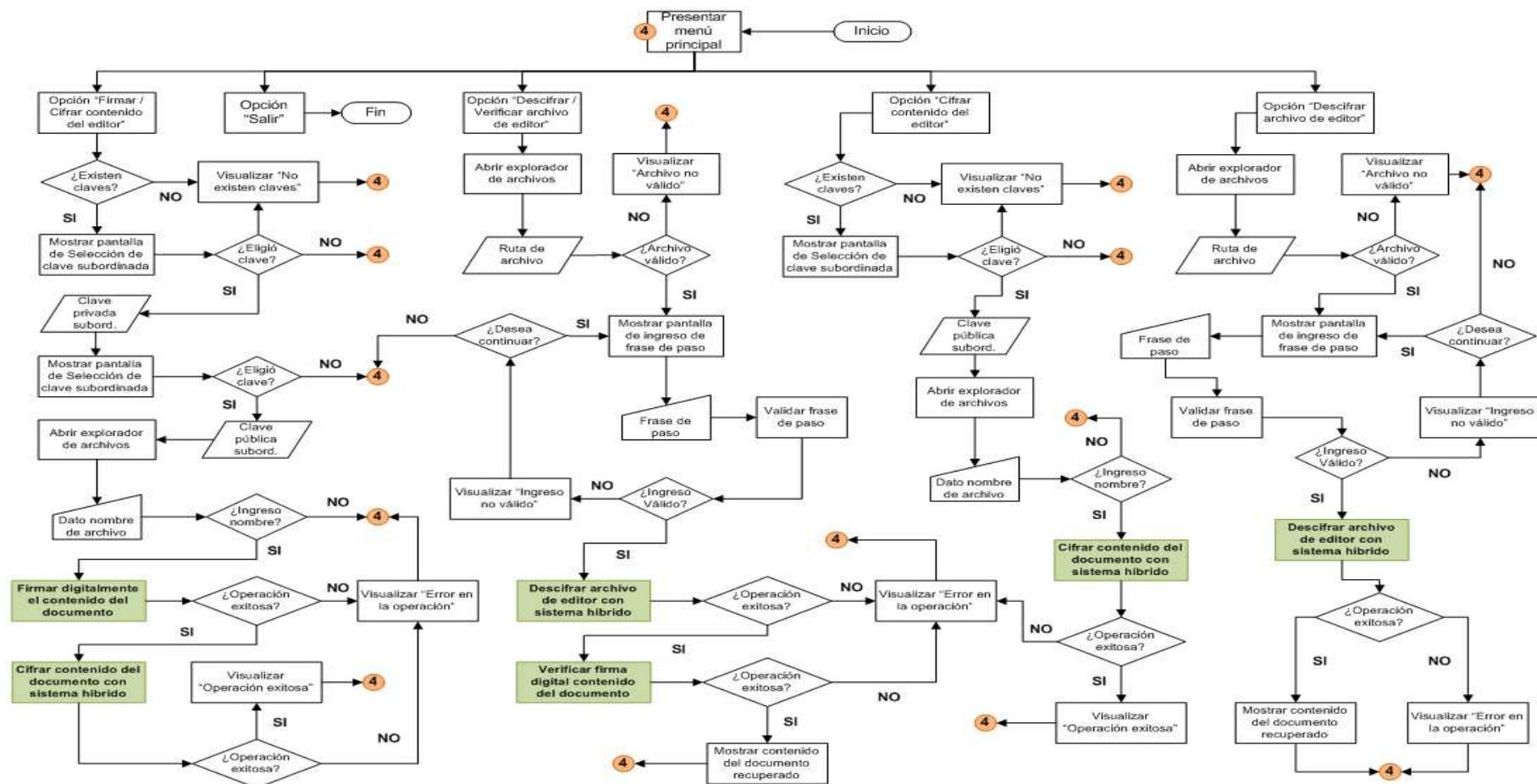


Figura 3.7: Diagrama de flujo de Editor SICFA (ESIC).

3.2 Descripción de los subsistemas

Este apartado comprende un diseño detallado de dos de los cinco subsistemas que conforman la herramienta de seguridad a implementar. Esta información adicional es de gran importancia para ahondar en el proceso de desarrollo e implementación del sistema.

3.2.1 ADMINISTRADOR DE CLAVES (AC)

Paquetes

Durante el análisis se han definido siete tipos de paquetes con los que trabajará SICFA-DTX235: Paquete de clave pública, paquete de clave privada, paquete de identidad de usuario, paquete de clave subordinada, paquete de confianza, paquete de revocación y paquete de firma digital. Aquí se describen los datos que contienen cada uno de estos paquetes.

La figura 3.8 muestra el ícono representativo de cada paquete.



Figura 3.8: Iconos representativos de paquetes.

Paquete de clave pública

- Tag. Etiqueta que identifica el contenido del paquete.
- Fecha de creación. Fecha de creación del paquete.
- Fecha de caducidad. Fecha de caducidad de la clave pública.
- Algoritmo. Algoritmo asimétrico o de clave pública.
- Clave pública. Parte pública de una pareja de claves principal.
- Id. Identificador único de la clave pública y por lo tanto del paquete.
- Huella digital. Huella digital única para identificar la clave pública principal.
- Revocada. Indicador de clave revocada.

Paquete de clave privada

- Tag. Etiqueta que identifica el contenido del paquete.
- Id. Identificador único de la clave privada y por lo tanto del paquete.
- Clave privada. Parte privada (cifrada) de una pareja de claves.
- Algoritmo. Algoritmo asimétrico o de clave pública.

Paquete de identidad de usuario

- Tag. Etiqueta que identifica el contenido del paquete.
- Fecha de creación. Fecha de creación del paquete.
- Nombre. Nombre del usuario.
- Correo. Correo electrónico del usuario.
- Comentario. Información adicional del usuario.

- Id. Identificador único de identidad de usuario y por lo tanto del paquete.
- Id propietario. Identificador único de clave pública principal asociada con este paquete.
- Primario. Indicador de identidad de usuario primario o secundario.

Paquete de clave subordinada

- Tag. Etiqueta que identifica el contenido del paquete.
- Fecha de creación. Fecha de creación del paquete.
- Fecha de caducidad. Fecha de caducidad de la clave subordinada.
- Algoritmo. Algoritmo asimétrico o de clave pública.
- Clave pública. Parte pública de una pareja de claves subordinadas.
- Id. Identificador único de la clave subordinada y por lo tanto del paquete.
- Id propietario. Identificador único de clave pública principal asociada con este paquete.
- Id usuario. Identificador único de identidad de usuario asociado con este paquete.
- Revocada. Indicador de clave revocada.

Paquete de confianza

- Tag. Etiqueta que identifica el contenido del paquete.
- Confianza. Nivel de confianza en una clave pública principal.
- Id propietario. Identificador único de clave pública principal asociada con este paquete.
- Ruta. Ruta de imagen asociada con el propietario de la clave pública principal.
- Id. Identificador único de confianza y por lo tanto del paquete.

Paquete de revocación

- Tag. Etiqueta que identifica el contenido del paquete.
- Fecha de creación. Fecha de creación del paquete.
- Motivo. Motivo por el cual se revocó la pareja de claves principal.
- Observación. Observación con respecto al motivo.
- Id. Identificador único de revocación y por lo tanto del paquete.
- Id propietario. Identificador único de clave pública principal asociada con este paquete.

Paquete de firma digital

- Tag. Etiqueta que identifica el contenido del paquete.
- Fecha de creación. Fecha de creación del paquete.
- Tipo firma. Identificador del contenido firmado.
- Algoritmo. Algoritmo asimétrico o de clave pública.
- Id firmante. Identificador único de clave privada principal con la cual se firmó digitalmente los datos del paquete.
- Id propietario. Identificador único de clave pública principal asociada con este paquete.
- Firma. Firma digital obtenida del proceso de firmado digital de datos de un paquete.

Anillos

Cada usuario tendrá 2 anillos de claves, uno para las claves públicas y otro para las claves privadas. El primero contendrá las claves públicas propias del usuario, así como las claves públicas que éste importa de otros usuarios. El segundo contendrá las claves privadas propias del usuario. Existe un tercer archivo con el que trabaja la aplicación que es el archivo o anillo de confianzas, el cual contiene los datos de confianza que se tiene en las claves públicas importadas de otros usuarios. Aquí se describe la estructura interna de cada uno de los archivos.

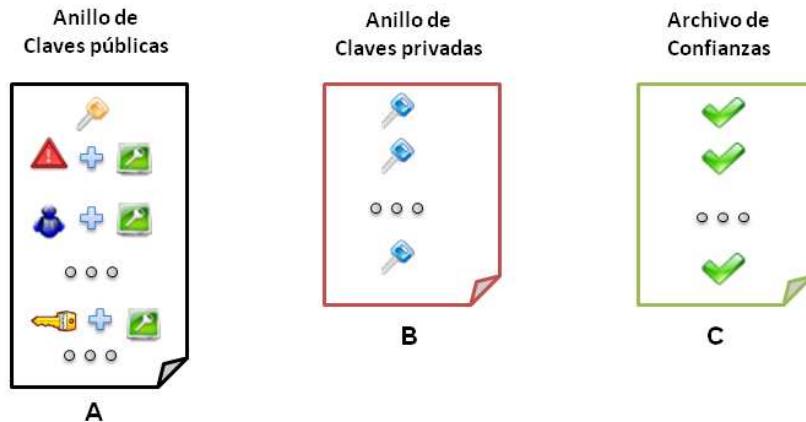


Figura 3.9: Estructura de archivos: A: Anillo de claves públicas, B: anillo de claves privadas, C: anillo de confianzas.

Anillo de claves públicas. Este archivo tendría la estructura de la figura 3.9.A.

Este archivo contiene uno o varios paquetes de clave pública principal (propio y de otros usuarios). Cada paquete de clave pública principal lleva asociado: un paquete de revocación (si existe) y su respectivo paquete de firma digital; cero o varios paquetes de claves públicas subordinadas y sus respectivos paquetes de firma digital; uno o varios paquetes de identidad de usuario y sus respectivos paquetes de firma digital. Todos los paquetes mencionados forman parte de la estructura de este archivo.

Anillo de claves privadas. Este archivo tendría la estructura de la figura 3.9.B.

Este archivo contiene el paquete de clave privada principal (propio). Este paquete de clave privada principal lleva asociado cero o varios paquetes de claves privadas subordinadas.

Anillo de confianzas. Este archivo tendría la estructura de la figura 3.9.C.

Este archivo contiene uno o varios paquetes de confianza. Cada paquete de confianza está asociado con un paquete de clave pública principal (ajeno) para indicar la confianza que el usuario tiene en esa clave y por lo tanto en el usuario que figura como su dueño.

Estructuras de datos

Listas doblemente enlazadas

En la figura 3.10 se propone una estructura de datos, específicamente listas doblemente enlazadas, para la gestión en memoria de los paquetes que utilizará el sistema. Las operaciones a realizar sobre la estructura de datos son: inserción de paquetes, localización de paquetes, actualización de paquetes y eliminación de paquetes.



Figura 3.10: Listas doblemente enlazadas con paquetes del sistema.

Árbol

La figura 3.11 muestra una estructura de datos, específicamente una estructura árbol, para la presentación de datos en pantalla. Con la ayuda de esta estructura se pretende visualizar en pantalla los datos relevantes de algunos de los paquetes para dar a conocer al usuario final parte de la información contenida en sus anillos.

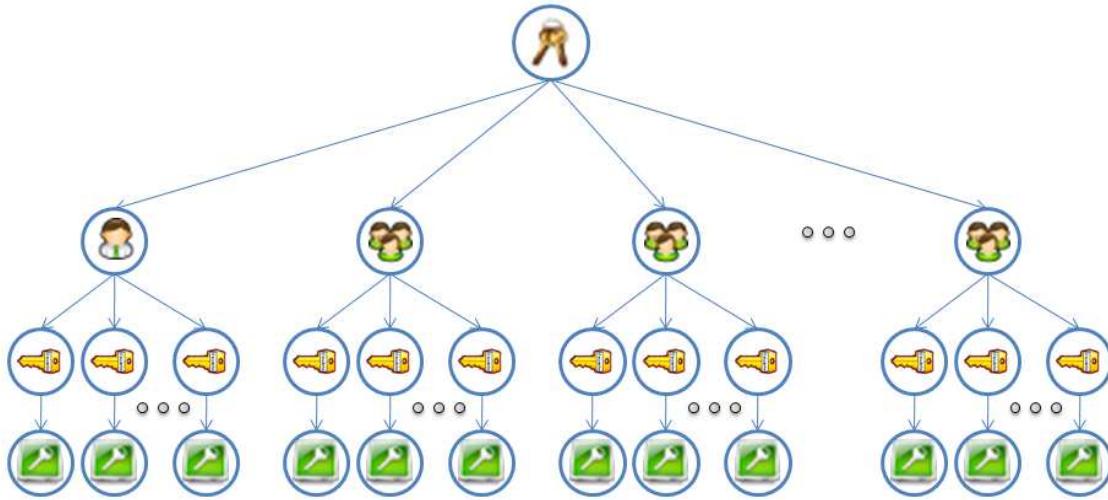


Figura 3.11: Estructura árbol para la presentación de datos por pantalla.

El árbol se compone de los siguientes nodos: un nodo raíz, nodos de clave pública principal (propia y ajenas). Cada nodo de clave pública principal lleva asociados cero o más nodos de clave pública subordinada y sus respectivos nodos de firma digital. Los datos contenidos en cada uno de los nodos son presentados en la figura 3.12.

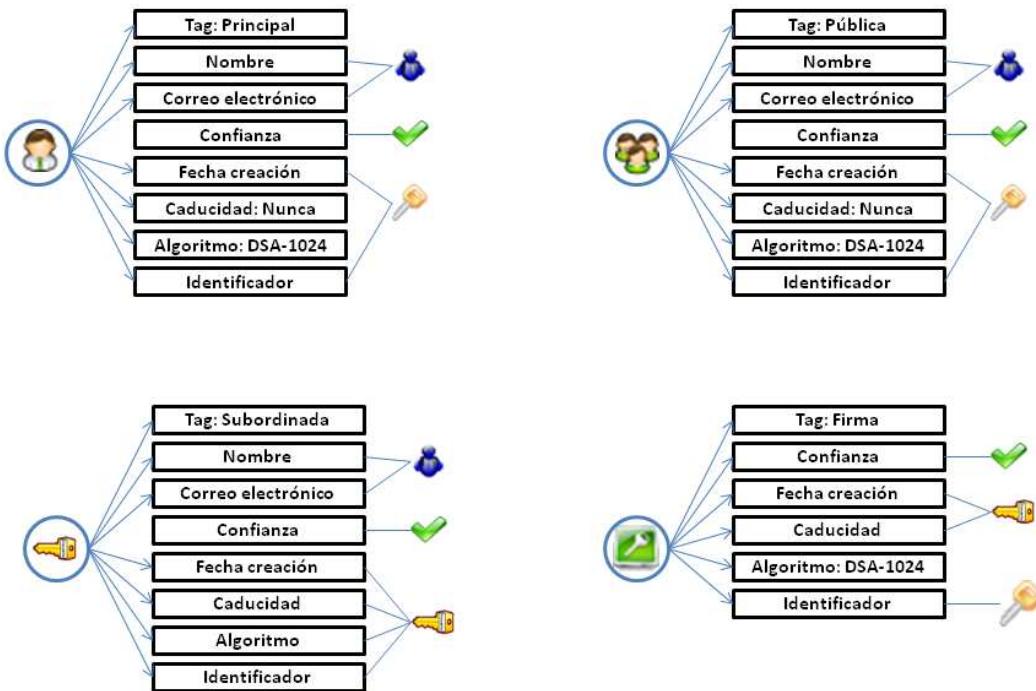


Figura 3.12: Datos contenidos en cada nodo de la estructura árbol.

Como se observa en la figura 3.12 cada nodo se compone de datos obtenidos de varios paquetes, el objetivo de esto es conseguir mostrar la relación que existe entre unos componentes y otros.

3.2.2 MECANISMOS DE SEGURIDAD (MS)

En este apartado se describen los detalles de algunas de las funciones que realizará el subsistema Mecanismos de Seguridad (MS). Resulta importante conocer el detalle del conjunto de operaciones realizadas para la ejecución de una función, para lograr con ello una implementación más rápida y eficiente.

Cifrado y descifrado híbrido de archivo

El cifrado híbrido combina cifrado simétrico y cifrado asimétrico para garantizar totalmente la confidencialidad de la comunicación, y mejorar la velocidad de los procesos de cifrado y descifrado.

Procedimiento de cifrado

01. El emisor elige un archivo digital.
02. El sistema genera un número aleatorio que será la clave secreta de sesión.
03. La clave secreta de sesión es cifrada utilizando un algoritmo asimétrico (RSA) y la clave pública RSA del destinatario. La clave secreta de sesión cifrada se añade al archivo de salida.
04. El sistema comprime el archivo digital.
05. El sistema cifra el archivo digital utilizando un algoritmo simétrico y la clave secreta de sesión. El criptograma se añade al archivo de salida.
06. El sistema también añade al archivo de salida el identificador único de la clave pública del destinatario.

La figura 3.13 presenta el esquema que muestra el procedimiento de cifrado híbrido de archivo.

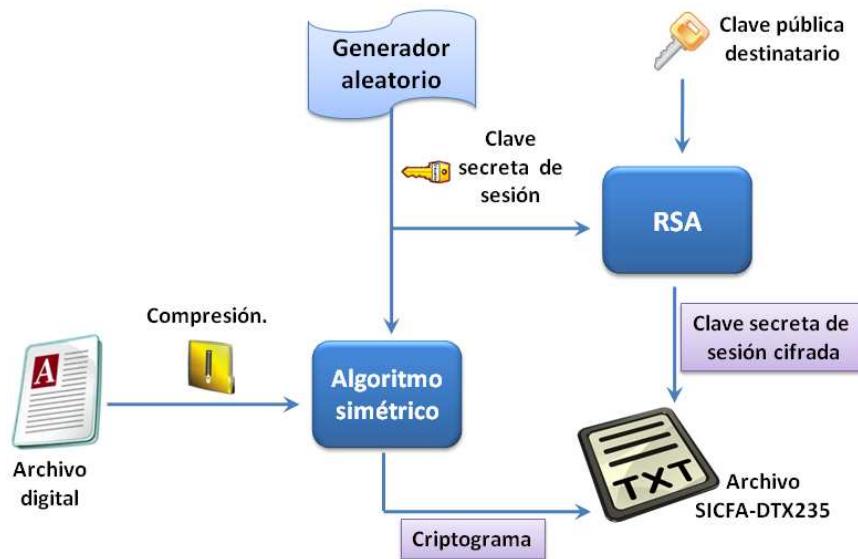


Figura 3.13: Esquema de procedimiento de cifrado híbrido de archivo.

Procedimiento de descifrado

01. El receptor elige el archivo SICFA-DTX235 recibido.
02. El sistema recupera, de la cabecera del archivo recibido, el identificador único de clave pública que identifica al receptor.
03. El sistema busca la clave privada, con ese identificador único, en el anillo de claves privadas.
04. El sistema descifra la clave privada, utilizando un algoritmo simétrico PBE y la clave binaria obtenida a partir de una frase de paso conocida por el receptor y un algoritmo Hash.
05. Con un algoritmo asimétrico (RSA) y la clave privada se descifra la clave secreta de sesión.
06. El sistema descifra el criptograma utilizando un algoritmo simétrico y la clave secreta de sesión, y recupera el archivo digital original comprimido.
07. El sistema descomprime el archivo digital original.

La figura 3.14 presenta el esquema que muestra el procedimiento de descifrado híbrido de archivo.

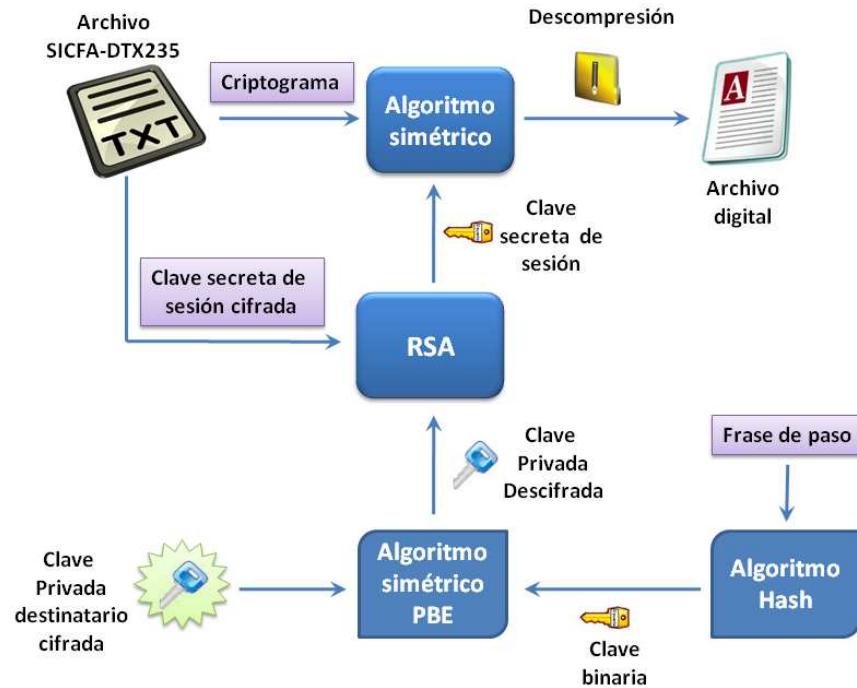


Figura 3.14: Esquema de procedimiento de descifrado híbrido de archivo.

Generación y verificación de firma digital de archivo

La forma de calcular firmas digitales consiste en emplear una combinación de cifrado asimétrico y funciones resumen.

Procedimiento de generación de firma digital

01. El emisor elige un archivo digital.
02. El sistema busca la clave privada del emisor en el anillo de claves privadas.
03. El sistema descifra la clave privada utilizando un algoritmo simétrico PBE y la clave binaria obtenida a partir de una frase de paso conocida por el emisor y un algoritmo Hash.
04. El sistema calcula un valor hash del archivo digital utilizando un algoritmo Hash.
05. El sistema genera una firma digital, a partir del valor hash, utilizando un algoritmo asimétrico y la clave privada del emisor. La firma digital se añade al archivo de salida.
06. El sistema también añade al archivo de salida el identificador único de la clave privada del emisor.

La figura 3.15 presenta el esquema que muestra el procedimiento de generación de firma digital de archivo.

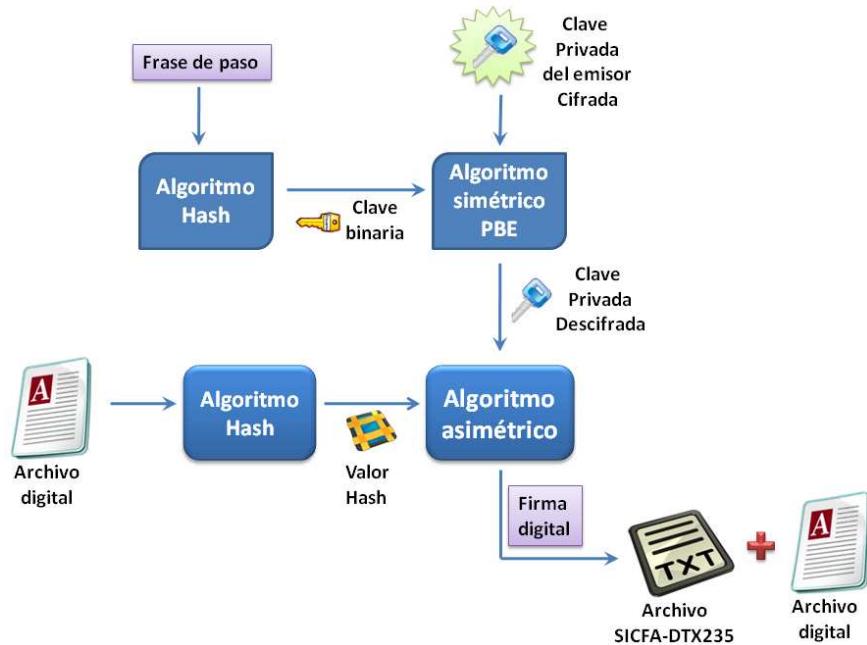


Figura 3.15: Esquema de procedimiento de generación de firma digital de archivo.

Procedimiento de verificación de firma digital

01. El receptor elige el archivo SICFA-DTX235 recibido. El archivo digital, que se firmó, debe estar en la misma carpeta del archivo recibido para comprobar la firma digital.
02. El sistema recupera, de la cabecera del archivo recibido, el identificador único de clave privada que identifica al emisor.
03. El sistema busca la clave pública, con ese identificador único, en el anillo de claves públicas.
04. El sistema calcula un valor hash del archivo digital original utilizando un algoritmo Hash.
05. El sistema recupera el valor hash enviado, a partir de la firma digital, utilizando un algoritmo asimétrico y la clave pública del emisor.
06. El sistema compara el valor hash calculado con el valor hash enviado. Si coinciden, el archivo digital es aceptado como auténtico.

La figura 3.16 presenta el esquema que muestra el procedimiento de verificación de firma digital de archivo.

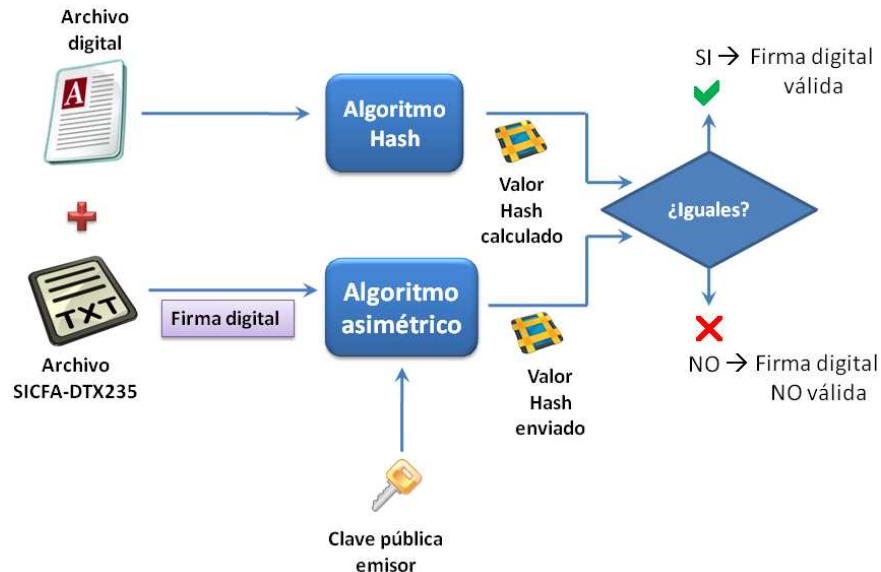


Figura 3.16: Esquema de procedimiento de verificación de firma digital de archivo.

Cifrado y descifrado simétrico PBE de archivo

Procedimiento de cifrado

01. El emisor elige un archivo digital.
02. El sistema solicita una frase de paso al emisor: ésta debe ser lo suficientemente larga como para evitar ataques por combinaciones.
03. El sistema aplica un algoritmo Hash a la frase de paso o contraseña para obtener una clave binaria.
04. El sistema comprime el archivo digital.
05. El sistema cifra el archivo digital utilizando un algoritmo simétrico PBE y la clave binaria generada. El criptograma se añade al archivo de salida.

La figura 3.17.A presenta el esquema que muestra el procedimiento de cifrado simétrico PBE de archivo.

Procedimiento de descifrado

01. El receptor elige el archivo SICFA-DTX235 recibido.
02. El sistema recupera la clave binaria, utilizada en el proceso de cifrado, utilizando un algoritmo Hash y una frase de paso conocida por el receptor.
03. El sistema descifra el criptograma utilizando un algoritmo simétrico PBE y la clave binaria recuperada, y recupera el archivo digital original comprimido.
04. El sistema descomprime el archivo digital original.

La figura 3.17.B presenta el esquema que muestra el procedimiento de descifrado simétrico PBE de archivo.

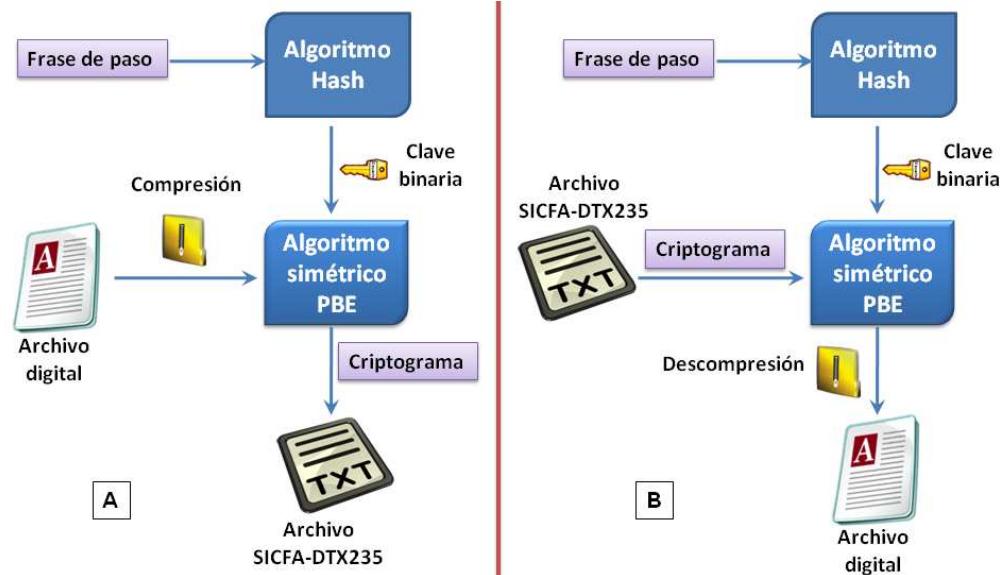


Figura 3.17: Esquema de procedimiento de cifrado (A) y descifrado simétrico PBE (B) de archivo.

Generación y verificación de valor hash MAC de archivo

Procedimiento de generación de valor hash MAC

01. El emisor elige un archivo digital.
02. El sistema inicializa un algoritmo Hash, el cual utilizando una función de autenticación reduce el archivo digital original, junto con una clave ingresada por el emisor, a una secuencia de bits que lo identifica y que se denomina valor hash. El valor hash se añade al archivo de salida.
03. El sistema también añade al archivo de salida el identificador del algoritmo Hash utilizado.

La figura 3.18.A presenta el esquema que muestra el procedimiento de generación de valor hash MAC de archivo.

Procedimiento de verificación de valor hash MAC

01. El receptor elige el archivo SICFA-DTX235 recibido. El archivo digital original, al que se le calculó el valor hash recibido, debe estar en la misma carpeta del archivo recibido para probar la integridad del contenido y la autenticación del origen.
02. El sistema recupera de la cabecera del archivo recibido el identificador del algoritmo Hash utilizado.
03. El sistema inicializa un algoritmo Hash, el cual utilizando una función de autenticación reduce el archivo digital original, junto con una clave ingresada por el receptor, a una secuencia de bits que lo identifica y que se denomina valor hash.
04. El sistema compara el valor hash calculado con el valor hash enviado. Si coinciden, entonces el archivo digital original no ha sufrido ninguna modificación y que proviene de quien dice ser su emisor, ya que sólo el emisor y receptor conocen la clave.

La figura 3.18.B presenta el esquema que muestra el procedimiento de verificación de valor hash MAC de archivo.

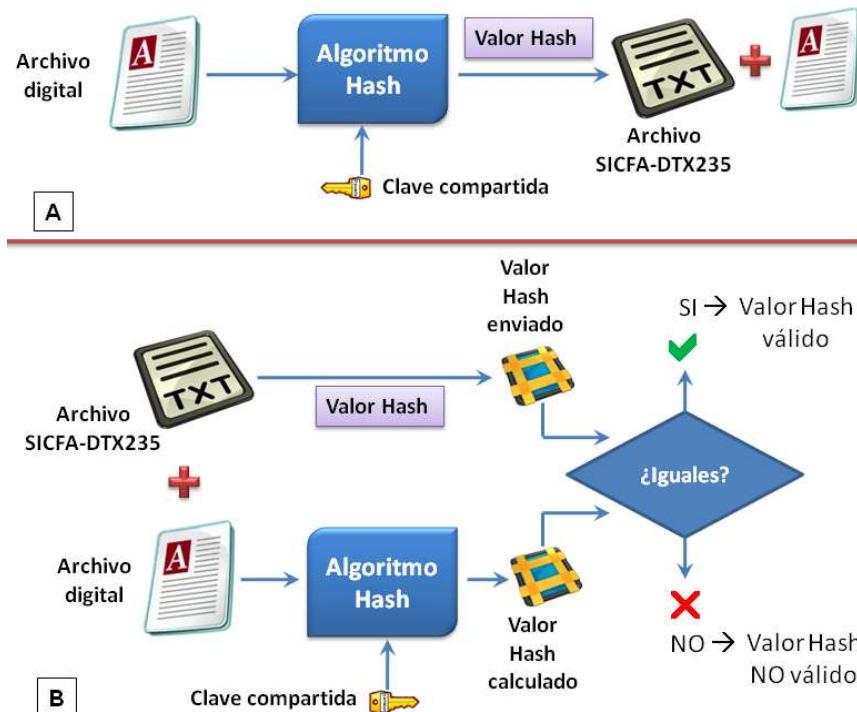


Figura 3.18: Esquema de procedimiento de generación (A) y verificación (B) de valor hash MAC de archivo.

Generación de firma digital y cifrado híbrido, de documento de editor SICFA

Procedimiento de generación de firma digital, y cifrado

01. El emisor crea un mensaje en el editor SICFA.
02. El sistema busca la clave privada del emisor en el anillo de claves privadas.
03. El sistema descifra la clave privada utilizando un algoritmo simétrico PBE y la clave binaria obtenida a partir de una frase de paso conocida por el emisor y un algoritmo Hash.
04. El sistema genera un valor hash del documento del editor utilizando un algoritmo Hash.
05. El sistema genera una firma digital, a partir del valor hash, utilizando un algoritmo asimétrico y la clave privada del emisor. La firma digital se añade al archivo de salida.
06. El sistema también añade al archivo de salida el identificador único de la clave privada del emisor.
07. El sistema genera un número aleatorio que será la clave secreta de sesión.
08. La clave secreta de sesión es cifrada utilizando un algoritmo asimétrico (RSA) y la clave pública RSA del destinatario. La clave secreta de sesión cifrada se añade al archivo de salida.
09. El sistema comprime el documento del editor.
10. El sistema cifra el documento utilizando un algoritmo simétrico y la clave secreta de sesión. El criptograma se añade al archivo de salida.
11. El sistema también añade al archivo de salida el identificador único de la clave pública del destinatario.

La figura 3.19 presenta el esquema que muestra el procedimiento de generación de firma digital, y cifrado, de documento de editor SICFA.

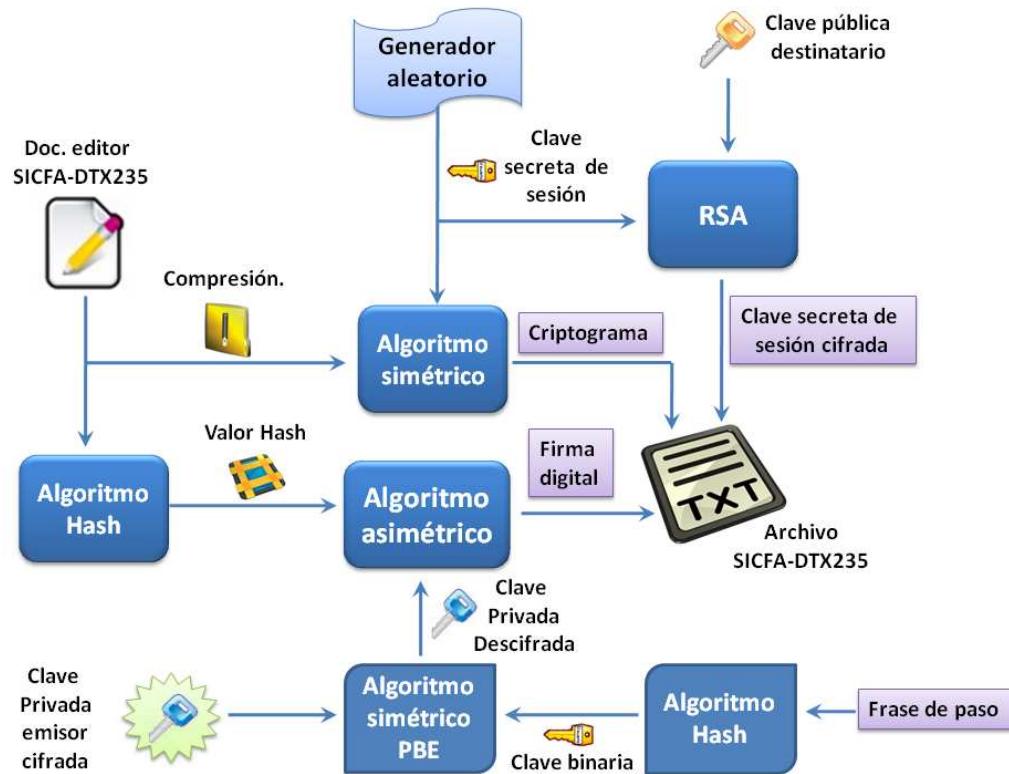


Figura 3.19: Esquema de procedimiento de generación de firma digital, y cifrado, de documento de editor SICFA.

Conversión a Base64

El software permitirá representar en caracteres ASCII (codificación Base64) cualquiera de sus salidas (criptograma, claves públicas extraídas de algún anillo, firmas digitales, valores hash, etcétera).

Para transformar datos en una codificación Base64 imprimible, el primer byte se sitúa en los 8 bits más significativos de un búfer de 24 bits, el siguiente en los 8 de en medio y el tercero en los de menor peso. Si hay menos de 3 bytes por codificar, el resto del búfer se rellena con ceros. En este punto, el búfer se usa de forma que se van extrayendo 6 bits desde la parte más significativa para ser usados como índice dentro de la tabla 3.1, de forma que el carácter indicado será la salida.

Este proceso se repite con los datos restantes hasta que queden menos de 4 octetos. Si quedan 3, se procesan de forma normal, mientras que si quedan menos de 3 bytes (24 bits), la entrada se llenará con ceros por la derecha para formar un múltiplo de 6 bits.

Después de codificar los datos, si en el paso anterior sólo quedaba 1 octeto por codificar, entonces añade el carácter '=' al final de la salida; si quedaban 2 octetos, se concatenarán 2 caracteres '='. Esto avisa al decodificador de que los bits 0 que se hayan añadido de relleno no deben formar parte de los datos reconstruidos.

Valor	Carácter	Valor	Carácter	Valor	Carácter	Valor	Carácter
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/

Tabla 3.1: Tabla ASCII codificación Base64.

SICFA-DTX235 necesita que todas las líneas codificadas estén formadas exactamente por 64 caracteres imprimibles, con la excepción de la última, que puede contener menos.

Ejemplo: Codificación de la palabra **Sol** en Base64.

Pasos:

01. Para cada letra o byte se busca su código de representación en ASCII.
S → 83; o → 111; l → 108.
02. El código ASCII representativo de cada letra se convierte a base binaria.
83 → 01010011; 111 → 01101111; 108 → 01101100.
03. Se forma un búfer de 24 bits.
Búfer → 010100110110111101101100.
04. Se toman bloques de 6 bits y se convierte cada bloque a base decimal.
010100 → 20; 110110 → 54; 111101 → 61; 101100 → 44.
05. Se codifican en Base64 cada bloque a partir del índice obtenido.
20 → U; 54 → 2; 61 → 9; 44 → s.

La figura 3.20 ilustra el proceso de conversión a Base64 de la palabra Sol.

Texto de entrada	S	o	l
ASCII	83	111	108
Bits	0 1 0 1 0 0 1 1 0 1 1 0 1 1 1 1 0 1 1 0 1 1 0 0		
Índice	20	54	61
Resultado en Base64	U	2	9

Figura 3.20: Proceso de conversión a Base64 de la palabra Sol.

Por tanto, 3 bytes sin codificar (en este caso, caracteres ASCII) entran y 4 caracteres ASCII codificados surgen como resultado.

3.3 Diagramas de secuencias

En un sistema funcional los objetos interactúan entre sí, y tales interacciones suceden con el tiempo. La idea es que las interacciones entre objetos se realicen en una secuencia establecida y que la secuencia se tome su tiempo en ir del principio a fin. Para visualizar estas interacciones es necesario especificar la secuencia, y para ello, se utilizará al diagrama correspondiente.

A continuación se describe en detalle la secuencia de pasos que se siguen en cada diagrama de secuencias relacionado con alguna de las funcionalidades que ofrece el sistema.

3.3.1 Diagrama de secuencias para: “Cargar claves propias”

La figura 3.21 presenta un diagrama de secuencias que captura las interacciones que se realizan a través del tiempo entre el usuario, la interfaz gráfica de usuario, el subsistema AC y el subsistema MS (representados como rectángulos en la parte superior del diagrama) para cargar claves desde los anillos de claves.

La secuencia sería la siguiente:

01. El usuario ejecuta la herramienta de software.
02. El sistema presenta al usuario la interfaz inicial con 2 opciones: “Crear pareja de claves principal” y “Cargar claves propias”.
03. El usuario elige la opción: “Cargar claves propias”.
04. El sistema muestra la interfaz de exploración de directorios.
05. El usuario busca y selecciona el directorio deseado.
06. El sistema muestra la interfaz para el ingreso de la clave de sesión.
07. El usuario ingresa su clave de sesión.
08. El sistema solicita al subsistema AC validar la clave de sesión ingresada y, le pasa la ruta del directorio y la clave de sesión.
09. El subsistema AC solicita al subsistema MS generar un valor hash de la clave de sesión ingresada y, le pasa la clave de sesión.
10. El subsistema MS inicializa un generador de valores hash.
11. El subsistema MS actualiza el generador con la clave de sesión.
12. El subsistema MS genera el valor hash a partir de la clave de sesión.
13. El subsistema MS envía el valor hash generado al subsistema AC.
14. El subsistema AC compara el valor hash, de clave de sesión, obtenido con un valor hash, de clave de sesión, que tiene almacenado en memoria. Si ambos coinciden, entonces se continúa con el proceso.
15. El subsistema AC lee el archivo de claves públicas. Mientras tenga paquetes (el propio y los ajenos) de clave pública principal, el subsistema realizará los pasos 16, 17, 18, y 19.
16. El subsistema AC recupera el paquete de clave pública principal y lo agrega a la estructura.

17. El subsistema AC recupera el paquete de revocación (si existe), y su paquete de firma digital y los agrega a la estructura.
18. El subsistema AC recupera cada paquete de identidad de usuario con su paquete de firma digital y los agrega a la estructura.
19. El subsistema AC recupera cada paquete de clave pública subordinada con su paquete de firma digital y los agrega a la estructura.
20. El subsistema AC lee el archivo de claves privadas.
21. El subsistema AC recupera el paquete de clave privada principal y los agrega a la estructura.
22. El subsistema AC recupera cada paquete de clave privada subordinada (si existen), y los agrega a la estructura.
23. El subsistema AC lee el archivo de confianzas.
24. El subsistema AC recupera cada paquete de confianza y los agrega a la estructura.
25. Si la lectura de los archivos ha sido correcta, se continúa con el proceso. En caso contrario, el sistema termina su ejecución.
26. El subsistema AC envía la estructura para que sea presentada al usuario en la interfaz gráfica de usuario.
27. El sistema actualiza la interfaz gráfica.
28. Fin del proceso.

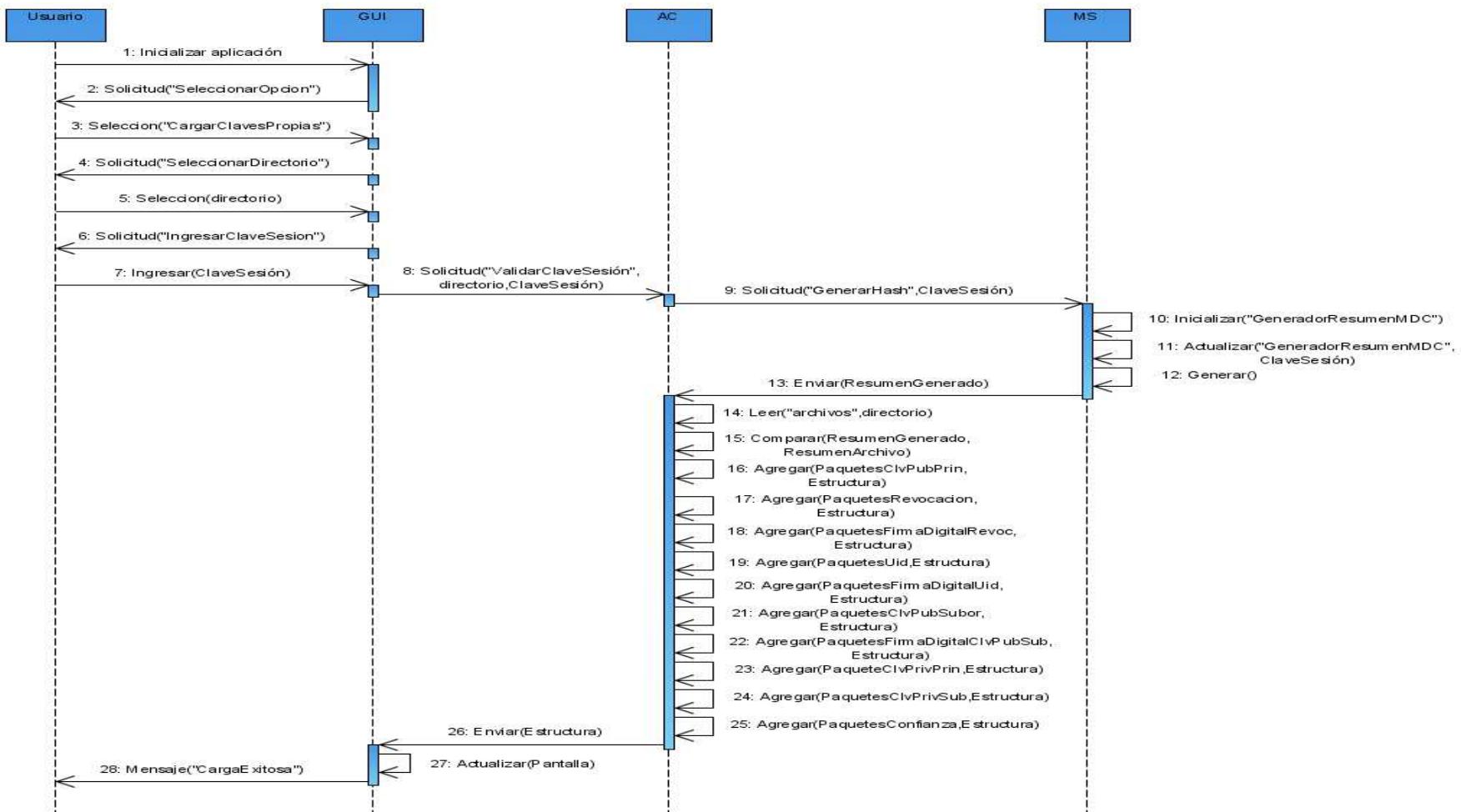


Figura 3.21: Diagrama de secuencias para: “Cargar claves propias”.

3.3.2 Diagrama de secuencias para: “Exportar clave(s) pública(s)”

La figura 3.22 presenta un diagrama de secuencias que captura las interacciones que se realizan a través del tiempo entre el usuario, la interfaz gráfica de usuario y el subsistema AC (representados como rectángulos en la parte superior del diagrama) para compartir las claves públicas propias con otros usuarios.

La secuencia sería la siguiente:

01. El sistema presenta al usuario la interfaz del administrador de claves que contiene un menú desplegable.
02. El usuario selecciona del menú la opción: “Exportar clave(s) pública(s)”.
03. El sistema solicita al subsistema AC una lista de las claves públicas principales disponibles en la estructura que las gestiona en memoria.
04. El subsistema AC envía la lista con las claves públicas principales para que sea presentada al usuario en la interfaz gráfica de usuario.
05. El sistema actualiza la interfaz gráfica, solicitando al usuario la selección de una de las claves presentadas.
06. El usuario selecciona la clave pública principal deseada.
07. El sistema solicita al subsistema AC la exportación de la clave pública principal seleccionada y de todos los datos asociados con ella (claves públicas subordinadas e identidades de usuarios) y, le pasa la clave pública principal seleccionada.
08. El subsistema AC crea un nuevo archivo digital.
09. El subsistema AC escribe paquete de clave pública principal en archivo.
10. El subsistema AC escribe paquete de revocación (si existe) con su paquete de firma digital en el archivo.
11. El subsistema AC escribe cada paquete de identidad de usuario con su paquete de firma digital en el archivo.
12. El subsistema AC escribe cada paquete de clave pública subordinada con su paquete de firma digital en el archivo.
13. El subsistema AC informa a la interfaz gráfica del resultado de la exportación.
14. El sistema informa al usuario del resultado de la exportación a través de la interfaz gráfica.
15. Fin del proceso.

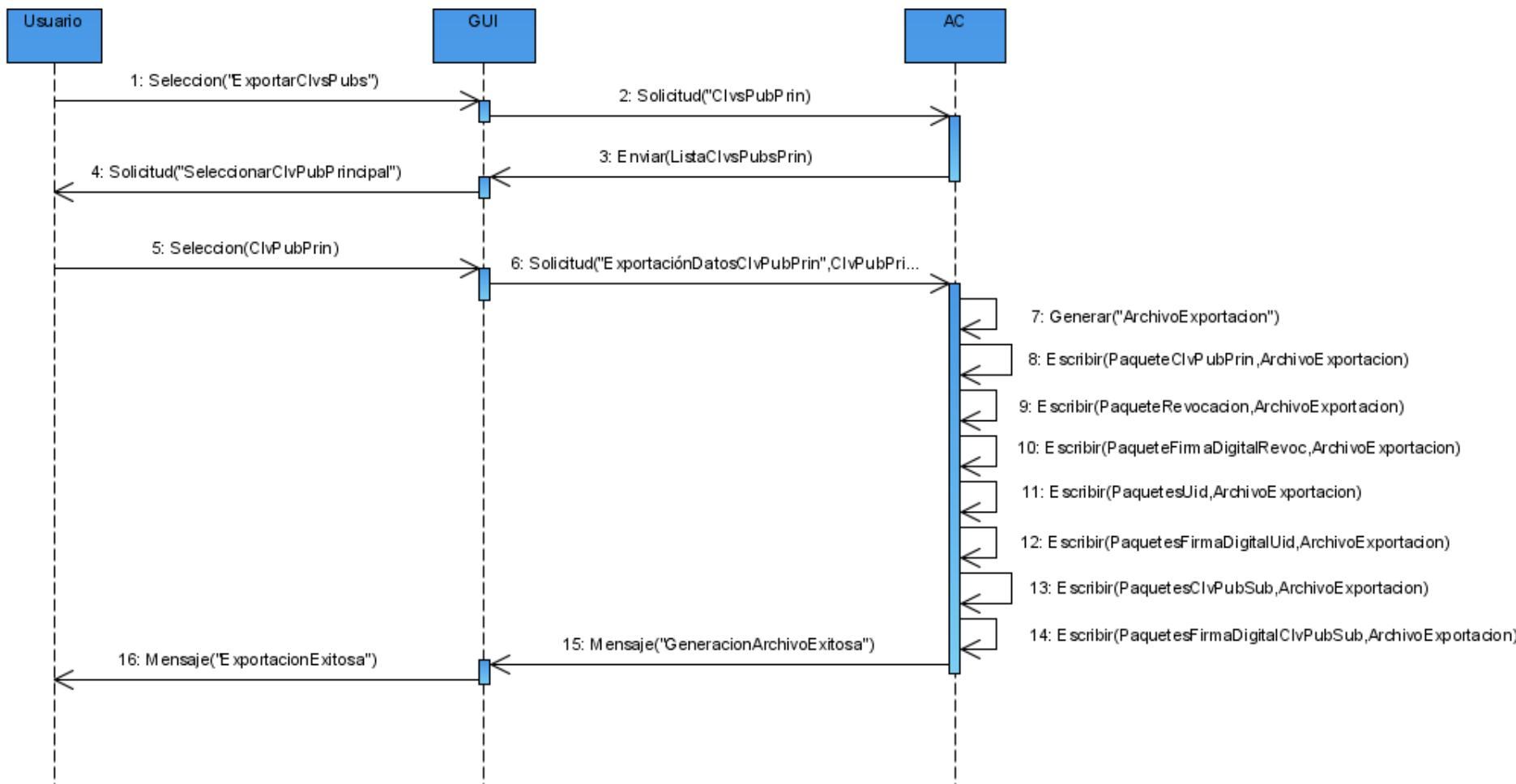


Figura 3.22: Diagrama de secuencias para: “Exportar clave(s) pública(s)”.

3.3.3 Diagrama de secuencias para: “Generar pareja de claves subordinadas”

La figura 3.23 presenta un diagrama de secuencias que captura las interacciones que se realizan a través del tiempo entre el usuario, la interfaz gráfica de usuario, el subsistema MS y el subsistema AC (representados como rectángulos en la parte superior del diagrama) para la generación de una nueva pareja de claves subordinadas.

La secuencia sería la siguiente:

01. El sistema presenta al usuario la interfaz del administrador de claves que contiene un menú desplegable.
02. El usuario selecciona del menú la opción: “Generar pareja de claves”.
03. El sistema muestra la interfaz para el ingreso de datos.
04. El usuario ingresa los datos de pareja de claves subordinadas y selecciona identidad de usuario con la cual asociará la pareja de claves subordinadas.
05. El sistema muestra la interfaz para el ingreso de la frase de paso.
06. El usuario ingresa la frase de paso (esta frase de paso es la que ingresa el usuario al momento de crear su pareja de claves principal).
07. El sistema solicita al subsistema MS generar una pareja de claves subordinadas y, le pasa la frase de paso.
08. El sistema envía los datos de pareja de claves subordinadas, identificador de identidad de usuario al subsistema AC.
09. El subsistema MS genera la pareja de claves (pública y privada).
10. El subsistema MS envía pareja de claves subordinadas al subsistema AC.
11. El subsistema AC crea el paquete de clave pública subordinada.
12. El subsistema AC solicita al subsistema MS una firma digital para los datos de clave pública subordinada y, le pasa el paquete de clave pública subordinada y la clave privada principal cifrada.
13. El subsistema MS inicializa el descifrador simétrico PBE con clave binaria obtenida a partir de la frase de paso.
14. El subsistema MS descifra la clave privada con el descifrador simétrico PBE.
15. El subsistema MS firma digitalmente los datos de clave pública subordinada y envía la firma digital generada al subsistema AC.
16. El subsistema AC crea el paquete de firma digital de clave pública subordinada.
17. El subsistema AC crea el paquete de clave privada subordinada.
18. El subsistema AC agrega los paquetes a una estructura de datos para almacenarlos en memoria y así tener un rápido acceso a ellos.
19. El subsistema AC envía la estructura para que sea presentada al usuario en la interfaz gráfica de usuario.
20. El sistema actualiza la interfaz gráfica.
21. Fin del proceso.

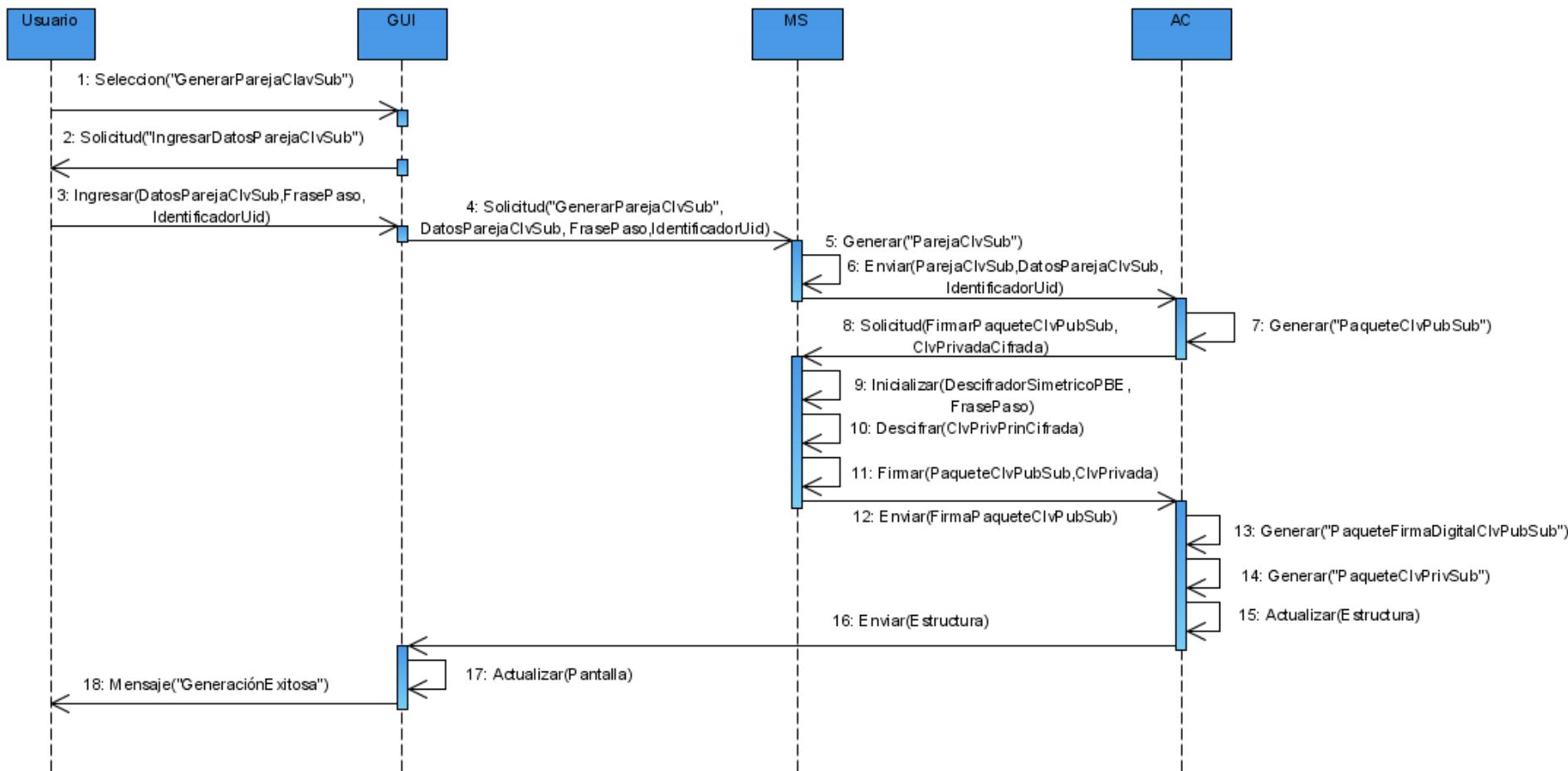


Figura 3.23: Diagrama de secuencias para: “Generar pareja de claves subordinadas”.

3.3.4 Diagrama de secuencias para: “Cifrar archivo con algoritmo simétrico”

La figura 3.24 presenta un diagrama de secuencias que captura las interacciones que se realizan a través del tiempo entre el usuario, la interfaz gráfica de usuario y el subsistema MS (representados como rectángulos en la parte superior del diagrama) para cifrar un archivo (cifrado simétrico).

La secuencia sería la siguiente:

01. El sistema presenta al usuario la interfaz de complementos simétricos, que contiene una serie de botones.
02. El usuario hace clic, con el mouse, en el botón: “Cifrador simétrico”.
03. El sistema muestra la interfaz de exploración de archivos.
04. El usuario busca y selecciona el archivo que desea cifrar.
05. El sistema presenta al usuario la interfaz de selección de algoritmo.
06. El usuario busca y selecciona el algoritmo simétrico con el que desea cifrar.
07. El sistema solicita al subsistema MS cifrar el archivo elegido y, le pasa la ruta del archivo junto con el algoritmo simétrico seleccionado.
08. El subsistema MS genera una clave secreta de sesión para algoritmo simétrico.
09. El subsistema MS inicializa el cifrador simétrico con algoritmo y la clave secreta de sesión.
10. El subsistema MS crea un nuevo archivo digital.
11. El subsistema MS escribe en archivo el identificador de algoritmo simétrico usado.
12. El subsistema MS lee los datos desde el archivo original.
13. El subsistema MS cifra los datos con el cifrador simétrico.
14. El subsistema MS escribe los datos cifrados en el archivo.
15. El subsistema MS informa a la interfaz gráfica del resultado de la operación de cifrado y, le pasa la clave secreta de sesión utilizada.
16. El sistema informa al usuario del resultado de la operación de cifrado a través de la interfaz gráfica. Además, le muestra la clave secreta de sesión utilizada.
17. Fin del proceso.

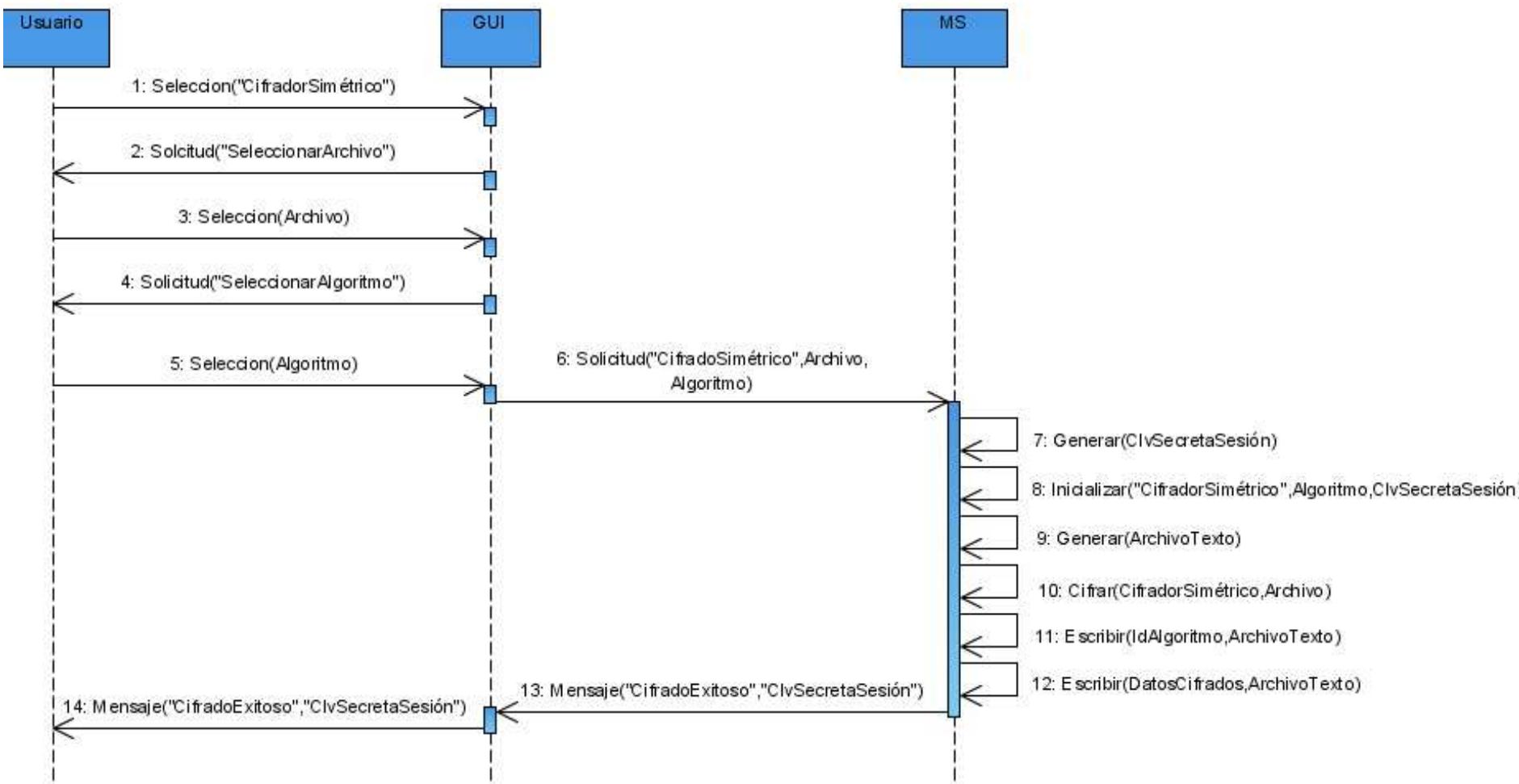


Figura 3.24: Diagrama de secuencias para: “Cifrar archivo con algoritmo simétrico”.

3.3.5 Diagrama de secuencias para: “Cifrar contenido del editor”

La figura 3.25 presenta un diagrama de secuencias que captura las interacciones que se realizan a través del tiempo entre el usuario, la interfaz gráfica de usuario, el subsistema AC, el subsistema ESIC y el subsistema MS (representados como rectángulos en la parte superior del diagrama) para cifrar el contenido de un documento del editor SICFA-DTX235 (combinación de cifrado asimétrico y cifrado simétrico).

La secuencia sería la siguiente:

01. El sistema presenta al usuario la interfaz del editor SICFA-DTX235 que contiene un menú desplegable.
02. El usuario selecciona del menú la opción: “Cifrar contenido”.
03. El sistema presenta una interfaz para el ingreso de un nombre de archivo.
04. El usuario ingresa el nombre del archivo que contendrá el contenido cifrado.
05. El sistema solicita al subsistema AC una lista con las claves públicas subordinadas disponibles en la estructura que las gestiona en memoria.
06. El subsistema AC envía la lista con las claves públicas subordinadas para que sea presentada al usuario en la interfaz gráfica de usuario.
07. El sistema actualiza la interfaz gráfica, solicitando al usuario la selección de una de las claves presentadas.
08. El usuario selecciona la clave pública subordinada deseada.
09. El sistema solicita al subsistema ESIC cifrar el contenido del editor y, le pasa el nombre de archivo y la clave pública subordinada seleccionada.
10. El subsistema ESIC solicita al subsistema MS cifrar el contenido del editor y, le pasa el contenido del editor, el nombre de archivo y la clave pública subordinada seleccionada.
11. El subsistema MS genera una clave secreta de sesión para algoritmo simétrico.
12. El subsistema MS inicializa el cifrador simétrico con la clave secreta de sesión.
13. El subsistema MS inicializa el cifrador asimétrico con la clave pública subordinada seleccionada.
14. El subsistema MS cifra la clave secreta de sesión con el cifrador asimétrico.
15. El subsistema MS crea un nuevo archivo digital con nombre de archivo elegido.
16. El subsistema MS escribe en archivo el identificador único de la clave pública subordinada y la clave secreta de sesión cifrada.
17. El subsistema MS cifra contenido del editor con el cifrador simétrico.
18. El subsistema MS escribe los datos cifrados en el archivo.
19. El subsistema MS informa a la interfaz gráfica del resultado de la operación de cifrado.
20. El sistema informa al usuario del resultado de la operación de cifrado a través de la interfaz gráfica.
21. Fin del proceso.

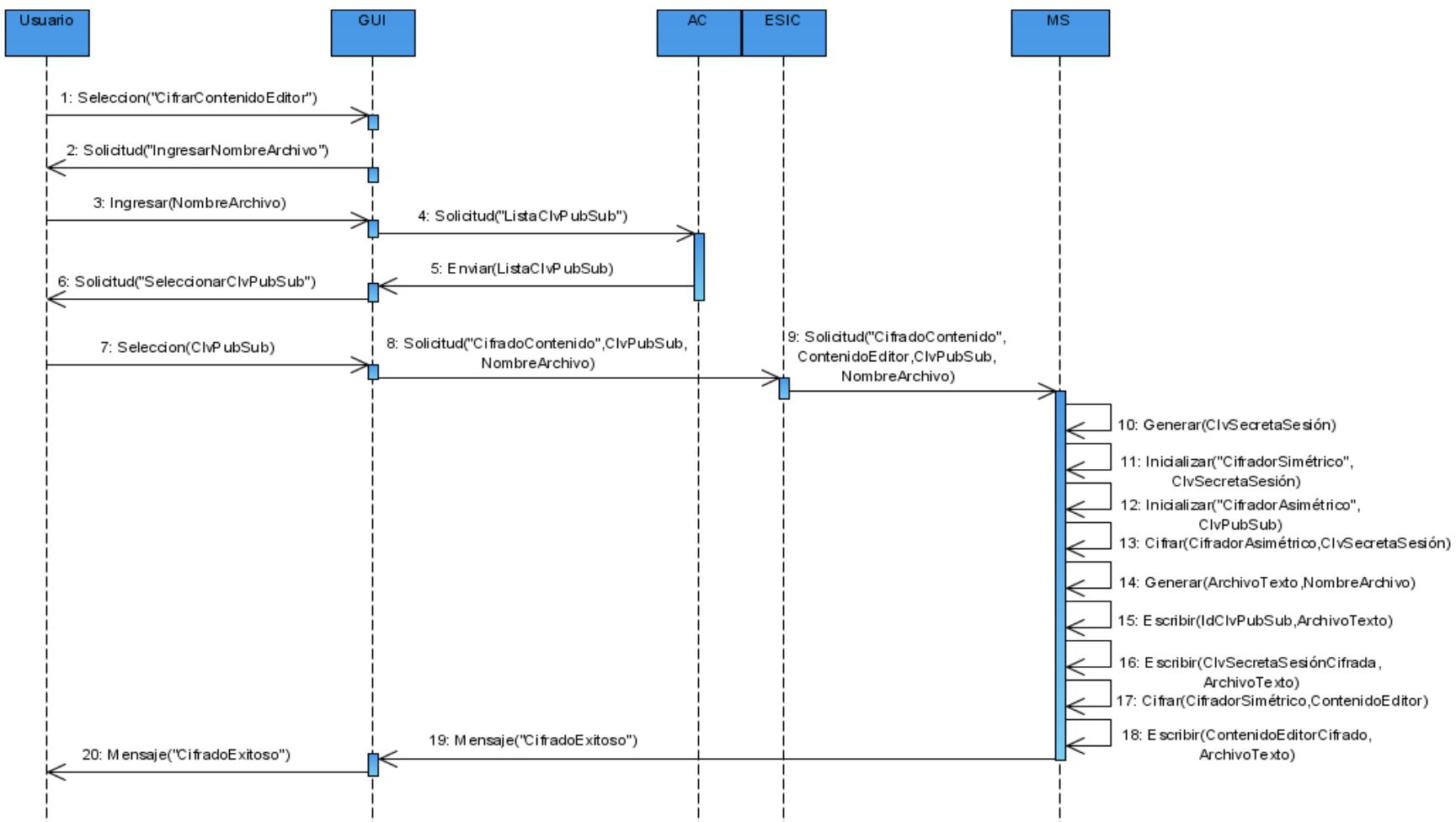


Figura 3.25: Diagrama de secuencias para: “Cifrar contenido del editor”.

3.3.6 Diagrama de secuencias para: “Copiar datos al portapapeles”

La figura 3.26 presenta un diagrama de secuencias que captura las interacciones que se realizan a través del tiempo entre el usuario, la interfaz gráfica de usuario, el subsistema GC y el subsistema MS (representados como rectángulos en la parte superior del diagrama) para copiar datos de un registro al portapapeles. Sólo se solicita la frase de paso para registro cuando el dato a copiar es la contraseña o clave segura.

La secuencia sería la siguiente:

01. El sistema presenta al usuario la interfaz del llavero personal, que contiene una serie de botones.
02. El usuario hace clic, con el mouse, en el botón: “Copiar datos al portapapeles”.
03. El sistema solicita al subsistema GC una lista de los registros disponibles en la estructura que los gestiona en memoria.
04. El subsistema GC envía la lista con los registros para que sea presentada al usuario en la interfaz gráfica de usuario.
05. El sistema actualiza la interfaz gráfica, solicitando al usuario la selección de uno de los registros.
06. El usuario selecciona el registro deseado.
07. El sistema solicita al subsistema GC los datos asociados con el registro seleccionado.
08. El subsistema GC envía los datos asociados con el registro seleccionado a la interfaz gráfica y solicita al usuario seleccionar el dato que desea copiar al portapapeles.
09. El usuario selecciona el dato a copiar.
10. El sistema muestra la interfaz para el ingreso de la frase de paso para registro.
11. El usuario ingresa la frase de paso (esta frase de paso es la que ingresa el usuario al momento de crear el registro).
12. El sistema solicita al subsistema GC copiar el dato elegido por el usuario al portapapeles y, le pasa el dato a copiar junto con la frase de paso para registro.
13. El subsistema GC solicita al subsistema MS descifrar el dato a copiar elegido por el usuario y, le pasa el dato a copiar junto con la frase de paso para registro.
14. El subsistema MS inicializa el descifrador simétrico PBE con clave binaria obtenida a partir de la frase de paso.
15. El subsistema MS descifra el dato a copiar con el descifrador simétrico PBE.
16. El subsistema MS envía al subsistema GC el dato a copiar ya descifrado.
17. El subsistema MS copia el dato al portapapeles.
18. El subsistema GC informa a la interfaz gráfica del resultado de la operación de copia de dato al portapapeles.
19. El sistema informa al usuario del resultado de la operación de copia de dato al portapapeles.
20. Fin del proceso.

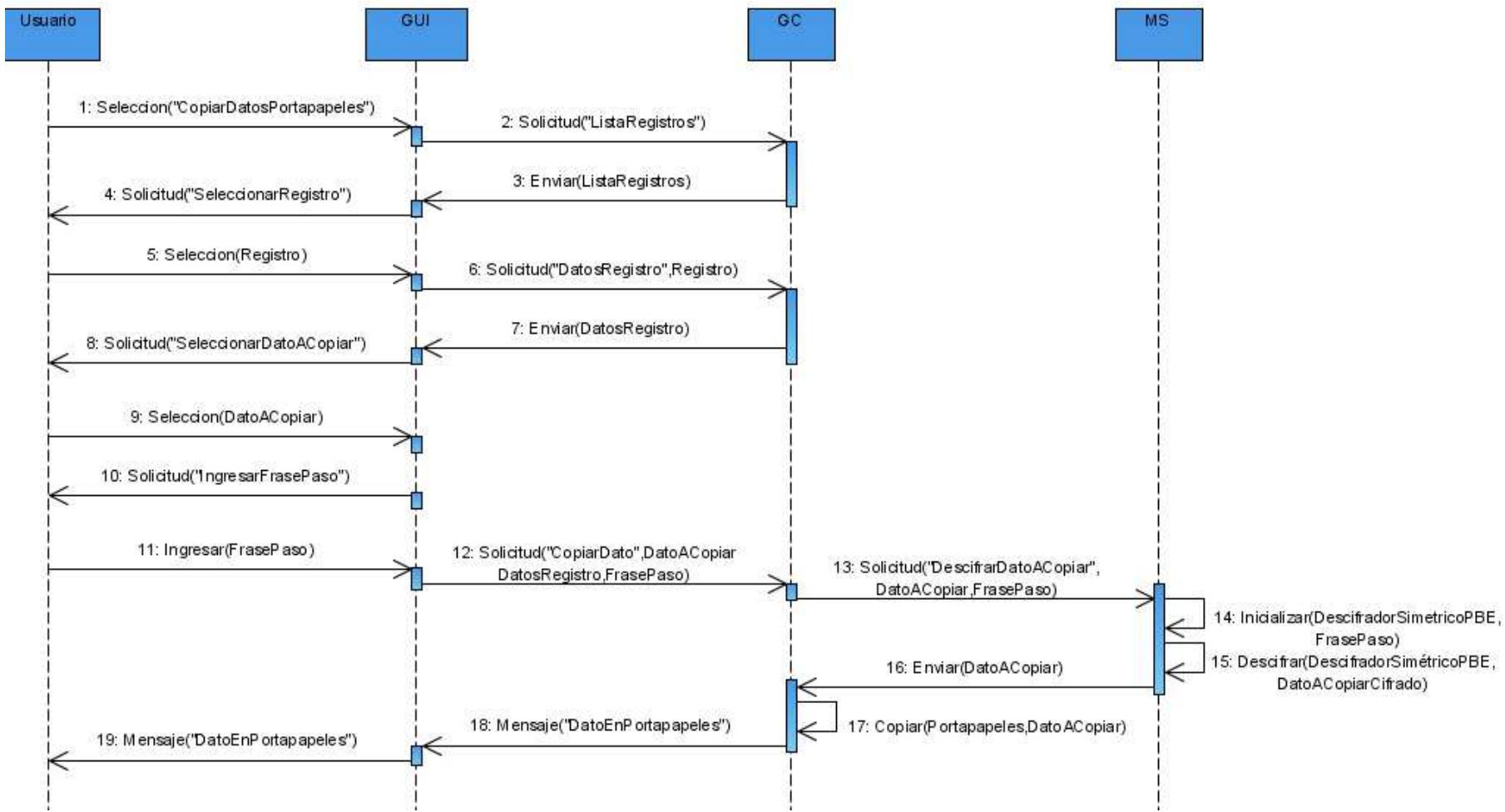


Figura 3.26: Diagrama de secuencias para: “Copiar datos al portapapeles”.

Capítulo 4 | SICFA: Implementación del Sistema

"No hay garantía de que la alta tecnología y la riqueza nos vayan a traer la felicidad. Pero traen dos importantes cosas: seguridad creciente y mayor capacidad de elección"

Kahn, Herman.

En este capítulo se presentan las herramientas, técnicas y librerías utilizadas en el proceso de desarrollo e implementación del sistema. Además, un resumen de los módulos implementados por subsistema y sus métodos más relevantes.

4.1 Herramientas y técnicas utilizadas

4.1.1 Lenguaje de programación Java

Java2, versión 1.6, es un lenguaje de programación orientado a objetos creado por Sun Microsystems Open Source, lo que significa que su uso es libre. Java es un potente lenguaje actualmente muy extendido con gran incidencia en la elaboración de aplicaciones Web y la informática en general donde su principal característica es la independencia de la plataforma donde se use.

El porque de la selección de Java como lenguaje de programación para el proyecto SICFA está relacionado con el amplio conocimiento en este lenguaje por parte de los integrantes del grupo. Además, Java proporciona las librerías de seguridad como JCA y JCE las cuales proporcionan servicios criptográficos.

4.1.2 Entorno de desarrollo NetBeans IDE 6.0

Herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java, pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el IDE NetBeans. El entorno de desarrollo integrado IDE NetBeans es un producto libre y gratuito sin restricciones de uso.

La elección de NetBeans para el desarrollo de SICFA-DTX235, a parte de las características antes mencionadas, se debe principalmente a que proporciona un ambiente grato con un gran número de funcionalidades que facilitan la labor del programador especialmente en la creación de ventanas graficas.

Por último se puede mencionar la facilidad con la que genera la documentación del código (JavaDoc), lo cual permite una mejor mantención y comprensión del código fuente.

4.1.3 Herramientas de Diagramas UML: Visual Paradigm.

Visual Paradigm for UML es una herramienta de software que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientado a objetos, construcción, pruebas y despliegue. El software de modelado UML, ayuda a una rápida construcción de aplicaciones de calidad y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación [WEB-12].

La elección de esta herramienta se debe a que los integrantes del equipo tenían conocimientos previos del uso de esta herramienta. Se utilizó la versión gratuita de Visual Paradigm for UML (Community Edition).

4.1.4 Base64

Base64 es un sistema de numeración posicional que usa 64 como base. Es la mayor potencia de 2 que puede ser representada usando únicamente los caracteres imprimibles de ASCII. Esto ha propiciado su uso para codificación de correos electrónicos, PGP y otras aplicaciones. Una aplicación de Base64 debe utilizar el rango de caracteres A-Z, a-z y 0-9 en este orden para los primeros 62 dígitos, pero los símbolos escogidos para los últimos 2 dígitos varían considerablemente de una implementación a otra.

4.2 Librerías utilizadas

4.2.1 JCA (Java Cryptography Architecture)

JCA es un marco de trabajo (framework) que provee servicios criptográficos bajo el paquete de Java Security y que forma parte del JDK desde la versión 1.2. Es una plataforma que define las clases que funcionan como interfaces para definir operaciones estándar y que contienen una arquitectura de proveedores donde son éstos los que implementan los diferentes algoritmos de seguridad entre los cuales están las funciones resumen (message digests), firmas digitales (digital signatures) y certificados digitales, por nombrar algunos. JCA fue diseñada bajo los siguientes 3 principios:

Independencia de implementación

Anteriormente se mencionó que JCA contiene una arquitectura de proveedores lo que permite que una misma funcionalidad pueda ser implementada de distintas maneras y por diferentes proveedores, haciendo que los desarrolladores usen los servicios sin la necesidad de conocer los detalles de la implementación de los algoritmos.

Interoperabilidad de implementación

Este principio se basa en que varias implementaciones trabajen con otras implementaciones. Por ejemplo, si dos proveedores implementan el mismo algoritmo criptográfico de cifrado simétrico, entonces lo que se cifre con uno puede ser descifrado por el otro.

Extensibilidad de algoritmos

JCA contiene una serie de algoritmos de seguridad incorporada en su versión estándar. Hay veces que desarrolladores, necesitan utilizar en sus aplicaciones algoritmos de seguridad que no estén implementados en esta versión. Este problema es resuelto debido a que Java soporta la incorporación de nuevos proveedores.

4.2.2 JCE (Java Cryptography Extension)

Debido a restricciones de exportación que impuso el gobierno de los E.E.U.U al software que incorporaba librerías criptográficas, SUN se vio en la obligación de separar en dos su arquitectura criptográfica. Por una parte estaba JCA, la cual incorporaba los servicios sin restricción y de la cual se habló anteriormente, y por otra JCE que incorpora los algoritmos de cifrado y descifrado y sobre los cuales estaban afectos las restricciones;

es por esta razón que SUN sólo exportaba en su versión estándar JCA en su totalidad (clases y proveedores) y JCE (sólo las clases). A partir de la versión 1.4 de Java, SUN pudo exportar JCE (clases y proveedores) como parte de la máquina virtual debido al cambio en las leyes de exportación. El JDK incluye los siguientes dos componentes de software:

1. El framework que define y soporta servicios criptográficos que son implementados por proveedores. Este framework, incluye paquetes como `java.security`, `javax.crypto`, `javax.crypto.spec` y `javax.crypto.interfaces`.
2. Los actuales proveedores que se incluyen son: Sun, SunRsaSign, SunJCE; los cuales contienen las implementaciones criptográficas actuales.

Servicios criptográficos de JCA (Tabla 4.1)

La arquitectura JCA provee los siguientes servicios:

Clase JCA 1.6	Descripción
<code>Java.security.SecureRandom</code>	Se utiliza para generar números aleatorios o pseudoaleatorios.
<code>Java.security.MessageDigest</code>	Cálculo de resumen de mensajes (hash).
<code>Java.security.Signature</code>	Firmado de datos y verificación de firmas.
<code>Java.crypto.Cipher</code>	Cifrado y descifrado.
<code>Java.crypto.Mac</code>	Autentificación de mensajes
<code>Java.security.KeyFactory</code>	Convierte claves de formato criptográfico a especificaciones de claves y viceversa.
<code>Java.crypto.SecretKeyFactory</code>	Representa una factoría de claves secretas.
<code>Java.security.KeyPairGenerator</code>	Generar pares de claves (pública y privada) para un algoritmo.
<code>Java.crypto.KeyGenerator</code>	Proporciona un generador de claves simétricas.
<code>Java.crypto.KeyAgreement</code>	Proporciona un protocolo de intercambio de claves.
<code>Java.security.AlgorithmParameters</code>	Gestiona los parámetros de un algoritmo incluyendo codificación y decodificación.
<code>Java.security.AlgorithmParameterGenerator</code>	Genera un conjunto de parámetros para un algoritmo.
<code>Java.security.KeyStore</code>	Crear y gestionar un almacén de claves.
<code>Java.security.certificate.CertificateFactory</code>	Crear certificados de clave pública y listas de revocación (CRLs).
<code>Java.security.certificate.CertPathBuilder</code>	Usado para construir cadenas de certificados (también conocida como ruta de certificación).
<code>Java.security.certificate.CertPathValidator</code>	Usado para validar cadenas de certificados.
<code>Java.security.certificate.CertStore</code>	Usado para obtener certificados y CRLs desde un repositorio.

Tabla 4.1: Descripción de las clases de JCA 1.6.

4.2.3 Compresión en Java: Paquete java.util.zip

Hay muchos beneficios con la compresión de los datos. No obstante, la ventaja principal es reducir requisitos de almacenamiento. También, para comunicaciones de los datos, la transferencia de datos comprimidos aumenta la cantidad de información transmitida.

Java proporciona el paquete `java.util.zip` para trabajar con archivos compatibles con el formato zip. Este paquete proporciona clases que permiten leer, crear, y modificar archivos con los formatos zip y gzip. También proporciona clases para comprobar las sumas de control de flujos de entrada y puede ser usado para validar entradas de datos; este paquete proporciona una interface, 14 clases, y 2 clases de excepciones como se muestra en la tabla 4.2.

Elemento	Tipo	Descripción
Checksum	Interface	Representa un dato de la suma de control (data checksum). Implementado por las clases Adler32 y CRC32.
Adler32	Clase	Usado para calcular la suma de control (checksum) de un flujo de datos.
CheckInputStream	Clase	Un flujo de entrada que guarda la suma de control de los datos que están siendo leídos.
CheckOutputStream	Clase	Un flujo de salida que guarda la suma de control de los datos que están siendo escritos.
CRC32	Clase	Usado para calcular la suma de control CRC32 de un flujo de datos.
Deflater	Clase	Soporte para compresión usando la librería ZLIB.
DeflaterOutputStream	Clase	Un flujo de salida filtrado para la compresión de datos con el formato de compresión deflate.
GZIPInputStream	Clase	Un flujo de salida filtrado para la compresión de datos con el formato GZIP.
GZIPOutputStream	Clase	Un flujo de salida filtrado para escribir datos comprimidos con el formato GZIP.
Inflater	Clase	Soporte para descompresión usando librería ZLIB.
InflaterInputStream	Clase	Un flujo de entrada filtrado para descompresión de datos con el formato de compresión deflate.
ZipEntry	Clase	Representa la entrada de un archivo ZIP.
ZipFile	Clase	Usado para leer entradas de un archivo ZIP.
ZipInputStream	Clase	Un filtro de entrada para leer archivos con el formato ZIP.
ZipOutputStream	Clase	Un filtro de salida para escribir archivos con el formato ZIP.
DataFormatException	Clase	Lanza una excepción para señalar un error en el formato de los datos.
ZipException	Clase	Lanza una excepción para señalar un error en el formato ZIP.

Tabla 4.2: Paquete `java.util.zip`.

4.3 Módulos Principales.

4.3.1 ADMINISTRADOR DE CLAVES (AC)

Este subsistema permite: gestionar claves públicas y privadas, propias; gestionar claves públicas de otros usuarios; gestionar identidades de usuario, que asocian una clave pública con una persona real; ingreso de datos para la ejecución de operaciones de cifrado/descifrado de archivo (cifrado híbrido); ingreso de datos para la generación/verificación de firma digital de archivo; ingreso de datos para la generación/verificación de firma digital de claves públicas subordinadas propias e identidades de usuario propias.

En los procedimientos de gestión de las claves se tienen en cuenta las siguientes actividades: generación de las claves, distribución de las claves, activación y utilización de las claves, almacenamiento y recuperación de las claves, destrucción y revocación de las claves.

Algunas de las clases, organizadas por funcionalidad, asociadas con este subsistema son presentadas a continuación.

Gestión de paquetes en memoria

- ❖ **Packet:** Es la clase padre de las clases representativas de los paquetes utilizados por el sistema. Estas clases son: ClavePublica, ClavePrivada, Revocación, ClaveSubordinada, Usuariold, Confianza y Firma. La función de esta clase es la de asignar un identificador a cada uno de los paquetes y proveer de funcionalidades en común a cada uno de ellos. Algunos de sus métodos relevantes son:
 - **Public Packet():** Constructor que permite asignar un identificador a un paquete.
 - **public void asignaTag(int opción):** Método que asigna un Tag a un paquete para su posterior identificación y distinción con otros. Ingresa por parámetro un identificador del Tag que se le asignará al paquete.
 - **public byte[] huellaid(byte[] resumenMsj):** Método que obtiene una porción de una huella digital, lo que equivale a un identificador único. Ingresa por parámetro un arreglo binario que contiene un valor hash de 20 bytes.
- ❖ **ClavePublica:** Esta clase permite la creación de un paquete SICFA que contiene en su interior la clave pública de una pareja de claves principal. Esta clase se extiende de Packet. Algunos de sus métodos relevantes son:
 - **public ClavePublica(int tipoPacket):** Constructor de la clase ClavePublica. Ingresa por parámetro un identificador del tipo de paquete, para asignarle un Tag.
 - **public boolean getRevocada():** Método que indica si la clave pública principal y su clave privada complementaria han sido revocadas.
 - **public byte[] getClvPub():** Método que obtiene un arreglo binario con la clave pública de la pareja de claves principal.
 - **private void asignald():** Método que asigna un identificador único al paquete de clave pública principal, solicitando un valor hash a un objeto de la clase MessageDigestSicfa perteneciente al subsistema MS.

- `public void escribeArchivo(DataOutputStream dio)`: Método que escribe datos del paquete de clave pública principal en el anillo de claves públicas. Ingresa por parámetro un flujo de salida de datos.
 - `public void leeArchivo(DataInputStream dis)`: Método que recupera datos del paquete de clave pública principal que se encuentra en el anillo de claves públicas. Ingresa por parámetro un flujo de entrada de datos.
- ❖ **ClavePrivada:** Esta clase permite la creación de un paquete SICFA que contiene en su interior una clave privada de una pareja de claves generada (principal o subordinada). Esta clase se extiende de Packet. Algunos de sus métodos relevantes son:
- `ClavePrivada(int tipoPacket)`: Constructor de la clase ClavePrivada. Ingresa por parámetro un identificador del tipo de paquete, para asignarle un Tag.
 - `public int getAlgoritmo()`: Método que obtiene el identificador del algoritmo asociado con la clave privada.
 - `public byte[] getClvPrivCfrd()`: Método que obtiene un arreglo binario con la clave privada cifrada, de una pareja de claves, contenida en el paquete.
 - `public void cifraClvPriv(String frasePaso)`: Método que permite cifrar la clave privada contenida en el paquete. Ingresa por parámetro una frase de paso que permite obtener una clave binaria para inicializar un cifrador simétrico PBE con el cual se cifra la clave privada.
 - `public byte[] descifraClvPriv(String frasePaso)`: Método que permite descifrar la clave privada contenida en el paquete, utilizando un algoritmo simétrico PBE y una clave binaria obtenida a partir de una frase de paso conocida por el usuario
 - `public PrivateKey recuperaClvPriv(byte[] clvPri)`: Método que permite recuperar la clave privada a partir de su codificación (se guarda codificada en un arreglo binario para poder transportarla y ocultar parte de su información). Ingresa por parámetro un arreglo binario que contiene la clave privada codificada.
 - `public void escribeArchivo(DataOutputStream dio)`: Método que escribe datos del paquete de clave privada en el anillo de claves privadas. Ingresa por parámetro un flujo de salida de datos.
 - `public void leeArchivo(DataInputStream dis)`: Recupera datos del paquete de clave privada que se encuentra en el anillo de claves privada. Ingresa por parámetro un flujo de entrada de datos.
- ❖ **ClaveSubordinada:** Esta clase permite la creación de un paquete SICFA que contiene en su interior la clave pública de una pareja de claves, esta pareja es subordinada de otra pareja de claves principal. Esta clase se extiende de Packet. Algunos de sus métodos relevantes son:
- `public ClaveSubordinada(int tipoPacket)`: Constructor de la clase ClaveSubordinada. Ingresa por parámetro un identificador del tipo de paquete, para asignarle un Tag.
 - `public void setIdUsuario(byte[] idUsua)`: Método que permite cambiar el valor del identificador único de identidad de usuario asociado con esta clave pública subordinada. Ingresa por parámetro un arreglo binario que contiene un nuevo identificador único de identidad de usuario.

- public boolean getRevocada(): Método que indica si la pareja de claves subordinadas está revocada.
 - public boolean getCaducada(): Método que indica si la pareja de claves subordinadas ha caducado.
 - public byte[] getIdProp(): Método que permite obtener el identificador único de la pareja de claves principal propietaria de esta clave.
 - protected void setRevocada(Listas listas): Método que permite revocar la clave pública subordinada dentro del paquete. También se revoca su clave privada subordinada complementaria. Ingresa por parámetro la estructura de datos que gestiona los paquetes en memoria.
 - private void asignald(): Método que asigna un identificador único al paquete de clave pública subordinada, solicitando un valor hash a un objeto de la clase MessageDigestSicfa perteneciente al subsistema MS.
 - public void escribeArchivo(DataOutputStream dio): Método que escribe datos del paquete de clave pública subordinada en el anillo de claves públicas. Ingresa por parámetro un flujo de salida de datos.
 - public void leeArchivo(DataInputStream dis): Método que recupera datos del paquete de clave pública subordinada que se encuentra en el anillo de claves pública. Ingresa por parámetro un flujo de entrada de datos.
- ❖ Usuariold: Esta clase permite la creación de un paquete SICFA que contiene en su interior los datos de una identidad de usuario, el cual puede estar asociado a cero o varias parejas de claves. Esta clase se extiende de Packet. Algunos de sus métodos relevantes son:
- public Usuariold(int tipoPacket): Constructor de la clase Usuariold. Ingresa por parámetro un identificador del tipo de paquete, para asignarle un Tag.
 - public byte[] getIdProp(): Método que permite obtener el identificador único de la pareja de claves principal propietaria de esta identidad de usuario.
 - private void asignald(): Método que asigna un identificador único al paquete de identidad de usuario, solicitando un valor hash a un objeto de la clase MessageDigestSicfa perteneciente al subsistema MS.
 - public void asignaPrimario(int opcion): Método que permite asignar un valor para distinguir si un usuario corresponde al usuario primario, distinguiéndolo de sus alias. Ingresa por parámetro un identificador que indica si el usuario es primario.
 - public boolean esPrimario(): Método que indica si un usuario es primario o no.
 - public void escribeArchivo(DataOutputStream dio): Método que escribe datos del paquete de identidad de usuario en el anillo de claves públicas. Ingresa por parámetro un flujo de salida de datos.
 - public void leeArchivo(DataInputStream dis): Método que recupera datos del paquete de identidad de usuario que se encuentra en el anillo de claves pública. Ingresa por parámetro un flujo de entrada de datos.
- ❖ Confianza: Esta clase permite la creación de un paquete SICFA, que contiene en su interior la confianza que el usuario tiene en una pareja de claves principal propia o ajena. Esta clase se extiende de Packet. Algunos de sus métodos relevantes son:

- public Confianza(int tipoPacket): Constructor de la clase Confianza. Ingresa por parámetro un identificador del tipo de paquete, para asignarle un Tag.
 - public byte[] getId(): Método que permite obtener el identificador único del paquete de confianza.
 - public byte[] getIdProp(): Método que permite obtener el identificador único de la pareja de claves principal propietaria de esta confianza.
 - public void asignaConfianza(int opcion): Método que permite asignar un indicador del nivel de confianza que tendrá una pareja de claves en particular, esta confianza la asigna el usuario. Ingresa por parámetro un indicador del nivel confianza (nula, marginal, parcial, total).
 - public void escribeArchivo(DataOutputStream dio): Método que escribe datos del paquete de confianza en el anillo de confianza. Ingresa por parámetro un flujo de salida de datos.
 - public void leeArchivo(DataInputStream dis): Método que recupera datos del paquete de confianza que se encuentra en el anillo de confianza. Ingresa por parámetro un flujo de entrada de datos.
- ❖ Revocación: Esta clase permite la creación de un paquete SICFA que contiene en su interior los datos de revocación de la pareja de claves principal. Esta clase se extiende de Packet. Algunos de sus métodos relevantes son:
- public Revocacion(int tipoPacket): Constructor de la clase Revocacion. Ingresa por parámetro un identificador del tipo de paquete, para asignarle un Tag.
 - public byte[] getId(): Método que permite obtener el identificador único del paquete de revocación.
 - public byte[] getIdProp(): Método que permite obtener el identificador único de la pareja de claves principal propietaria de esta revocación.
 - private void asignarMotivo(int opcion): Método que permite asignar el motivo por el cual se ha producido la revocación de la pareja de claves principal. Ingresa por parámetro un identificador de revocación (ninguno, clave robada, clave retirada temporalmente y clave retirada definitivamente).
 - public void escribeArchivo(DataOutputStream dio): Método que escribe datos del paquete de revocación en el anillo de claves públicas. Ingresa por parámetro un flujo de salida de datos.
 - public void leeArchivo(DataInputStream dis): Método que recupera datos del paquete de revocación que se encuentra en el anillo de claves públicas. Ingresa por parámetro un flujo de entrada de datos.
- ❖ Firma: Esta clase permite la creación de un paquete SICFA que contiene en su interior la firma digital de los datos de un paquete en particular (identidad de usuario, revocación, clave pública subordinada). Esta clase se extiende de Packet. Algunos de sus métodos relevantes son:
- public Firma(int tipoPacket): Constructor de la clase Firma. Ingresa por parámetro un identificador del tipo de paquete, para asignarle un Tag.
 - public byte[] getIdFirmante(): Método que permite obtener el identificador único de la pareja de claves principal que generó la firma digital contenida en el paquete.

- `public byte[] getIdProp():` Método que permite obtener el identificador único del paquete firmado, propietario de este paquete de firma.
 - `private void asignaTipoFirma(int opcion):` Método que permite asignar un identificador que especifica el tipo de firma. Ingresa por parámetro un identificador de tipo de firma (firma revocación, firma identidad de usuario y firma clave subordinada).
 - `private void asignaAlgClvPub(int opcion):` Método que permite asignar un identificador del algoritmo de clave pública que se usará para firmar. Ingresa por parámetro un identificador de algoritmo de clave pública (DSA, RSA).
 - `public void generaFirma(byte[] datosConcat, byte[] clvPriv, int algorit):` Método que permite generar una firma digital para un paquete en particular utilizando un objeto de la clase SignatureSicfa del subsistema MS. Ingresa por parámetro un arreglo con los datos que se van a firmar; la clave privada principal que se utilizará para firmar y el algoritmo asociado con la clave privada con el que se generará la firma.
 - `public boolean verificaFirma(byte[] datosConcat, byte[] clvPub, int algorit):` Método que indica si una firma digital asociada a un paquete en particular es válida o no. Ingresa por parámetro un arreglo binario con los datos del paquete a firmar y datos de firma digital; clave pública principal para verificar firma y el identificador del algoritmo utilizado para generar la firma.
 - `public void escribeArchivo(DataOutputStream dio):` Método que escribe datos del paquete de firma en el anillo de claves públicas. Ingresa por parámetro un flujo de salida de datos.
 - `public void leeArchivo(DataInputStream dis):` Método que recupera datos del paquete de firma que se encuentra en el anillo de claves públicas. Ingresa por parámetro un flujo de entrada de datos.
- ❖ Listas: Esta clase permite la creación de un conjunto de listas, cada una de las cuales contendrá un conjunto de paquetes de un tipo en particular. Algunos de sus métodos relevantes son:
- `public void addClvPub(Object clvPublica):` Método que permite añadir un paquete de clave pública a la lista correspondiente. Ingresa por parámetro un objeto de ClavePublica.
 - `public void addRevYFir(Object revacion, Object firma):` Método que permite añadir un paquete de revocación y su paquete de firma digital a la lista correspondiente. Ingresa por parámetro un objeto de Revacion y un objeto de Firma.
 - `public void addUidYFir(Object usuario, Object firma):` Método que permite añadir un paquete de identidad de usuario y su paquete de firma digital a la lista correspondiente. Ingresa por parámetro un objeto de Usuariold y un objeto de Firma.
 - `public void addClvSubYFir(Object clvSubordinada, Object firma):` Método que permite añadir un paquete de clave pública subordinada y su paquete de firma digital a la lista correspondiente. Ingresa por parámetro un objeto de ClaveSubordinada y un objeto de Firma.
 - `public void addClvPriv(Object clvPrivada):` Método que permite añadir un paquete de clave privada a la lista correspondiente. Ingresa por parámetro un objeto de ClavePrivada.

- `public void addConfianza(Object confianza)`: Método que permite añadir un paquete de confianza a la lista correspondiente. Ingresa por parámetro un objeto de Confianza.
- `public void leeArchivoClvPub()`: Método que permite recuperar paquetes públicos desde un archivo digital (anillo de claves públicas) y almacena cada uno de ellos en su correspondiente lista.
- `public void leeArchivoConfianza()`: Método que permite recuperar paquetes de confianza desde un archivo digital (anillo de confianzas) y almacena cada uno de ellos en su correspondiente lista.
- `public void leeArchivoClvPriv()`: Método que permite recuperar paquetes de clave privada desde un archivo digital (anillo de claves privadas) y almacena cada uno de ellos en su correspondiente lista.
- `public boolean eliminaClvPub(byte[] id)`: Método que permite buscar una clave pública en la lista de claves públicas y la elimina si la encuentra. Ingresa por parámetro un arreglo binario que contiene el identificador único del paquete de clave pública.
- `public boolean eliminaRevocYFir(byte[] id)`: Método que permite eliminar una revocación y su firma digital de sus listas correspondientes. Ingresa por parámetro un arreglo binario que contiene el identificador único del paquete de revocación.
- `public boolean eliminaUidYFir(byte[] id)`: Método que permite eliminar una identidad de usuario y su firma digital de sus listas correspondientes. Ingresa por parámetro un arreglo binario que contiene el identificador único del paquete de identidad de usuario.
- `public boolean eliminaClvSubYFir(byte[] id)`: Método que permite eliminar una clave pública subordinada y su firma digital de sus listas correspondientes. Si la clave pública subordinada es propia, entonces se elimina de igual forma su clave privada complementaria. Ingresa por parámetro un arreglo binario que contiene el identificador único del paquete de clave subordinada.
- `public boolean eliminaConfianza(byte[] idProp)`: Método que permite eliminar una confianza de la lista de confianzas. Ingresa por parámetro un arreglo binario que contiene el identificador único del paquete de confianza.
- `public boolean eliminaClvPriv(byte[] id)`: Método que permite eliminar una clave privada de la lista de claves privadas. Ingresa por parámetro un arreglo binario que contiene el identificador único del paquete de clave privada.
- `public void escribeArchClvPub()`: Método que permite escribir los datos de los paquetes de clave pública principal propio y ajenos en un archivo digital (anillo de claves públicas). Además de los datos de sus paquetes asociados.
- `public void escribeArchConfianza()`: Método que permite escribir los datos de los paquetes de confianza, existentes en la lista de confianzas, en un archivo digital (anillo de confianzas).
- `public void escribeArchClvPriv()`: Método que permite escribir los datos de los paquetes de clave privada, existentes en la lista de claves privadas, en un archivo digital (anillo de claves privadas).
- `public void exportaClvPub(byte[] id, boolean alFinal)`: Método que permite exportar a un archivo digital datos de paquetes de clave pública principal y datos de sus correspondientes paquetes asociados, para compartirlos con otros usuarios.

Ingrasa por parámetro un arreglo binario que contiene el identificador único del paquete de clave pública a exportar y un valor que indica si se debe sobrescribir el archivo de salida o no.

- `public void importaClvPub()`: Método que permite importar desde un archivo digital datos de paquetes de clave pública principal y datos de sus correspondientes paquetes asociados, compartidos por otros usuarios.
- `public FileNode creaTree()`: Este método permite crear una estructura de árbol que contiene las claves públicas principales propia y ajenas. Además de sus respectivas claves subordinadas, si existen, con sus firmas. Cada uno de estos nodos generados llevan asociados datos de otros paquetes (paquetes de usuario y paquetes de confianza).

Iniciar sistema

- ❖ `Sicfa_GUI`: Esta clase implementa la interfaz gráfica necesaria para comenzar a utilizar el sistema, en la cual el usuario dispondrá de dos opciones. (Ver Figura 4.1).

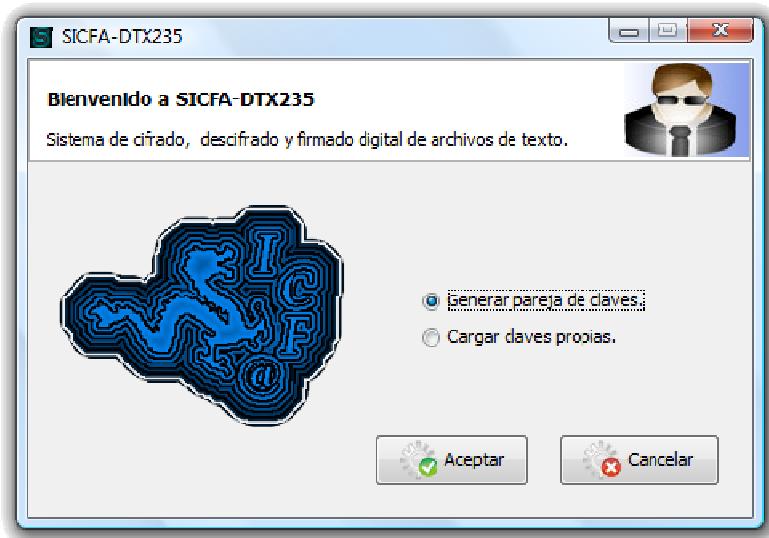


Figura 4.1: Interfaz gráfica para iniciar sistema.

- ❖ `ClaveInicial_GUI`: Esta clase implementa la interfaz gráfica para el ingreso de los datos necesarios para la generación de una nueva pareja de claves principal. Se requiere el ingreso de datos de usuario, datos de claves y datos de clave de sesión. (Ver Figura 4.2). Esta clase se extiende de `javax.swing.JFrame`.
- ❖ `FrasePaso_GUI`: Esta clase implementa la interfaz gráfica para el ingreso de los datos de una frase de paso. La cual es muy importante debido a que con ella se protege el acceso a la clave privada de una pareja de claves propia. Además, esta clase permite la creación de los paquetes necesarios para comenzar a utilizar la aplicación. Esta clase se extiende de `javax.swing.JDialog`. Algunos de sus métodos relevantes son:
 - `public void creaClaves()`: Este método permite: la creación de las estructuras de datos (listas), necesarias para la gestión de paquetes en memoria, y la creación de los paquetes relacionados con la pareja de claves principal que está siendo creada. Estos paquetes son: clave pública principal, identidad de usuario, firma digital de identidad de usuario, clave privada principal y confianza.

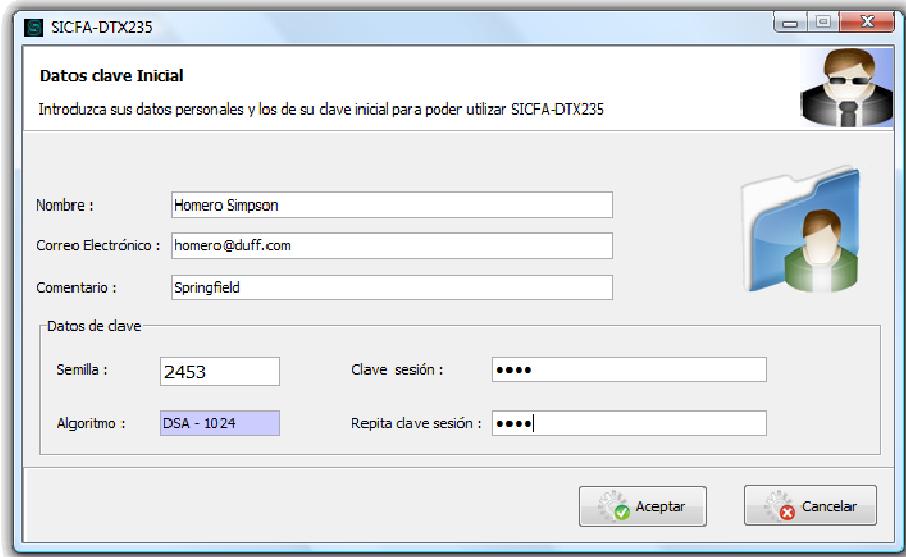


Figura 4.2: Interfaz gráfica para ingreso de datos de pareja de claves principal.

- ❖ **ClaveSesión_GUI:** Esta clase implementa la interfaz gráfica para el ingreso de la clave de sesión, la cual es ingresada por el usuario al momento de crear su pareja de claves principal. Si esta clave de sesión es válida, entonces se despliega la interfaz gráfica implementada por la clase **Base_GUI**. Esta clase se extiende de *javax.swing.JFrame*. Algunos de sus métodos relevantes son:

- **private void inicializaciones():** Este método permite cargar las estructuras de datos (listas) con sus paquetes correspondientes. Estos paquetes se encuentran en los archivos digitales (anillos) que requiere el sistema.

Ventana principal

- ❖ **Base_GUI:** Esta clase implementa la interfaz gráfica principal. Aquí el usuario puede ver el detalle de sus parejas de claves propias y el detalle de las claves públicas compartidas por otros usuarios; así como los datos de otros paquetes asociados con ellas. Esta interfaz presenta al usuario un menú, en el cual se encuentran la totalidad de las funcionalidades que brinda el programa: administración de claves propias y ajenas, administración de usuarios propios, cifrado y descifrado de archivos, firmado digital de archivos, opciones de configuración del sistema. (Ver Figura 4.3). Esta clase se extiende de *javax.swing.JFrame*. Algunos de sus métodos relevantes son:

- **private void cargaArbol():** Este método permite cargar la estructura de árbol que contiene las claves públicas principales propia y ajenas. Además de sus respectivas claves subordinadas, si existen, con sus firmas. Esta estructura es presentada por pantalla al usuario gracias a la implementación de un *TreeTable*.
- **private FileNode arbol():** Este método permite obtener la estructura de árbol que contiene las claves públicas principales propia y ajenas. Además de sus respectivas claves subordinadas, si existen, con sus firmas. Cada uno de estos nodos generados llevan asociados datos de otros paquetes (paquetes de usuario y de confianza).

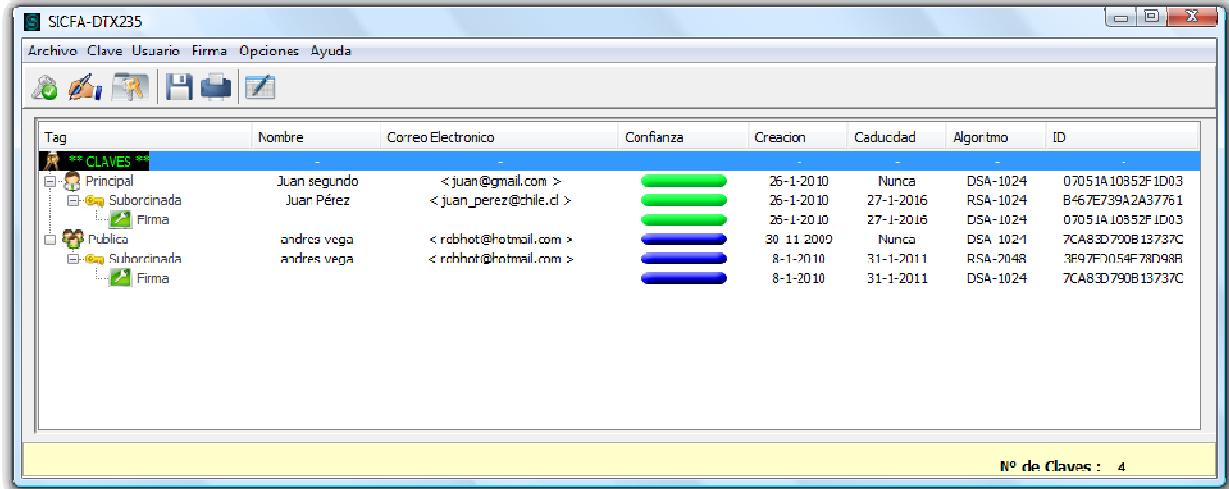


Figura 4.3: Interfaz gráfica del Administrador de claves.

Administración de las claves

- ❖ GenerarParClaves_GUI: Esta clase implementa la interfaz gráfica para el ingreso de los datos necesarios para la generación de una pareja de claves subordinadas. Se requiere el ingreso de datos de claves y, selección de un identificador único de identidad de usuario. Además, instancia un objeto de la clase Frase_GUI para el ingreso de la frase de paso necesaria para liberar la clave privada principal y así firmar digitalmente los datos de la clave pública subordinada. (Ver Figura 4.4). Esta clase se extiende de *javax.swing.JDialog*. Algunos de sus métodos relevantes son:
 - public void creaSubord(): Método que permite la creación de los paquetes relacionados con la pareja de claves subordinadas que está siendo creada. Estos paquetes son: clave pública subordinada, firma digital de clave pública subordinada y clave privada subordinada.

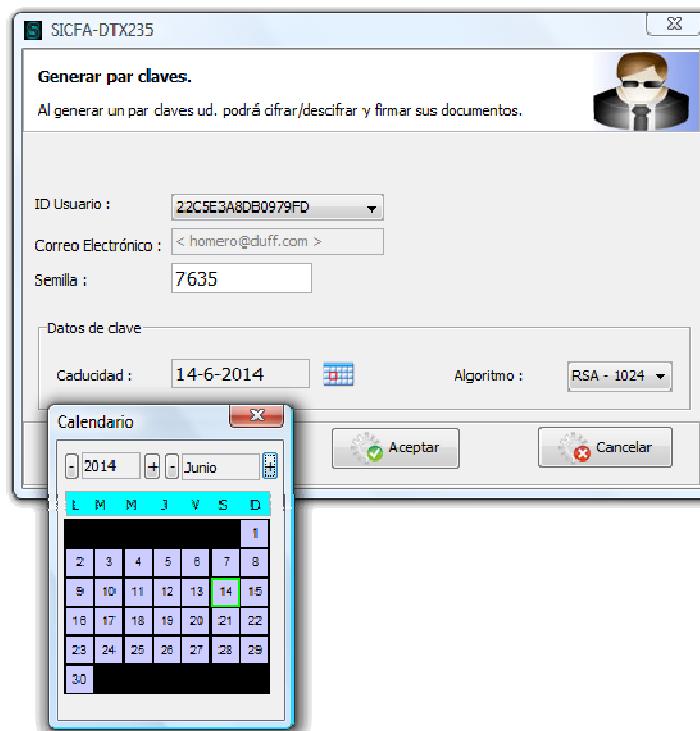


Figura 4.4: Interfaz gráfica para el ingreso de datos de pareja de claves subordinadas.

- ❖ NuevoUsuario_GUI: Esta clase implementa la interfaz gráfica que permite el ingreso de los datos para la creación de una nueva identidad de usuario. Además, instancia un objeto de la clase Frase_GUI para el ingreso de la frase de paso necesaria para liberar la clave privada principal y así firmar digitalmente los datos de identidad de usuario. Esta clase se extiende de *javax.swing.JDialog*. Algunos de sus métodos relevantes son:
 - `private void creaUsuario()`: Método que permite la creación de un paquete de identidad de usuario y su paquete de firma digital asociado.
- ❖ RevocacionClvPrincipal_GUI: Esta clase implementa la interfaz gráfica que permite el ingreso de los datos para la revocación de una pareja de claves principal y sus parejas de claves subordinadas (si existen) asociadas. Además, instancia un objeto de la clase Frase_GUI para el ingreso de la frase de paso necesaria para liberar la clave privada principal y así firmar digitalmente los datos de revocación. Esta clase se extiende de *javax.swing.JDialog*. Algunos de sus métodos relevantes son:
 - `private void creaRevocacion()`: Método que permite la creación del paquete de revocación y su paquete de firma digital asociado.
- ❖ EditorClave_GUI: Esta clase implementa la interfaz gráfica que permite la modificación de los datos (identificador único de identidad de usuario y fecha de caducación) de una pareja de claves subordinadas (propia). Esta clase se extiende de *javax.swing.JDialog*. Algunos de sus métodos relevantes son:
 - `private void generaDatosArreglo()`: Método que permite recuperar los datos asociados a cada pareja de claves subordinadas existentes en la estructura, para su posterior presentación en pantalla.
 - `private void guardarCambios()`: Método que instancia un objeto de la clase Clave_GUI para el ingreso de la clave de sesión necesaria para validar la operación. Si la clave de sesión es correcta, se actualizan los paquetes en la estructura.
- ❖ EditorUsuario_GUI: Esta clase implementa la interfaz gráfica que permite cambiar el estado de una identidad de usuario de secundaria a primaria. Esta clase se extiende de *javax.swing.JDialog*. Algunos de sus métodos relevantes son:
 - `private void generaDatosArreglo()`: Método que permite recuperar los datos asociados a las identidades de usuarios propios existentes en la estructura, para su posterior presentación en pantalla.
 - `private void guardarCambios()`: Método que instancia un objeto de la clase Clave_GUI para el ingreso de la clave de sesión necesaria para validar la operación. Si la clave de sesión es correcta, se actualizan los paquetes en la estructura.
- ❖ EdicionClave_GUI: Esta clase implementa la interfaz gráfica que permite modificar los datos (nivel de confianza y la foto que identifica al propietario) de una pareja de claves principal (propia o ajena). (Ver Figura 4.5). Esta clase se extiende de *javax.swing.JDialog*. Algunos de sus métodos relevantes son:
 - `private void generaDatosClave(String cadenaId)`: Método que permite mostrar en pantalla los datos asociados a una pareja de claves principal propia o ajena. Ingresa por parámetro la cadena de texto con el identificador único de la clave pública principal.

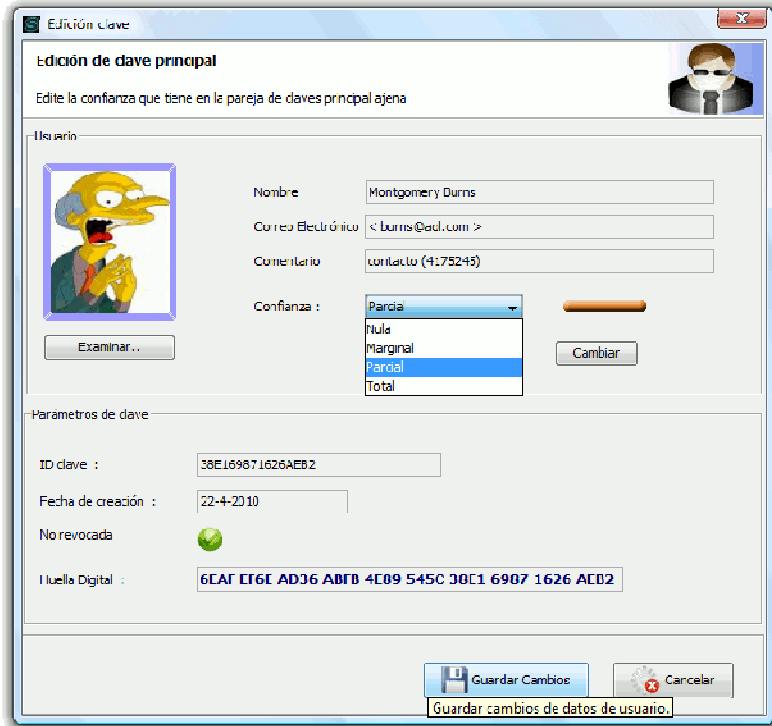


Figura 4.5: Interfaz gráfica para la edición de datos de una pareja de claves principal.

- ❖ **TablaClvsSubord_GUI:** Esta clase implementa la interfaz gráfica necesaria para mostrar una serie de claves públicas subordinadas, disponibles en la estructura de datos que contiene las claves públicas. Una de estas claves presentadas, es seleccionada por el usuario para realizar alguna operación de eliminación de la clave. Esta clase se extiende de *javax.swing.JDialog*. Algunos de sus métodos relevantes son:
 - `private void generaDatosTabla():` Método que permite recuperar los datos de cada clave pública subordinada, existente en la estructura, y sus datos asociados (nombre de usuario, confianza, identificador único de la clave, revocación), para su posterior presentación en pantalla a través de una tabla.
 - `private void eliminarClvSub():` Método que instancia un objeto de la clase Clave_GUI para el ingreso de la clave de sesión necesaria para validar la operación. Si la clave de sesión es correcta, se elimina la clave de la estructura.
- ❖ **TablaUsuarios_GUI:** Esta clase implementa la interfaz gráfica necesaria para mostrar una serie de identidades de usuario, disponibles en la estructura de datos que contiene las identidades de usuario. Una de estas identidades presentadas, es seleccionada por el usuario para realizar alguna operación de eliminación de la identidad de usuario. Esta clase se extiende de *javax.swing.JDialog*. Algunos de sus métodos relevantes son:
 - `private void generaDatosTabla():` Método que permite recuperar los datos de cada identidad de usuario, existente en la estructura, y sus datos asociados (nombre, correo electrónico, fecha de creación, identificador único de usuario e indicador primario), para su posterior presentación en pantalla a través de una tabla.
 - `private void eliminarUsuario():` Método que instancia un objeto de la clase Clave_GUI para el ingreso de la clave de sesión necesaria para validar la operación. Si la clave de sesión es correcta, se elimina la identidad de usuario de la estructura.

- ❖ TablaClvsPrincipal_GUI: Esta clase implementa la interfaz gráfica necesaria para mostrar una serie de claves públicas principales ajenas, disponibles en la estructura de datos que contiene las claves públicas principales. Una de estas claves presentadas, es seleccionada por el usuario para realizar alguna operación de eliminación de la clave. Esta clase se extiende de *javax.swing.JDialog*. Algunos de sus métodos relevantes son:
 - private void generaDatosTabla(): Método que permite recuperar los datos de cada clave pública principal ajena, existente en la estructura, y sus datos asociados (nombre de usuario, confianza, fecha de creación, identificador único de la clave, indicador de revocación), para su posterior presentación en pantalla a través de una tabla.
 - private void eliminaClvPrincipalAjena(): Método que instancia un objeto de la clase Clave_GUI para el ingreso de la clave de sesión necesaria para validar la operación. Si la clave de sesión es correcta se elimina la clave de la estructura.
- ❖ Clave_GUI: Esta clase implementa la interfaz gráfica para el ingreso de la clave de sesión, la cual es creada por el usuario al momento de crear su pareja de claves principal. Además, implementa los métodos necesarios para validar la clave de sesión ingresada. Esta clase se extiende de *javax.swing.JDialog*. Algunos de sus métodos relevantes son:
 - private void obtieneHashsesion(): Método que instancia un objeto de MessageDigestSicfa del subsistema MS para obtener un valor hash representativo de la clave de sesión ingresada por el usuario.
 - private boolean esValidaClaveSesion(): Método que valida la clave de sesión ingresada.

Exportación de claves públicas

- ❖ TablaClvsExpor_GUI: Esta clase implementa la interfaz gráfica necesaria para mostrar una serie de claves públicas principales, disponibles en la estructura de datos que contiene las claves públicas principales. Una de estas claves presentadas, es seleccionada por el usuario para realizar alguna operación de exportación de claves. (Ver Figura 4.6). Esta clase se extiende de *javax.swing.JDialog*. Algunos de sus métodos relevantes son:
 - private void generaDatosTablaExp(): Método que permite recuperar los datos de cada clave pública principal propia y ajena, existente en la estructura, y sus datos asociados (nombre de usuario, confianza, identificador único de la clave, revocación), para su posterior presentación en pantalla a través de una tabla.
 - private void exportaClv(): Método que permite buscar una clave pública principal en memoria y luego solicita a un objeto de Listas que todos los datos asociados con esta clave sean escritos en un archivo digital.

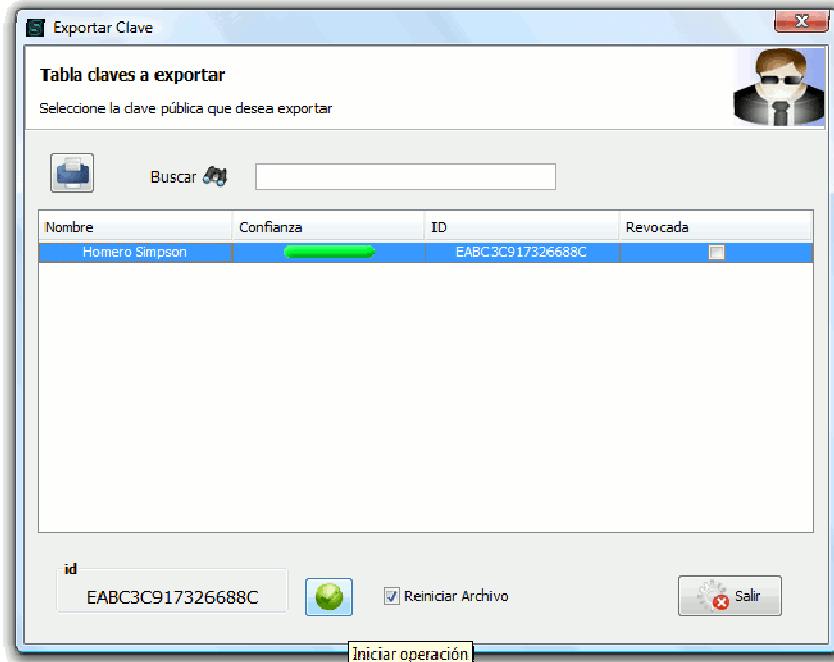


Figura 4.6: Interfaz gráfica para la exportación de claves públicas.

Cifrado de archivos

- ❖ TablaClvsCfr_GUI: Esta clase implementa la interfaz gráfica necesaria para mostrar una serie de claves públicas subordinadas (RSA, propias y ajena), disponibles en la estructura de datos que contiene las claves públicas subordinadas. Una de estas claves presentadas, es seleccionada por el usuario para realizar alguna operación de cifrado. Esta clase se extiende de *javax.swing.JDialog*. Algunos de sus métodos relevantes son:
 - private void generaDatosTablaCfr(): Método que permite recuperar los datos de cada clave pública subordinada RSA, existente en la estructura, y sus datos asociados (nombre de usuario, identificador único de la clave, identificador único del propietario, revocación, caducidad), para su posterior presentación en pantalla a través de una tabla.
 - private void cifraConClv(): Método que permite buscar una clave pública subordinada RSA en memoria y luego solicita a un objeto del subsistema MS una operación de cifrado para el archivo digital seleccionado por el usuario.

Firma de archivos

- ❖ TablaClvsFmr_GUI: Esta clase implementa la interfaz gráfica necesaria para mostrar una serie de claves públicas subordinadas (propias), disponibles en la estructura de datos que contiene las claves públicas subordinadas. Una de estas claves presentadas, es seleccionada por el usuario para realizar alguna operación de generación de firma digital. Esta clase se extiende de *javax.swing.JDialog*. Algunos de sus métodos relevantes son:
 - private void generaDatosTablaFmr(): Método que permite recuperar los datos de cada clave pública subordinada, existente en la estructura, y sus datos asociados (nombre de usuario, identificador único de la clave, revocación, caducidad), para su posterior presentación en pantalla a través de una tabla.

- public void firmaConclv(String strId): Método que permite buscar una clave pública subordinada en memoria y luego solicita a un objeto del subsistema MS una operación de generación de firma digital para el archivo digital seleccionado por el usuario. Ingresá por parámetro una cadena de texto con el identificador único de la clave subordinada.

Validación de las claves e identidades de usuario

❖ VerificadorFirma_GUI: Esta clase implementa la interfaz gráfica necesaria para mostrar una serie de claves públicas subordinadas e identidades de usuario, disponibles en la estructura. Una de estas claves o identidades de usuario, presentadas, es seleccionada por el usuario para verificar si una firma digital generada a partir de sus datos es válida. (Ver Figura 4.7). Esta clase se extiende de *javax.swing.JDialog*. Algunos de sus métodos relevantes son:

- private void generaDatosTablaClaves(): Método que permite recuperar los datos de cada clave pública subordinada, existente en la estructura, y sus datos asociados (nombre de usuario, confianza, identificador único de la clave, identificador único de propietario, revocación), para su posterior presentación en pantalla a través de una tabla.
- private void generaDatosTablaUsuarios(): Método que permite recuperar los datos de cada identidad de usuario, existente en la estructura, y sus datos asociados (nombre de usuario, correo electrónico, comentario, identificador único propietario e indicador primario), para su posterior presentación en pantalla a través de una tabla.
- private void verificaFirmaClave(): Método que busca en la estructura de datos la clave seleccionada, a través de su identificador único, y verifica si la firma digital asociada con la clave es válida utilizando un objeto de Firma.
- private void verificaFirmaUsuario(): Método que busca en la estructura de datos la identidad de usuario seleccionada, a través de su identificador único, y verifica si la firma digital asociada con la identidad de usuario es válida utilizando un objeto de Firma.

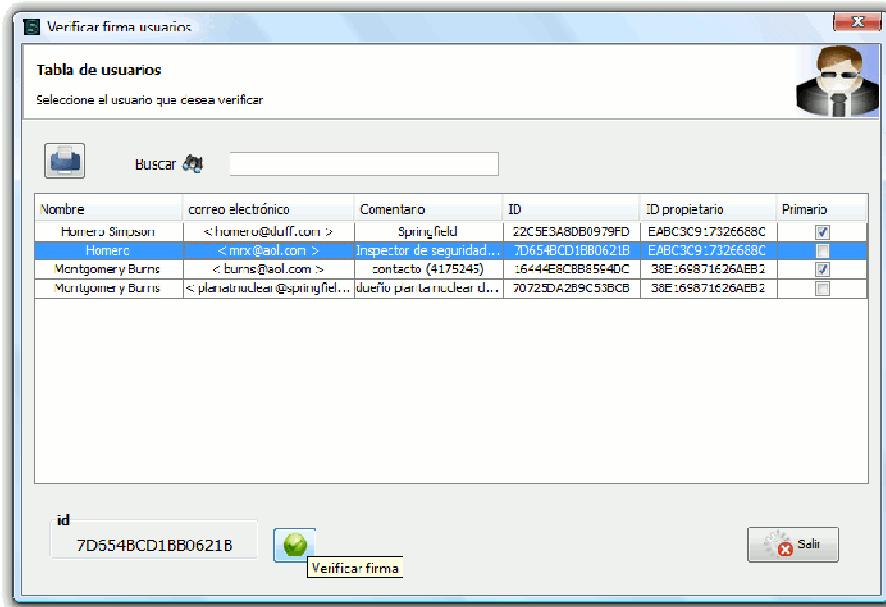


Figura 4.7: Interfaz gráfica para validación de identidad de usuario.

Configuración de preferencias

❖ Configurador_GUI: Esta clase implementa la interfaz gráfica que permite cambiar opciones de configuración, adaptando el sistema a las condiciones que el usuario estime conveniente. (Ver Figura 4.8). Esta clase se extiende de *javax.swing.JDialog*. Algunos de sus métodos relevantes son:

- public int getCfrAscii(): Método que indica si la opción de cifrado en ASCII blindado está activado.
- public String getIdClvCfr(): Método que indica si se ha dejado por defecto una clave pública subordinada para realizar una operación de cifrado de archivo.
- public int getFmrAscii(): Método que indica si la opción de firmado digital en ASCII blindado está activado.
- public void setIdClvFmr(): Método que indica si se ha dejado por defecto una clave pública subordinada para realizar una operación de generación de firma digital de archivo.

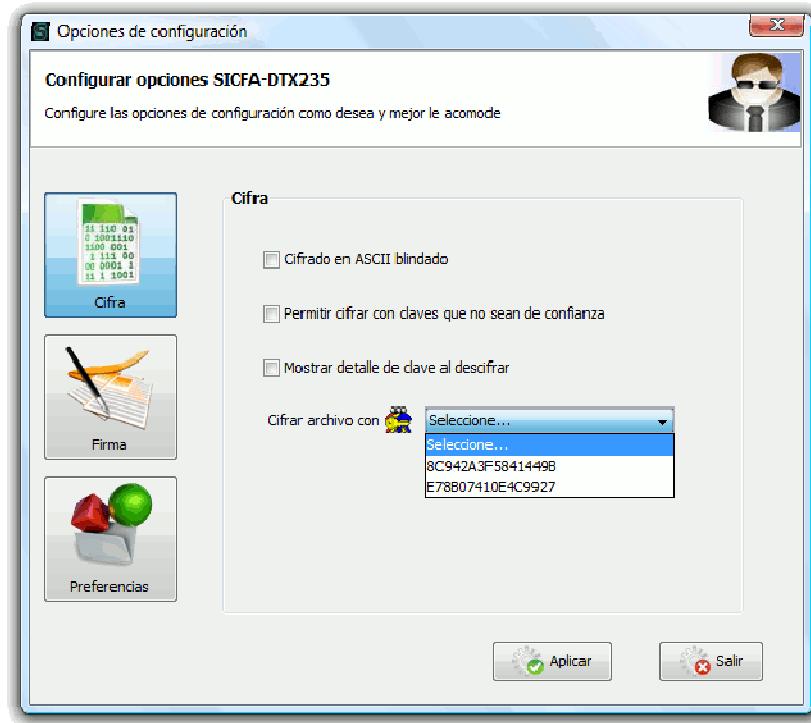


Figura 4.8: Interfaz gráfica de opciones de configuración del sistema.

4.3.2 LLAVERO PERSONAL O GESTOR DE CLAVES SEGURAS (GC)

Este subsistema permite: creación de registros; edición de registros; eliminación de registros; almacenamiento de registros en archivo digital; recuperación de registros desde archivo digital para su gestión en memoria; almacenamiento de clave segura o contraseña, cifrada; copia de clave segura o contraseña al portapapeles para un traslado seguro. Cada registro contiene en su interior datos asociados con una clave segura generada por el sistema o contraseña ingresada por el usuario.

Algunas de las clases, organizadas por funcionalidad, asociadas con este subsistema son presentadas a continuación.

Gestión de registros en memoria

❖ Registro: Esta clase permite almacenar los datos de un nuevo registro que será incorporado al llavero personal de un usuario. Esta clase se extiende de la clase Packet. Algunos de sus métodos relevantes son:

- `public byte[] getId():` Este método permite obtener el identificador único del registro.
- `private void asignaUsuario(String usuario):` Este método permite asignar un nombre de usuario al registro creado. Ingresa por parámetro una cadena de texto que contiene el nombre de usuario.
- `private void asignaDominio(String dominio):` Este método permite asignar un dominio al registro creado. Ingresa por parámetro una cadena de texto que contiene el dominio.
- `public void asignaObservacion(String observac):` Este método permite asignar una observación con respecto al dominio del registro. Ingresa por parámetro una cadena de texto que contiene una observación.
- `public void asignaConfianza(int opcion):` Método que permite asignar un indicador del nivel de confianza en el dominio del registro. Ingresa por parámetro un indicador del nivel confianza (nula, marginal, parcial, total).
- `private void asignaClave(String clave):` Método que permite asignar la clave segura o contraseña a un registro en particular.
- `private void asignaId():` Método que permite asignar un identificador único al registro, solicitando un valor hash a un objeto de la clase MessageDigestSicfa perteneciente al subsistema MS.
- `private byte[] cifraClave():` Método que permite cifrar la clave segura o contraseña del registro utilizando un objeto de la clase PBE del subsistema MS.
- `public byte[] descifraClave():` Método que permite descifrar la clave segura o contraseña del registro utilizando un objeto de la clase PBE del subsistema MS.
- `public void escribeArchivo(Base64EncoderOutputStream out):` Método que permite escribir datos del registro en un archivo digital. Ingresa por parámetro un flujo de salida con formato Base64.
- `public void leeArchivo(InputStream is):` Método que permite recuperar datos de un registro desde un archivo digital. Ingresa por parámetro un flujo entrada.

❖ ListaRegistros: Esta clase permite la creación de una lista, la cual contiene cada uno de los registros del Llavero personal. Algunos de sus métodos relevantes son:

- `public void addRegistro(Object registro):` Método que permite añadir un registro a la lista. Ingresa por parámetro un objeto de Registro.
- `public Object buscaRegistro(byte[] id):` Método que permite buscar un registro en la lista a partir de un identificador único y lo retorna si lo encuentra. Ingresa por parámetro un arreglo binario que contiene el identificador único del registro.
- `public boolean eliminaRegistro(byte[] id):` Método que permite eliminar un registro en la lista a partir de su identificador único. Ingresa por parámetro un arreglo binario que contiene el identificador único del registro.

- public void leeArchivoRegistros(): Método que permite recuperar registros desde un archivo digital (archivo de Llavero personal) y almacena cada uno de ellos en la lista de registros.
- public void escribeArchivoRegistros(): Método que permite escribir los datos de cada uno de los registros contenidos en la lista de registros, en un archivo digital (archivo de Llavero personal).

Ventana principal

- ❖ Llavero_GUI: Esta clase implementa la interfaz gráfica que permite administrar registros con información confidencial, proporcionando un acceso rápido y seguro a ellos cuando el usuario los necesite. (Ver Figura 4.9). Esta clase se extiende de *javax.swing.JFrame*. Algunos de sus métodos relevantes son:
- private void generaDatosTabla(): Método que permite recuperar los datos asociados a cada registro, existentes en la estructura, para su posterior presentación en pantalla a través de una tabla.
 - public void portapapeles(String cadenaTexto): Método que permite copiar al portapapeles del sistema cualquier cadena de texto. Ingresa por parámetro una cadena de texto.

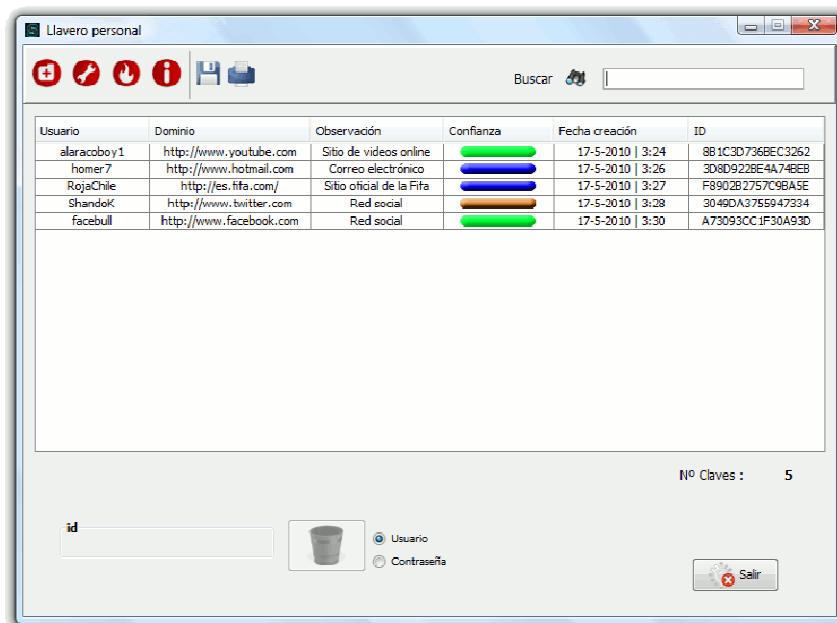


Figura 4.9: Interfaz gráfica de Llavero personal o Gestor de claves seguras.

Administración de los registros

- ❖ NuevoRegistro_GUI: Esta clase implementa la interfaz gráfica que permite el ingreso de los datos para la creación de un nuevo registro. (Ver Figura 4.10). Esta clase se extiende de *javax.swing.JDialog*. Algunos de sus métodos relevantes son:
- private void botonGeneraClaveSegura(): Este método permite generar una clave segura para un registro, utilizando un objeto de la clase GeneradorDeClaves del subsistema MS.

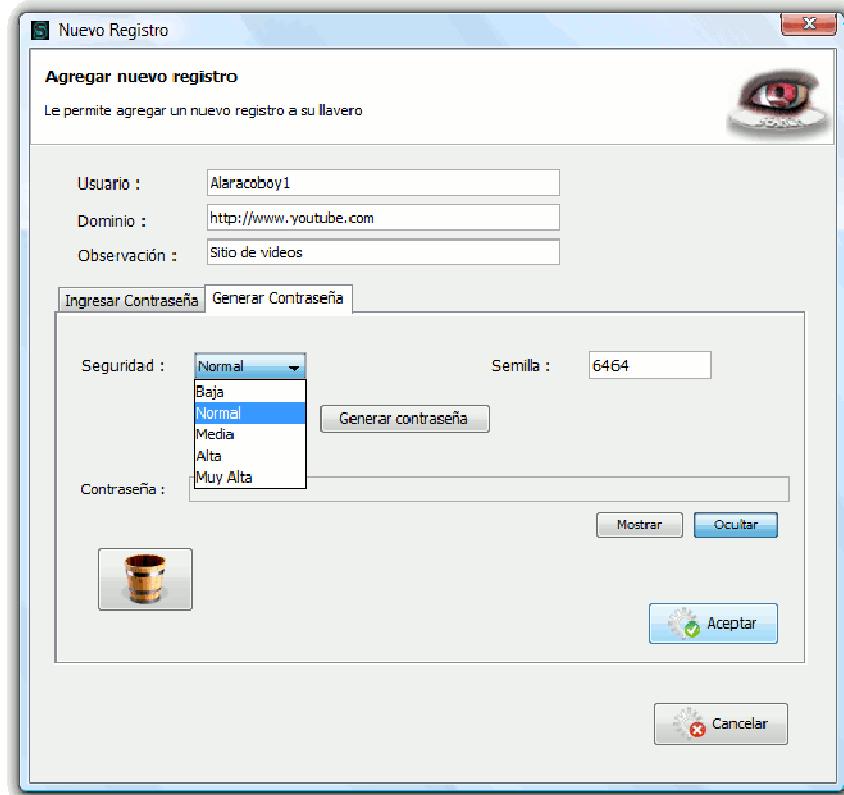


Figura 4.10: Interfaz gráfica para el ingreso de datos de un nuevo registro.

- ❖ **EdicionRegistro_GUI:** Esta clase implementa la interfaz gráfica que permite la edición de los datos asociados a un registro. Esta clase se extiende de *javax.swing.JDialog*. Algunos de sus métodos relevantes son:
 - **private void generaDatosRegistro():** Método que permite generar y presentar en pantalla los datos de un registro seleccionado por el usuario.

4.3.3 COMPLEMENTOS SIMÉTRICOS (CS)

Este subsistema permite: borrado seguro de archivo; generación de archivo auto-descifrable; ingreso de datos para la ejecución de operaciones de cifrado/descifrado de archivo (cifrado simétrico); ingreso de datos para la ejecución de operaciones de cifrado/descifrado de archivo (cifrado simétrico PBE); ingreso de datos para la ejecución de operaciones de generación/verificación valor hash MDC; ingreso de datos para la ejecución de operaciones de generación/verificación valor hash MAC.

Algunas de las clases, organizadas por funcionalidad, asociadas con este subsistema son presentadas a continuación.

Ventana principal

- ❖ **ComplementosSimetricos_GUI:** Esta clase implementa la interfaz gráfica necesaria para permitir a un usuario ingresar los datos necesarios para llevar a cabo alguna operación de: cifrado/descifrado simétrico, cifrado/descifrado simétrico PBE, generación/verificación de valores hash MDC, generación/verificación de valor hash MAC, borrado seguro de archivo, y generación de archivo auto-descifrable. (Ver Figura 4.11). Esta clase se extiende de *javax.swing.JFrame*.

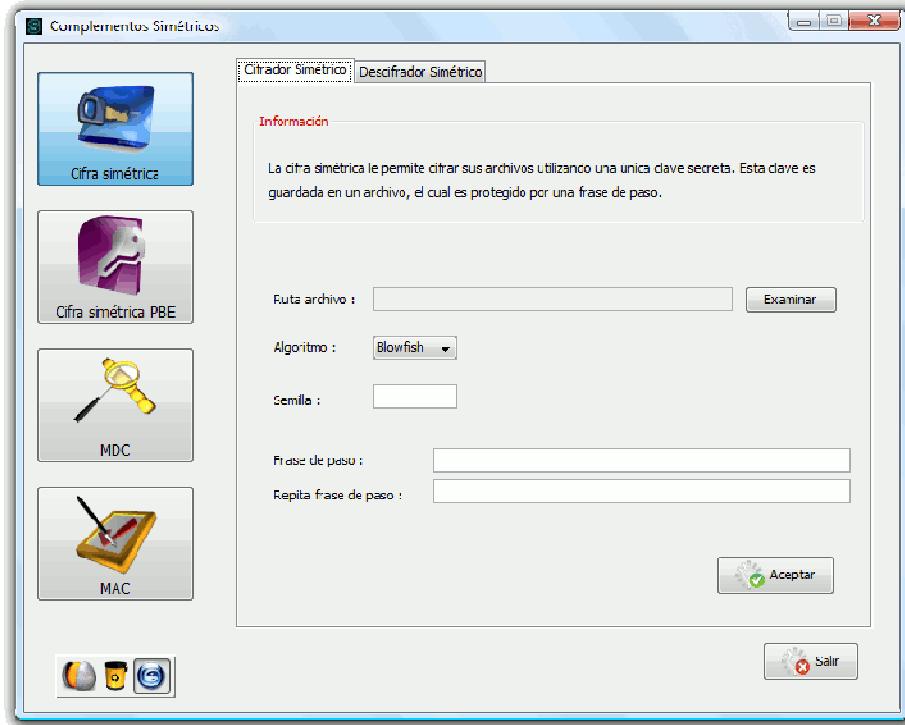


Figura 4.11: Interfaz gráfica de Complementos Simétricos.

Generación de archivo auto-descifrable

- ❖ CifradoPBEJar_GUI: Esta clase implementa la interfaz gráfica para el ingreso de datos y posterior generación de un archivo auto-descifrable. (Ver Figura 4.12). Esta clase se extiende de *javax.swing.JFrame*. Algunos de sus métodos relevantes son:
 - private void escribeArchivoJar: Método que permite la generación de un archivo Jar, que contiene en su interior un archivo cifrado y los archivos con las clases necesarias para descifrar este archivo y recuperar su contenido original.

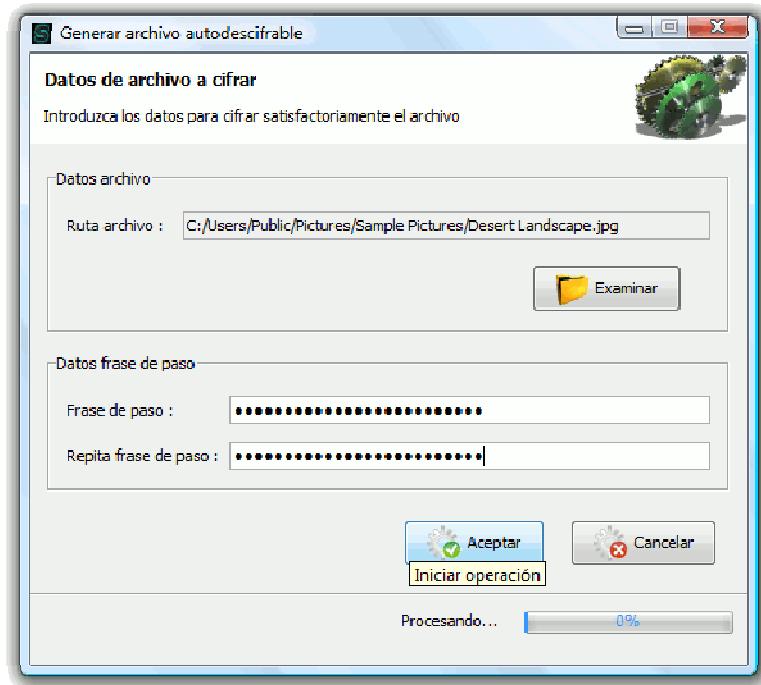


Figura 4.12: Interfaz gráfica para el ingreso de datos de generación de un archivo auto-descifrable.

Borrado seguro de archivos

- ❖ **BorradoSeguro_GUI:** Esta clase implementa la interfaz gráfica para el ingreso de los datos necesarios para llevar a cabo una operación de borrado seguro de archivo. El archivo es borrado una vez que se validan los datos. (Ver Figura 4.13). Esta clase se extiende de *javax.swing.JFrame*. Algunos de sus métodos relevantes son:
 - `private void borradoDOD5220():` Método que permite eliminar un archivo utilizando el método de borrado seguro American DoD 5220-22.M Standard Wipe, utilizado por el departamento de defensa de E.E.U.U (DOD), basado en el NISPO (National Industrial Security).

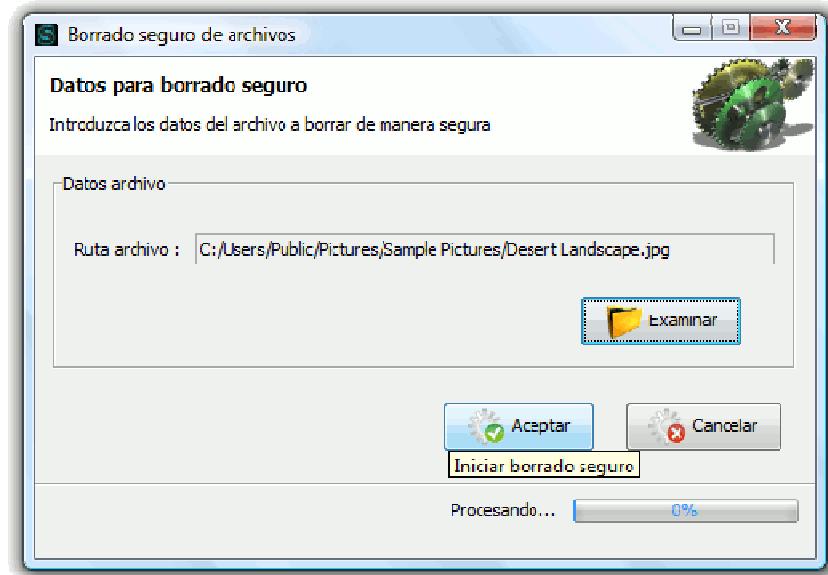


Figura 4.13: Interfaz gráfica para el ingreso de datos de borrado seguro de archivos.

4.3.4 EDITOR SICFA (ESIC)

Este subsistema permite: crear mensajes, en los que además de texto se puede insertar imágenes; ingreso de datos para la ejecución de operaciones de cifrado/descifrado de mensaje (cifrado híbrido); ingreso de datos para la ejecución de operaciones de generación de firma digital y cifrado de mensaje/descifrado y verificación de firma digital de mensaje.

Algunas de las clases, organizadas por funcionalidad, asociadas con este subsistema son presentadas a continuación.

Ventana principal

- ❖ **Editor_GUI:** Esta clase implementa la interfaz gráfica de editor de texto que permite al usuario crear mensajes, en los que además de texto puede insertar imágenes. Estos mensajes nunca son almacenados en claro en el disco duro, sino que antes se le aplica una operación de cifrado o firmado/cifrado. También dispone de operaciones de descifrado o descifrado/verificación. Esta clase se extiende de *javax.swing.JDialog*. (Ver Figura 4.14). Algunos de sus métodos relevantes son:
 - `private void escribeObjetoAArreglo():` Método que permite comprimir el contenido del documento del editor para luego pasarlo a un arreglo binario.

- `private void recuperaObjetoDeArreglo(byte[] entrada)`: Método que permite recuperar el contenido de un arreglo binario, lo descomprime, y lo convierte en objeto para posteriormente visualizar el contenido en un nuevo documento del editor de texto. Ingresá por parámetro un arreglo binario con contenido comprimido.

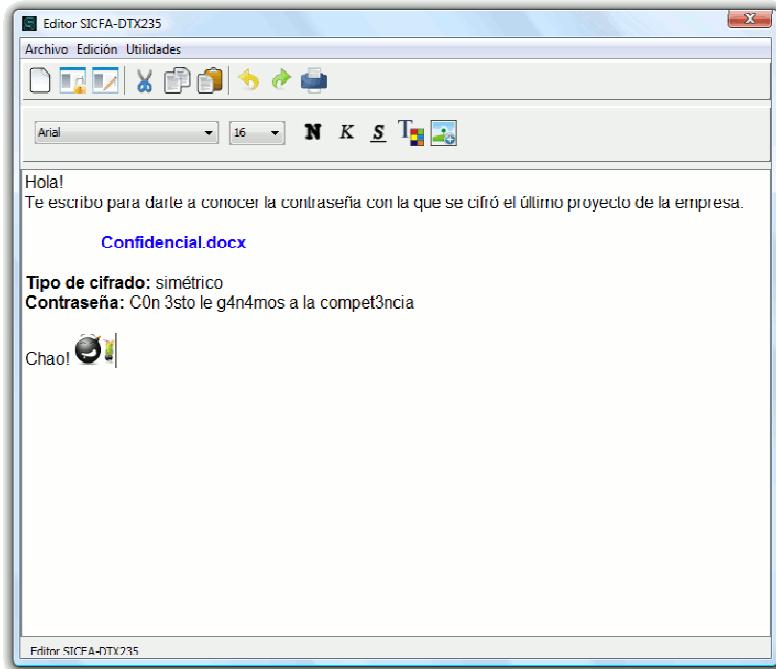


Figura 4.14: Interfaz gráfica del Editor SICFA.

Cifrado o firma de mensajes

- ❖ `TablaClvsEditor_GUI`: Esta clase implementa la interfaz gráfica necesaria para mostrar una serie de claves públicas subordinadas, disponibles en la estructura de datos. Una de estas claves presentadas, es seleccionada por el usuario para realizar alguna operación de cifrado o generación de firma digital. (Ver Figura 4.15). Esta clase se extiende de `javax.swing.JDialog`. Algunos de sus métodos relevantes son:
 - `private void generaDatosTablaCfr()`: Método que permite recuperar los datos de cada clave pública subordinada RSA, existente en la estructura, y sus datos asociados (nombre de usuario, identificador único de la clave, identificador único del propietario, revocación, caducidad), para su posterior presentación en pantalla a través de una tabla.
 - `private void generaDatosTablaFmr()`: Método que permite recuperar los datos de cada clave pública subordinada, existente en la estructura, y sus datos asociados (nombre de usuario, identificador único de la clave, revocación, caducidad), para su posterior presentación en pantalla a través de una tabla.
 - `private void cifraConClv()`: Método que permite buscar una clave pública subordinada RSA en memoria y luego solicita a un objeto del subsistema MS una operación de cifrado para el contenido del documento del editor.
 - `public void firmaConClv()`: Método que permite buscar una clave pública subordinada en memoria y luego solicita a un objeto del subsistema MS una operación de generación de firma digital para el contenido del documento del editor.

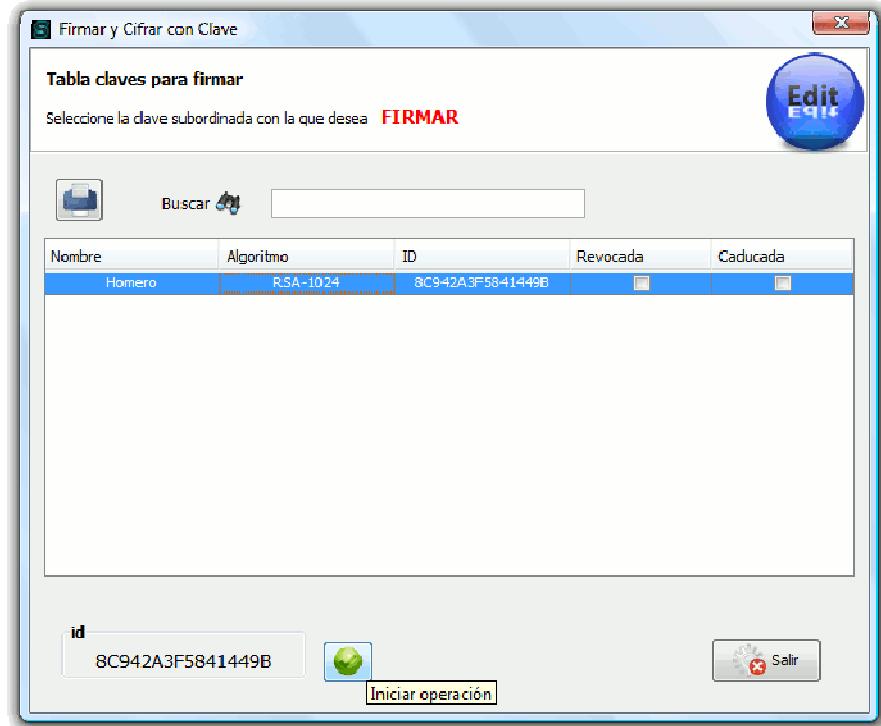


Figura 4.15: Interfaz gráfica para la selección de clave pública para una operación de generación de firma digital.

4.3.5 MECANISMOS DE SEGURIDAD (MS)

Este subsistema colabora con todos los demás subsistemas y permite la ejecución de operaciones criptográficas sobre un flujo de bits. Las operaciones que se pueden destacar son: generación de parejas de claves (pública y privada), cifrado/descifrado simétrico y asimétrico, generación/verificación de valor hash MDC y MAC, generación/verificación de firma digital y generación de números pseudoaleatorios. Dada la complejidad y requerimientos de seguridad que conlleva la implementación de algoritmos criptográficos, éstos son proveídos por terceros por intermedio de la arquitectura de seguridad de java.

Algunas de las clases, organizadas por funcionalidad, asociadas con este subsistema son presentadas a continuación.

Generación de claves

- ❖ KeyPairGeneratorSicfa: Esta clase permite generar una pareja de claves (pública y privada).
- ❖ GeneradorDeClaves: Esta clase permite obtener una clave binaria aleatoria de una determinada longitud en formato ASCII. Algunos de sus métodos relevantes son:
 - public void ObtieneValores(int alg, int semillaUsuario): Método que permite establecer los valores iniciales para obtener un generador de claves binarias. Ingresan por parámetro: el identificador de algoritmo y valor de semilla ingresada por el usuario.
 - private void generaClvSesion(): Método que permite generar una clave binaria aleatoria y segura.

- `private void convierteClvABase64()`: Método que permite convertir la clave segura generada, de formato binario a formato ASCII.

Cifrado híbrido de archivos

- ❖ `CifradorSicfa`: Esta clase permite la creación de un archivo SICFA que contiene en su interior datos cifrados (cifrado híbrido) de un archivo. Algunos de sus métodos relevantes son:
 - `public void obtieneValores(byte[] clvPub, int semillaUsuario, byte[] idProp, String rutaFicheroCfr)`: Método que permite establecer los valores iniciales para obtener un cifrador híbrido. Ingresan por parámetro: la clave pública subordinada, una semilla de usuario, identificador único de propietario y ruta del archivo a cifrar.
 - `private void escribeArchivo()`: Método que permite cifrar un archivo y el contenido cifrado lo va guardando en un nuevo archivo (formato binario o ASCII), el cual es almacenado en la misma ruta del archivo original.
 - `public byte[] cifraAsimet(byte[] entrada)`: Método que permite cifrar un arreglo binario de datos utilizando cifrado asimétrico y retorna el resultado en otro arreglo binario, de tamaño mayor que el arreglo de entrada. El parámetro entrada, corresponde al arreglo binario con datos en claro.
 - `public byte[] cifraSimet(byte[] entrada)`: Método que permite cifrar un arreglo binario de datos utilizando cifrado simétrico y retorna el resultado en otro arreglo binario, de tamaño mayor que el arreglo de entrada. El parámetro entrada, corresponde al arreglo binario con datos en claro.
- ❖ `CifradorEditor`: Esta clase permite la creación de un archivo SICFA que contiene en su interior datos cifrados de un documento del editor. Algunos de sus métodos relevantes son:
 - `public void obtieneValores(byte[] clvPub, int semillaUsuario, byte[] idProp, String rutaArchivo)`: Método que permite establecer los valores iniciales para obtener un cifrador híbrido. Ingresan por parámetro: la clave pública subordinada, una semilla de usuario, identificador único de propietario y ruta del archivo a cifrar.
 - `public void obtieneFirmaObjeto(byte[] arregloFirma)`: Método que permite obtener la firma digital del documento del editor que está siendo cifrado. El parámetro de entrada es un arreglo binario con la firma digital.
 - `public void escribeArchivo(byte[] objetoDoc)`: Método que permite escribir en un archivo (formato binario o ASCII) definido por el usuario, el contenido firmado y/o cifrado de un mensaje digitado en el editor. El parámetro de entrada corresponde al documento del editor.
 - `public byte[] cifraAsimet(byte[] entrada)`: Método que permite cifrar un arreglo binario de datos utilizando cifrado asimétrico y retorna el resultado en otro arreglo binario, de tamaño mayor que el arreglo de entrada. El parámetro entrada, corresponde al arreglo binario con datos en claro.
 - `public byte[] cifraSimet(byte[] entrada)`: Método que permite cifrar un arreglo binario de datos utilizando cifrado simétrico y retorna el resultado en otro arreglo binario, de tamaño mayor que el arreglo de entrada. El parámetro entrada, corresponde al arreglo binario con datos en claro.

La figura 4.16 muestra un mensaje digitado en el editor de texto SICFA y su correspondiente armadura ASCII, luego de aplicarle al mensaje una operación de cifrado.

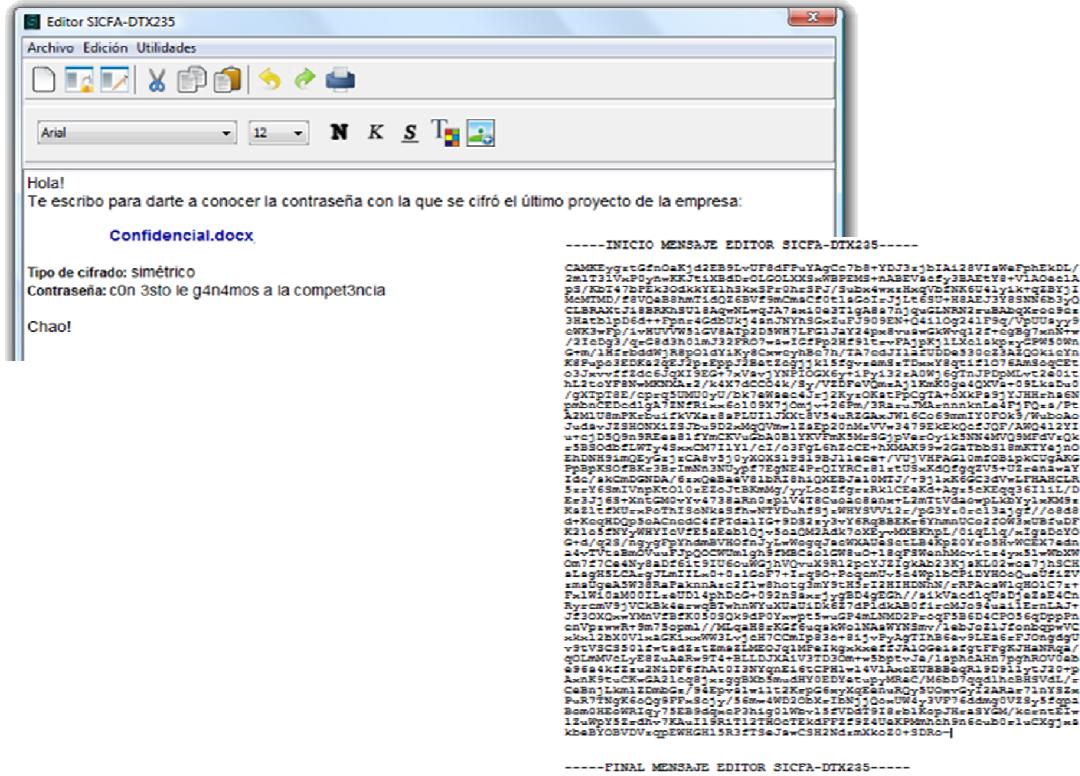


Figura 4.16: Mensaje digitado en editor SICFA y su salida en formato ASCII.

Cifrado simétrico de archivos

- ❖ **CifradorSimétrico:** Esta clase permite la creación de un archivo SICFA que contiene en su interior datos cifrados (cifrado simétrico) de un archivo. Algunos de sus métodos relevantes son:

- public void obtieneValores(byte[] clvPub, int semillaUsuario, byte[] idProp, String rutaFicheroCfr): Método que permite establecer los valores iniciales para obtener un cifrador simétrico único. Ingresan por parámetro: la ruta del archivo a cifrar, identificador del algoritmo para inicializar el cifrador simétrico y semilla de usuario.
 - private void escribeArchivo(): Método que permite cifrar un archivo y el contenido cifrado lo va guardando en un nuevo archivo (formato binario o ASCII), el cual es almacenado en la misma ruta del archivo original.
 - public byte[] cifraSimet(byte[] entrada, int longitud): Método que permite cifrar un arreglo binario de datos utilizando cifrado simétrico y retorna el resultado en otro arreglo binario, de tamaño mayor que el arreglo de entrada. Ingresan por parámetro: el arreglo binario, con datos en claro, y el largo del arreglo.

Cifrado simétrico PBE de archivos

- ❖ **CifradorSimetricoPBE:** Esta clase permite la creación de un archivo SICFA que contiene en su interior datos cifrados (cifrado simétrico PBE, clave binaria generada a partir de una contraseña o frase de paso) de un archivo. Algunos de sus métodos relevantes son:

- public void obtieneValores(String rutaFicheCfr, int alg, int semillaUsuario, String frasePaso): Método que permite establecer los valores iniciales para obtener un cifrador simétrico PBE único. Ingresan por parámetro: la ruta del archivo a cifrar, identificador del algoritmo para inicializar el cifrador simétrico PBE, semilla de usuario y la frase de paso.
- private void escribeClvEnArchivo(String rutaFichero): Método que permite cifrar un archivo y el contenido cifrado lo va guardando en un nuevo archivo (formato binario o ASCII), el cual es almacenado en la misma ruta del archivo original. Ingresan por parámetro una cadena de texto que contiene la ruta del archivo original.
- public byte[] cifraSimet(byte[] entrada, int longitud): Método que permite cifrar un arreglo binario de datos utilizando cifrado simétrico PBE y retorna el resultado en otro arreglo binario, de tamaño mayor que el arreglo de entrada. Ingresan por parámetro: el arreglo binario, con datos en claro, y el largo del arreglo.

Generación de firma digital de datos

- ❖ SignatureSicfa: Esta clase permite firmar digitalmente un arreglo binario de datos. Además, permite la verificación de una firma digital.
- ❖ FirmaDigitalSicfa: Esta clase permite la creación de un archivo SICFA que contiene en su interior la firma digital de un archivo. Algunos de sus métodos relevantes son:
 - public void obtieneValores(byte[] clvPriv, int alg, byte[] idProp, String rutaFichFirmar): Método que permite establecer los valores iniciales para obtener un generador de firma digital. Ingresan por parámetro: la clave privada subordinada, identificador de algoritmo para inicializar generador, identificador único de propietario y ruta del archivo a firmar digitalmente.
 - private void escribeArchivoFirma(): Método que permite firmar digitalmente un archivo y la firma se guarda en un nuevo archivo (formato binario o ASCII), el cual es almacenado en la misma ruta del archivo original.

La figura 4.17 presenta, a modo de ejemplo, la salida en formato ASCII de un archivo SICFA que contiene la firma digital de un archivo digital.

```
-----INICIO FIRMA DIGITAL SICFA-DTX235-----  
pRUHCgIHK4yUKj9YQUSBAlAUg8yrCEMsJmnxqkMzo9Hs8rde1g/6pa2Iu4Ddy+qQ  
KJDFnuOfSFdMlf5b5j3IfYzzIo9ZNoYTwtjsgasr2JscGnyFxAr3f69lg1zh3u67  
HPWFeBPJ8NDDfkOQixEij0LGKJO7OT4vvB//i5FUUmYneFzzzI7bWNULshFoNNZ3N  
KA==  
-----FINAL FIRMA DIGITAL SICFA-DTX235-----
```

Figura 4.17: Firma digital de archivo digital, representada en formato ASCII.

- ❖ FirmaDigitalEditor: Esta clase permite la creación de un arreglo binario que contiene en su interior la firma digital de los datos de un documento del editor. Algunos de sus métodos relevantes son:
 - public void obtieneValores(byte[] clvPriv, int alg, byte[] idProp): Método que permite establecer los valores iniciales para obtener un generador de firma digital. Ingresan por parámetro: la clave privada subordinada, identificador de algoritmo para inicializar generador e identificador único de propietario.

- `public void generaArregloFirma(byte[] arregloDatos)`: Método que permite firmar digitalmente un arreglo binario de datos y su firma digital la almacena en otro arreglo binario. Ingresa por parámetro el arreglo binario con los datos que serán firmados.

Generación de valor hash de datos

- ❖ `MessageDigestSicfa`: Esta clase permite generar un valor hash a partir de un arreglo binario de datos.
- ❖ `MDCSicfa`: Esta clase permite obtener, a partir de un archivo, un valor hash que lo representa y asegura su integridad. Algunos de sus métodos relevantes son:
 - `private void obtieneValores(String rutaFiche, int alg)`: Método que permite establecer los valores iniciales para obtener un generador MDC único. Ingresan por parámetro: la ruta del archivo a calcular su valor hash e identificador del algoritmo para inicializar el generador.
 - `private void escribeArchivoMDC()`: Método que permite obtener el valor hash de un archivo y lo guarda en un nuevo archivo (formato binario o ASCII), el cual es almacenado en la misma ruta del archivo original.
- ❖ `MACSicfa`: Esta clase permite obtener, a partir de un archivo y una contraseña, un valor hash que lo representa y asegura su autenticidad e integridad. Algunos de sus métodos relevantes son:
 - `private void obtieneValores(String rutaFiche, int alg, String password)`: Método que permite establecer los valores iniciales para obtener un generador MAC único. Ingresan por parámetro: la ruta del archivo a calcular su valor hash, identificador del algoritmo para inicializar el generador y la contraseña.
 - `private void escribeArchivoMAC()`: Método que permite obtener el valor hash a partir de un archivo y una contraseña; el resultado se guarda en un nuevo archivo (formato binario o ASCII), el cual es almacenado en la misma ruta del archivo original.

Capítulo 5 | SICFA-DTX235: Pruebas

“Las contraseñas son como la ropa interior. No puedes dejar que nadie la vea, debes cambiarla regularmente y no debes compartirla con extraños”

Chris Pirillo.

Además de realizar en cada iteración algunas pruebas de caja blanca, caja negra y de integración se hicieron 2 tipos de pruebas las cuales fueron documentadas, éstas son:

- Pruebas de validación. Las pruebas de validación empiezan tras la culminación de la prueba de integración, cuando se han ejercitado los componentes individuales, se ha terminado de ensamblar el software como paquete y se han descubierto y corregido los errores de interfaz. La prueba se concentra en las acciones visibles para el usuario y en la salida del sistema que éste puede reconocer [PRE-05].
- Pruebas de rendimiento. Las pruebas de rendimiento permiten determinar lo rápido que realiza una tarea el sistema en condiciones particulares.

5.1 Pruebas de validación

En las pruebas de validación, a partir de los casos de prueba se puede establecer si los requerimientos del sistema han sido cumplidos en su totalidad y satisfacen las expectativas del cliente. Cada uno de los casos de prueba, fueron documentados de forma separada siguiendo la siguiente plantilla.

ID Caso de prueba	Índice correlativo del caso de prueba.
ID requerimientos	IDs de los requerimientos involucrados en los casos de prueba.
Nombre de caso de prueba	Nombre representativo del caso de prueba.
Descripción	Breve descripción del desarrollo y objetivos del caso de prueba.
Condiciones de ejecución	Descripción de las condiciones de ejecución que se deben cumplir antes de iniciar el caso de prueba, por ejemplo, que se haya realizado correctamente el login en el sistema.
Entrada	Descripción, paso a paso, de la ejecución del caso de prueba.
Resultado esperado	Descripción del resultado que se esperaba de la prueba de test en la aplicación.
Evaluación de la prueba	Estado del caso de prueba, que puede ser por ejemplo: propuesta, pendiente de evaluación, realizada y satisfactoria, etcétera.

Tabla 5.1: Plantilla de casos de prueba para pruebas de validación.

Los casos de prueba más relevantes, que permiten verificar el correcto funcionamiento y cumplimiento de los requerimientos, en los que mayor interés se ha puesto, se presentan a continuación.

ID Caso de prueba	03
ID requerimientos	02
Nombre de caso de prueba	Ingreso de datos de pareja de claves principal.
Descripción	Se llenan los campos, desplegados en la interfaz gráfica, relacionados con la creación de una pareja de claves principal.
Condiciones de ejecución	Que el usuario haya seleccionado la opción: "Crear pareja de claves principal". Se debe haber seleccionado un directorio donde guardar las claves.
Entrada	Se inicializa la aplicación y se selecciona la opción: "Crear pareja de claves principal". Se despliega una nueva interfaz gráfica con todos los campos, de ingreso de datos, vacíos. Se llenan todos los campos de forma incorrecta. Nombre: Correo electrónico: juan_perezchile.cl Comentario: Semilla: 123 Clave de sesión: hola Repetición clave de sesión: holas Se presiona el botón "Aceptar".
Resultado esperado	El sistema despliega los mensajes correspondientes a los campos ingresados erróneamente y la ventana de ingreso permanece abierta mientras no se presione el botón "Cancelar". Por lo tanto se puede seguir llenando los campos.
Evaluación de la prueba	Realizada y satisfactoria.

Tabla 5.2: Prueba de validación 03.

ID Caso de prueba	07
ID requerimientos	23
Nombre de caso de prueba	Cargar claves propias.
Descripción	Se despliega una interfaz gráfica para validar el ingreso de la clave de sesión, se ingresa la clave de sesión incorrecta por lo que no se debe acceder a los anillos.
Condiciones de ejecución	Que el usuario haya seleccionado la opción: "Cargar claves propias".
Entrada	Se despliega una nueva interfaz gráfica para seleccionar el directorio donde se encuentran los anillos de claves públicas, claves privadas y confianzas. El sistema despliega una interfaz gráfica para ingresar la clave de sesión, la cual fue creada al momento de la generación de la pareja de claves principal. Se ingresa la clave de sesión: chao. Se presiona el botón "Aceptar".

Resultado esperado	El sistema muestra un mensaje donde indica que la clave de sesión ingresada es errónea Se mantiene la ventana de ingreso de la clave de sesión.
Evaluación de la prueba	Realizada y satisfactoria.

Tabla 5.3: Prueba de validación 07.

ID Caso de prueba	09
ID requerimientos	09 - 14 – 26 - 27 – 28 – 29 – 30
Nombre de caso de prueba	Generar pareja de claves subordinadas.
Descripción	Se ingresan los datos, para la creación de una pareja de claves subordinadas, en los siguientes campos. Semilla, Caducidad y Algoritmo.
Condiciones de ejecución	Que el usuario haya seleccionado desde el menú del administrador de claves la opción: “Clave→Generar pareja de claves”. Que exista al menos una identidad de usuario con la cual asociar la nueva pareja de claves.
Entrada	Se despliega una nueva interfaz grafica para ingresar los siguientes datos: Identidad de usuario: 82CCCC6F0B7B5173 Semilla: 1234. Caducidad: 26 – 1 – 2015. Algoritmo: RSA 1024. Se presiona el botón “Aceptar”. El sistema valida los datos correctamente y despliega la ventana para el ingreso de la frase de paso necesaria para firmar digitalmente la nueva pareja de claves subordinadas. Se ingresa la frase de paso: mifrasesdepaso .
Resultado esperado	El sistema muestra un mensaje indicando el éxito de la operación. El sistema actualiza el administrador de claves.
Evaluación de la prueba	Realizada y satisfactoria.

Tabla 5.4: Prueba de validación 09.

ID Caso de prueba	25
ID requerimientos	15 – 47
Nombre de caso de prueba	Firmado digital de archivo en formato ASCII.
Descripción	Se firma digitalmente un archivo y su firma digital se guarda en un archivo de texto con formato ASCII con extensión sigdtxa.
Condiciones de ejecución	Que desde el menú: “Opciones→Configurar” se haya activado la casilla “firmado en ASCII blindado”. Que se haya seleccionado desde el menú del administrador de claves la opción: “Firma→Firmar archivo”. Que exista al menos una pareja de claves subordinadas. Que exista un archivo al cual firmar.

Entrada	<p>Se despliega una interfaz gráfica para seleccionar un archivo.</p> <p>Se selecciona el archivo imagen.png</p> <p>Se presiona el botón “Abrir”.</p> <p>El sistema despliega una interfaz para seleccionar la clave con la cual se desea firmar.</p> <p>Se selecciona la clave RSA-1024 con ID: B467E739A2A37761.</p> <p>Se presiona el botón “Iniciar operación”.</p> <p>El sistema muestra un mensaje para confirmar o cancelar el proceso de firmado.</p> <p>Se confirma la operación.</p> <p>El sistema despliega la interfaz para el ingreso de la frase de paso.</p> <p>Se rellena el campo Frase de paso: mifrasedepaso.</p> <p>Se presiona el botón “Aceptar”.</p>
Resultado esperado	<p>El sistema muestra un mensaje donde se informa de que el firmado se realizó correctamente.</p> <p>Se genera el archivo imagen.png.sigdtxa en el directorio donde se encuentra el archivo.</p>
Evaluación de la prueba	Realizada y satisfactoria.

Tabla 5.5: Prueba de validación 25.

ID Caso de prueba	32
ID requerimientos	15 – 45
Nombre de caso de prueba	Cifrar Archivo.
Descripción	Se cifra un archivo cualquiera y se obtiene un archivo cifrado en formato ASCII con extensión sicfadtxa .
Condiciones de ejecución	Que exista al menos una clave pública subordinada. Que exista al menos un archivo al cual cifrar.
Entrada	<p>Se selecciona desde el menú del administrador de claves la opción: “Opciones→Configurar”.</p> <p>Se despliega la interfaz gráfica para ingresar la clave de sesión.</p> <p>Se ingresa la clave de sesión: hola</p> <p>Se presiona sobre el botón Cifra y se activa la casilla “Cifrado en ASCII blindado”.</p> <p>Se presiona el botón “Aplicar”.</p> <p>Se selecciona desde el menú del administrador de claves la opción Archivo→Cifrar archivo.</p> <p>Se despliega una interfaz gráfica para seleccionar un archivo.</p> <p>Se selecciona el archivo biometría.pdf.</p> <p>Se presiona el botón “Abrir”.</p> <p>Se despliega una interfaz gráfica para seleccionar la clave pública subordinada con la cual se cifrará el archivo.</p> <p>Se selecciona la clave con ID: B467E739A2A37761 cuyo propietario es Juan Pérez.</p> <p>Se ingresa la semilla 1234 necesaria para generar la</p>

	clave secreta de sesión. Se presiona el botón “Iniciar operación”. Se despliega un mensaje para confirmar o cancelar la operación de cifrado. Se confirma la operación.
Resultado esperado	El sistema muestra un mensaje donde se informa que el cifrado ha sido exitoso. Se crea el archivo biometría.pdf.sicfadtxa en el mismo directorio donde se encuentra el archivo original.
Evaluación de la prueba	Realizada y satisfactoria.

Tabla 5.6: Prueba de validación 32.

ID Caso de prueba	39
ID requerimientos	53 – 54 – 57
Nombre de caso de prueba	Crear nuevo registro
Descripción	Se ingresan los datos para la creación de un nuevo registro con los siguientes campos. Usuario, Dominio, Observación, Contraseña y Repita contraseña
Condiciones de ejecución	Que se haya seleccionado desde la bandeja del sistema la opción: “Llavero personal”. Que se haya presionado el botón “Agregar nuevo registro”.
Entrada	Se despliega una nueva interfaz grafica para ingresar los siguientes datos: Usuario: Alarazboy. Dominio: http://www.youtube.com . Observación: sitio de videos. Contraseña: 123456 Repita contraseña: 123456 Se presiona el botón “Aceptar”. El sistema valida los datos correctamente y despliega la ventana para el ingreso de los datos de la frase de paso necesaria para cifrar la contraseña creada. Se ingresa la frase de paso: protegiendo pass Se presiona el botón “Aceptar”.
Resultado esperado	El sistema muestra un mensaje indicando que el registro ha sido creado correctamente. Se actualiza el llavero personal.
Evaluación de la prueba	Realizada y satisfactoria.

Tabla 5.7: Prueba de validación 39.

ID Caso de prueba	45
ID requerimientos	53 – 62
Nombre de caso de prueba	Copiar clave segura al portapapeles.
Descripción	Se copia una clave segura o contraseña, almacenada en uno de los registros, al portapapeles para posteriormente copiarla en el dominio asociado.
Condiciones de ejecución	Que se haya seleccionado desde la bandeja del sistema la opción: “Llavero personal”. Que exista, al menos, un registro.
Entrada	Se selecciona el registro con ID: 603DC8087B9985FC asociado al usuario alaracoboy1 del dominio http://www.youtube.com . Se selecciona la casilla contraseña. Se presiona el botón “Copiar al portapapeles”. El sistema despliega la interfaz gráfica para el ingreso de la frase de paso. Se ingresa la frase de paso: protegiendo pass Se presiona el botón “Aceptar”.
Resultado esperado	El sistema despliega un mensaje indicando que la clave ha sido descifrada y copiada al portapapeles. Se ingresa al dominio a través del navegador. Se pega en el campo correspondiente la contraseña.
Evaluación de la prueba	Realizada y satisfactoria.

Tabla 5.8: Prueba de validación 45.

Las pruebas de validación resultaron ser muy beneficiosas en el proceso de desarrollo e implementación de la herramienta de software, ya que permitieron encontrar errores en la especificación de requerimientos, lo que llevó al grupo de trabajo a incorporar nuevos requerimientos y a replantear otros ya existentes. A continuación, se presentan los errores más recurrentes en cada una de las iteraciones del modelo de ciclo de vida seleccionado.

En la primera iteración se encontraron errores, principalmente de validación de datos de entrada, que afectaban al formato de los archivos con los que trabaja el sistema y eran la causa del mal funcionamiento de algunas de sus funcionalidades, como es el caso del firmado digital de archivos, por nombrar alguna.

En la segunda iteración se presentaron conflictos con el sistema operativo Windows, el cual restringía el acceso a ciertos sectores del disco e impedía que ciertos archivos se vieran afectados por alguna de las operaciones criptográficas ofrecidas por la aplicación. La solución fue agregar nuevas excepciones para la detección de conflictos y errores de ejecución e informar por pantalla al usuario final, mediante ventanas de diálogo, del error producido.

En la tercera iteración no se presentan mayores problemas en la ejecución de los casos de prueba; sólo se destaca un mal funcionamiento al aplicar formatos al texto que estaba siendo digitado en el editor SICFA en ese momento.

Para una revisión más detallada de las pruebas de validación, véase el anexo digital: “Pruebas de validación SICFA”, que se entrega junto con este documento de memoria de título.

5.2 Pruebas de rendimiento

Prueba de rendimiento 1: cifrado de datos en SICFA-DTX235

La siguiente prueba de rendimiento tiene como objetivo medir los tiempos de CPU que toma cada uno de las operaciones de cifrado simétrico que realiza SICFA-DTX235. El experimento consistió en tomar 5 archivos de diferentes tamaños y extraer, para cada uno de los algoritmos de cifrado disponibles en SICFA-DTX235, 5 muestras de tiempo, las cuales son promediadas sacando un valor central para cada operación de cifrado (ver tabla 5.11).

Se debe tomar en cuenta que la operación de cifrado que se realiza arroja como salida un archivo binario y no uno con armadura ASCII, ya que estos últimos aumentan el tiempo de ejecución debido a que existe un proceso de conversión a caracteres imprimibles (codificación Base64).

Las características de la máquina donde se realizaron las pruebas; así como los tamaños de los archivos se muestran en la tabla 5.9 y 5.10 respectivamente.

Características	
Procesador	Intel core 2 duo
Modelo del procesador	T7250
Velocidad de procesador	2.0 Ghz
Memoria Ram	2.0 Gb
Sistema Operativo	Microsoft Windows XP
JVM	JDK 1.6

Tabla 5.9: Características de la máquina donde se realizaron las pruebas de rendimiento.

Archivo	Tamaño del archivo
Archivo 1	80 MB
Archivo 2	34,6 MB
Archivo 3	22,2 MB
Archivo 4	15,1 MB
Archivo 5	7,48 MB

Tabla 5.10: Descripción del tamaño de los archivos.

Algoritmo	A - 7,48 MB	A - 15,1 MB	A - 22,2 MB	A - 34,6 MB	A - 80 MB
Blowfish	903,2	1690,4	2599,8	3965,6	9887,4
TripleDES	2218,6	4481,2	6537,4	10228,2	25122
DES	1162,4	2369	3447	5468,8	12893,8
AES	940,8	1784,4	2662,4	4078,2	9134,2

Tabla 5.11: Resultados de las mediciones (mseg) de las operaciones de cifrado SICFA.

Algoritmo	A - 7,8 MB	A - 15,1 MB	A - 22,2 MB	A - 34,6 MB	A - 80 MB	Promedio	Desviación Estándar
Blowfish	120,7487	111,947	117,1081	114,6127	123,5925	117,6018	4,6622
TripleDES	296,6043	296,7682	294,4775	295,6127	314,025	299,4975	8,1725
DES	155,4011	156,8874	155,2703	158,0578	161,1725	157,3578	2,4212
AES	125,7754	118,1721	119,9279	117,8670	114,1775	119,1840	4,2374

Tabla 5.12: Estimaciones de tiempo (mseg) para el cifrado de 1 MB en SICFA-DTX235.

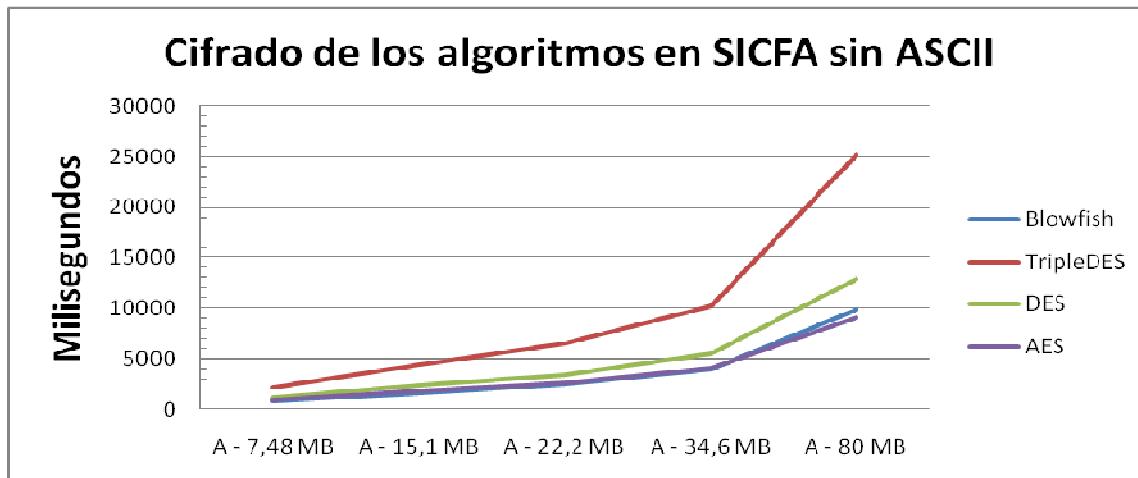


Figura 5.1: Gráfico de mediciones de las operaciones de cifrado SICFA-DTX235.

De los datos presentados en tabla 5.11 se puede decir que el resultado de las mediciones es el esperado, siendo Blowfish y AES los más rápidos, ambos con un tiempo de ejecución bastante parecido. Al contrario, TripleDES es el algoritmo más lento; esto se explica porque es un DES aplicado 3 veces. De la tabla 5.12, que muestra las estimaciones de tiempo para el cifrado de 1 Megabyte, se puede extraer que:

- El tiempo de cifrado para un algoritmo cualquiera es homogéneo, esto es, que independiente del tamaño del archivo, el costo de tiempo para cifrar 1 megabyte es relativamente el mismo, esto se puede apreciar de mejor manera al mirar el dato correspondiente al promedio y la desviación estándar para cada algoritmo.
- El tiempo de cifrado es directamente proporcional al tamaño del archivo, es decir, a mayor tamaño del archivo mayor es el tiempo que ocupa el cifrado, así lo muestra el gráfico de la figura 5.1 construido a partir de los datos de la tabla 5.11, el cual muestra las curvas con pendiente positiva de cada uno de los algoritmos utilizados en las operaciones de cifrado simétrico de SICFA-DTX235.

Prueba de rendimiento 2: descifrado de datos en SICFA-DTX235

Las pruebas de rendimiento realizadas a las operaciones de descifrado de SICFA-DTX235 permiten medir el tiempo de CPU (en milisegundos) que toma descifrar los archivos, con contenido cifrado, que se generaron en la prueba de rendimiento número 1. Al igual como en la prueba de rendimiento número 1, se tomaron 5 muestras para cada archivo cifrado con cada uno de los algoritmos disponibles. Por último, se forma una tabla

con los resultados finales y se construye un gráfico asociado. Las mediciones y los resultados finales se muestran a continuación.

Algoritmo	A - 7,48 MB	A - 15,1 MB	A - 22,2 MB	A - 34,6 MB	A - 80 MB
Blowfish	546,8	940,8	1349,8	2059	4887,2
TripleDES	1978,2	3821	5622	8928,2	20234,2
DES	865,6	1640,4	2340,8	3599,8	8409,4
AES	500,2	912,6	1318,8	1965,4	4690,6

Tabla 5.13: Resultados de las mediciones (mseg) de las operaciones de descifrado SICFA.

Algoritmo	A - 7,8 MB	A - 15,1 MB	A - 22,2 MB	A - 34,6 MB	A - 80 MB	Promedio	Desviación Estándar
Blowfish	70,1026	62,3046	60,8018	59,5087	61,09	62,7615	4,2224
TripleDES	253,6154	253,0464	253,2432	258,0405	252,9275	254,1746	2,1768
DES	110,9744	108,6358	105,4414	104,0405	105,1175	106,8419	2,8757
AES	64,1282	60,4371	59,4054	56,8035	58,6325	59,8813	2,7204

Tabla 5.14: Estimaciones de tiempo (mseg) para el descifrado de 1 MB en SICFA-DTX235.

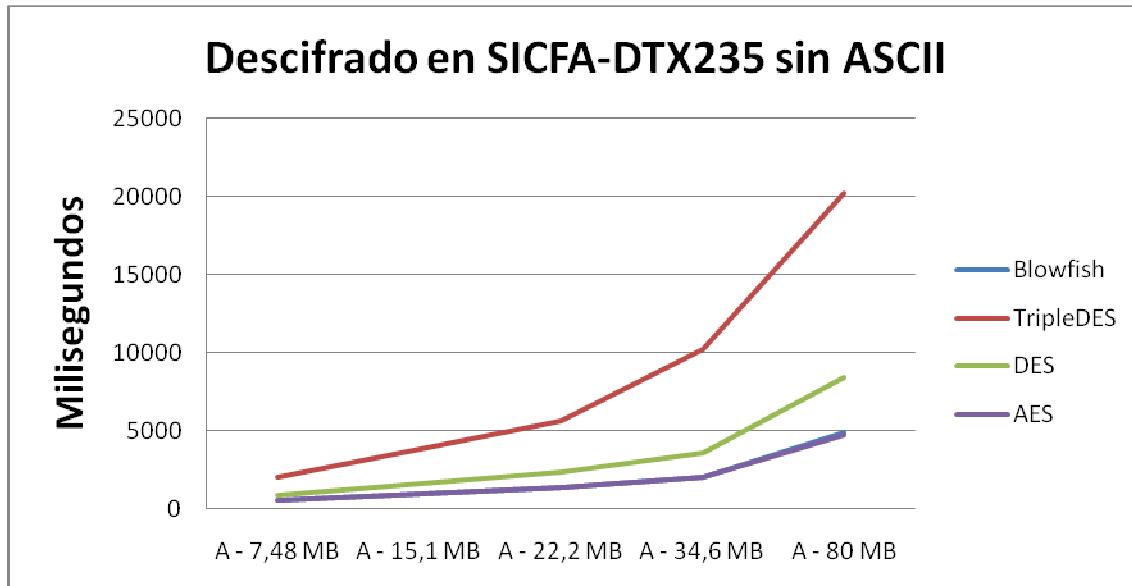


Figura 5.2: Gráfico de mediciones de las operaciones de descifrado SICFA-DTX235.

De acuerdo a los datos de la tabla 5.13, se puede apreciar que la velocidad de descifrado para cada uno de los algoritmos, Blowfish, TripleDES, DES y AES, sigue la misma tendencia que los resultados de la prueba de rendimiento para las operaciones de cifrado, en donde los algoritmos Blowfish y AES son los más rápidos. A partir de los datos de la tabla 5.14 se puede notar que el tiempo de descifrado es homogéneo y directamente proporcional al tamaño del archivo, lo que también se puede ver al observar las pendientes positivas de las curvas del gráfico construido con los datos de la tabla 5.13 y presentado en la figura 5.2.

Como resultado, al momento de observar los datos de las tablas 5.11 y 5.13, es posible apreciar que el tiempo de descifrado es menor que el tiempo de cifrado para un mismo archivo, alrededor de un 56% del tiempo de cifrado toma el descifrado para el caso de Blowfish, 85% para TripleDES, 68% para DES y un 50% para AES.

Prueba de rendimiento 3: generación de valores hash MDC en SICFA-DTX235

Las pruebas de rendimiento realizadas a las funciones de integridad MDC de SICFA-DTX235, permiten medir el tiempo de CPU (en milisegundos) que toma en obtener un resumen de bits asociado a un archivo. Al igual como en las pruebas anteriores se usaron los mismos archivos, en donde a cada uno, se le tomaron 5 muestras para cada uno de los algoritmos Hash disponibles para generar resúmenes MDC. Por último, se forma una tabla con los resultados finales y se construye un gráfico asociado. Las mediciones y los resultados finales se muestran a continuación.

Archivos	Tamaño archivo en Kilobytes	Tiempo ejecución algoritmos (mseg)		
		MD5	SHA-1	SHA-256
Archivo 80,0(MB)	82.023	940,4	1673,2	2397
Archivo 34,6(MB)	35.479	418,8	787,4	1121,8
Archivo 22,2(MB)	22.738	259,6	515,6	750
Archivo 15,1(MB)	15.507	162,8	347,2	537,6
Archivo 7,48(MB)	7.665	109,2	153	318,6

Tabla 5.15: Resultados de mediciones de la generación de resúmenes MDC SICFA-DTX235.

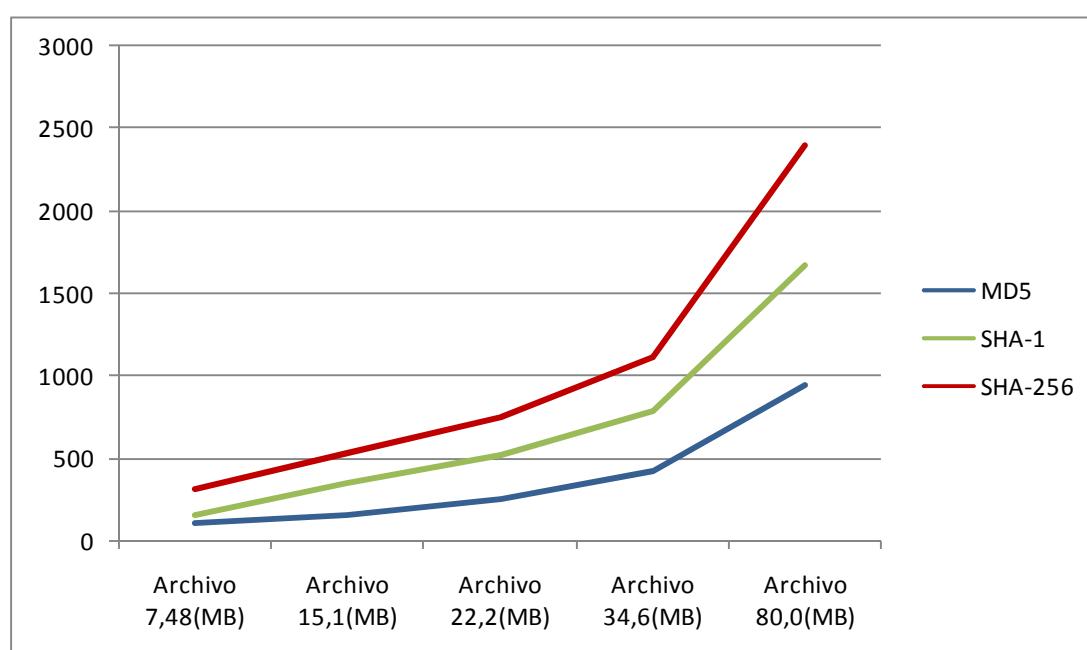


Figura 5.3: Gráfico de mediciones de la generación de resúmenes MDC SICFA-DTX235.

De los datos de la tabla 5.15, la cual contiene los resultados de las mediciones hechas a los algoritmos Hash con los cuales se generan los resúmenes MDC, se puede observar que MD5 es más rápido que SHA-1 y SHA-256, esto se explica porque MD5, para generar los compendios de mensajes, realiza 64 (4x16) operaciones frente a las 80 (4x20) operaciones de SHA-1.

De la figura 5.3, que muestra el gráfico construido a partir de los datos de la tabla 5.15, se puede concluir que al igual que en las operaciones de cifrado y descifrado, el tiempo para generar un resumen o compendio de mensaje es proporcional al tamaño del archivo, así lo muestran las pendientes positivas de cada una de las curvas, y esto se debe a que los algoritmos Hash procesan bloques de mensajes de 512 bits, sobre los cuales aplican operaciones matemáticas. Por lo tanto, a mayor tamaño de archivo mayor será la cantidad de bloques que se deben procesar.

Prueba de rendimiento 4: carga de registros al llavero personal y recuperación de claves seguras o contraseñas en SICFA-DTX235

Las pruebas de rendimiento realizadas al llavero personal de SICFA-DTX235 permiten medir el tiempo de CPU (en milisegundos) que toma en cargar registros al llavero personal (estructura de datos), desde un archivo digital. Además, se mide el tiempo promedio de recuperación de una clave segura o contraseña desde la estructura de datos. Las mediciones y los resultados finales se muestran a continuación.

Cantidad de registros	Tiempo carga registros en milisegundos	Tiempo carga registros en segundos	Tiempo carga contraseña al portapapeles (en mseg)	Tiempo carga contraseña al portapapeles (en seg)
1	5,4	0,0054	4,2	0,0042
10	21	0,021	4,6	0,0046
20	31,2	0,0312	4,8	0,0048
30	46,4	0,0464	5,1	0,0051
40	51,6	0,0516	4,6	0,0046
50	66,4	0,0664	4,6	0,0046
60	80,4	0,0804	4,9	0,0049
70	96,4	0,0964	4,2	0,0042
80	111,4	0,1114	4,9	0,0049
90	127,8	0,1278	4,7	0,0047
100	153,6	0,1536	5,5	0,0055

Tabla 5.16: Resultados de mediciones de la carga de registros en el llavero personal.



Figura 5.4: Gráfico de mediciones de la carga de registros en el llavero personal.

De los datos presentados en la tabla 5.16, se obtiene que el promedio de carga de un registro, que incluye la lectura desde el archivo hasta la inserción en la estructura de datos, es de aproximadamente de 0,00185 seg (1,85 mseg).

De la figura 5.4, que muestra el gráfico construido a partir de los datos de la tabla 5.16 correspondientes al tiempo de carga de los registros, se puede apreciar que la curva tiene pendiente positiva y casi constante en todo momento, lo cual implica que el tiempo de carga es proporcional a la cantidad de registros contenidos en el archivo, es decir, que a mayor cantidad de registros, mayor será el tiempo de carga en la estructura de datos del llavero personal.

Capítulo 6| Conclusiones

Al momento de culminar esta memoria de titulación es importante resaltar aquellas experiencias que para los autores son importantes de mostrar y dar a conocer; esto es, presentar aquellos aspectos que resultaron interesantes en el desarrollo de este trabajo ya sea en la investigación sobre el tema o en el desarrollo de la aplicación y trabajos futuros que se le puedan hacer a SICFA-DTX235.

Con respecto a la investigación

Hasta mediados del 2007 el grupo de trabajo no encontró información abundante sobre seguridad informática y criptografía. En Internet se podían encontrar sólo algunos papers que discutían algunos temas de seguridad. A partir de fines del 2007 se observó un crecimiento de información disponible, lo que lleva a pensar que en ese periodo el tema de seguridad sufrió un auge y por consiguiente hoy en día es un asunto que concierne a todo el mundo y sobre todo a los expertos en computación.

El estudio sobre seguridad y sobre todo el de criptografía, no fue un proceso fácil, ya que implica otra ciencia como las matemáticas en la cual se sustenta. Resulta importante indicar que el aprendizaje total de la materia es imposible de conseguir en una memoria de título debido a la gran cantidad de temas que involucra, pero sí resultó posible, elaborar estrategias que nos permitieron clasificar y enfocarnos en aquellas materias que tienen mayor relevancia.

Sobre seguridad informática se puede decir, que llama la atención la cantidad de amenazas (intencionales y otras no) a la que están sometidos los sistemas informáticos actualmente, algunas desconocidas e impensadas para la mayoría de las personas. También, resulta asombroso que una gran parte de las amenazas están sostenidas en el descuido de las personas y en el desconocimiento que éstas tienen de los potenciales peligros a los que se enfrentan con el sólo hecho de manejar información a través de una computadora. Por otra parte, existe un conjunto de mecanismos de seguridad (prevención, detección y recuperación) que luchan día a día para hacer frente a tan diverso espectro de estrategias inescrupulosas. Toda esta guerra entre amenazas y seguridad se ha dado por el constante crecimiento tecnológico, por lo tanto, se ha vuelto un procedimiento repetitivo en el tiempo, lo que implica que cuando hay un avance tecnológico sirve tanto para atacantes como para los expertos en seguridad. Por esta razón la seguridad informática no es un producto como tal, sino que es un proceso que involucra un desarrollo constante y compromete tanto a organizaciones como a personas en un trabajo de concientización (véase figura 6.1) de los peligros y riesgos, estandarización de los protocolos y del manejo de información, actualización continua de los mecanismos de seguridad y estudio de las nuevas amenazas que van apareciendo.

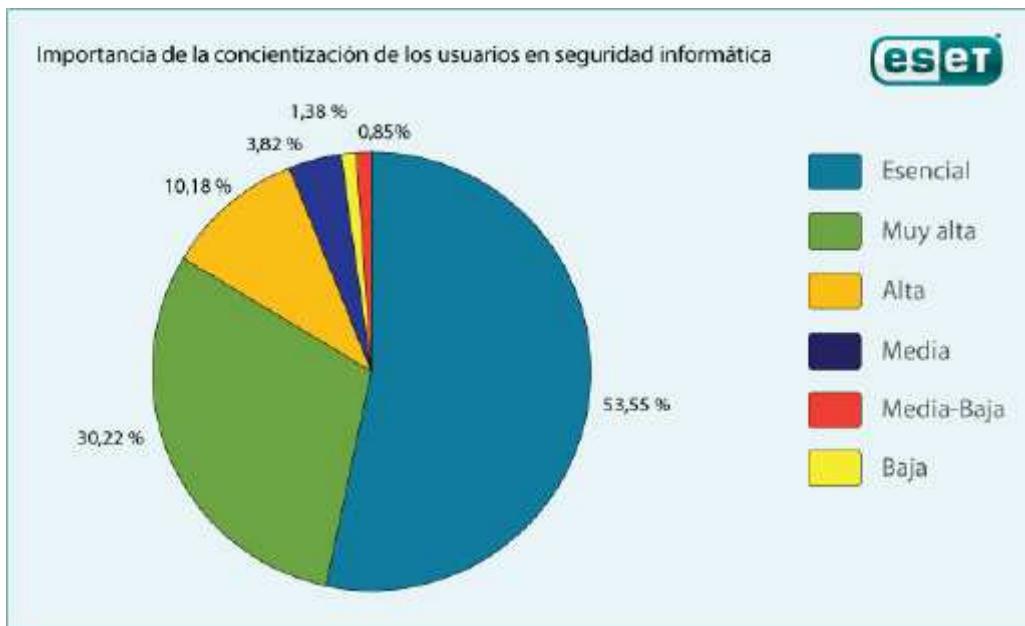


Figura 6.1: Importancia de la concienciación de los usuarios en seguridad informática.
Encuesta por Eset a 947 profesionales de seguridad informática.

La criptografía resultó ser un tema apasionante y a la vez difícil de estudiar. Resulta curioso que en sus inicios y hasta mediados del siglo XX sólo era usada con fines militares lo que indica que desde tiempos remotos el hombre ha tenido la necesidad de ocultar o mejor dicho de proteger información. También es importante señalar que la criptografía por su naturaleza siempre será objeto de discusión en las naciones del mundo, ya que en ciertos casos (Francia y E.E.U.U), es tratada como arma terrorista, por ende estará sujeta a restricciones. Pensar en prohibir la criptografía es irrisorio, restringirla de acuerdo al tamaño de la clave para cifrados fuertes sería más factible, pero con lo avanzada que está la tecnología y los múltiples métodos en que pueden ser distribuidos y luego implementados los algoritmos, ninguna de las opciones sería eficaz.

Ahora, hablando de los sistemas criptográficos, como son: los criptosistemas simétricos y asimétricos; podemos mencionar que los primeros basan su seguridad en el largo de clave, considerada actualmente de 128 bits. Los segundos se sustentan sobre problemas matemáticos difíciles de tratar como la factorización de números grandes y el problema del logaritmo discreto. Ambos criptosistemas, simétricos y asimétricos tienen un factor en común y es que son computacionalmente intratables para las condiciones actuales, pero esto lleva a hacerse la siguiente pregunta ¿qué pasaría si hubiese un gran avance que permitiera a un computador aumentar muchas veces su capacidad de procesamiento?

Actualmente se está trabajando en los llamados sistemas cuánticos, los cuales en un futuro permitirán el aumento de la capacidad de proceso de los computadores en forma considerable, reduciendo así, el tiempo que se tomaría en encontrar una clave en un sistema de cifrado simétrico, por ejemplo. Entonces, no es descabellado pensar en buscar nuevas técnicas criptográficas que puedan hacer frente a los cambios que se avecinan. Una de las últimas innovaciones que se han hecho en criptografía (véase [WEB-16]), es aprovechar las propiedades de la física cuántica para implementar lo que se conoce como criptografía cuántica, pero lamentablemente hay indicios que este tipo de sistema ya ha sido roto. (véase [WEB-19]).

Con respecto al desarrollo del software

El proceso de desarrollo de SICFA-DTX235 resultó gratamente satisfactorio ya que la meta y todos los objetivos se cumplieron en su totalidad. El ciclo de vida incremental permitió tener siempre un software funcional desde la primera iteración e ir añadiendo nuevos requerimientos a los ya establecidos en el anteproyecto de memoria, por ende incorporando nuevas funcionalidades.

Un único problema al utilizar el ciclo de vida iterativo cascada incremental, es el hecho de que no incorpora una etapa de análisis de riesgos, lo que hubiera sido de gran utilidad al momento de enfrentar de manera adecuada los distintos problemas (personales y de trabajo) a los que se vio expuesto el equipo de desarrollo al momento de desarrollar este proyecto.

Al culminar este trabajo, se logró obtener una aplicación de seguridad con las siguientes características:

- Cifrado, firmado y funciones de integridad para cualquier tipo de archivo, independiente de su extensión.
- Portabilidad de la aplicación. SICFA-DTX235 puede ser usado en cualquier plataforma ya que esta desarrollado en el lenguaje de programación Java.
- Cifrado, firmado y funciones de integridad de archivos optimizadas.
- Portabilidad de las claves. Permite transportar las claves y usarla en diferentes computadoras que tengan instalado SICFA-DTX235.
- ASCII. El uso de armaduras ASCII en los archivos con contenido cifrado permite la portabilidad, ya que es posible enviar el contenido por medios de comunicación como correo electrónico y MSN.

Mejoras a SICFA-DTX235 y trabajos futuros

En este apartado se dan a conocer algunas mejoras e ideas que pueden ser consideradas para futuras versiones de SICFA-DTX235, todo con el objetivo de mejorar la aplicación.

Implementaciones propias de algoritmos criptográficos

El hecho de que SICFA-DTX235 utilice algoritmos implementados por terceros (JCA y JCE), para desarrollar las distintas técnicas criptográficas, ha traído varios beneficios, como por ejemplo, el ahorro de tiempo en implementación y el posterior testeo de éstos. Pero, si bien estos algoritmos han sido de gran utilidad, también hay que estar sujeto a las restricciones que éstos tienen y que en este caso tienen relación con la limitación de la longitud de clave para los algoritmos simétricos, la cual actualmente no sobrepasa los 128 bits.

De acuerdo a las razones antes mencionadas, un avance importante para el desarrollo de la aplicación, seria la implementación propia de los algoritmos para SICFA-DTX235, evitando con esto, la dependencia y las limitaciones actuales, al hacer uso de proveedores de algoritmos criptográficos.

Certificados digitales en SICFA-DTX235

La utilización de certificados digitales supondría una utilidad adicional en la firma de archivos, ya que daría la opción a los usuarios de firmar utilizando certificados. Hay que recordar que Java da soporte para los certificados x509, por ende la implementación de éstos dentro de SICFA-DTX235 sería viable.

Implementar una herramienta para envío de correo seguro

Esta idea contempla la implementación de un plug-in para aplicaciones de correo electrónico, que permita al usuario crear mensajes cifrados y/o firmados digitalmente y enviarlos a una dirección de correo electrónico (Hotmail, Yahoo, Gmail, etcétera), pudiendo también adjuntar algún tipo de archivo, el cual también podría estar afecto a alguna operación criptográfica.

Editor SICFA-DTX235

Actualmente el contenido cifrado, firmado/cifrado del editor de SICFA se almacena en un archivo digital. Sería una buena idea mostrar el contenido cifrado en una ventana para poder dar la opción al usuario de copiarlo o cortarlo para enviarlo por correo electrónico a algún destino aprovechando la armadura ASCII.

Como palabras finales se puede decir que sí es posible construir una aplicación de seguridad basada en técnicas criptográficas, ya que se cuenta con todas las herramientas para hacerlo. Un trabajo serio de investigación, permitió identificar las mejores técnicas y algoritmos, que implementan los mecanismos para garantizar los servicios básicos de seguridad, que se formularon como meta para este proyecto de titulación. Los conocimientos adquiridos a lo largo de la carrera, posibilitaron la planificación del proyecto y la clasificación de las técnicas más apropiadas para el desarrollarlo del software. La plataforma Java, usada para la codificación de la aplicación y, la experiencia que se tiene en ella, facilitó enormemente este proceso.

Glosario

Algoritmo criptográfico: Es una función matemática usada para cifrado y descifrado (generalmente, hay dos funciones relacionadas: una para cifrado y otra para descifrado). Si la seguridad del algoritmo reside en el no conocimiento de este, entonces se dice que es un algoritmo privado. En el otro caso si el algoritmo es conocido por todo el mundo se denominada “algoritmo público”.

Archivo de claves: Conjunto de claves. Cada usuario tiene dos tipos de archivos de claves: un archivo de claves públicas y un archivo de claves privadas.

Archivo de claves privadas: Conjunto de una o más claves privadas, todas ellas pertenecientes al propietario del archivo.

Archivo de claves públicas: Conjunto de claves públicas. Su archivo de claves públicas contiene además su(s) propia(s) clave(s) pública(s).

Autenticación de origen de datos: Comprobación de que la fuente de los datos recibidos es la afirmada. (ISO 7498-2).

Cifrado [encryption]: Método de modificar la información para hacerla ilegible para todos menos para el destinatario deseado, quien debe descifrarla para poder leerla.

Cifrado convencional [conventional encryption]: Cifrado que se basa en una contraseña común, en vez de criptografía de clave pública. El archivo se cifra usando una clave secreta de sesión, que a su vez es cifrada usando una contraseña que se le pedirá elija.

Confidencialidad: Propiedad de la información que hace que ésta no sea disponible o descubierta a individuos, entidades o procesos no autorizados. (ISO 7498-2).

Criptografía de clave pública [public-key cryptography]: Criptografía en la que se utiliza una pareja de claves, compuesta por una clave pública y una clave privada, y que no necesita seguridad en el canal de comunicación en sí.

Criptograma: Cualquier información que se encuentre convenientemente cifrada y no resulte legible ni comprensible más que para el destinatario legítimo de la misma.

Descifrado [decryption]: Método de recomposición de la información cifrada para volverla legible de nuevo. Para descifrar se usa la clave privada del destinatario.

Huella [fingerprint]: Cadena identificadora única, de números y caracteres, usada para autenticar claves públicas. Esta es la manera principal de comprobar la autenticidad de una clave.

Huella de clave [key fingerprint]: Cadena identificadora única, de números y caracteres, usada para autenticar claves públicas. Por ejemplo, usted puede llamar por teléfono al propietario de una clave pública y pedirle que le lea la huella asociada a esa clave de modo que usted pueda compararla con la huella de su copia de esa clave pública para ver si coinciden. Si la huella no coincide, entonces usted sabrá que tiene una clave falsa.

Identificador [ID] de clave [key ID]: Código legible que identifica de manera única a una pareja de claves. Dos parejas de claves pueden tener el mismo ID de usuario, pero tendrán diferentes IDs de clave.

Identificador [ID] de usuario [user ID]: Código legible que identifica de manera única a un usuario. El ID de usuario ayuda a los usuarios a identificar al propietario de una pareja de claves.

Integridad de datos: Propiedad de los datos que garantiza que éstos no han sido alterados o destruidos de modo no autorizado. (ISO 7498-2).

Pareja de claves: Clave pública y su clave privada complementaria. En sistemas de criptografía de clave pública como SICFA-DTX235, cada usuario tiene como mínimo una pareja de claves.

Repudio: Denegación, por una de las entidades implicadas en una comunicación, de haber participado en todo o parte de dicha comunicación. (ISO 7498-2).

Servicio de seguridad de la información: Método para proveer algún aspecto específico de seguridad. Por ejemplo, la integridad de los datos es un objetivo de seguridad, y un método para asegurar este aspecto es un servicio de seguridad de la información.

Texto con armadura ASCII [ASCII-Armoured Text]: Información binaria que ha sido codificada utilizando un juego de caracteres normales e imprimibles ASCII, por su conveniencia para transportar la información a través de sistemas de comunicaciones.

Texto claro [plaintext]: Cualquier información que resulte legible y comprensible por sí misma. Un texto claro sería cualquier información antes de ser cifrada o después de ser descifrada. Se considera que cualquier información es vulnerable si se encuentra en este estado.

Verificación [verification]: Acto de comparar una firma creada con una clave privada usando la clave pública complementaria. La verificación demuestra que la información fue enviada realmente por el firmante, y que posteriormente el mensaje no ha sido alterado por nadie.

Bibliografía

- [ARC-02] Arcert, Coordinación de Emergencia en Redes Teleinformáticas.
Manual de Seguridad en Redes, 2000.
- [CAB-03] Caballero Gil, Pino.
Introducción a la criptografía.
Ra-Ma, 2003.
- [BRA-99] Braicovich, Gustavo Ariel.
Firma digital.
Universidad Nacional del Comahue, Argentina, 1999.
- [DIH-76] Diffie Whitfield, Hellman Martin E.
New Directions in Cryptography, IEEE Transactions on Information Theory.
vol. IT-22, 1976, páginas 644-654.
- [ELG-85] ElGamal, Taher.
A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms, IEEE Transactions on Information Theory.
vol. IT-31, 1985, páginas 469-472.
- [FHM-01] Fúster Amparo, Hernández Luis, Montoya Fausto, Muñoz Jaime.
Técnicas Criptográficas de protección de datos.
Ra-Ma, 2001.
- [GOM-07] Gómez Vieites, Álvaro
Enciclopedia de la seguridad informática.
Ra-Ma, 2007.
- [GLO-05] García Eugenio, López Miguel Ángel, Ortega Jesús.
Una introducción a la criptografía.
Universidad de Castilla, España, 2005.
- [GLP-02] García Carlos, López Josep, Puigjaner Dolors.
Matemática discreta.
Prentice Hall, 2002.
- [LOP-07] Lopéz Hernández, Fernando.
Seguridad, Criptografía y comercio electrónico con Java.
Macprogramadores, 2007.
- [LRG-06] Lehtinen Rick, Russell Deborah y Gangemi G.T.
Computer Security Basics, Second Edition.
O'Reilly Media, 2006.
- [LUC-09] Lucena López, Manuel José.
Criptografía y seguridad en computadores, versión 4-0.7.53.
Universidad de Jaén, España, 2009.

- [MCM-07] Martos Navarro, Calvo Prieto Jesus, Muñoz Labiano Álvaro.
Cuerpo Técnicos Auxiliares de Informática de la Administración del Estado.
CEP Editorial, 2007.
- [MEN-97] Menezes Alfred, Van Oorschot Paul, Vanstone Scott.
Handbook Applied Cryptography.
CRC Press LLC. 1997
- [MIT-07] Mitnick Kevin D, Simon William L.
El arte de la intrusión.
Ra-ma, 2007.
- [MOG-07] Mogollon, Manuel.
Cryptography and Security Services: Mechanisms and Applications.
Cybertech Publishing, 2007.
- [MOR-02] Morales Vázquez, José María.
SSL y Otros Protocolos Seguros para el Comercio Electrónicos.
Universidad Politécnica de Madrid, España, 2002
- [MOR-05] Moreno, Salvador.
Algoritmo RSA.
Universidad de Valencia, 2005.
- [NOR-03] Instituto Nacional de Normalización, INN – Chile.
Tecnología de la Información – Código de prácticas para la gestión de
seguridad de la información, 2003.
- [PAL-02] Palacios, Manuel.
Grupos, anillos y cuerpos.
Universidad de Zaragoza, 2002.
- [PFL-07] Pfleeger, Charles.
Security in Computing.
Pearson Education, 2007.
- [PON-00] Ponts Martorell, Manuel.
Control de accesos.
Escuela Universitaria Politécnica de Mataró, España, 2000.
- [PRE-05] Pressman, Roger S.
Ingeniería del Software, un enfoque práctico.
McGraw-Hill, 2005.
- [RAM-06] Ramió Aguirre, Jorge.
Libro Electrónico de seguridad informática y criptografía, versión 4.1
Universidad Politécnica de Madrid, España, 2006.
- [RAM-99] Ramió Aguirre, Jorge.
Criptosistemas clásicos.
Universidad Politécnica de Madrid, España, 1999.

- [RAS⁺-07] Richardson W. Clay, Avondolio Donald, Schrager Scot, Mitchell Mark and Scanlon Jeff.
Profesional Java JDK 6 Edition.
Wiley Publishing Ltd. 2007
- [SCH-96] Schneier, Bruce.
Applied Cryptography, Second Edition: Protocols, Algorithms.
John Wiley & Sons Ltd. 1996.
- [SHA-48] Shannon, Claude E.
A Mathematical Theory of Communications, Bell System technical Journal,
vol. 27, 1948, páginas 379-423, 623-656.
- [SHA-49] Shannon, Claude E.
Communication Theory of Secrecy Systems, Bell System Technical Journal,
vol. 28, 1949, páginas 656-715.
- [TAN-97] Tanenbaum, Andrew S.
Redes de Computadoras, Tercera edición.
Pearson, 1997.
- [VIL-02] Villalón Huerta, Antonio.
Seguridad en Unix y Redes, versión 2.1.
GNU Free Documentation, 2002.
- [YOU-03] Young Rhee, Man.
Internet Security-Cryptographic principles, algorithms and protocols.
John Wiley & Sons Ltd. 2003
- [ZUK-05] Zukowski, John.
The Definitive Guide to Java Swing, Third Edition.
Apress, 2005.
- [WEB-01] <http://foro.elhacker.net>
Seguridad informática y criptografía.
- [WEB-02] <http://rijmenants.blogspot.com>.
Información histórica y técnica sobre máquinas criptográficas.
- [WEB-03] <http://www.kriptolis.org>
Criptografía, privacidad en Internet.
- [WEB-04] <http://seguridad-information.blogspot.com>
Blog dedicado al estudio de la seguridad de la información.
- [WEB-05] <http://java.sun.com/javase/6/docs/technotes/guides/security>
Documentación de la arquitectura criptográfica de Java.
- [WEB-06] http://es.wikipedia.org/wiki/Seguridad_informática
Conceptos sobre seguridad informática.

- [WEB-07] <http://www.lpsi.eui.upm.es/SInformatica/SInformatica.htm>
Conceptos sobre seguridad informática.
- [WEB-08] http://es.wikipedia.org/wiki/NetBeans_IDE
Información sobre el IDE de programación NetBeans.
- [WEB-09] http://netbeans.org/index_es.html
Página oficial de NetBeans.
- [WEB-10] <http://www.galeon.com/rc4>
Información sobre el algoritmo RC4.
- [WEB-11] <http://www.openssl.org/>
Página oficial de OpenSSL.
- [WEB-12] <http://www.visual-paradigm.com/product/vpuml/>
Página oficial de Visual Paradigm.
- [WEB-13] <http://alerta-antivirus.red.es/seuridad>
Actualidad en seguridad informática.
- [WEB-14] <http://maestros.unitec.edu/~efutch/prolog-AI-ponencia-v2.pdf>
Información sobre Inteligencia Artificial.
- [WEB-15] <http://news.bbc.co.uk/2/hi/technology/8552410.stm>
BBC News – Mapping the growth of the internet
- [WEB-16] <http://www.textoscientificos.com/criptografia/cuantica>
Conceptos sobre criptografía cuántica.
- [WEB-17] <http://www.gnupg.org/>
Página oficial de GPG.
- [WEB-18] <http://java.sun.com/docs/books/jls/>
The Java Language Specification.
- [WEB-19] <http://www.cienciakanija.com/2010/05/17/hackeado-un-sistema-comercial-de-criptografia-cuantica/>
Página de noticias de seguridad informática.

Anexos

1. Especificación de Requerimientos.
2. Informe Casos de uso SICFA.
3. UML SICFA Diagramas casos de uso.
4. Diagramas de secuencias.
5. Esquemas subsistema MS.
6. Manual de usuario HTML.
7. Pruebas de validación SICFA.
8. Anexo digital Seguridad Informática.
9. Anexo digital Criptografía Moderna.
10. Anexo digital Criptografía Clásica.
11. CD-Rom con software SICFA-DTX235.