APLICACIÓN DE FIRMA DIGITAL

FABIO DURAN ORTIZ

MILTON EDUARDO HERNANDEZ ZAKZUK

CORPORACION UNIVERSITARIA TECNOLOGICA DE BOLIVAR

FACULTAD DE INGENIERIA DE SISTEMAS

CARTAGENA D.T y C

2000

APLICACIÓN DE FIRMA DIGITAL

FABIO DURAN ORTIZ

MILTON EDUARDO HERNANDEZ ZAKZUK

Trabajo Presentado Como Requisito Para Optar Por El Titulo De Ingeniero De Sistemas

Director
GIOVANNY VASQUEZ
Ingeniero De Sistemas

CORPORACION UNIVERSITARIA TECNOLOGICA DE BOLIVAR

FACULTAD DE INGENIERIA DE SISTEMAS

CARTAGENA D.T y C

2000

Cartagena, noviembre 29 de 1999

Señores
Comité de proyecto de grado
Facultad de Ingeniería de Sistemas
Corporación Universitaria Tecnológica De Bolívar

Respetados señores

Por medio de la presente, le presento el proyecto de grado presentado por los alumnos **Milton Eduardo Hernández Zakzuk** y **Fabio Durán Ortíz**, que lleva por titulo **Aplicación De Firma Digital** para que sea estudiado y aprobado.

Cordialmente

_____

Giovanny Vásquez

Cartagena, noviembre 29 de 1999

Señores

Comité de proyecto de grado

Facultad de Ingeniería de Sistemas

Corporación Universitaria Tecnológica De Bolívar

Apreciados señores

La presente es para someter a vuestra consideración el proyecto de grado titulado **"Aplicación de firma digital"**, para que mediante su aprobación podamos optar por el titulo de ingeniero de sistemas.

Cordialmente

_____

Milton Eduardo Hernández Zakzuk

_____

Fabio Durán Ortíz

Cartagena, noviembre 29 de 1999

Ingeniero

Juan Carlos Mantilla

Decano de la Facultad de Ingeniería de Sistemas

Corporación Universitaria Tecnológica De Bolívar

Apreciado Ingeniero

Por medio de la presente le hacemos entrega formal del proyecto de grado titulado **"APLICACIÓN DE FIRMA DIGITAL"** para su aprobación.

Cordialmente

_____

Milton Eduardo Hernández Zakzuk

_____

Fabio Durán Ortíz

La institución se reserva el derecho de propiedad e intelectual de todos los trabajos de grado aprobados, los cuales no pueden ser explotados comercialmente sin su aprobación.

NOTA DE ACEPTACION

_____

_____

_____

_____

Presidente del jurado

_____

Calificador

_____

Calificador

Cartagena, 29 de Noviembre de 1999

AGRADECIMIENTOS

A Dios por haberme dado la sabiduría para realizar este trabajo.

A mis padres por haber creído y apoyado en mis estudios.

A mi amigo y compañero Fabio por estar siempre apoyándome.

A Giovanny nuestro director por habernos dirigido, guiado y acompañado desde el inicio hasta la culminación del proyecto.

MILTON

AGRADECIMIENTOS

A todas la personas que directa o indirectamente se vieron involucradas en el desarrollo de este proyecto.

A Giovanny nuestro director por habernos dirigido, guiado y acompañado desde el inicio hasta la culminación del proyecto.

A Gustavo Ortíz por brindarme su apoyo y comprensión.

A mis padres por su apoyo incondicional y su gran esfuerzo para que este resultado se diera.

FABIO

DEDICATORIA

Este trabajo lo dedico a Dios y a mis padres por haberme traído a este mundo.

MILTON

DEDICATORIA

A Dios . . .

A mis padres . . .

A mis tíos y Abuelos . . .

FABIO

CONTENIDO

PAG

LISTA DE FLUJOS

LISTA DE ALMACENES DE DATOS

LISTA DE ESTRUCTURAS DE DATOS

LISTA DE PROCESOS

LISTAS DE FIGURAS.

LISTA DE ANEXOS

RESUMEN

La aplicación Firma Digital es una herramienta desarrollada con el fin de brindar el servicio de privacidad y autenticidad ó únicamente autenticidad según sea el caso, todo esto aplicado a documentos digitales.

A través de esta aplicación una persona puede firmar y cifrar o solo firmar documentos, que luego se pueden enviar a otros usuarios por medio de una red utilizando una aplicación de correo, también sirve para aclarar o revisar documentos firmados y cifrados o solo firmados, recibidos. Para su utilización se necesita un par de claves, publica y privada que debe ser suministrada por una autoridad certificada.

Para su implementación se utilizaron algoritmos utilizados en criptosistemas criptográficos como son: LUC criptosistema de clave publica, IDEA criptosistema de clave privada y RIPEMD-160 como función Hash o resumen. Los cuales se escogieron por presentar la misma o más seguridad que los demás algoritmos de igual categoria, y además por no presentar problemas de patente.

Según el diseño criptográfico para firmar un documento primero se utiliza la función Hash RIPEMD—160 para generar un resumen de 160-bits (firma), y este se cifra con la clave privada del emisor. A su vez para el proceso de revisado de firma, se utilizara la clave publica del emisor para descifrar el resumen cifrado y luego se generara un nuevo resumen del mensaje recibido (sin la firma), y se comparan los dos resúmenes obtenidos y si estos son iguales la firma será la correcta.

El proceso de firmado y cifrado diseñado funciona de la siguiente manera, primero se toma el mensaje firmado y lo cifra utilizando el algoritmo IDEA, con una clave de sesión generada, luego esta clave de sesión es cifrada con Lucas usando la clave publica del receptor. Para su desencripción el receptor necesitará su clave privada, para desencriptar la clave de sesión cifrada, con la cual se descifra el mensaje firmado y cifrado obteniendo solo el mensaje firmado, al cual se le aplica el mismo proceso de revisado de firma

SUMARY

Digital asignature aplication is a tool development to give a service of privacity and autenticity or only autenticity that depends the case, all applied to digitals documents.

Throught this application a person can sign and cipher or only sign documents, that then you cansend to other user throught a net using a mail application clear or ilustrate signed documentsand ciphered or only signed, received too. For its use you need a pair of keys, public and private that must be suministrated for a certificated autority.

For its implementation we used algoritms employed in cripto systems criptographics like: LUC criptosystem public key, IDEA and RIPEMD-160 like a funtion Hash or sumary.

Theywere choosen to present problems of patent according to criptographic design to sign a document, first we use the funtion Hash RIPEMD-160 to generate a summary of 160 bits (signaturre), and we cipher with the private key of checking of sign, we're going to use the public key of the emitty to decipher the ciphered summary and then it will generate a new summary of the received

message (without the sign) after that, we compare both obtained summaries and if they are the same the sign is the correct.

The process of signed and ciphered woncs the following forms: first, take the signed message and cipher it using the algoritmi IDEA, with a key of generated sesión, then this key of sesión is cipherred

INTRODUCCION

La firma manual impregnada en un documento, indica que una persona manifiesta su conformidad y aceptación del contenido de este, por lo cual debe ser casi imposible de falsificar.

La firma digital, se podría decir que es la “versión computarizada de la firma manual”, e intenta proporcionar los mismos servicios que una firma manual, esto se realiza a través de algoritmos criptográficos seguros.

Remontándonos a los años cincuenta, se observa que los volúmenes de información en el mundo han aumentaron de forma exponencial, por lo que se inventaron métodos más rápidos y precisos para poder administrarla; Es aquí donde aparecen las primeras computadoras y con esto la comunicación entre ellas (REDES DE COMPUTADORAS), que hasta nuestros días sigue siendo una tecnología en crecimiento y hace parte esencial de nuestra educación. Con el desarrollo de esta tecnología, se crean una serie de servicios bastantes funcionales para la vida moderna y, algunas, como el correo electrónico, forman la cotidianidad de las organizaciones.

La Corporación Universitaria Tecnológica De Bolívar no podría ser la excepción a estos adelantos tecnológicos, por esto cuenta con su propia red, por lo que podemos suponer que a medida que pase el tiempo los volúmenes de información a manejar serán gigantescos y se deberán crear herramientas que ayuden al manejo de esta de una forma segura y confiable.

No se puede determinar con exactitud el instante en el que comiencen a aparecer los primeros delitos electrónicos, pero lo cierto es que desde ese momento hacen su aparición en escena, los científicos computacionales que tienen como objetivo minimizar los riesgos en los servicios que se prestan tomando como base las telecomunicaciones y, en general, las redes. La firma digital toma uno de estos aspectos de seguridad, mas concretamente el de autenticidad en los mensajes que son enviados por un usuario A y recibidos por un usuario B.

El diseño de la aplicación garantiza autenticidad y privacidad, utilizando a Lucas como algoritmo de clave publica, la función Hash RIPEMD-160 y el cifrador de bloques Idea, los cuales son algoritmos criptográficos basados en la teoría de números y aun no han sido quebrantados.

Para utilizar esta herramienta necesitará un par de claves (Publica y Privada), por medio de las cuales usted podrá: Autenticar y Cifrar o solo Autenticar un documento para luego por medio una aplicación de correo enviarlo a un receptor; O Descifrar y confirmar la autenticidad de dicho documento.

Con este trabajo, se coloca un granito de arena en la creación de una infraestructura que brinde mayor seguridad a la comunicación a través de documentos por medio de una red insegura.

El documento está organizado en dos tres capítulos y dos anexos. En él primer capitulo se describen algunos aspectos de seguridad en redes, y como la criptografía contribuye a brindar este servicio, también se describe con detalle cada uno de los algoritmos escogidos.

El capitulo 2 describe el diseño de la firma digital y su diseño de flujo de datos además del diccionario de datos, que sirvieron como base para la implementación.

En el capitulo 3, se detallan los pasos seguidos en el desarrollo de esta investigación, hasta llegar al resultado final.

El anexo A, llamado manual del usuario, describe como utilizar esta aplicación.

El anexo B, documenta detalladamente la composición de la aplicación y su código original. Constituye lo que se denomina el manual del sistema.

1. SEGURIDAD Y CRIPTOGRAFÍA

1.1 SEGURIDAD

1.1.1 Antecedentes. El crecimiento exponencial de los usuarios y organizaciones conectadas a Internet ha originado el tránsito a través de ella de informaciones de todo tipo, desde noticias, hasta complejas transacciones que requieren medidas específicas de seguridad que garanticen la confidencialidad, la integridad y la constatación del origen de los datos.

Se supone la existencia de un emisor y un receptor, los cuales quieren intercambiar mensajes. El posible enemigo que quiere interferir de algún modo en la comunicación se denomina intruso. Este intruso puede ser pasivo, si sólo escucha la comunicación, o activo si trata de alterar los mensajes.

Es aquí donde aparece la criptografía con objeto de proporcionar comunicaciones seguras sobre canales inseguros. Los mensajes sin transformar de ninguna manera se denominan **texto en claro** (también llamado, según una no muy adecuada traducción, **texto plano**). El proceso

mediante el cual la información contenida en el mensaje es ocultada se denomina **cifrado**. Un mensaje cifrado se denomina, obviamente, **texto cifrado**. El proceso mediante el que se revierte el proceso de ocultación, obteniendo el texto en claro a partir del texto cifrado se denomina **descifrado.**

1.1.2 Servicios De Seguridad. Dentro del modelo de referencia OSI, se define una arquitectura de seguridad ("*Information Processing Systems". OSI Reference Model - Part 2: Security Arquitecture*", ISO/IEC IS 7498-2). De acuerdo con esta arquitectura, para proteger las comunicaciones de los usuarios a través de una red, es necesario dotar a las mismas con una serie de servicios, que se conocen como servicios de seguridad:

- **Autenticación de la entidad par**: este servicio verifica la fuente de los datos. La autenticación puede ser sólo de la entidad origen, de la entidad destino o de ambas a la vez.
- **Control de acceso**: este servicio verifica que los recursos son utilizados por quien tiene derecho a hacerlo.
- **Confidencialidad de los datos**: este servicio evita que se revelen, deliberada o accidentalmente, los datos de una comunicación.
- **Integridad de los datos**: este servicio verifica que los datos de una comunicación no se alteren, esto es, que los datos recibidos por el receptor coincidan por los enviados por el emisor.

- **No repudio (irrenunciabilidad)**: este servicio proporciona la prueba, ante una tercera parte, de que cada una de las entidades han participado, efectivamente, en la comunicación. Puede ser de dos tipos:

    - **Con prueba de origen o emisor**: el destinatario tiene garantía de quien es el emisor concreto de los datos.
    - **Con prueba de entrega o receptor**: el emisor tiene prueba de que los datos de la comunicación han llegado íntegramente al destinatario correcto en un instante dado.

1.1.3 Mecanismos De Seguridad. Para proporcionar los servicios de seguridad citados, es necesario incorporar en los niveles adecuados del modelo de referencia OSI los siguientes mecanismos de seguridad:

- **cifrado**: el cifrado puede hacerse mediante el uso de criptosistemas simétricos o asimétricos y puede aplicarse extremo a extremo o a cada enlace del sistema de comunicaciones.
  El mecanismo de cifrado soporta el servicio de confidencialidad de los datos y puede complementar a otros mecanismos para conseguir diversos servicios de seguridad.

- **Firmado digital**: la firma digital se puede definir como un conjunto de datos que se añaden a una unidad de datos de modo que protejan a ésta contra cualquier falsificación, permitiendo al receptor comprobar el origen y la integridad de los datos. Para ello, se cifra la unidad de datos junto con alguna componente secreta del firmante, y se obtiene un valor de control ligado al resultado cifrado.

  El mecanismo de cifrado digital soporta los servicios de integridad de los datos, autenticación del emisor y no repudio con prueba de origen. Para que se pueda proporcionar el servicio de no repudio con prueba de entrega, hay que forzar al receptor para que envíe un acuse de recibo firmado digitalmente.

- **Control de acceso**: se usa para autentificar las capacidades de una entidad para acceder a un recurso dado. El control de acceso se puede llevar a cabo en el origen o en un punto intermedio, y se encarga de asegurar que el emisor está autorizado a comunicarse con el receptor o a usar los recursos de comunicación.
  El mecanismo de control de acceso soporta el servicio de control de acceso.

- **Integridad de datos**: hay que distinguir entre la integridad de una unidad de datos individual y la integridad de una secuencia de unidades de datos. Para lograr integridad de una unidad de datos, el emisor añade datos suplementarios a la unidad de datos. Estos datos suplementarios se obtienen en función de la unidad de datos y, generalmente, se cifran. El receptor genera los mismos datos suplementarios a partir de la unidad original y los compara con los recibidos.

  Para proporcionar integridad a una secuencia de unidades de datos se requiere, adicionalmente, algún mecanismo de ordenación, tal como el uso de números de secuencia, un sello temporal o un encadenamiento criptográfico entre las unidades.

  Los mecanismos de integridad de datos soportan el servicio de integridad de datos.

- **Intercambio de autenticación**, que tiene dos grados:

  - **Autenticación simple**: el emisor envía su identificador y una contraseña al receptor, el cual los comprueba.

- **Autenticación fuerte**: utiliza propiedades de los criptosistemas de clave pública. Un usuario se autentifica mediante su identificador y su clave privada. Su interlocutor debe verificar que aquel, efectivamente, posee la clave privada, para lo cual debe obtener, de algún modo, la clave pública del primero. Para ello deberá obtener su certificado. Un certificado es un documento firmado por una Autoridad de Certificación (una tercera parte de confianza) y válido durante el periodo de tiempo determinado, que asocia una clave pública a un usuario. El mecanismo de intercambio de autenticación soporta el servicio de autenticación de entidad par.

1.2 CRIPTOGRAFÍA

Las raíces etimológicas de la palabra criptografía son KRYPTOS, que significa oculto y GRAPHOS, que se traduce como escribir, lo que da lugar a su definición clásica:

” Arte de escribir mensajes en clave secreta o enigmáticamente ”

Fue considerada un arte hasta 1949 cuando Shannon publicó, “La teoría de las comunicaciones secretas”, que fue aplicada por el NBS (National Bureau of Standards) de los Estados Unidos para desarrollar el sistema criptográfico DES. Desde entonces la criptografía empezó a ser considerada como una ciencia aplicada, debido a su relación con otras ciencias como la estadística, la teoría de números, la teoría de la información, y la teoría de la complejidad computacional.

1.2.1 Criptosistema Convencional Las técnicas de clave única, secreta o simétrica tienen fundamentos de complejidad diversa, pero todas usan una misma clave k que es conocida por el remitente de los mensajes y por el receptor, y mediante la cual se cifra y descifra el mensaje que se quiere proteger (existe una variante según la cual la clave de descifrado puede obtenerse de la de cifrado utilizando unos recursos y en un tiempo razonable).

Los cifradores simétricos pueden dividirse en dos grupos:

- **cifradores de flujo**, los cuales cifran un único bit del texto en claro cada vez.

- **cifradores de bloque**, que toman un grupo de bits (un valor típico es 64 bits) y lo cifran como si se tratase de una unidad.

Las ventajas de la utilización de criptografía de clave simétrica es la existencia de algoritmos muy rápidos y eficientes, especialmente si se implementan en *hardware*. Si k es lo bastante larga (típicamente se usan valores de 56 a 128 bits), es imposible reventarlas usando la fuerza bruta.

Cualquier Criptosistema, en el sentido clásico del termino, se compone de 5 elementos que vamos a ver a continuación:

a. **Espacio de mensajes en claro:**

Se trata de un conjunto que contiene todos los mensajes en claro posibles.

Esto significa que en principio puede parecer excesivo (imaginemos en español cuántos mensajes posibles podemos escribir... ¡infinitos!), En la práctica no lo es tanto, ya que en algunos sistemas la longitud del mensaje es limitada (comunicaciones entre un banco y un cajero, por ejemplo).

Se denota normalmente con la letra **M**

b. **Espacio de mensajes cifrados:**

Conjunto que contiene todos los posibles mensajes cifrados. Se sigue el mismo razonamiento que en el punto anterior.

Los mensajes cifrados pueden estarlo en el mismo alfabeto que los mensajes en claro o no. Hoy en día se puede considerar que es el mismo alfabeto, si tenemos en cuenta que al fin y al cabo todo se traduce al sistema binario.

Para hablar de este conjunto usaremos la letra **C**.

c. Espacio de claves:

Son todas las claves posibles que se pueden usar.

Se usa la letra **K** para referirse a este conjunto.

d. Transformación de cifrado:

Son los distintos algoritmos que se pueden usar para cifrar los mensajes en claro.

Se hace referencia al algoritmo de cifrado como **E**.

e. Transformación de descifrado:

Algoritmos que se pueden emplear para descifrar los mensajes cifrados.

Evidentemente el algoritmo de cifrado debe estar relacionado con el de descifrado.

Se trata de la transformación **D** en la mayoría de las fórmulas.

Siempre se ha de cumplir que:

$$\mathbf{E_k(D_k(c)) = c}$$

Lo cual se lee de la siguiente manera:

Si tenemos un mensaje cifrado c (por supuesto este pertenece al conjunto C), y lo desciframos usando el algoritmo D con la clave k, y a su vez el resultado lo volvemos a cifrar con E y con la misma clave, pues como es lógico debe dar el mensaje cifrado original c.

Estos sistemas proporcionan seguridad en la integridad de la información, ya que no puede modificarse sin el conocimiento de la clave. El algoritmo **IDEA** es un ejemplo moderno de estos sistemas.

La fuerza del Criptosistema de clave privada radica en la **clave**.

Algunas veces se confía en la confidencialidad del algoritmo empleado. La historia nos dice que esto es mala práctica, y que incluso suele utilizarse cuando el algoritmo usado es poco seguro.

1.2.1.1 Algoritmos De Clave Simétrica. Dentro de los algoritmos de clave simétrica más conocidos se encuentran:

DES con varios modos de operación (ECB, CBC, CFB, OFB), Triple-DES, RC-2, RC-4, Skipjack, Blowfish, AES ,IDEA y otros, una información mas detallada sobre cada uno de ellos mas información se puede encontrar en [1].

Dentro de estos algoritmos se estima que él mas seguro hasta el momento es el IDEA que aun no ha sido violado, y a continuación se describe con detalle, este algoritmo se selecciono para la aplicación; además, puede ser utilizado fuera de los Estados Unidos sin ningún problema de patente.

1.2.1.1.1 El Bloque Cifrador Idea

1.2.1.1.1.1 Antecedentes. La primera personalización del cifrador de IDEA fue hecha por Xuejia Lai y James Massey, en 1990. Bajo el nombre de PES (Proposed Encryption Standard).

Al año siguiente, Biham y Shamir demostraron un criptoanálisis diferencial, fortalecieron el cifrado contra los ataques y llamaron al nuevo algoritmo IPES (

Improved Proposed Encryption Standard). En 1992 IPES cambio de nombre a IDEA (International Data Encryption Algorithm).

La estructura del cifrador fue escogida para facilitar su implementación tanto en hardware como en software.

1.2.1.1.1.2 DESCRIPCION DEL ALGORITMO IDEA. IDEA es un cifrador de bloques, opera sobre bloques de texto plano de 64 bits. La clave es de una longitud de 128 bits. El mismo algoritmo es usado para la encripción y desencripción. El diseño inicial de la filosofía del algoritmo es el de:

"*MEZCLAR OPERACIONES DE DIFERENTES GRUPOS ALGEBRAICOS*".

- *XOR*
- *SUMA MODULO* 2^{16}
- *MULTIPLICACION MODULO* $2^{16} + 1$

Todas esta operaciones operan sobre bloques de 16 bits. El XOR opera bit a bit y es denotado por $\oplus$; La suma modulo 2^{16} es denotada por $\boxplus$; En la multiplicación si un sub-bloque consta de puros ceros es tratado como representación 2^{16} ; el resultado de esta operación es denotado como $\odot$.

Como un ejemplo de estas operaciones, note que:

$$(0, ..., 0) \odot (1, 0, ..., 0) = (1, 0, ..., 0, 1)$$

porque

$$2^{16}2^{15} \bmod (2^{16} + 1) = 2^{15} + 1.$$

El bloque de texto plano de 64 bits X es particionado en 4 sub-bloques de 16 bits X_1, X_2, X_3 y X_4; por ejemplo $X = (X_1, X_2, X_3, X_4)$. Los cuatros sub-bloques del texto plano son entonces transformados en texto cifrado de sub-bloques de 16 bits Y_1, Y_2, Y_3, Y_4. Entonces el texto cifrado es $Y = (Y_1, Y_2, Y_3, Y_4)$ bajo el control de 52 claves sub-bloques de 16 bits las cuales son formadas de la clave secreta de 128 bits. Todo esto es explicado en la figura 1.

Símbolos Utilizados en la Figura 1:

X_i : Bloques del texto plano de 16 bits

Y_i : Bloques del texto cifrado de 16 bits

$Z_i^{(r)}$: Clave de sub-bloque de 16 bits

$\boxplus$: Suma modulo 2^{16} de enteros de 16 bits

$\odot$: Multiplicación modulo $2^{16} + 1$

$\oplus$: OR exclusivo bit a bit en bloques de 16 bits.

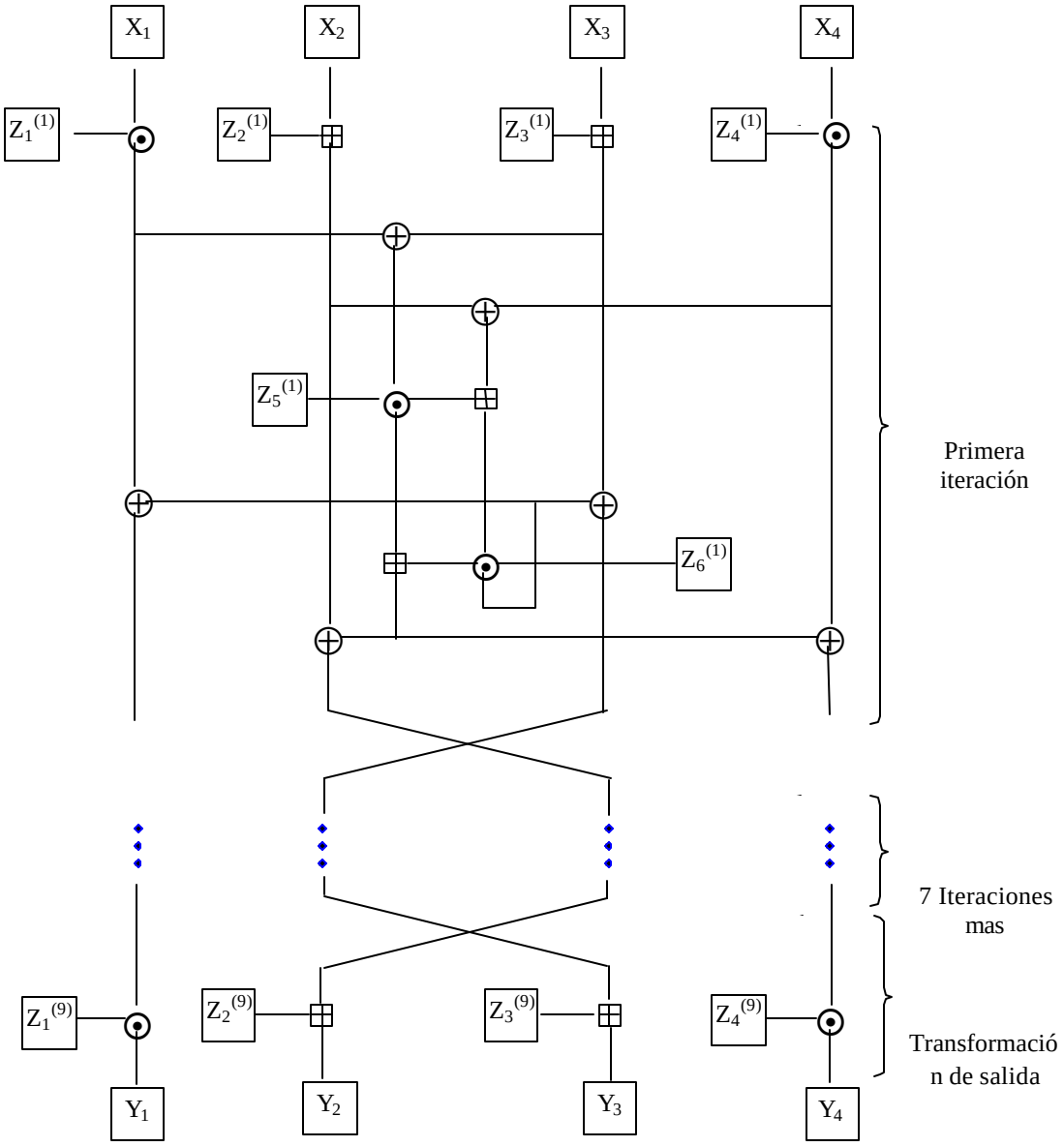


Figura 1. Diagrama Del Algoritmo IDEA

Según la figura 1: El bloque de datos de 64 bits es dividido en 4 sub-bloques de 16 bits X_1, X_2, X_3 y X_4. Estos 4 sub-bloques se convierten en la entrada de la primera iteración del algoritmo. (8 en total).

En cada iteración a los cuatro sub-bloques se les hace XOR, multiplicaciones y sumas con otras 6 sub-claves de 16 bits. En 2 iteraciones, el segundo y tercer sub-bloque son cambiados, finalmente los 4 sub-bloques son combinados con 4 sub-claves en una salida transformada. En cada iteración la secuencia es como sigue:

- Se multiplica X_1 y la primera sub-clave.
- Suma de X_2 y la segunda sub-clave.
- Suma de X_3 y la tercera sub-clave.
- Multiplicación de X_4 y la cuarta sub-clave.
- XOR los resultados de los pasos 1 y 3.
- XOR los resultados de los pasos 2 y 4.
- Multiplicación del paso 5 con la quinta sub-clave.
- Suma de los resultados 6 y 7.
- Multiplicación del resultado del paso 8 con la sexta sub-clave.
- Suma de los pasos 7 y 9.
- XOR los resultados de los pasos 1 y 9.
- XOR los resultados de los pasos 3 y 9.
- XOR los resultados de los pasos 2 y 10.
- XOR los resultados de los pasos 4 y 10.

La salida de la iteración son los 4 sub-bloques que dan los resultados de los pasos 11 al 14. Cambiar los 2 bloques internos (excepto en la ultima iteración). Y esto es la entrada de la siguiente iteración.

Después de las 8 iteraciones:

- Multiplicar X_1 por la primera sub-clave.
- Sumar X_2 y la segunda sub-clave.
- Sumar X_3 y la tercera sub-clave.
- Multiplicar X_4 y la cuarta sub-clave.

Finalmente los 4 sub-bloques son reunidos para producir el texto cifrado.

Generación De Las Sub-claves De Desencripción: La clave de 128 bits es dividida en 8 sub-claves de 16 bits. Esas son las primeras 8 sub-claves para el algoritmo (las seis primeras para la primera iteración y las 2 primeras para la segunda iteración):

Después, la clave es rotada 25 bits a la izquierda y de nuevo es dividida en 8 sub-claves. Las primeras 4 son usadas en al segunda iteración y las ultimas 4 en la tercera iteración.

La clave es rotada otros 25 bits a la izquierda para las siguientes 8 sub-claves, y así hasta el final del algoritmo.

Para desencriptar es exactamente lo mismo, excepto que las claves son lo contrario y ligeramente diferentes.

La sub-clave de desencripción son también la suma o multiplicación inversa de la sub-clave de encripción.

La tabla 6 muestra las sub-claves de encripción y su correspondiente sub-clave para desencriptar.

Tabla 6 IDEA. Sub-Claves De Encripción Y De Desencripción.

ITERACION	SUB-CLAVE DE ENCRIPCIÓN	SUB-CLAVE DE DESENCRIPCIÓN
1-ra ITERACION	$Z_1^{(1)}$, $Z_2^{(1)}$, $Z_3^{(1)}$, $Z_4^{(1)}$, $Z_5^{(1)}$, $Z_6^{(1)}$	$Z_1^{(9)-1}$, $-Z_2^{(9)}$, $-Z_3^{(9)}$, $Z_4^{(9)-1}$, $Z_5^{(8)}$, $Z_6^{(8)}$
2-da ITERACION	$Z_1^{(2)}$, $Z_2^{(2)}$, $Z_3^{(2)}$, $Z_4^{(2)}$, $Z_5^{(2)}$, $Z_6^{(2)}$	$Z_1^{(8)-1}$, $-Z_3^{(8)}$, $-Z_2^{(8)}$, $Z_4^{(8)-1}$, $Z_5^{(7)}$, $Z_6^{(7)}$
3-ra ITERACION	$Z_1^{(3)}$, $Z_2^{(3)}$, $Z_3^{(3)}$, $Z_4^{(3)}$, $Z_5^{(3)}$, $Z_6^{(3)}$	$Z_1^{(7)-1}$, $-Z_3^{(7)}$, $-Z_2^{(7)}$, $Z_4^{(7)-1}$, $Z_5^{(6)}$, $Z_6^{(6)}$
4-ta ITERACION	$Z_1^{(4)}$, $Z_2^{(4)}$, $Z_3^{(4)}$, $Z_4^{(4)}$, $Z_5^{(4)}$, $Z_6^{(4)}$	$Z_1^{(6)-1}$, $-Z_3^{(6)}$, $-Z_2^{(6)}$, $Z_4^{(6)-1}$, $Z_5^{(5)}$, $Z_6^{(5)}$
5-ta ITERACION	$Z_1^{(5)}$, $Z_2^{(5)}$, $Z_3^{(5)}$, $Z_4^{(5)}$, $Z_5^{(5)}$, $Z_6^{(5)}$	$Z_1^{(5)-1}$, $-Z_3^{(5)}$, $-Z_2^{(5)}$, $Z_4^{(5)-1}$, $Z_5^{(4)}$, $Z_6^{(4)}$
6-ta ITERACION	$Z_1^{(6)}$, $Z_2^{(6)}$, $Z_3^{(6)}$, $Z_4^{(6)}$, $Z_5^{(6)}$, $Z_6^{(6)}$	$Z_1^{(4)-1}$, $-Z_3^{(4)}$, $-Z_2^{(4)}$, $Z_4^{(4)-1}$, $Z_5^{(3)}$, $Z_6^{(3)}$
7-ma ITERACION	$Z_1^{(7)}$, $Z_2^{(7)}$, $Z_3^{(7)}$, $Z_4^{(7)}$, $Z_5^{(7)}$, $Z_6^{(7)}$	$Z_1^{(3)-1}$, $-Z_3^{(3)}$, $-Z_2^{(3)}$, $Z_4^{(3)-1}$, $Z_5^{(2)}$, $Z_6^{(2)}$
8-va ITERACION	$Z_1^{(8)}$, $Z_2^{(8)}$, $Z_3^{(8)}$, $Z_4^{(8)}$, $Z_5^{(8)}$, $Z_6^{(8)}$	$Z_1^{(2)-1}$, $-Z_3^{(2)}$, $-Z_2^{(2)}$, $Z_4^{(2)-1}$, $Z_5^{(1)}$, $Z_6^{(1)}$
TRANSFORMACIÓN DE LA SALIDA	$Z_1^{(9)}$, $Z_2^{(9)}$, $Z_3^{(9)}$, $Z_4^{(9)}$	$Z_1^{(1)-1}$, $-Z_2^{(1)}$, $-Z_3^{(1)}$, $Z_4^{(1)-1}$

Donde Z^{-1} denota la multiplicación inversa (modulo $2^{16} + 1$) de Z, o sea, $Z \odot Z^{-1} = 1$. Y - Z denota la suma inversa (modulo 2^{16}) de Z, o sea, - Z ⊞ Z = 0.

1.2.1.1.1.3 La Lista De La Clave. Los 52 sub-bloques de 16 bits de las claves usadas en el proceso de encripción son generadas desde la clave de sesión del usuario de 128 bits como sigue: La clave es particionada en 8 sub-bloques que son directamente usados como las primeras 8 sub-claves.

1.2.2 Criptosistema De Clave Publica. Su principal característica es que cada usuario posee dos claves distintas: una pública, conocida por todos, y otra privada, que se mantiene en secreto.

Las posibilidades son mucho mayores en estos sistemas, prestando servicios como la autenticación o la firma digital, lo que a su vez deriva en muchos otros usos.

En la práctica, el uso más seguro que se le da a los criptosistemas es combinando la criptografía de clave pública con la de clave privada. Los algoritmos de clave pública son lentos, y para mensajes largos es mejor cifrar

con algoritmos utilizados en criptografía de clave privada. Para intercambiar esta clave única (de longitud corta) se usan los algoritmos de clave pública.
El principal inconveniente estriba en la necesidad de que todas las partes conozcan k. Esta clave es distribuida mediante una transacción separada y diferente a la transmisión del mensaje cifrado. Es aquí, precisamente, donde se halla el punto vulnerable del mecanismo: la distribución de la clave. Si la clave es interceptada se pone en peligro todo el mecanismo.

La clave debe ser transmitida por un canal seguro para poder asegurar la eficacia del sistema criptográfico. Este canal es generalmente externo a Internet, ya que, en caso contrario, surgiría la pregunta "*Si tenemos un canal seguro, ¿para qué necesitamos criptografía?*".

La solución a este problema apareció en 1976. En ese año, Whitfield Diffie y Martín Hellman demostraron la posibilidad de construir sistemas criptográficos que no precisaban la transferencia de una clave secreta entre emisor y receptor, evitando así los problemas derivados de la búsqueda de canales seguros para tal transferencia. Se trataba de la "criptografía de clave asimétrica o pública".

En 1976, **Whitfield Diffie** y **Martín Hellman** publican la idea de que cada usuario tenga dos claves: una pública (P_i, conocida por cualquiera) y otra privada (S_i, sólo conocida por su dueño).

Entre estas claves existe una relación particular que permite a una de ellas cifrar un mensaje mientras que la otra es empleada para descifrarlo. Las claves privadas deben ser conservadas por su propietario del modo más seguro posible (almacenadas en el disco duro del ordenador de su propietario, cifradas con una contraseña, o en tarjetas magnéticas o inteligentes que se inserten en *hardware* especial conectado al ordenador).

Supongamos un algoritmo de cifrado E y otro de descifrado D, aplicados a un mensaje M. Debe cumplirse que:

$$\mathbf{D(E(M, P_i), S_i) = M}$$

Supongamos que A quiere mandarle un mensaje a B:

A busca la clave pública de B (P_m).

A cifra el mensaje M con la clave pública de B y le envía el resultado, **C.**

$$\mathbf{C = E(M, P_m)}$$

B, al recibir el mensaje C lo descifra con su clave privada.

$$\mathbf{M = D(C, S_m)}$$

La seguridad de un sistema de este tipo depende de que las funciones de cifrado y descifrado, E y D, cumplan una serie de condiciones:

1. Dados el mensaje M y la clave pública P que vayamos a utilizar, el mensaje cifrado, C, debe ser fácil de calcular.

2. Dado C, el mensaje original, M, no debe ser obtenible de forma sencilla.
3. Dados C y la clave privada, S, debe ser sencillo descifrar el mensaje original.
4. para que sea práctico el uso de criptografía de clave asimétrica, debe ser sencillo calcular parejas aleatorias de claves P y S.

Aunque Diffie y Hellman definieron los principios de la criptografía de clave asimétrica, fueron **Ron Rivest**, **Adi Shamir** y **Leonard Adleman**, investigadores del MIT, los primeros que, en 1978, encontraron las funciones que satisfacían los requisitos citados.

La criptografía de clave asimétrica posee, sin embargo, dos inconvenientes:

- El primero se refiere a la velocidad. Los sistemas basados en clave asimétrica son notablemente más lentos que sus equivalentes de clave simétrica (por lo general y como mínimo, unos dos órdenes de magnitud). Por tanto estos sistemas no suelen ser adecuados para el cifrado masivo de datos.

- El segundo está relacionado con la validación de la clave. La discusión sobre la fortaleza de un algoritmo de clave asimétrica es irrelevante sin una discusión previa sobre el protocolo de validación de las claves.

1.2.2.1 Autenticación Mediante Criptografía De Clave Asimétrica. La criptografía de clave pública puede ser utilizada para identificar sin ambigüedades al remitente de un mensaje. Esto es posible teniendo en cuenta que, si el remitente cifra con su clave privada el mensaje que envía, éste solamente puede ser descifrado en destino utilizando la clave pública del remitente.

Si el mensaje no puede ser descifrado con la clave pública de quien afirma ser el remitente (si el resultado es basura), éste no ha sido el remitente del mensaje. La posibilidad de descifrar el mensaje da prueba fehaciente de la identidad del remitente.

La probabilidad de que dos personas diferentes tengan la misma combinación clave pública/clave privada es insignificante.

La desventaja de la utilización de cualquier algoritmo de clave pública es su lentitud, por lo que resulta poco práctico el cifrado asimétrico del mensaje entero.

1.2.2.2 Algoritmos De Clave Asimétrica. Dentro de los algoritmos de clave publica tenemos RSA, Algoritmo el Gamal, LUC y otros.

El algoritmo LUC es utilizado en la aplicación, no presenta problemas de patente como el RSA, y es tan seguro como los demás algoritmos de clave publica, e incluso se cree que este es criptograficamente más difícil de violar que el RSA.

1.2.2.2.1 Sistema De Clave Publica LUC. Las propiedades de la función Lucas se reflejan en la exponenciación, los procesos de claves públicas y privadas se pueden desarrollar de una manera análoga al RSA.

Esto facilita demostrar que un prospero ataque contra el sistema LUC puede tener un mayor éxito en el sistema RSA.

1.2.2.2.2 Función Lucas. La relación de segundo orden lineal se considera como:

$$T_n = PT_{n-1} - QT_{n-2} \qquad (1)$$

P y Q pueden ser primos relativos. Si se hace P = 1 = -Q, entonces la secuencia de números obtenidos por elección $T_0 = 0$ y $T_1 = 1$ es la famosa serie de Fibonacci.

La forma general de una secuencia obtenida desde una relación lineal de segundo orden repetitiva puede ser encontrada fácilmente.
Si se escoge a α y β como las raíces de la ecuación polinomial:

$$X^2 - PX + Q = 0 \qquad (2)$$

si c_1 y c_2 son números enteros, entonces la secuencia $\{ c_1\alpha^n + c_2\beta^n \}$ tiene la propiedad de :

$$P(c_1\alpha^{n-1} + c_2\beta^{n-1}) - Q(c_1\alpha^{n-2} + c_2\beta^{n-2}) = c_1\alpha^{n-2}(P\alpha - Q) + c_2\beta^{n-2}(P\beta - Q)$$

$$= c_1\alpha^{n-2}(\alpha^2) + c_2\beta^{n-2}(\beta^2)$$

por la ecuación 1

$$= c_1\alpha^n + c_2\beta^n$$

Así esta ecuación satisface la relación lineal de segundo orden repetitiva de la ecuación 1, y de igual forma se puede comprobar una secuencia T_n donde

$$T_0 = c_1 + c_2 \text{ y } T_1 = c_1\alpha + c_2\beta.$$

Nótese que si T_0 y T_1 son enteros, entonces por (1), todos los términos en la secuencia serán enteros, aunque α, β, c_1 y c_2 no sean enteros y todos no pueden ser reales.

Existen dos soluciones particulares en la relación lineal de segundo orden repetitivo las cuales son de particular interés. Son denotadas por U_n y V_n y son definidas por:

$$U_n = (\alpha^n - \beta^n) / (\alpha - \beta) \text{ luego } c_1 = 1/ (\alpha - \beta) = - c_2$$

$$V_n = \alpha^n + \beta^n \text{ luego } c_1 = c_2 = 1$$

Ambas son una secuencia de enteros ya que:

$$U_0 = 0,\ U_1 = 1,\ V_0 = 2 \text{ y } V_1 = P.$$

esta secuencia depende sólo de P y Q, y los términos son llamados la función de Lucas de P y Q.

Se puede también escribir de la forma $U_n (P, Q)$ y $V_n (P, Q)$.

1.2.2.2.3 Relaciones De La Función Lucas. Puesto que α y β son las raíces de (2) satisfacen las siguientes ecuaciones:

$$\alpha + \beta = P \qquad y \qquad \alpha\beta = Q$$

No es difícil de encontrar muchas relaciones entre la función Lucas U_n y V_n, y el coeficiente de relación repetitiva (1), P y Q. El discriminante de (2),

$D = P^2 - 4Q$, puede ser expresado en términos de α y β por:

$$D = (\alpha - \beta)^2$$

de igual manera para:

$$V_{2n} = V_n^2 - 2Q^n \qquad (3)$$

$$V_{2n-1} = V_n V_{n-1} - PQ^{n-1} \qquad (4)$$

$$V_{2n+1} = P V_n^2 - Q V_n V_{n-1} - PQ^n \qquad (5)$$

$$V_n^2 = DU_n^2 + 4Q^n \qquad (6)$$

$$2 V_{n+m} = V_n V_m + DU_n U_m \qquad (7)$$

$$2Q^m V_{n-m} = V_n V_m - DU_n U_m \qquad (8)$$

Considerando la relación lineal repetitiva con V_k (P, Q) por P y Q^k por Q se tiene que:

$$T_n = V_k (P, Q) T_{n-1} - Q^k T_{n-2}$$

Las raíces de la correspondiente ecuación cuadrática, α' y β' debe satisfacer:

$$\alpha' + \beta' = V_k(P, Q) = \alpha^k + \beta^k \quad y$$

$$\alpha' \beta' = Q^k = \alpha^k \beta^k ,$$

así se obtiene que $\alpha' = \alpha^k$ y $\beta' = \beta^k$ esto quiere decir :

$$V_n (V_k (P, Q) , Q^k) = (\alpha^k)^n + (\beta^k)^n = V_{nk} (P, Q).$$

Este arreglo resultante es crucial; es una clara generalización de la regla para composiciones de potencia, con la sub-escritura de una función Lucas jugando el papel de una potencia. Si se hace Q = 1, entonces se tiene la relación simple:

$$V_{nk} (P, 1) = V_n (V_k (P, 1), 1). \qquad (9)$$

La función Lucas será usada para probar resultados de divisibilidad, porque hay relaciones entre la sub-escritura de una función de Lucas y el divisor de este valor.

Para explicar esto se define el símbolo de Legendre:

(D/p) = 0 si p divide a D

(D/p) = 1 si existe un número x tal que $D \equiv x^2 \bmod p$ ó

(D/p) = -1 sino existe x.

Si P es un número primo, el cual no divide a Q ó D, y ε es (D/p), entonces por la demostración de Lehmer:

$$U_{k(p-\varepsilon)}(P, Q) \equiv 0 \bmod p \qquad (10)$$

para algún entero k. También es verdadero para:

$$V_{k(p-\varepsilon)}(P, Q) \equiv 2Q^{k(1-\varepsilon)/2} \bmod p \qquad (11)$$

Para un entero k. Para k = 1; y el resultado para cualquier k es deducido de:

$V_{(k+1)(p-\varepsilon)} = (\frac{1}{2})(V_{k(p-\varepsilon)}\ V_{p-\varepsilon} + DU_{k(p-\varepsilon)}\ U_{p-e}\ )$ por (8):

$V_{k(p-\varepsilon)}\ Q^{(1-\varepsilon)/2} \bmod p$ por (11).

Con una secuencia de la función Lucas definida por números primos P y Q , hay una generalización de la función totalizadora de Euler para la función Lucas, la función totalizadora de Lehmer. Esta definición general, y una explicación de esta relación para la función totalizadora de Euler no es concerniente aquí. Sólo se puede necesitar para ser aplicada a números N de la forma N = pq, con p y q primos diferentes. En este caso la función totalizadora de Lehmer N es:

$$T(N) = (p - (D/ p))* (q - (D/q)).$$

Justo como en el caso de la función totalizadora de Euler, el producto no es necesario, y solamente se necesita el lcm (mínimo común múltiplo) de los factores; se define:

$$S(N) = \text{lcm} ((p - (D/p)), (q - (D/q))).$$

Puesto que S(N) es un producto de (p - (D/p)) y (q - (D/q)) el resultado de (10) y (11) muestra que, cuando N = pq, p y q números primos no dividiendo D = P^2 - 4 en este caso:

$$U_{k\,S(N)}\ (P, 1) \equiv 0 \bmod N \text{ para un entero } k \qquad (12)$$

y

$$V_{k\,S(N)}\ (P, 1) \equiv 2 \bmod N \text{ para un entero } k \qquad (13).$$

Si N = pq es un producto de dos primos diferentes, P < N es primo relativo de N, e es un primo relativo de S(N), y d es encontrado (por el algoritmo de Euclides extendido) así que :

$$de = kS(N) + 1$$

para algún entero k, entonces se obtiene:

$$V_d\ (V_e\ (P, 1), 1) = V_{de}\ (P, 1) \qquad \text{por (9)}$$

$$= V_{k\,S(N)\,+\,1}\ (P, 1)$$

$$= PV_{k\,S(N)}\ (P, 1) - V_{k\,S(N)\,-\,1}\ (P, 1) \qquad \text{por (1)}$$

$$= P\,V_{k\,S(N)}\ (P, 1) - (\,1/2\,)(\,V_{k\,S(N)}\ (P, 1)V_1\ (P, 1)$$

$$- D\ U_{k\,S(N)}\ (P, 1)U_1\ (P, 1) \qquad \text{por (8)}$$

$$\equiv (2P - (1/2)(2P - 0)) \bmod N \quad \text{por (12) y (13)}$$

$$= P \qquad (14)$$

O sea: $V_d (V_e (P, 1), 1) = P$ (14)

Una aparente diferencia con la situación en el caso de potencias es una aparente falta de simetría entre la función $V_e (P, 1)$ y su inverso $V_d (R, 1)$. Los números d y e son relacionados por la función S(N), el cual contiene o envuelve el residuo cuadrático con respecto a D, el cual es definido en términos de P. por simetría, deberá ser esperado que S(N) puede tener el mismo valor, considerando que sí P ó V_e (P,1) es usado en su definición.

Por (7) se tiene que:

$$V_e^2 (P,1) - 4 = DU_e^2 (P, 1)$$

También, $(D/p) = (DU_e^2 / p)$, ya que el símbolo de Legendre no es cambiado por los cuadrados, como puede ser fácilmente observado de la definición. Esto quiere decir:

$$(D/p) = ((P^2 - 4)/ p) = ((V_e^2(P, 1) - 4) / p).$$

Luego el valor de S(N) es el mismo, considerando de sí esto es computado por la función directa o la función inversa.

1.2.2.2.4 El Nuevo Sistema De Clave Publica De LUC. Usando el resultado anterior, un sistema de clave publica puede ser desarrollado por analogía con el sistema RSA.

Suponga que N y e son dos números escogidos, con N el producto de dos primos diferentes, p y q. El número e puede ser escogido así: primo relativo de (p-1)(q-1)(p+1)(q+1).

Dado un M el cual es un mensaje que es menor que N y primo relativo de N aun que no necesariamente. Se define:

$$f_{LUC}(M) = V_e(M,1) \bmod N$$

Donde V_e es una función de Lucas.

Para definir de igual manera el proceso de clave privada, se necesita un número d tal que:

$$de \equiv 1 \bmod S(N)$$

donde S(N) = lcm ((p - (D/p)), (q -(D/q))),

Donde $D = (M')^2 - 4$, y (D/p), (D/q) son el símbolo de Legendre de D con respecto a p y q. Se Puede asumir que D es primo relativo de N, así el símbolo de Legendre es +1 ó -1.

Desde aquí la escogencia original de e asegura que es primo relativo de S(N), así el número d puede ser encontrado fácilmente por el algoritmo de Euclides extendido. El proceso de clave privada es el mismo como el proceso de clave publica, con e reemplazado por d. por (14) y (2) y el factor de M < N,

$$M = V_d (V_e (M, 1) \bmod N, 1) \bmod N,$$

Y el proceso de clave privada y clave publica son inversos por la simetría de e y d. hay dos formas en el cual este proceso parece tener dificultades; primeramente, la computación de V_e y V_d son extremadamente largos, por valores grandes de e o d, y segundo, el número d de la clave privada necesita ser recomputado por cada mensaje.

Ambas dificultades no son serias, como se explica más adelante.

La computación de la función de Lucas puede ser hecha por una doble técnica sucesiva, como en el “Russian Peasant” (campesino Ruso) método de multiplicación. Toda la aritmética es hecha modulo N, y Q es 1 para la función Lucas, así la ecuación (3) puede ser escrita:

$$V_{2n}(M, 1) \bmod N = ((V_n(M, 1) \bmod N)^2 - 2) \bmod N$$

Y de igual forma para las ecuaciones (5) y (6).

Si e tiene la expansión binaria:

$$e = \sum_{i=0}^{t} x_i 2^{t-i} \qquad (x_0 = 1,\ x_i = 0 \text{ ó } 1 \text{ sí } i > 0)$$

entonces dado e_k será la suma principal

$$\text{sum} \sum x_i 2^{t-i}$$, así que e_t es e y e_0 es 1.

Definiendo:

$$R_k = V_{2ek-1}(M, 1) \bmod N, \quad S_k = V_{2ek}(M,1) \bmod N$$

$$T_k = V_{2ek+1}(M, 1) \bmod N.$$

Entonces para cada k, R_k, S_k y T_k puede ser computada de $V_{ek}(M, 1) \bmod N$ y $V_{ek-1}(M, 1) \bmod N$, usando la forma modificada de (3), (4) y (5). El valor de $V_{ek+1}(M, 1) \bmod N$ y $V_{ek} - 1(M, 1) \bmod N$ puede ser entonces obtenido de R_k, S_k y T_k usando el factor que $e_{k+1} = 2e_k$, $e_{k+1} - 1 = 2e_k - 1$ si $x_{k+1} = 0$, mientras $e_{k+1} = 2e_k + 1$, $e_{k+1} - 1 = 2e_k$ si $x_{k+1} = 1$

Este método asegura que V_e y V_d pueden ser computadas en casi una misma fracción de tiempo.

Existen optimizaciones en el cálculo lo cual quiere decir que la suma total de computación es sólo cerca del 50% más que la necesitada para el RSA.

Ya que la clave privada correcta necesita ser obtenida para cada desencripción, existe un pequeño trabajo más complicado en el sistema de clave privada LUC que en el sistema de clave privada RSA.

Primeramente hay que notar que la "*recomputación*" de la clave privada número d es más aparente que real, como hay sólo cuatro valores posibles para d:

lcm((p + 1), (q + 1)),

lcm((p + 1), (q - 1)),

lcm((p - 1), (q + 1)) y

lcm((p - 1), (q - 1)).

Estos valores son todos conocidos en el tiempo que N es creado, así cuatro diferentes valores de d pueden ser calculados al tiempo.

Dado un mensaje para el cual aplica el proceso de clave privada, esto es necesario para encontrar el residuo cuadrático $(((M')^2 - 4)/p)$ y $(((M')^2 - 4)/q)$. Esto puede realizarse por un algoritmo análogo al algoritmo Euclidiano, el cual

puede hacerse al rededor de un tiempo de ejecución equivalente a: $O(\log_2 p) + O(\log_2 q)$ operaciones, cerca del mismo orden que se necesita para el cálculo de V_d (M', 1) mod N.

El cálculo de p y q es mucho más fácil, sin embargo, dado que p y q tienen cerca de la mitad de longitud que N, y en la practica el cálculo del residuo cuadrático hace crecer en un 20% del tiempo requerido para él calculo de la función Lucas. Una vez que el residuo cuadrático es conocido, el valor correcto de d puede ser usado en él calculo de la función Lucas, teniendo una suma total de trabajo computacional cerca del 80% más que el necesitado por el proceso de clave privada del RSA.

Una forma de evitar él computo del residuo cuadrático es definir una nueva función:

$$R(N) = \text{lcm}((p-1), (q-1), (p+1), (q+1)),$$

y escoger d tal que:

$$de \equiv 1 \bmod R(N).$$

puesto que R(N) es un múltiplo de todos los valores posibles de S(N), él calculo principal para (14) están inalterados (iguales), y el d obtenido en esta forma puede trabajar en todos los casos. Solamente el problema que esto representa es que probablemente pueda tener cerca del doble de dígitos que N, así el esfuerzo computacional para calcular V_d (M', 1) mod N para este d puede ser el doble.

1.2.2.2.5 Fortaleza Criptográfica De LUC. El proceso de clave privada puede ser revelado solamente si hay una forma de computación V_d (M', 1)mod N sin conocimiento de d, o si hay alguna manera de encontrar d desde e y N, el segundo problema es de mayor dificultad que el problema correspondiente para el RSA, porque existen cuatro valores diferentes de d para cada par (e, N), solamente un valor de d trabaja para un arbitrario M'.

El hecho de que las funciones de Lucas sean una generalización de potencia hace seguro que un afortunado ataque sobre el sistema de clave publica LUC automáticamente lo hace también seguro para el sistema RSA. Sin embargo la relación inversa no lo hace verdad.

1.2.2.2.6 El Proceso Completo LUC.

- Se seleccionan dos números primos p y q.
- Se calcula un N tal que: N = p*q.
- Se selecciona un entero e de tal manera que:
- Gcd[(p – 1)(q – 1)(p + 1)(q + 1), e] = 1
- Se calcula el determinante D tal que: $D = p^2 - 4$
- Se calcula un S(N) tal que: S(N) = lcm[[p – (D/p)], [q – (D/q)]]
- se calcula el entero d tal que: $d = e^{-1} \bmod S(N)$
- Se forma la clave publica KU con el par e y N así:

- KU = {e, N}
- Se forma la clave privada KR con el par d y N así:
- KR = {d, N}
- Sé encripta con el siguiente proceso:
- P < N donde P es el mensaje.
- $C = V_e (P, 1) \pmod N$
- C es el texto cifrado.
- Sé descifra con el siguiente proceso:

  Se recibe C LUEGO $P = V_d (C, 1) \pmod N$

  Se obtiene el mensaje P.

1.2.3 Funciones Resumen (O De *Hash*)

1.2.3.1 Definición. Una función resumen o de *hash* H es una transformación que, tomando como entrada una cadena x de bits de longitud variable, produce como salida una cadena h de bits de longitud fija (h = H(x)). Para que una función de este tipo pueda usarse con propósitos criptográficos, se debe cumplir una serie de requisitos:

1 La entrada puede tener cualquier longitud. Deben proveerse mecanismos para evitar el desbordamiento (*overflow*).

2 La salida debe ser de longitud fija, independientemente de cual fuera la longitud de la entrada.

3 Para cualquier entrada, su resumen (o valor de *hash*) debe ser sencillo de calcular.

4 La función resumen debe ser de un "único sentido", entendiendo por este concepto que, dado f(x), debe ser computacionalmente difícil encontrar un valor y (tal vez el mismo x) tal que f(y) = f(x).

5 Es difícil encontrar dos entradas x e y, tales que H(x) = H(y) (colisiones).

Al resumen o valor de *hash* de un mensaje *M* se le llama generalmente **huella digital de *M***. Si la salida de la función tiene una longitud de n bits, entonces existen $k = 2^n$ salidas diferentes. Las funciones resumen son también extensivamente utilizadas como parte de los mecanismos que generan números aleatorios.

Ejemplos de funciones resumen usadas en criptografía son **RIPEMD-128, MD2**, **MD4, RIPEMD-160**, **MD5** o **SHA**.

La función escogida para la aplicación es RIPEMD-160 que genera un resumen de 160 bits razón por lo que es mas difícil de violar que una de 128 bits, no presenta problemas de patentado y se puede utilizar fuera de los Estados Unidos sin ningún inconveniente, además fue propuesta por la persona que ha violado MD4, MD5 y otras funciones hash.

De las funciones resumen conocidas Ripemd-160 aun no ha sido violada.

1.2.3.2 RIPEMD. El consorcio RIPE propuso un portafolio de recomendaciones basado sobre sus evaluaciones independientes de MD4 y MD5, el consorcio propuso una versión fuerte del MD4, la cual fue llamada RIPEMD.

RIPEMD consta esencialmente de dos versiones paralelas de MD4, con algún perfeccionamiento para los desplazamientos y el orden de las palabras del mensaje; las dos instancias paralelas difieren sólo en las constantes de las iteraciones. En el final de la función de comprensión, las mitades de las palabras izquierda y derecha son concatenadas.

1.2.3.2.1 Motivación Para Una Nueva Versión De RIPEMD. La principal contribución de MD4 es que esta es la primera función Hash criptográfica la cual hace optima el uso de la estructura de los procesadores de 32 bits. El uso de operaciones consecutivas y el tratamiento favorable de arquitectura little-endian que muestra como MD4 es llevado hacia implementaciones en software.

Sin embargo introduciendo una nueva estructura en algoritmos criptográficos también compromete el riesgo de debilidades inesperadas. Esto conviene claramente a que existen técnicas tales como el cripto-análisis diferencial o lineal, y que un cripto-análisis exitoso puede requerir el desarrollo de una nueva técnica.

El ataque por B. den Boer y A. Bosselaers en dos iteraciones del MD4 y sobre la función de comprensión de MD5 indican que algunas propiedades de la estructura del algoritmo pueden ser explotadas, pero no provocando una serie de amenazas para el modo de trabajo del algoritmo. Mas recientemente, el ataque sobre MD4 fue desarrollado por S. Vaudenay produciendo dos resultados de Hash diferentes para un mismo mensaje. Los cuales difieren en unos pocos bits.

Anteriormente en 1995 H. Dobbertin encontró colisiones para las ultimas 3 iteraciones de RIPEMD. Esto no fue una amenaza inmediata para RIPEMD

con tres iteraciones, el ataque no fue divulgado, por otra parte, esto introdujo una nueva técnica para cripto-analizar este tipo de funciones.

En la caída de H. Dobbertin se puede extender esa técnica para producir colisiones por MD4.

P. Van Oorschot y M. Wiener presentaron un diseño para una máquina de U$10 millones para buscar una colisión en MD5 en 24 días. Si solamente un presupuesto de un millón de Dólares es disponible, y las memorias de unas computadoras en red son usadas, la computación pude requerir cerca de seis meses. Haciendo provecho del factor que el costo de computación y memoria es divido entre 4 cada tres años (esta observación es conocida como la ley Moore), uno puede concluir que una función Hash de resultado de 128 bits no ofrece suficiente protección para los próximos 10 años.

RIPEMD esta siendo usado en varias aplicaciones bancarias y es usualmente, bajo consideraciones, un candidato para la estandarización dentro de ISO / IEC JTC1/ SC27. Sin embargo, la situación normal nos lleva a la conclusión que esto puede ser prudente para un alto grado de implementaciones, y para considerar un esquema más seguro para estandarización. Por lo tanto el autor diseño una versión fuerte llamada RIPEMD-160 la cual puede ser segura para 10 años o más.

1.2.3.2.2 Descripción De Ripemd-160. Ripemd-160 consiste esencialmente en dos versiones paralelas de MD4, con algunas mejoras para las sustituciones, y el orden de las palabras de los mensajes, las dos instancias paralelas difieren solo en la iteración constante. Al final de la función de compresión, las palabras de la izquierda y derecha son añadidas.

El tamaño de los bits del resultado Hash y la cadena variable para RIPEMD160 son incrementadas para 160 bits (5 palabras de 32 bits), el número de iteraciones es incrementada de 3 a 5, y las dos ultimas son totalmente diferentes (no solamente las constantes son modificadas, sino también las funciones Booleanas y el orden de las palabras del mensaje).

Pasos y operaciones a seguir en el desarrollo del algoritmo:

1. **Operaciones en un solo paso.**

   $A := (A + f(B, C, D) + X + K)^{<< S} + E$ y

   $C := C^{<< 10}$

   << S denota rotación cíclica sobre S posiciones.

2. **Ordenando las Palabras del Mensaje.**

   Se realizan las siguientes permutaciones ρ:

l	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
r (l)	7	4	13	1	10	6	15	3	12	0	9	5	2	14	11	8

Adicionalmente define la permutación π como:

$p(i) = 9*i + 5 \ (mod \ 16).$

LINEA	ITERACIÓN 1	ITERACION 2	ITERACION 3	ITERACION 4	ITERACION 5
Izquierda	Id	ρ	ρ^2	ρ^3	ρ^4
Derecha	π	$\pi\rho$	$\pi\rho^2$	$\pi\rho^3$	$\pi\rho^4$

3. Funciones Booleanas.

Se definen las siguientes funciones Booleanas:

$$f_1 (x, y, z) = x \oplus y \oplus z,$$

$$f_2 (x, y, z) = (x \wedge y) \vee (\neg x \wedge z),$$

$$f_3 (x, y, z) = (x \vee \neg y) \oplus z,$$

$$f_4 (x, y, z) = (x \wedge z) \vee (y \wedge \neg z),$$

$$f_5 (x, y, z) = x \oplus (y \vee \neg z).$$

Estas funciones Booleanas son aplicadas así:

LINEA	ITERACION 1	ITERACION 2	ITERACION 3	ITERACION 4	ITERACION 5
Izquierda	F_1	f_2	f_3	f_4	f_5
Derecha	F_5	f_4	f_3	f_2	f_1

4. Rotaciones.

Para ambas líneas se hacen las siguientes relaciones:

ITERACIÓN	X_0	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	X_{11}	X_{12}	X_{13}	X_{14}	X_{15}
1	11	14	15	12	5	8	7	9	11	13	14	15	6	7	9	8
2	12	13	11	15	6	9	9	7	12	15	11	13	7	8	7	7
3	13	15	14	11	7	7	6	8	13	14	13	12	5	5	6	9
4	14	11	12	14	8	6	5	5	15	12	15	14	9	9	8	6
5	15	12	13	13	9	5	8	6	14	11	12	11	8	6	5	5

5. Constantes.

Hacer la parte entera de los siguientes números:

LINEA	ITERACION 1	ITERACION 2	ITERACION 3	ITERACION 4	ITERACION 5
Izquierda	0	$2^{30} * \sqrt{2}$	$2^{30} * \sqrt{3}$	$2^{30} * \sqrt{5}$	$2^{30} * \sqrt{7}$
Derecha	$2^{30} * \sqrt[3]{2}$	$2^{30} * \sqrt[3]{3}$	$2^{30} * \sqrt[3]{5}$	$2^{30} * \sqrt[3]{7}$	0

1.2.3.2.3 Motivaciones Del Diseño De Ripemd-160. El diseño básico de la filosofía de RIPEMD fue para tener dos iteraciones paralelas; las dos principales mejoras son: que el número de iteraciones son incrementadas de tres a cinco, y que las dos iteraciones paralelas son hechas diferentes. Desde el ataque sobre RIPEMD concluimos que teniendo sólo diferentes constantes de sumas en las dos líneas no es suficiente.

En RIPEMD-160, el orden de los bloques del mensaje en las dos iteraciones es completamente diferente; en la suma, el orden de las funciones Booleanas es reservado. Enfocando que en los próximos años puede ser posible un ataque en una de las dos líneas y en la 4 - 5 iteraciones de las dos líneas paralelas, pero en la combinación de las dos líneas paralelas pueden resistir el ataque.

Las operaciones para RIPEMD-160 sobre el registro A esta relacionado con MD5; la rotación del registro C tiene que ser adicionada para evitar el ataque hecho a MD5 el cual es enfocado al bit más significativo. El valor de 10 para el registro C se puede elegir ya que este no es usado para las otras iteraciones.

Las permutaciones de las palabras del mensaje de RIPEMD esta diseñado tal que dos palabras que están ‘cerradas’ en la iteración 1-2 son separadas en 2-3. Si estas permutaciones tienen que ser aplicadas en RIPEMD-160, este criterio no puede ser satisfecho (bloques del mensaje 2 y 13 desde un indeseable modelo debido a un ciclo de longitud 2). Por lo tanto, esto fue

decidido para cambiar los valores de 12 y 13, resultando en la permutación de ρ.

La permutación π fue escogida tal que dos palabras del mensaje son cerradas en la mitad izquierda pueden siempre ser por lo menos 7 posiciones a partir de la mitad derecha. Para las funciones Booleanas, fue decidido eliminar la mayoría de las funciones porque lo de sus propiedades simétricas eso representa desventajas. Las funciones Booleanas son las mismas usadas en MD5.

La rotación en RIPEMD fue escogida acordando una estrategia especifica, la cual sólo se documenta en un reporte interno.
El diseño del criterio es como sigue:

- Las rotaciones son escogidas entre 5 y 15 (también pequeñas/largas rotaciones son consideradas no muy buenas y una escogencia de 16 no ayuda mucho).
- Cada bloque de mensaje puede ser rotado sobre diferentes cantidades, no tienen la misma paridad.
- La rotación aplicada para cada registro puede no tener un modelo especial (por ejemplo, el total no puede ser divisible por 32).
- No muchas rotaciones constantes deben ser divisibles por 4.

2. FIRMA DIGITAL

2.1 ANTECEDENTES

Dos problemas aquejan a los documentos electrónicos: La Confidencialidad y la Autenticidad.

La confidencialidad se refiere a la capacidad de mantener un documento electrónico inaccesible a todos, excepto a una lista determinada de personas.

La autenticidad se refiere a la capacidad de determinar si una lista determinada de personas han establecido su reconocimiento y/o compromiso sobre el contenido del documento electrónico.

El problema de la autenticidad en un documento tradicional se soluciona mediante la firma autógrafa. Mediante su firma autógrafa, un individuo, o varios, manifiestan su voluntad de reconocer el contenido de un documento, y en su caso, a cumplir con los compromisos que el documento establezca para con el individuo.

Una firma digital es un bloque de caracteres que acompaña a un documento (o fichero), acreditando quién es su autor ("autenticación") y que no ha existido ninguna manipulación posterior de los datos ("integridad").

Para firmar un documento digital, su autor utiliza su propia clave secreta, a la que sólo él tiene acceso, lo que impide que pueda después negar su autoría (" no revocación "). De esta forma, el autor queda vinculado al documento que firma.

Cualquier persona puede verificar la validez de una firma si dispone de la clave pública del autor.

La firma manuscrita como medio para acreditar la identidad del firmante de un documento ha sido, y sigue siendo, ampliamente usada por las sociedades humanas desde hace siglos y debe cumplir con los siguientes requerimientos:

La firma es autentica. La firma convence al recibidor (receptor) que el firmador prudentemente ha firmado el documento.

La firma es no falsificable. La firma es evidencia que el firmante, y no otro, deliberadamente a firmado el documento.

La firma no es reusable. La firma es parte del documento; una persona inescrupulosa no puede mover la firma hacia otro documento.

El documento firmado es inalterable. Después que el documento es firmado, no puede ser alterado.

La firma no puede ser repudiada. La firma y el documento son uno solo. El firmante no pude después denunciar que él o ella no han firmado.

En realidad, ninguna de estas sentencias a cerca de la firma es completamente verdadera. Las firmas pueden ser falsificables, pueden ser quitadas de un papel y movidas a otros y, los documentos pueden ser alterados después de haber sido firmado. Sin embargo, estamos dispuestos a vivir con estos problemas por las dificultades en los engaños y riesgos de detección.

En este capitulo se hablara de la firma digital, su definición, sus requerimientos y, sus funciones de autenticación.

2.2 DEFINICION

“ La firma digital es la versión computarizada de la firma manual.”

Básicamente es el resultado de una transformación de un DOCUMENTO DIGITAL empleando un CRIPTOSISTEMA ASIMETRICO y un RESUMEN SEGURO, de forma tal que una persona que posea el DOCUMENTO DIGITAL inicial y la CLAVE PUBLICA del firmante pueda determinar con certeza:

1 Si la transformación se llevó a cabo utilizando la CLAVE PRIVADA que corresponde a la CLAVE PUBLICA del firmante,

2 Si el DOCUMENTO DIGITAL ha sido modificado desde que se efectuó la transformación.

La conjunción de los dos requisitos anteriores garantiza su NO REPUDIO y su INTEGRIDAD.

La firma digital debe ser :

Unica: Pudiéndola generar solamente el usuario legitimo.

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.

Depende tanto del mensaje como del autor: Debe ser así por que en otro caso el receptor podría modificar el mensaje y mantener la firma, produciendo así un fraude.

2.3 Autenticidad

Requiere que el originador del mensaje esté correctamente identificado, con la certeza de que la identidad no sea falsa.

Este servicio de seguridad protege al receptor del mensaje, garantizándole que dicho mensaje ha sido generado por la parte identificada en el documento como emisor del mismo, no pudiendo alguna otra entidad suplantar a un usuario del sistema. Este servicio de seguridad puede lograrse incluyendo en el mensaje transmitido un valor de autenticación (MAC = Message Autentication Code). El valor depende tanto del contenido del mensaje como de la llave secreta en poder del emisor. Este servicio puede incluir la "Integridad" del contenido del mensaje y puede ser obtenido como un subproducto de la "no-repudiación de origen".

2.4 Privacidad

Requiere que la información en un sistema de computo y la información transmitida sea accesible solamente por lectura para usuarios autorizados.

Es donde la parte receptora del mensaje será la única que pueda descifrar y entender de manera clara dicha información, aunque muchas personas puedan tener acceso a ésta, no podrán entender ni descifrar su contenido.

2.5 Integridad

Requiere que la información en un sistema de computo y la información transmitida solo pueda ser modificada por usuarios autorizados.

La integridad del mensaje es una protección contra la modificación de los datos, en forma intencional o accidental. De esta manera el emisor protege al mensaje, incorporándole a este un valor de control de integridad. Es un valor único, calculado a partir del contenido del mensaje.

El receptor calcula nuevamente este valor y lo compara con el enviado por el emisor. De coincidir, se concluye que el mensaje no ha sido modificado durante la transferencia. La integridad del contenido del mensaje se obtiene como un subproducto de la "Autenticidad del origen del mensaje" o de la "No-repudiación de origen".

Existe una diferencia sutil pero muy importante entre el concepto de autenticidad y el concepto de no-repudiación. Por ejemplo usted puede presenciar que un documento fue escrito por alguien pues lo vio en persona.

Si el documento no esta firmado autógrafamente usted estará absolutamente convencido de su autenticidad pero no podrá probarlo pues sin la firma autógrafa es imposible establecer él vinculo entre la voluntad de la persona y el contenido del documento. Si se puede probar a terceros que efectivamente el documento es autentico entonces se dice que el documento es no repudiable. Si un documento es no-repudiable es autentico pero no viceversa.

2.6 DISEÑO CRIPTOGRAFICO DE LA APLICACION FIRMA DIGITAL.

2.6.1 Diseño Base Del Proceso De Firma.

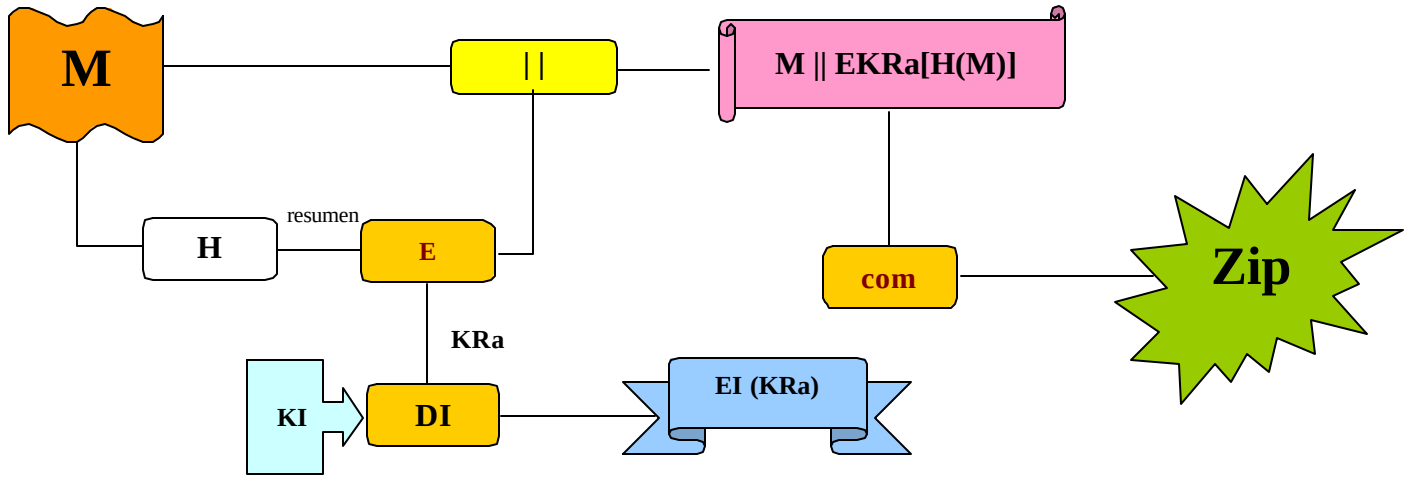


Figura 2 Proceso De Firma.

2.6.2 Diseño Base Del Proceso Revisado De Firma.

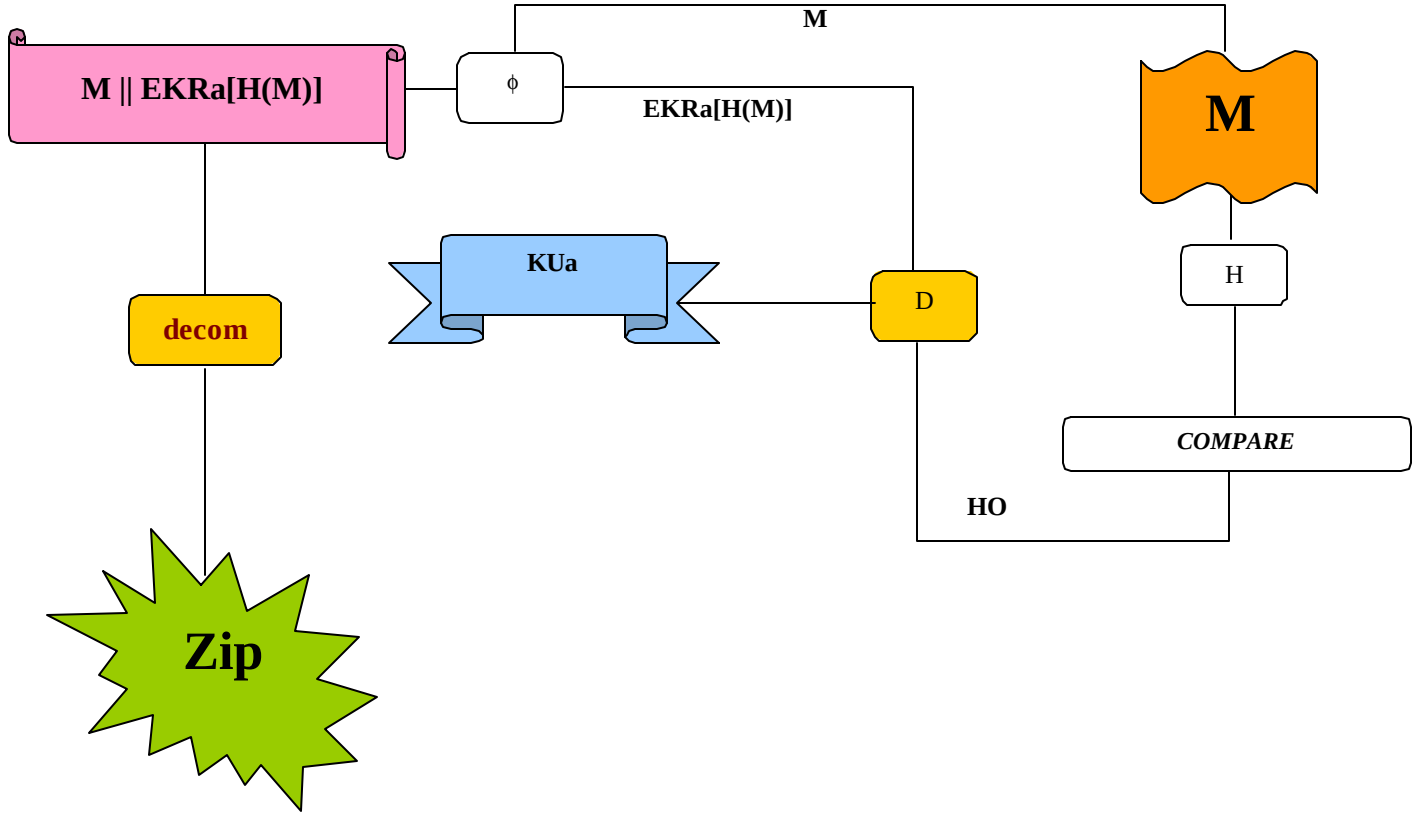


Figura 3. Proceso Revisado De Firma

2.6.3 Diseño Base Del Proceso De Firmado Y Cifrado.

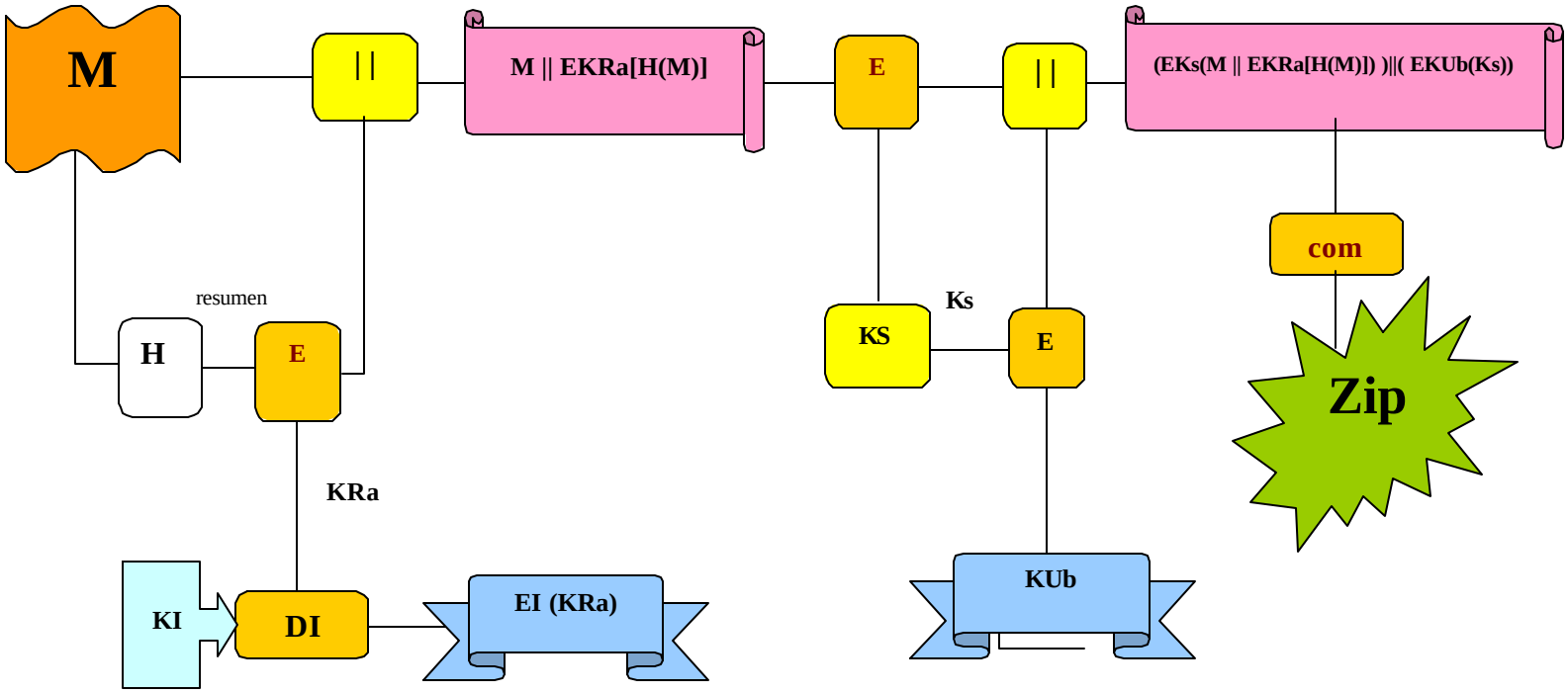


Figura 4. Proceso De Firmado y Cifrado

2.6.4 Diseño Base Proceso De Aclarar Documento Firmado Y Cifrado.

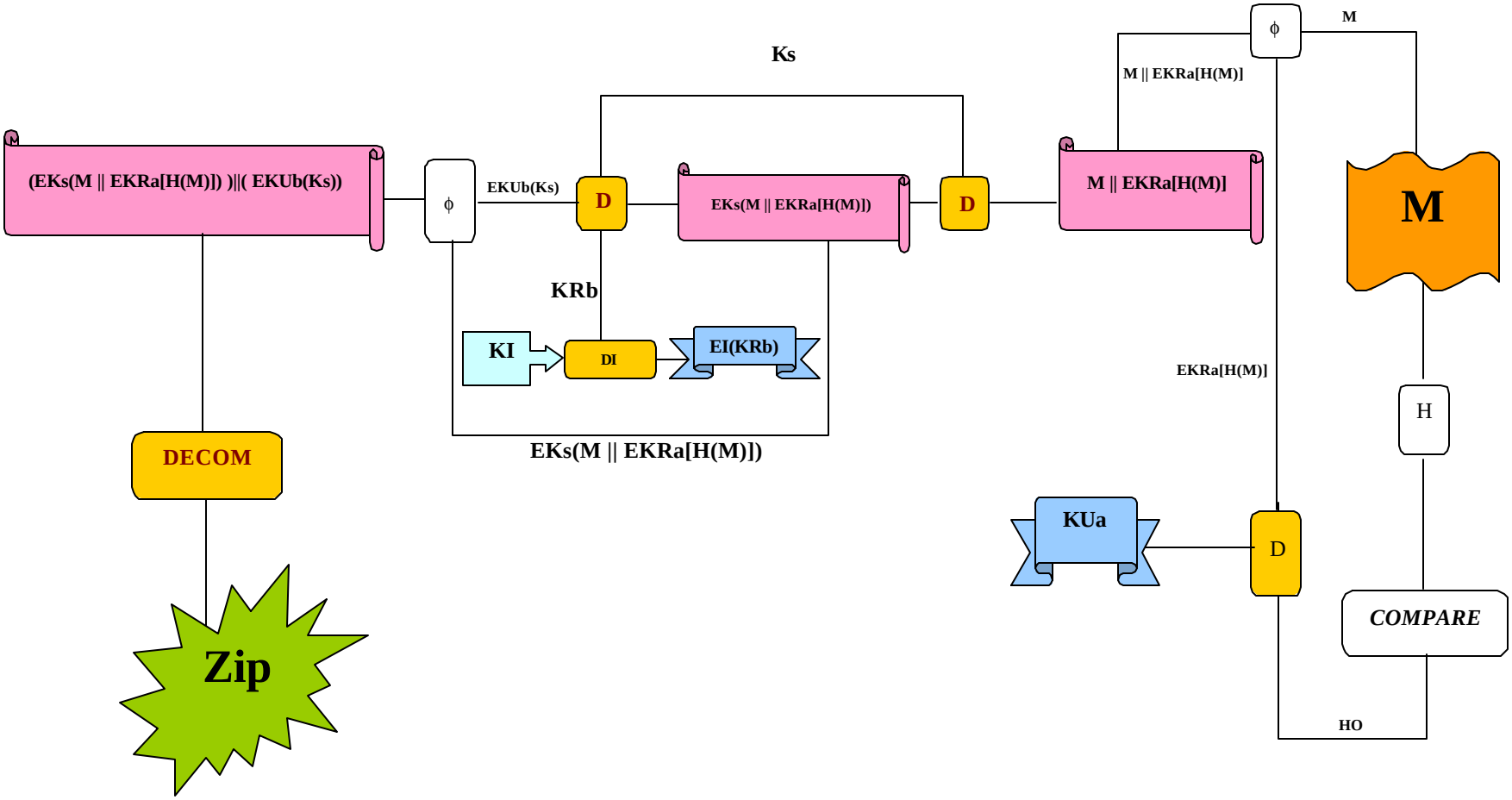
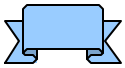


Figura 5. Proceso Aclarar Documento Firmado y Cifrado

2.6.5 Nomenclatura Utilizada:

- Clave IDEA Introducida manualmente


- Clave recuperada de archivo

M - Mensaje original

H - Función Hash.

E - Proceso de Encripción.

|| - Proceso de concatenación.


- Almacenamiento

KRa - Clave Privada de A.

KUa - Clave Publica de A.

KRb - Clave Privada de B.

KUb - Clave Publica de B.

D - Proceso de Desencripción.

DI - Proceso de Desencripción con IDEA

Ks - Clave de sesión.

COMPARE - Comparación de 2 resúmenes.

HO - Resumen obtenido, guardado.

H(M) - Resumen obtenido por la Función Hash del mensaje.

ϕ - Proceso de separación.

COM - Comprimir archivo

DECOM - Descomprimir Archivo

Zip - Archivo comprimido almacenado

2.6.6 Descripción Del Diseño Base Del Proceso De Firma.

Se obtiene el mensaje inicial M.

Se genera la función Hash por medio del algoritmo RIPEMD-160.

Se desencripta la función privada Lucas la cual fue encriptada con anterioridad utilizando el cifrador IDEA

Se encripta la función Hash obtenida, con la clave privada Lucas del emisor, esta clave es generada por medio del algoritmo de Lucas.

Se concatena el mensaje original y la función Hash Encriptada.

Se comprime con el algoritmo Zip.

El resultado obtenido es enviado al receptor por algún medio de transmisión.

2.6.7 Descripción Del Diseño Base Del Proceso Revisado De Firma.

1. Se descomprime con el algoritmo Zip.
2. Se obtiene el mensaje firmado.
3. Se separa el mensaje original y el resumen cifrado recibido.
4. Con la clave publica Lucas del emisor sé desencripta la función Hash original.
5. Se le halla la función Hash al mensaje original, obtenido en el paso uno.
6. Se comparan los dos resúmenes, para saber si el documento es autentico.

2.6.8 Descripción Del Diseño Base Del Proceso De Firmado Y Cifrado.

1. Se obtiene el mensaje inicial M.
2. Se genera la función Hash por medio del algoritmo RIPEMD-160.
3. Se desencripta la función privada Lucas que fue encriptada con anterioridad utilizando el cifrador IDEA
4. Se encripta la función Hash obtenida, con la clave privada Lucas del emisor, esta clave es generada por medio del algoritmo de Lucas.
5. Se concatena el mensaje original y la función Hash Encriptada.
6. Se genera la clave de sesión, este valor es obtenido teniendo en cuenta la hora en milisegundos a los cuales se la aplica una función matemática, para generar una clave de 128 bits.
7. Sé encripta lo obtenido en el paso cuatro por medio del bloque cifrador IDEA el cual trabaja con la clave de 128 bits obtenida en el paso seis.
8. La clave de sesión es encriptada con la clave publica del receptor utilizando Lucas.
9. Se concatena lo obtenido en el paso siete y ocho.
10. Se comprime con el algoritmo Zip
11. El resultado obtenido es enviado al receptor por algún medio de transmisión.

2.6.9 Descripción Del Diseño Base Del Proceso De Aclarar Documento Firmado Y Cifrado.

1 Se descomprime con el algoritmo zip.

2 Se obtiene el mensaje recibido que se encuentra encriptado.

3 Se separa la clave de sesión cifrada del resto del paquete cifrado.

4 Se desencripta la clave privada Lucas por medio del cifrador IDEA

5 La clave de sesión cifrada es desencriptada con la clave privada Lucas y utilizando el proceso de desencripción LUCAS.

6 Con la clave de sesión se desencripta la otra parte del mensaje recibido, obteniendo el mensaje original y la función Hash que fue encriptada con la clave privada Lucas del emisor.

7 Se separa el mensaje original y el resumen cifrado recibido.

8 Con la clave publica Lucas del emisor sé desencripta la función Hash original.

9 Se le halla la función Hash al mensaje original, obtenido en el paso tres.

10 Se comparan los resultados obtenidos en el paso siete y ocho, para determinar si el documento es autentico.

2.7.3 DISEÑO DE LA APLICACIÓN FIRMA DIGITAL

2.7.1 Diagrama Jerárquico De Procesos.

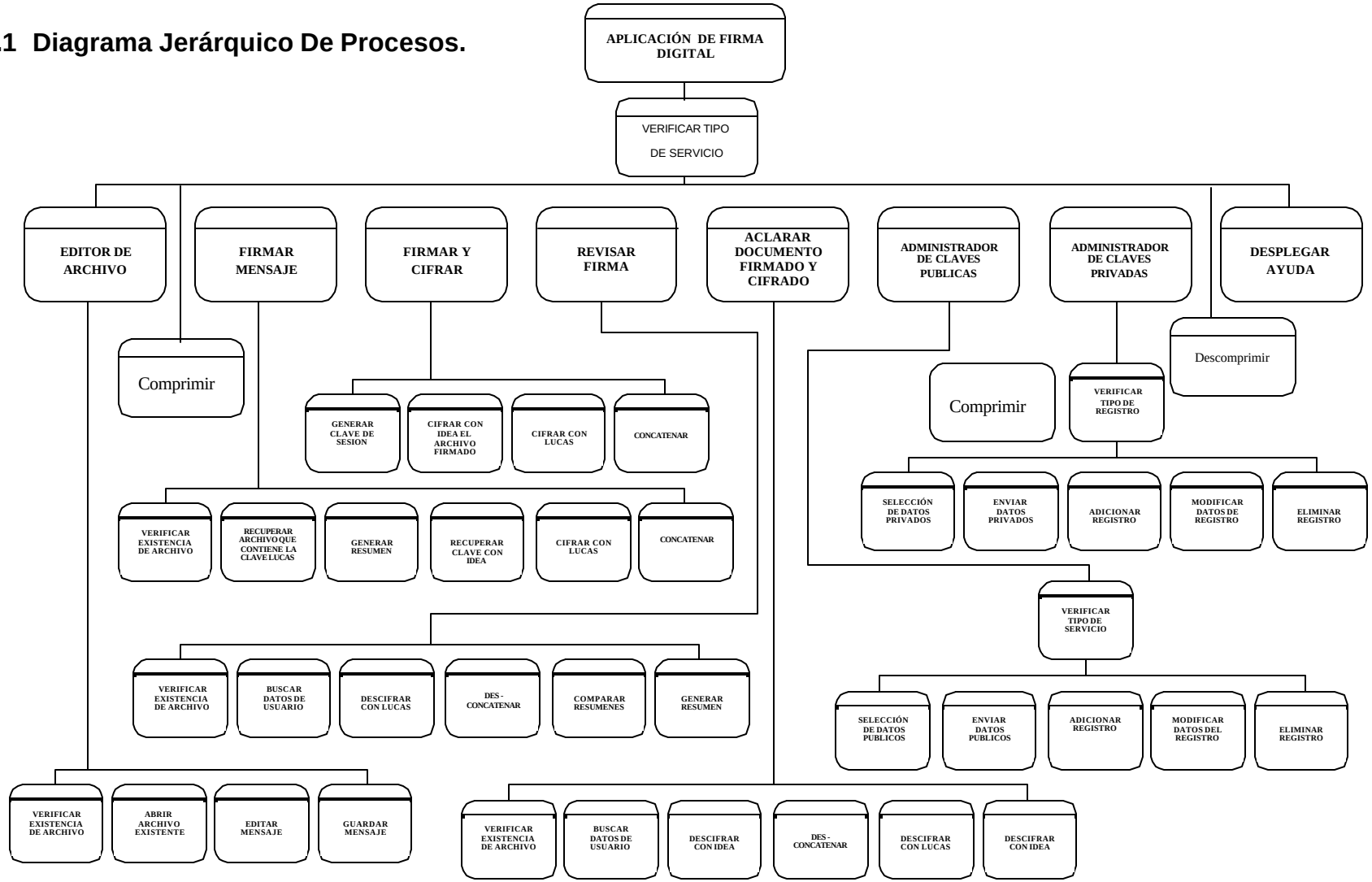


Figura 6. Diagrama Jerárquico De Procesos.

2.7.2 Diagramas De Flujos De Datos De La Aplicación.

2.7.2.1 Nivel De Contexto

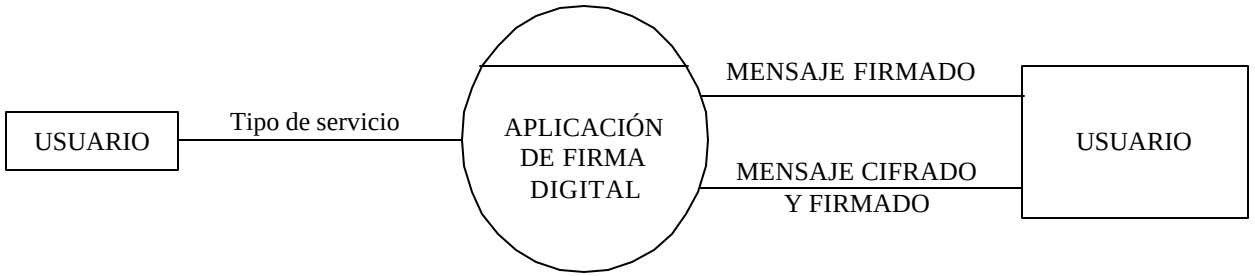


Figura 7. Nivel De Contexto

2.7.2.2Primer Nivel

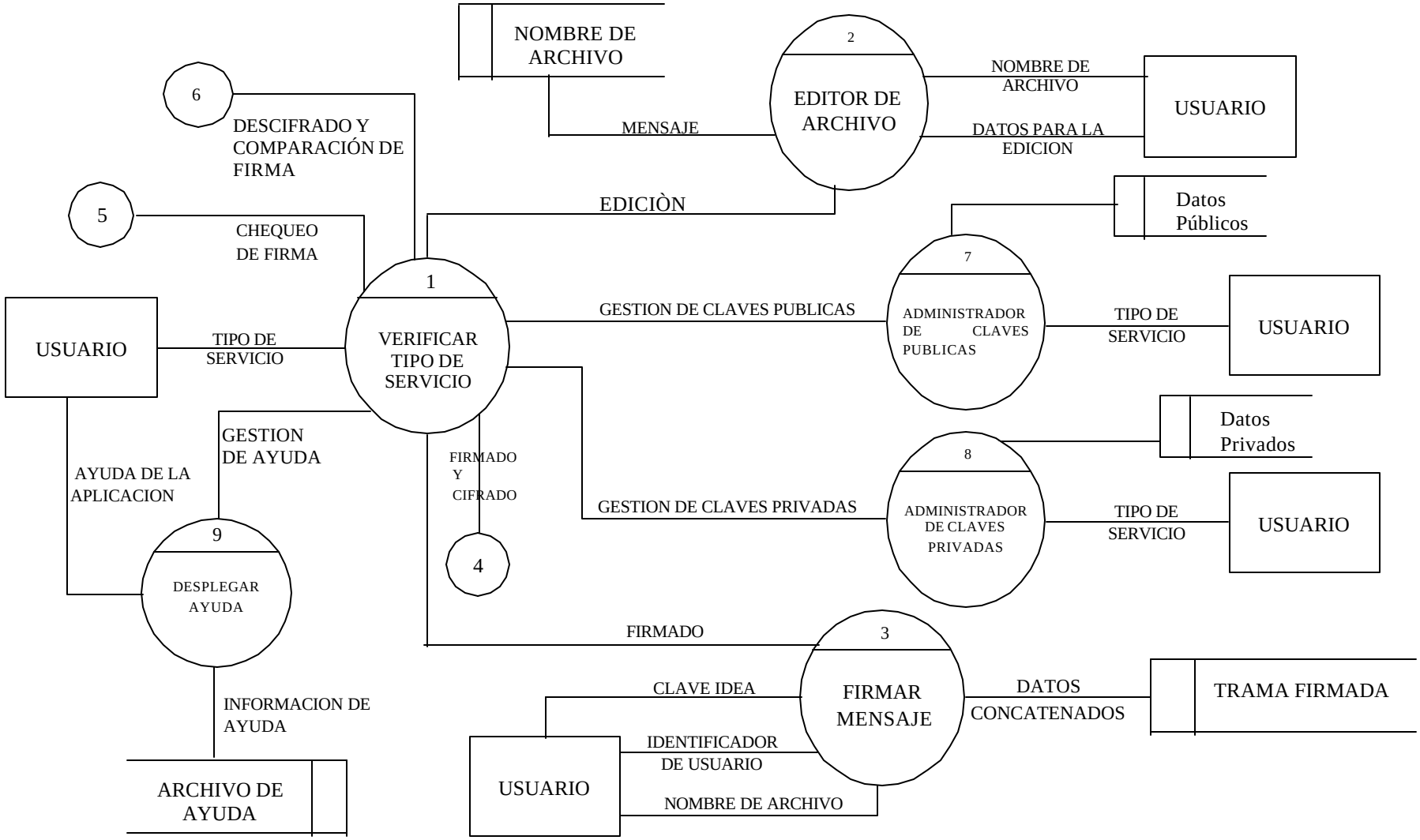


Figura 8. Primer Nivel

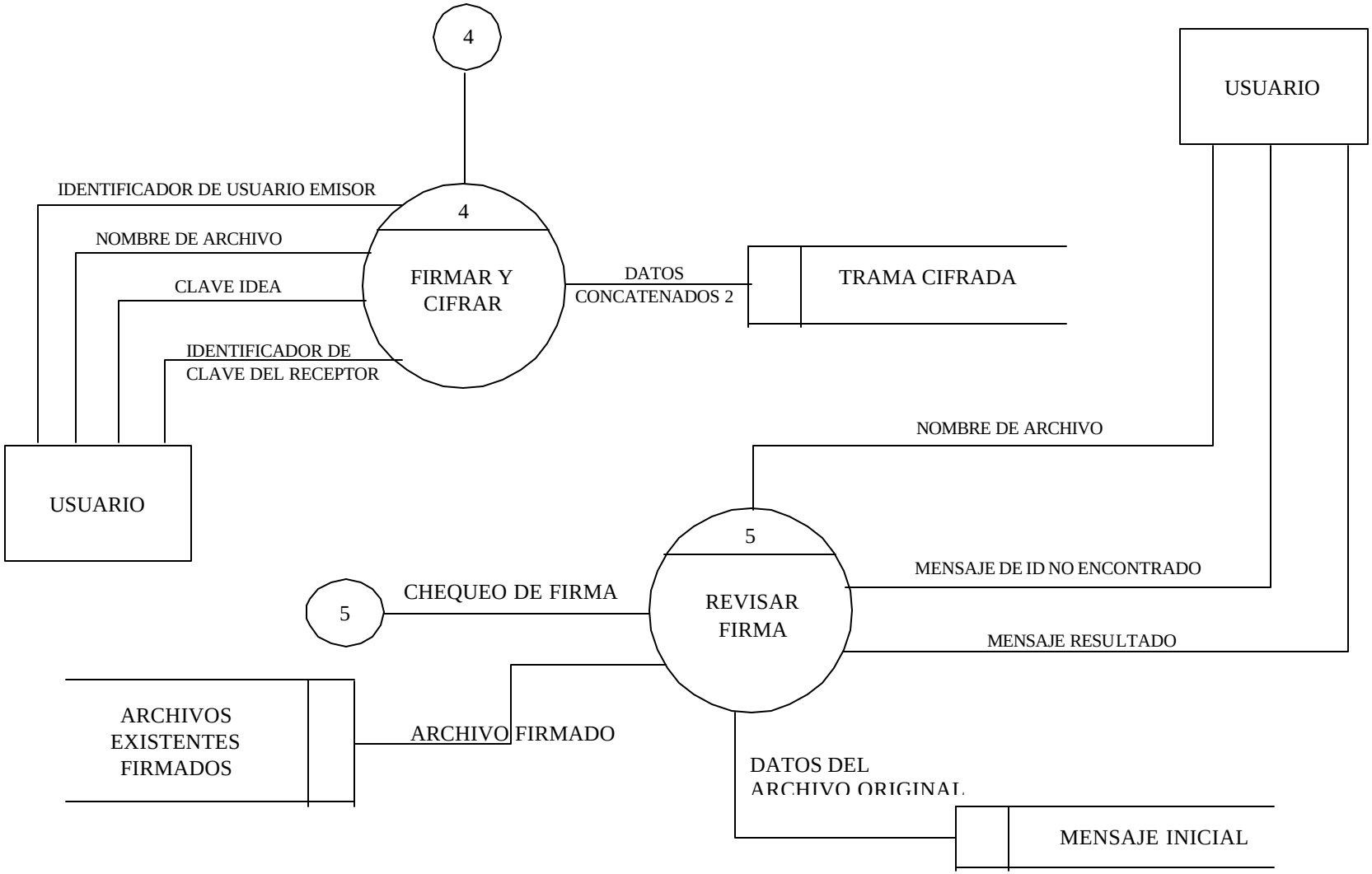


Figura 9. Continuación Primer Nivel

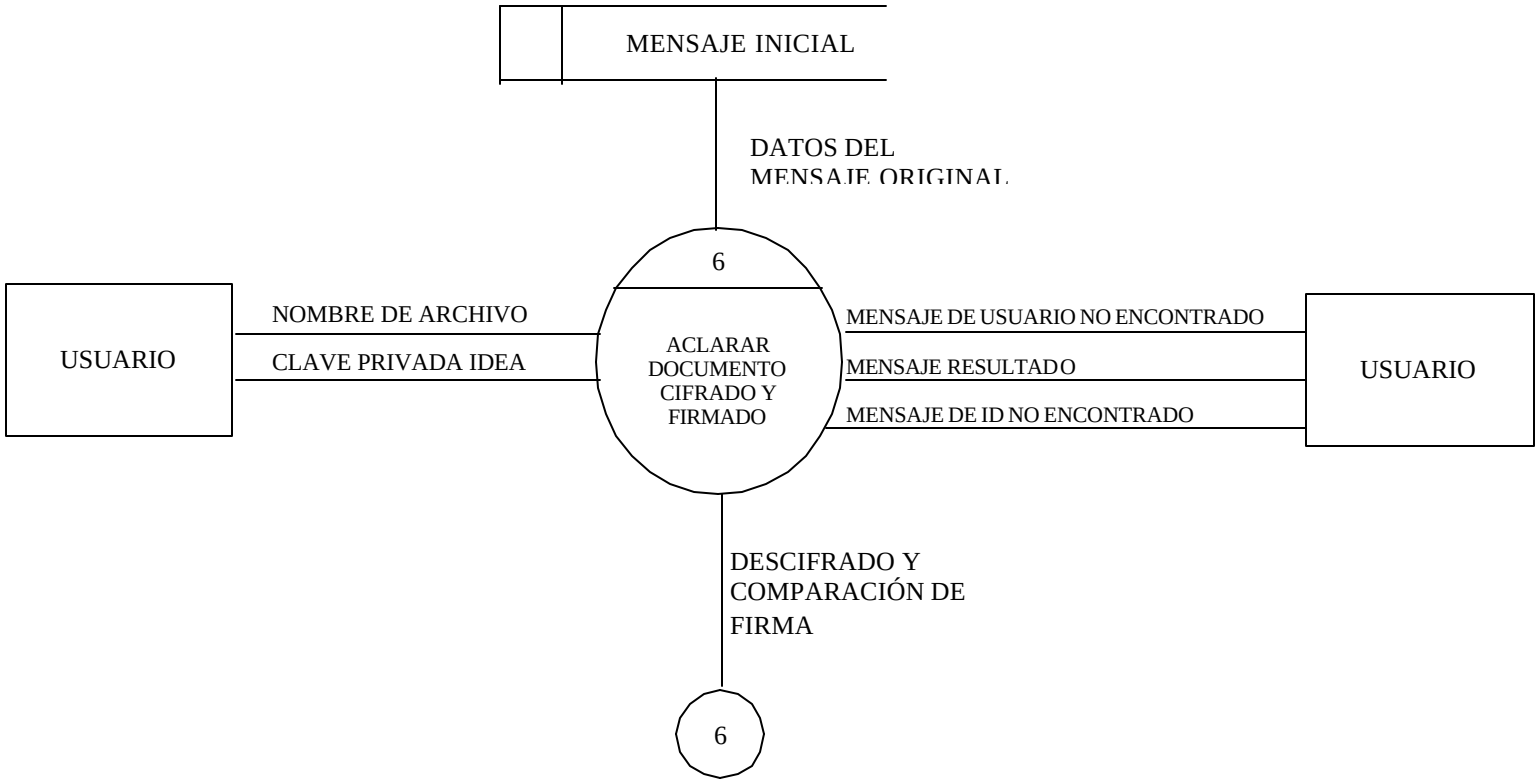


Figura 10. Continuación Primer Nivel

2.7.2.3 Diagrama Del Nivel 2 Del Proceso Editor De Archivo.(PROCESO 2)

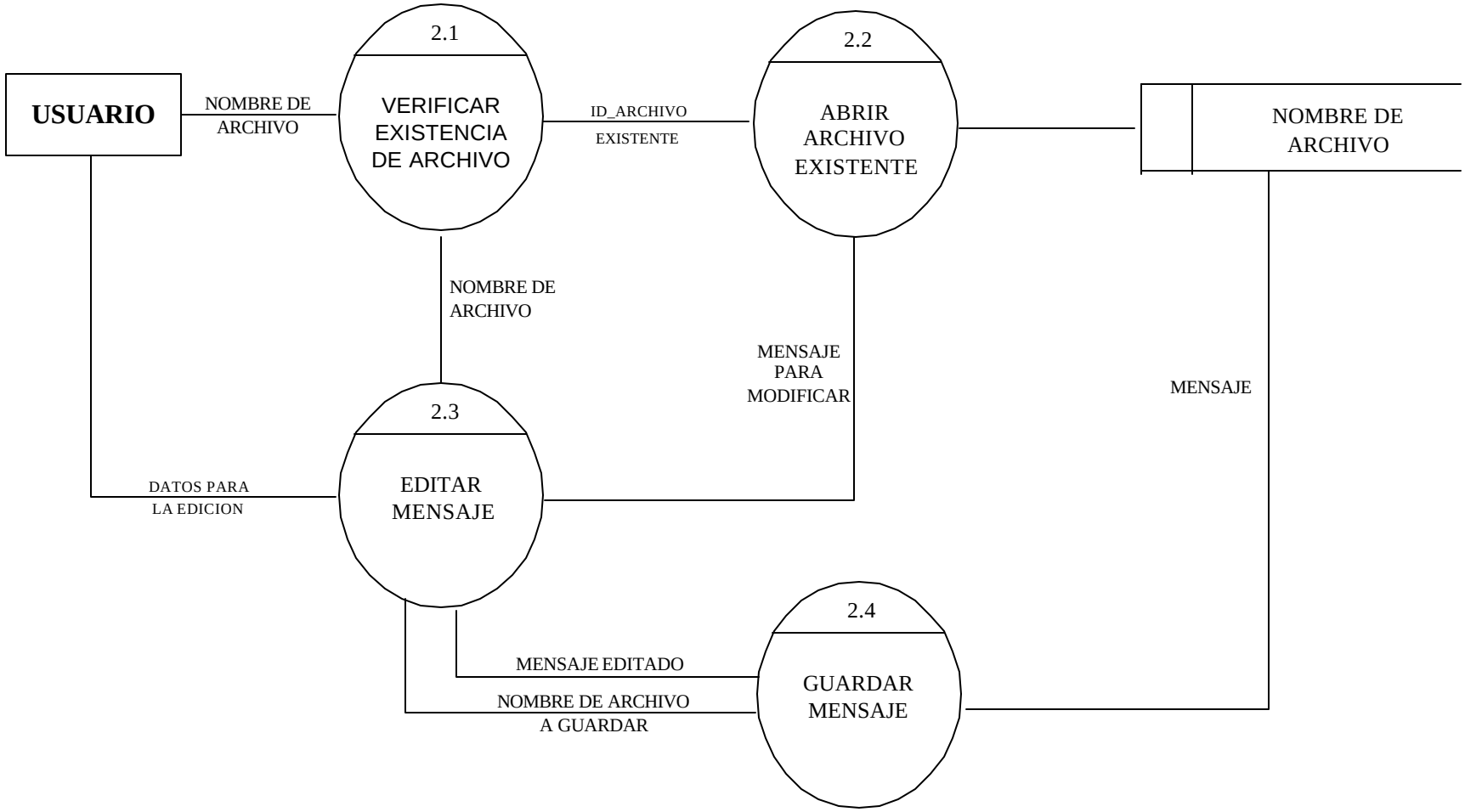


Figura 11. Proceso Editor De Archivo (Proceso 2)

2.7.2.4 DIAGRAMA DEL NIVEL 2 DEL PROCESO FIRMAR MENSAJE (PROCESO 3)

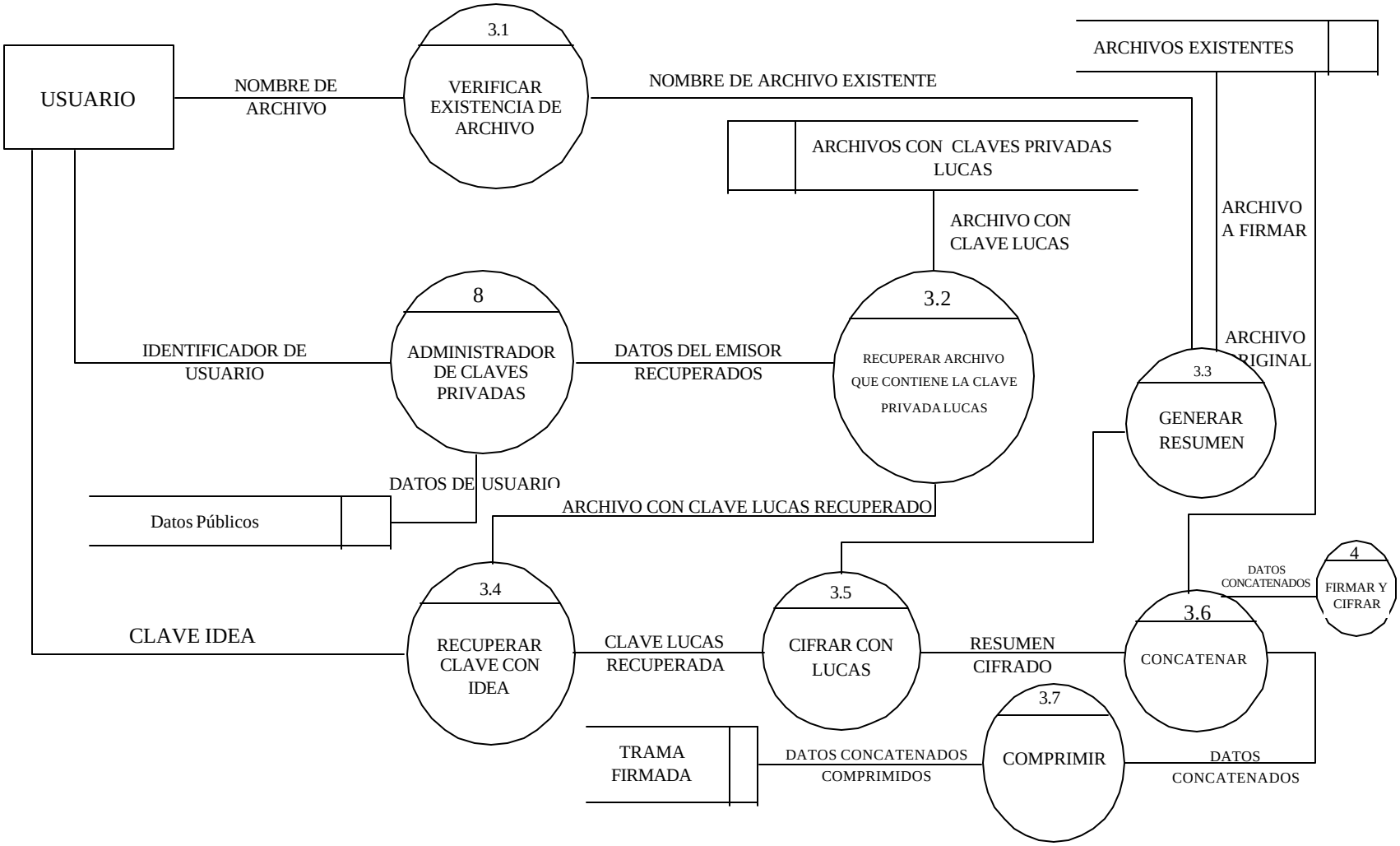


Figura 12. Proceso Firmar Mensaje (Proceso 3)

2.7.2.5 DIAGRAMA DEL NIVEL 2 DEL PROCESO FIRMAR Y CIFRAR MENSAJE (PROCESO 4)

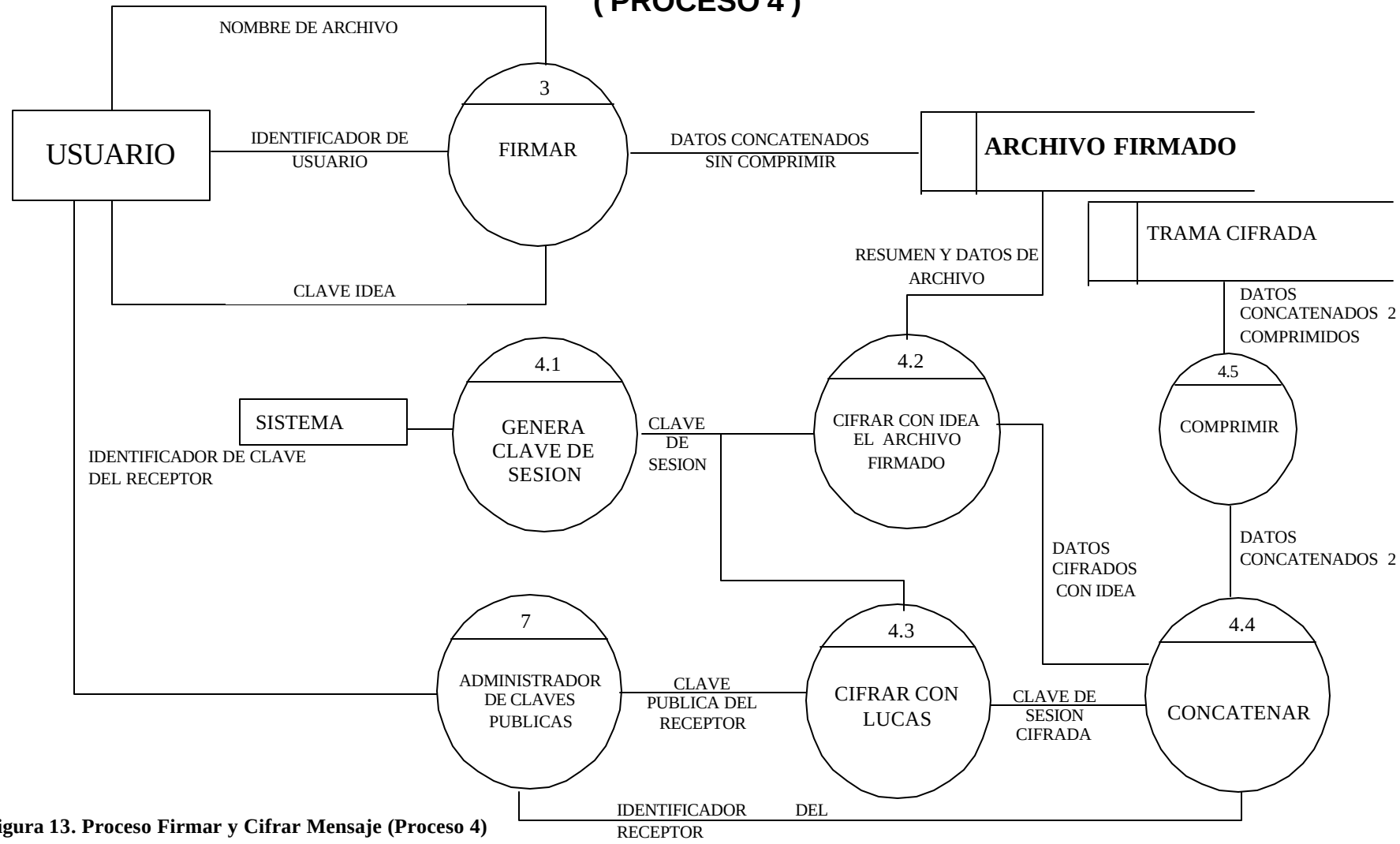


Figura 13. Proceso Firmar y Cifrar Mensaje (Proceso 4)

2.7.2.6 DIAGRAMA DE SEGUNDO NIVEL DEL PROCESO REVISAR FIRMA (PROCESO 5)

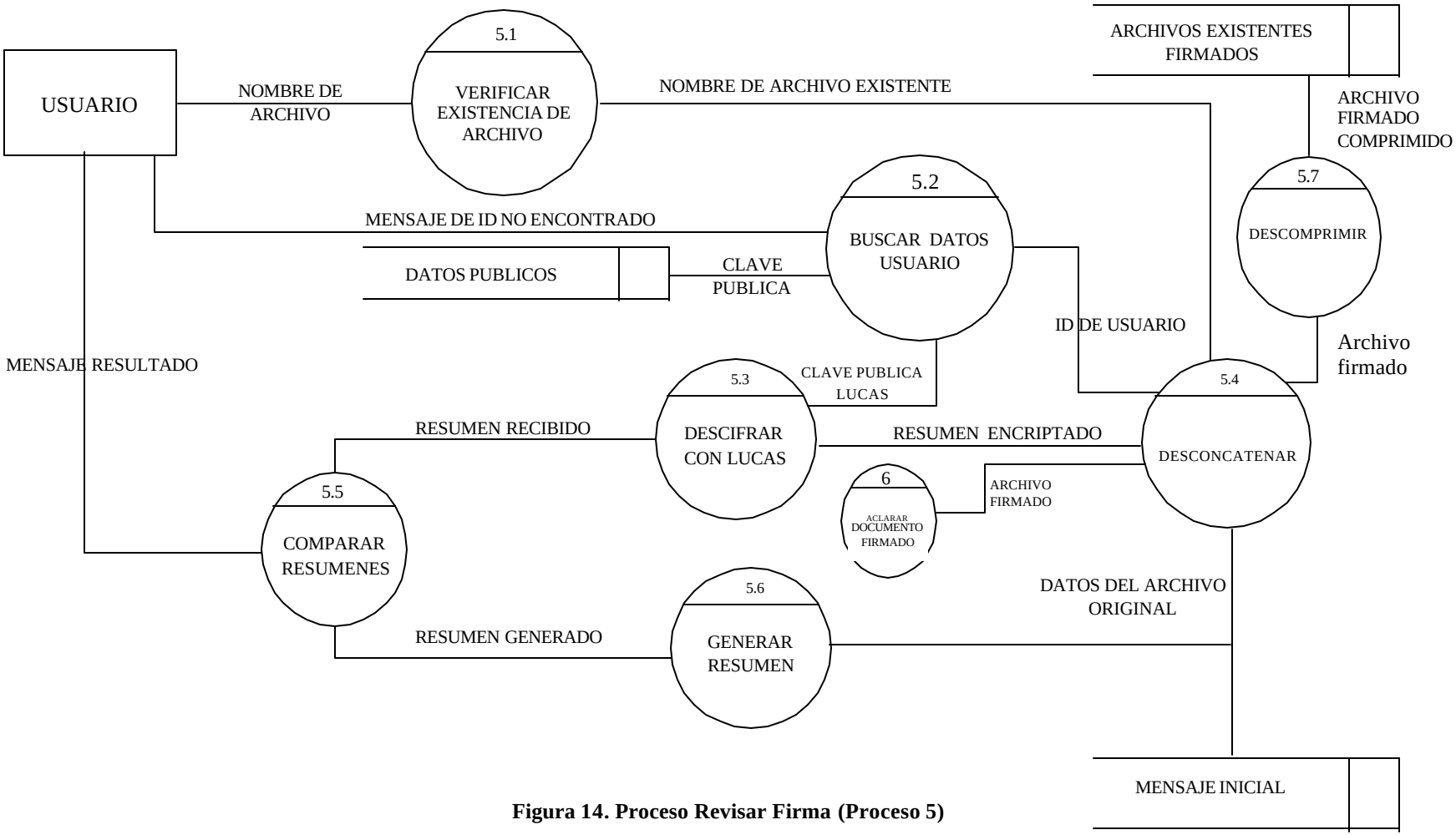


Figura 14. Proceso Revisar Firma (Proceso 5)

2.7.2.7 DIAGRAMA DE SEGUNDO NIVEL DEL PROCESO ACLARAR DOCUMENTO FIRMADO Y CIFRADO (PROCESO 6)

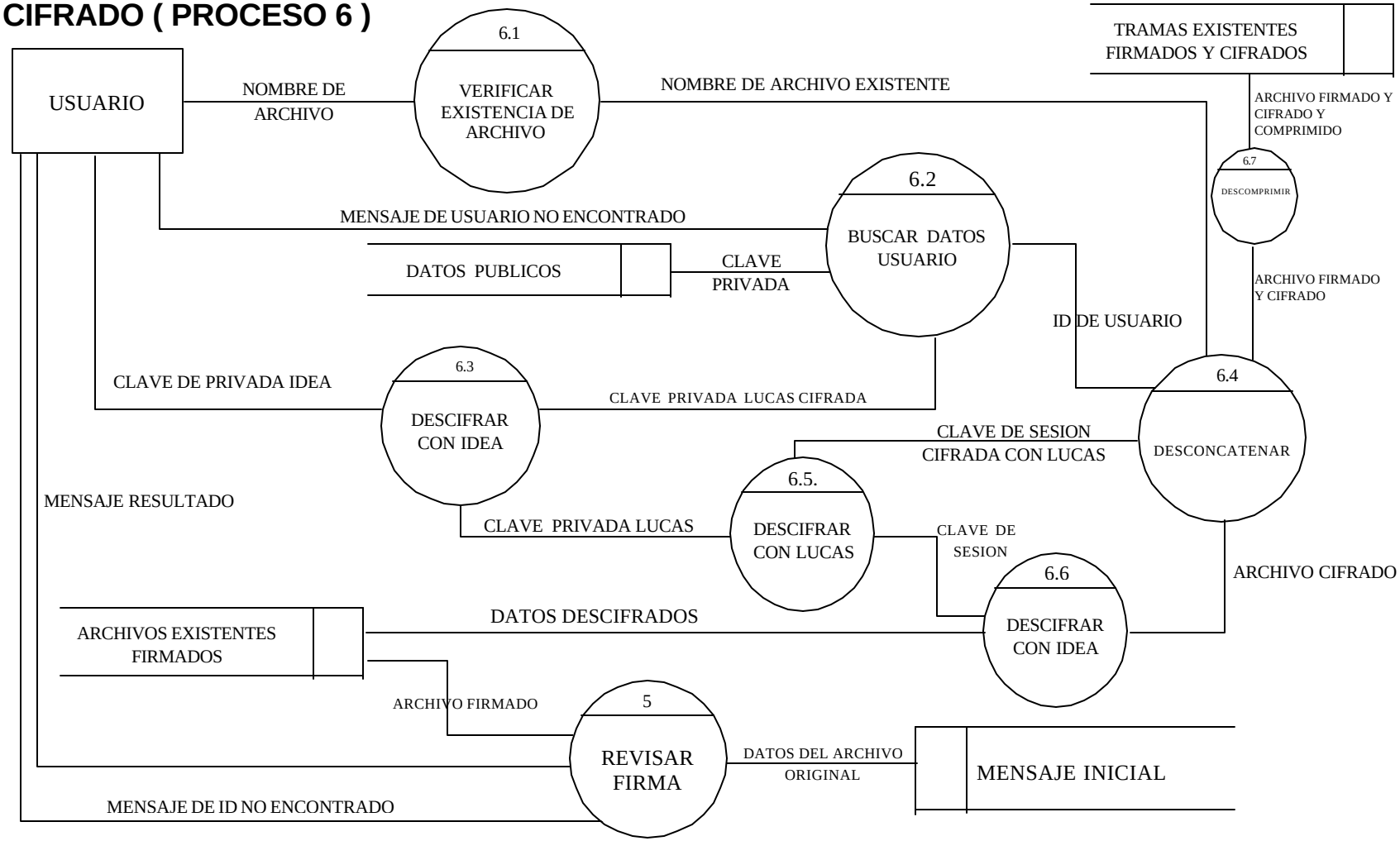


Figura 15. Proceso Aclarar Documento Firmado y Cifrado (Proceso 6)

2.7.2.8 DIAGRAMA DE SEGUNDO NIVEL DEL PROCESO ADMINISTRADOR DE CLAVES PUBLICAS (PROCESO 7)

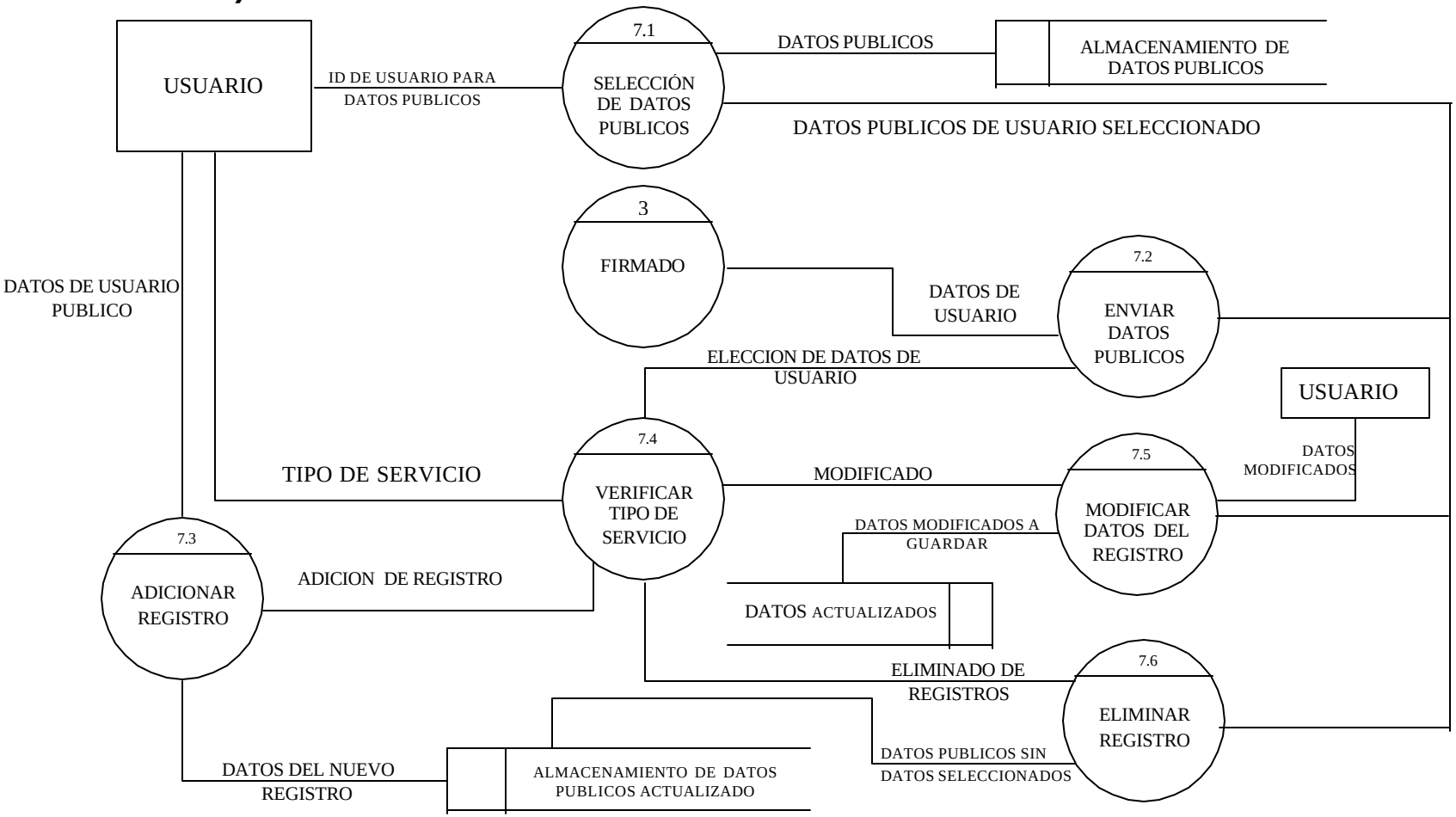


Figura 16. Proceso Administrador De Clave Publica (Proceso 7)

2.7.2.9 DIAGRAMA DE SEGUNDO NIVEL DEL PROCESO ADMINISTRADOR DE CLAVES PRIVADAS (PROCESO 8)

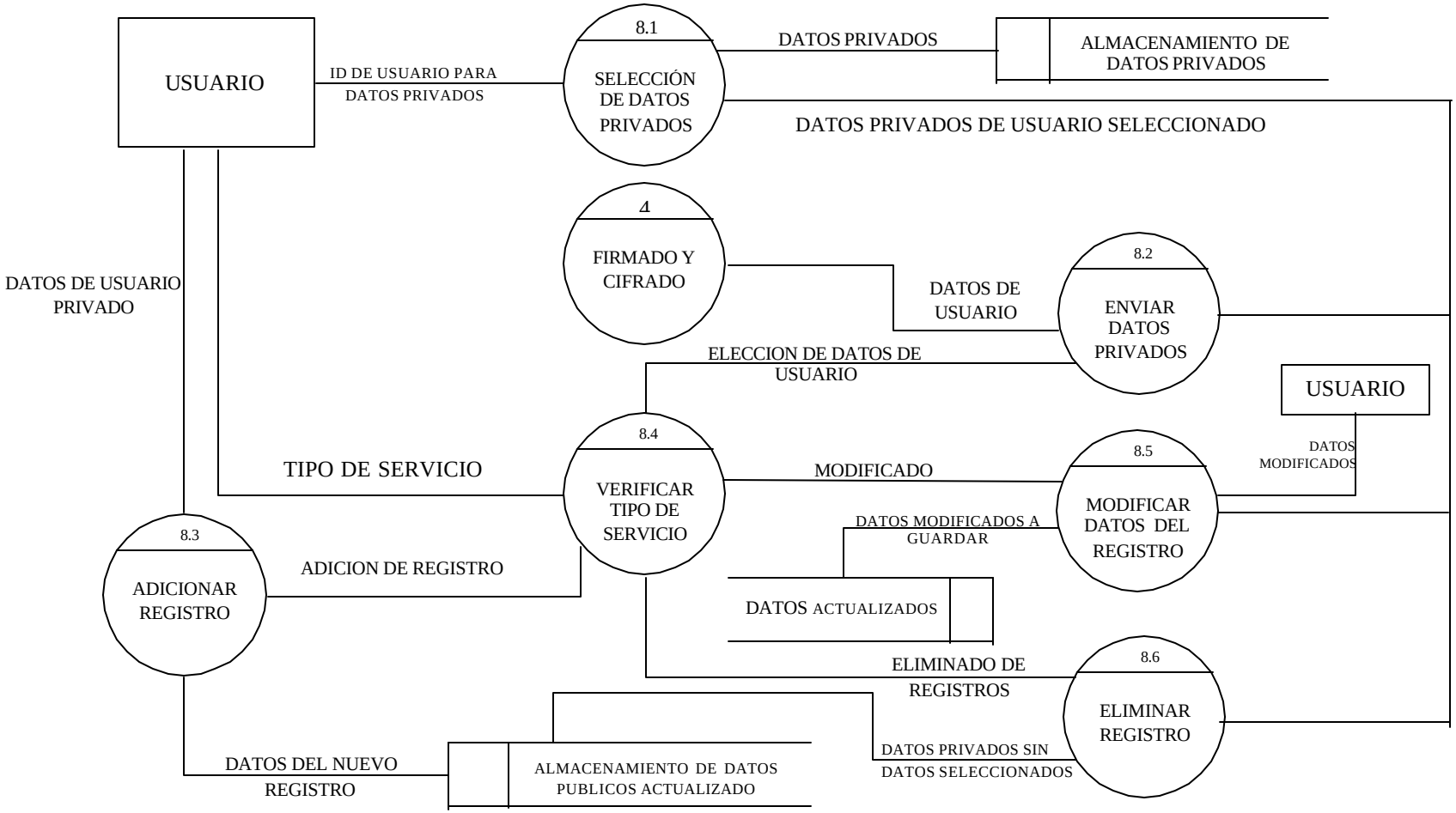


Figura 17. Proceso Administrador De Claves Privadas (Proceso 8)

2.7.3 Diccionario De Datos.

2.7.3.1 Flujos De Datos.

Flujo 1

Nombre	**TIPO DE SERVICIO**
Descripción	Servicio solicitado para que desarrolle la aplicación.
Para los procesos	1. Verificar de tipo de servicio.

Flujo 2

Nombre	**EDICIÓN**
Descripción	Indica a la aplicación que preste el servicio editar archivo.
Proveniente de los procesos	1. Verificar tipo de servicio
Para los procesos	2. Editor de archivo

Flujo 3

Nombre	**FIRMADO**
Descripción	Indica a la aplicación que preste el servicio firmar mensaje.
Proveniente de los procesos	1. Verificar tipo de servicio
Para los procesos	3. Firmar mensaje

Flujo 4

Nombre	**FIRMADO Y CIFRADO**
Descripción	Indica a la aplicación que preste el servicio de firmar y cifrar.
Proveniente de los procesos	1. Verificar tipo de servicio
Para los procesos	4. Firmar y Cifrar.

Flujo 5

Nombre	**CHEQUEO DE FIRMA**
Descripción	Indica a la aplicación que preste el servicio de revisar firma.
Proveniente de los procesos	1. Verificar tipo de servicio
Para los procesos	5. Revisar firma.

Flujo 6

Nombre	**DESCIFRADO Y COMPARACIÓN DE FIRMA**
Descripción	Indica a la aplicación que preste el servicio de aclarar documento firmado y cifrado.
Proveniente de los procesos	1. Verificar tipo de servicio
Para los procesos	6. Firmar y cifrar

Flujo 7

Nombre	**GESTIÓN DE CLAVES PUBLICAS**
Descripción	Indica a la aplicación que preste el servicio para administrar claves publicas.
Proveniente de los procesos	1. Verificar tipo de servicio
Para los procesos	7. Administrador de claves publicas

Flujo 8

Nombre	GESTIÓN DE CLAVES PRIVADAS
Descripción	Indica a la aplicación que preste el servicio para administrar claves privadas.
Proveniente de los procesos	1. Verificar tipo de servicio.
Para los procesos	8. Administrador de claves privadas
Estructura de datos	Claves Publicas

Flujo 9

Nombre	GESTIÓN DE AYUDA
Descripción	Indica a la aplicación que despierte el proceso de ayuda.
Proveniente de los procesos	1. Verificar tipo de servicio
Para los procesos	9. Desplegar ayuda.

Flujo 10

Nombre	DATOS PUBLICOS.
Descripción	Contiene los datos de usuario que actualizaran los datos guardados.
Proveniente de los procesos	7. Administrador de Claves Publicas

Flujo 11

Nombre	DATOS PRIVADOS.
Descripción	Contiene los datos privados de usuario que actualizaran los datos guardados.
Proveniente de los procesos	8. Administrador de Claves.Privadas

Flujo 12

Nombre	AYUDA DE LA APLICACIÓN.

Descripción	Contiene toda la ayuda de la aplicación, que será desplegada en el navegador predeterminado del sistema.
Proveniente de los procesos	9. Desplegar ayuda.

Flujo 13

Nombre	**NOMBRE DE ARCHIVO**
Descripción	Contiene el nombre del archivo sobre el cual el usuario desea solicitar un servicio.
Proveniente de los procesos	2.1 Verificar existencia de archivo 3.1 Verificar existencia de archivo 5.1 Verificar existencia de archivo 6.1 verificar existencia de archivo
Para los procesos	2.3 Editar mensaje

Flujo 14

Nombre	**ID_ARCHIVO EXISTENTE**
Descripción	Contiene el identificador que indica que un archivo existe.
Proveniente de los procesos	2.1 VERIFICAR EXISTENCIA DE ARCHIVO. VERIFICAR EXISTENCIA DE ARCHIVO. 5.1 VERIFICAR EXISTENCIA DE ARCHIVO 6.1 VERIFICAR EXISTENCIA DE ARCHIVO
Para los procesos	2.2 ABRIR ARCHIVO EXISTENTE. - GENERAR RESUMEN 5.4 DESCONCATENAR 6.4 DESCONCATENAR

Flujo 15

Nombre	**CONTENIDO DE ARCHIVO.**

Descripción	Son los datos que están guardados en un archivo.
Para los procesos	2.2 ABRIR ARCHIVO EXISTENTE

Flujo 16

Nombre	**MENSAJE PARA MODIFICAR**
Descripción	Son los datos leídos de un archivo y que podrían ser modificados por el usuario.
Proveniente de los procesos	2.2 ABRIR ARCHIVO EXISTENTE
Para los procesos	2.3 EDITAR MENSAJE

Flujo 17

Nombre	**DATOS PARA LA EDICIÓN.**
Descripción	Caracteres ASCII que inserta el usuario para ser editados.
Para los procesos	2.3 EDITAR MENSAJE

Flujo 18

Nombre	**MENSAJE EDITADO**
Descripción	Datos insertados editados y listos a ser guardados.
Proveniente de los procesos	2.3 MENSAJE EDITADO
Para los procesos	2.4 GUARDAR MENSAJE

Flujo 19

Nombre	**NOMBRE DE ARCHIVO A GUARDAR**
Descripción	Es el nombre del archivo donde se

	guardaran datos.
Proveniente de los procesos	2.3 EDITAR MENSAJE.
Para los procesos	2.4 GUARDAR MENSAJE.

Flujo 20

Nombre	**MENSAJE**
Descripción	Datos definitivos que se guardaran en el archivo.
Proveniente de los procesos	2.4 GUARDAR MENSAJE.

Flujo 21

Nombre	**ARCHIVO A FIRMAR.**
Descripción	Conjunto de bytes que conforman el archivo que se firmara.
Para los procesos	3.3 GENERAR RESUMEN

Flujo 22

Nombre	**ARCHIVO ORIGINAL.**
Descripción	Conjunto de datos iniciales a los cuales no se le han aplicado ninguna transformación.
Para los procesos	3.6 CONCATENAR

Flujo 23

Nombre	**ARCHIVO CON CLAVE LUCAS**
Descripción	Clave privada Lucas cifrada con IDEA.
Para los procesos	3.2 RECUPERAR ARCHIVO QUE

	CONTIENE LA CLAVE LUCAS
Estructuras de datos	ARCHIVO CON CLAVE PRIVADA CIFRADA

Flujo 24

Nombre	**DATOS DEL EMISOR RECUPERADOS.**
Descripción	Conjunto de datos privados que identifican al usuario con su par de claves publica y privada.
Proveniente de los procesos	8. ADMINISTRADOR DE CLAVES PRIVADAS.
Para los procesos	3.2 RECUPERAR ARCHIVO QUE CONTIENE LA CLAVE PRIVADA LUCAS
Estructuras de datos	DATOS PUBLICOS

Flujo 25

Nombre	**DATOS DE USUARIO.**
Descripción	Conjunto de datos privados guardados.
Para los procesos	8. ADMINISTRADOR DE CLAVES PRIVADAS.
Estructuras de datos	DATOS PRIVADOS

Flujo 26

Nombre	**CLAVE IDEA**
Descripción	Conjunto de caracteres que representan la clave IDEA con la que fue cifrado el archivo que contiene la clave privada Lucas.

Para los procesos	3.4 RECUPERAR CLAVE CON IDEA 6.3 DESCIFRAR CON IDEA

Flujo 27

Nombre	**CLAVE LUCAS RECUPERADA.**
Descripción	Clave privada Lucas en su formato numérico.
Proveniente de los procesos	3.4 RECUPERAR CLAVE CON IDEA
Para los procesos	CIFRAR CON LUCAS
Estructuras de datos	CLAVE LUCAS

Flujo 28

Nombre	**RESUMEN**
Descripción	160 bits que son el resultado de la función hash RIPEMD-160.
Proveniente de los procesos	3.3 GENERAR RESUMEN
Para los procesos	3.5 CIFRAR CON LUCAS

Flujo 29

Nombre	**RESUMEN CIFRADO**
Descripción	Resumen encriptado con la clave privada Lucas.
Proveniente de los procesos	3.5 CIFRAR CON LUCAS
Para los procesos	3.6 CONCATENAR

Flujo 30

Nombre	**DATOS CONCATENADOS**
Descripción	Son los datos que se guardaran en un archivo y que conforman el archivo firmado.
Proveniente de los procesos	3.6 CONCATENAR
Para los procesos	3.7 Comprimir
Estructuras de datos	ARCHIVO FIRMADO

Flujo 31

Nombre	**DATOS CONCATENADOS COMPRIMIDOS**
Descripción	Son los datos que se guardaran en un archivo y están comprimidos por medio de el algoritmo de compresión Zip .
Proveniente de los procesos	3.6 CONCATENAR
Estructuras de datos	TRAMA FIRMADA

Flujo 32

Nombre	**RESUMEN Y DATOS DE ARCHIVO**
Descripción	Datos guardados que conforman el archivo firmado.
Para los procesos	CIFRAR CON IDEA EL ARCHIVO FIRMADO
Estructuras de datos	ARCHIVO FIRMADO

Flujo 33

Nombre	CLAVE DE SESION.
Descripción	128 bits generados al azar, y que conforman la clave IDEA
Proveniente de los procesos	4.1 GENERAR CLAVE DE SESION. 6.5 DESCIFRAR CON LUCAS
Para los procesos	4.2 CIFRAR CON IDEA EL ARCHIVO FIRMADO. 6.6 DESCIFRAR CON IDEA

Flujo 34

Nombre	IDENTIFICADOR DEL RECEPTOR.
Descripción	Identificador único del par de claves.
Proveniente de los procesos	7 ADMINISTRADOR DE CLAVES PUBLICAS.
Para los procesos	4.4 CONCATENAR.

Flujo 35

Nombre	CLAVE PUBLICA DEL RECEPTOR
Descripción	Conjunto de caracteres que representan la clave publica Lucas
Proveniente de los procesos	7 ADMINISTRADOR DE CLAVES PUBLICAS.
Para los procesos	4.3. CIFRAR CON LUCAS
Estructuras de datos	CLAVE PUBLICA LUCAS

Flujo 36

Nombre	CLAVE DE SESION CIFRADA
Descripción	Clave de sesión cifrada con la clave publica Lucas.
Proveniente de los procesos	4.3. CIFRAR CON LUCAS
Para los procesos	4.4. CONCATENAR
Estructuras de datos	CLAVE PUBLICA LUCAS

Flujo 37

Nombre	DATOS CIFRADOS CON IDEA.
Descripción	Conjunto de bits que representan el archivo firmado después de ser cifrado con la clave de sesión.
Proveniente de los procesos	4.2. DATOS CIFRADOS CON IDEA
Para los procesos	4.4. CONCATENAR
Estructuras de datos	ARCHIVO FIRMADO Y CIFRADO

Flujo 38

Nombre	DATOS CONCATENADOS 2
Descripción	Son los datos que se guardaran en un archivo y que conforman el archivo firmado y cifrado.
Proveniente de los procesos	4.4 CONCATENAR
Estructuras de datos	ARCHIVO FIRMADO Y CIFRADO

Flujo 39

Nombre	DATOS CONCATENADOS 2 COMPRIMIDOS.
Descripción	Son los datos que se guardaran en un archivo y que se comprimirán en el archivo.
Proveniente de los procesos	4.4 CONCATENAR
Estructuras de datos	ARCHIVO FIRMADO Y CIFRADO COMPRIMIDO

Flujo 40

Nombre	MENSAJE DE ID NO ENCONTRADO
Descripción	Mensaje que se le envía al usuario de que, en la base de datos no se encuentra usuario con ese identificador.
Proveniente de los procesos	5.2 BUSCAR DATOS DE USUARIO.

Flujo 41

Nombre	CLAVE PUBLICA
Descripción	Conjunto de caracteres que representan la clave publica Lucas
Para los procesos	4.3. BUSCAR DATOS DE USUARIO
Estructuras de datos	CLAVE PUBLICA LUCAS

Flujo 42

Nombre	CLAVE PUBLICA LUCAS
Descripción	Formato numérico que conforman la clave blica Lucas

Proveniente de los procesos	5.2 BUSCAR DATOS DE USUARIO.
Para los procesos	5.4 DESCONCATENAR 1
Estructuras de datos	CLAVE PUBLICA LUCAS NUMERICA

Flujo 43

Nombre	**ID DE USUARIO**
Descripción	Identificador único del par de claves publica, privada.
Proveniente de los procesos	5.4 DESCONCATENAR 1 6.4 DESCONCATENAR
Para los procesos	5.2 BUSCAR DATOS USUARIO 6.2 BUSCAR DATOS USUARIO

Flujo 44

Nombre	**ARCHIVO FIRMADO**
Descripción	Archivo firmado que se encuentra guardado en disco.
Para los procesos	5.4 DESCONCATENAR
Estructuras de datos	ARCHIVOS FIRMADOS

Flujo 45

Nombre	**RESUMEN ENCRIPTADO**
Descripción	Resumen que fue cifrado por el emisor y ahora es recuperado de un archivo.
Proveniente de los procesos	5.4 DESCONCATENAR 1

Para los procesos	5.3 DESCIFRAR CON LUCAS

Flujo 46

Nombre	**RESUMEN RECIBIDO**
Descripción	Resumen que fue descifrado por el receptor con la clave publica Lucas del archivo firmado recibido.
Proveniente de los procesos	5.3 DESCIFRAR CON LUCAS
Para los procesos	5.5 COMPARAR RESUMENES

Flujo 47

Nombre	**DATOS DEL ARCHIVO ORIGINAL**
Descripción	Conjunto de datos que fueron recuperados por el receptor y que supuestamente fueron enviados por el emisor.
Proveniente de los procesos	5.4 DESCONCATENAR 1
Para los procesos	5.6 GENERAR RESUMEN

Flujo 48

Nombre	**RESUMEN**
Descripción	160 bits que son el resultado de la función hash RIPEMD-160.
Proveniente de los procesos	5.6 GENERAR RESUMEN
Para los procesos	5.5 CIFRAR CON LUCAS

Flujo 49

Nombre	**MENSAJE DE RESULTADO**
Descripción	Mensaje que se le envía al usuario para informarle que el mensaje fue violado
Proveniente de los procesos	5.5 COMPARAR RESUMENES.

Flujo 50

Nombre	**ARCHIVO FIRMADO Y CIFRADO**
Descripción	Archivo firmado y cifrado que se encuentra guardado en disco.
Para los procesos	6.4 DESCONCATENAR
Estructuras de datos	ARCHIVOS FIRMADOS Y CIFRADOS

Flujo 51

Nombre	**MENSAJE DE USUARIO NO ENCONTRADO**
Descripción	Mensaje que se le envía al usuario de que, en la base de datos no se encuentra usuario con ese identificador.
Proveniente de los procesos	6.2 BUSCAR DATOS DE USUARIO.

Flujo 52

Nombre	**CLAVE PRIVADA**
Descripción	Dirección donde se encuentra el archivo de clave privada Lucas
Para los procesos	6.2 BUSCAR DATOS DE USUARIO

Flujo 53

Nombre	**CLAVE PRIVADA LUCAS CIFRADA**
Descripción	Contenido del archivo donde se encuentra la clave privada Lucas cifrada.
Proveniente de los procesos	6.2 BUSCAR DATOS DE USUARIO

Para los procesos	6.3 DESCIFRAR CON IDEA

Flujo 54

Nombre	**CLAVE PRIVADA LUCAS**
Descripción	Clave en su formato numérico después de ser descifrada con IDEA.
Proveniente de los procesos	6.3 DESCIFRAR CON IDEA
Para los procesos	6.5 DESCIFRAR CON LUCAS
Estructuras de datos	CLAVE LUCAS

Flujo 55

Nombre	**CLAVE DE SESION CIFRADA CON LUCAS**
Descripción	Clave se sesión guardada y cifrada que fue enviada por el emisor.
Proveniente de los procesos	6.4 DESCONCATENAR
Para los procesos	6.5 DESCIFRAR CON LUCAS

Flujo 56

Nombre	**ARCHIVO CIFRADO**
Descripción	Archivo cifrado por el emisor.
Proveniente de los procesos	6.4 DESCONCATENAR
Para los procesos	6.6 DESCIFRAR CON IDEA

Estructuras de datos	ARCHIVO FIRMADO Y CIFRADO

Flujo 57

Nombre	**ID DE USUARIO PARA DATOS PUBLICOS**
Descripción	Identificador único para accesar datos públicos.
Para los procesos	7.1 SELECCIÓN DE DATOS PUBLICOS

Flujo 58

Nombre	**DATOS PUBLICOS**
Descripción	Datos guardados que corresponden a los datos públicos del usuario
Para los procesos	7.1 SELECCIÓN DA DATOS PUBLICOS
Estructuras de datos	DATOS PUBLICOS

Flujo 59

Nombre	**DATOS PUBLICOS DE USUARIO SELECCIONADO**
Descripción	Datos guardados que corresponden a los datos privados del usuario y que pueden ser modificados, eliminados o enviados.
Proveniente de los procesos	7.1 SELECCIÓN DA DATOS PÚBLICOS
Para los procesos	7.2 ENVIAR DATOS PÚBLICOS - MODIFICAR DATOS DEL REGISTRO - 7.5ELIMINAR REGISTRO
Estructuras de datos	DATOS PÚBLICOS

Flujo 60

Nombre	**DATOS DE USUARIO**

Descripción	Datos al proceso de FIRMADO y que corresponde a los datos públicos del usuario.
Proveniente de los procesos	7.2 ENVIAR DATOS PUBLICOS
Para los procesos	3 FIRMADO
Estructuras de datos	DATOS PUBLICOS

Flujo 61

Nombre	**DATOS DE USUARIO PUBLICO**
Descripción	Datos del nuevo usuario a adicionar en la Base de datos.
Para los procesos	7.3 ADICIONAR REGISTRO
Estructuras de datos	DATOS PUBLICOS

Flujo 62

Nombre	**DATOS DEL NUEVO REGISTRO**
Descripción	Datos del nuevo usuario a que se guardaran a disco.
Proveniente de los procesos	7.3 ADICIONAR REGISTRO
Estructuras de datos	DATOS PUBLICOS

Flujo 63

Nombre	**TIPO DE SERVICIO 2**
Descripción	Servicio solicitado a realizar sobre los datos públicos.
Para los procesos	7.4 Verificar tipo de servicio

Flujo 64

Nombre	ELIMINADO DE REGISTROS
Descripción	Activa el proceso de eliminar registro, para borrarlo de la base de datos.
Proveniente de los procesos	7.4 VERIFICAR TIPO DE SERVICIO.
Para los procesos	7.6 ELIMINAR REGISTRO

Flujo 65

Nombre	ADICIÓN DE REGISTRO
Descripción	Activa el proceso de adicionar registro.
Proveniente de los procesos	7.4 VERIFICAR TIPO DE SERVICIO.
Para los procesos	7.3 ADICIONAR REGISTRO

Flujo 66

Nombre	ELECCION DE DATOS DE USUARIO
Descripción	Activa el proceso para ENVIAR DATOS PUBLICOS.
Proveniente de los procesos	7.4 VERIFICAR TIPO DE SERVICIO.
Para los procesos	7.2 ENVIAR DATOS PUBLICOS

Flujo 67

Nombre	MODIFICADO
Descripción	Activa el proceso de MODIFICAR DATOS DEL REGISTRO.
Proveniente de los procesos	7.4 VERIFICAR TIPO DE SERVICIO.
Para los procesos	7.2 MODIFICAR DATOS DE REGISTRO

Flujo 68

Nombre	DATOS PUBLICOS SIN DATOS SELECCIONADOS
Descripción	Datos sin el registro eliminado y que se actualizan en la base de datos.
Proveniente de los procesos	7.6 ELIMINAR REGISTRO.
Estructuras de datos	DATOS PUBLICOS

Flujo 69

Nombre	DATOS MODIFICADOS
Descripción	Datos modificados por el usuario sobre los datos guardados.
Para los procesos	7.5 MODIFICAR DATOS DEL REGISTRO
Estructuras de datos	DATOS PUBLICOS

Flujo 70

Nombre	DATOS MODIFICADOS A GUARDAR
Descripción	Datos modificados por el usuario sobre los datos guardados y se guardaran ahora.
Proveniente de los procesos	7.5 MODIFICAR DATOS DEL REGISTRO
Estructuras de datos	DATOS PUBLICOS

Flujo 71

Nombre	ID DE USUARIO PARA DATOS PRIVADOS
Descripción	Identificador único para accesar datos privados.
Para los procesos	8.1 SELECCIÓN DE DATOS PRIVADOS

Flujo 72

Nombre	DATOS PRIVADOS
Descripción	Datos guardados que corresponden a los datos privados del usuario
Para los procesos	8.1 SELECCIÓN DA DATOS PRIVADOS
Estructuras de datos	DATOS PRIVADOS

Flujo 73

Nombre	DATOS PRIVADOS DE USUARIO SELECCIONADO
Descripción	Datos guardados que corresponden a los datos privados del usuario y que pueden ser modificados, eliminados o enviados.
Proveniente de los procesos	8.1 SELECCIÓN DA DATOS PUBLICOS
Para los procesos	8.2 ENVIAR DATOS PÚBLICOS 8.5 MODIFICAR DATOS DEL REGISTRO 8.6 ELIMINAR REGISTRO
Estructuras de datos	DATOS PRIVADOS

Flujo 74

Nombre	DATOS DE USUARIO
Descripción	Datos al proceso de FIRMADO y que Corresponde a los datos privados del usuario.
Proveniente de los procesos	8.2 ENVIAR DATOS PUBLICOS

Para los procesos	3 FIRMADO Y CIFRADO
Estructuras de datos	DATOS PRIVADOS

Flujo 75

Nombre	**DATOS DE USUARIO PRIVADO**
Descripción	Datos del nuevo usuario a adicionar en la Base de datos.
Para los procesos	8.3 ADICIONAR REGISTRO
Estructuras de datos	DATOS PRIVADOS

Flujo 76

Nombre	**DATOS DEL NUEVO REGISTRO**
Descripción	Datos del nuevo usuario a que se guardaran a disco.
Proveniente de los procesos	8.3 ADICIONAR REGISTRO
Estructuras de datos	DATOS PRIVADOS

Flujo 77

Nombre	**TIPO DE SERVICIO 3**
Descripción	Servicio solicitado a realizar sobre los datos privados.
Para los procesos	8.4 Verificar tipo de servicio

Flujo 78

Nombre	**ELIMINADO DE REGISTROS**
Descripción	Activa el proceso de eliminar registro, para borrarlo de la base de datos.

Proveniente de los procesos	8.4 VERIFICAR TIPO DE SERVICIO.
Para los procesos	8.6 ELIMINAR REGISTRO

Flujo 79

Nombre	**ADICIÓN DE REGISTRO**
Descripción	Activa el proceso de adicionar registro.
Proveniente de los procesos	8.4 VERIFICAR TIPO DE SERVICIO.
Para los procesos	8.3 ADICIONAR REGISTRO

Flujo 80

Nombre	**ELECCION DE DATOS DE USUARIO**
Descripción	Activa el proceso ENVIAR DATOS PRIVADOS.
Proveniente de los procesos	8.4 VERIFICAR TIPO DE SERVICIO.
Para los procesos	8.2 ENVIAR DATOS PRIVADOS

Flujo 81

Nombre	**MODIFICADO**
Descripción	Activa el proceso de MODIFICAR DATOS DEL REGISTRO.
Proveniente de los procesos	8.4 VERIFICAR TIPO DE SERVICIO.
Para los procesos	8.5 MODIFICAR DATOS DE REGISTRO

Flujo 82

Nombre	**DATOS PRIVADOS SIN DATOS SELECCIONADOS**

Descripción	Datos sin el registro eliminado y que se actualizan en la base de datos.
Proveniente de los procesos	8.6 ELIMINAR REGISTRO.
Estructuras de datos	DATOS PRIVADOS

Flujo 83

Nombre	**DATOS MODIFICADOS**
Descripción	Datos modificados por el usuario sobre los datos guardados.
Para los procesos	8.5 MODIFICAR DATOS DEL REGISTRO
Estructuras de datos	DATOS PRIVADOS

Flujo 84

Nombre	**DATOS MODIFICADOS A GUARDAR**
Descripción	Datos modificados por el usuario sobre los datos guardados y se guardaran ahora.
Proveniente de los procesos	8.5 MODIFICAR DATOS DEL REGISTRO
Estructuras de datos	DATOS PRIVADOS

2.7.3.2 Almacenes De Datos.

Almacen De Datos 1

ALMACEN DE DATOS	**ANILLO DE CLAVES PUBLICAS**
DESCRIPCION	Contiene los datos privados del usuario que utiliza la aplicación.
FLUJOS DE DATOS RECIBIDOS	- Datos modificados a guardar - Datos privados sin datos seleccionados
FLUJOS DE DATOS PROPORCIONADOS	- Datos privados - Clave privada - Archivo con clave Lucas
DESCRIPCION DE DATOS	**Nombre** - Nombre del usuario. **Identificador** - Identificador único que identifica este par de claves. **Clave Publica** - La clave publica correspondiente al par (N,d) en formato radix-64. **Clave privada Lucas** - Contiene la dirección donde se encuentra el archivo con la clave Lucas cifrada con IDEA
VOLUMEN	Aumentara a manera que aumenten los

	usuarios de aplicación, y de la actualización que le de, el usuario de la misma. Soportara dependiendo de la capacidad del disco.
ACCESO	Puede ser accesada por cualquier persona.

Almacen De Datos 2

ALMACEN DE DATOS	**ANILLO DE CLAVES PRIVADAS**
DESCRIPCION	Contiene los datos públicos del usuario que utiliza la aplicación.
FLUJOS DE DATOS RECIBIDOS	- Datos modificados a guardar - Datos públicos sin datos seleccionados.
FLUJOS DE DATOS PROPORCIONADOS	- Clave Publica. - Datos públicos.
DESCRIPCION DE DATOS	**Nombre** - Nombre del usuario. **Identificador** - Identificador único que identifica esta claves. **Clave Publica** – La clave publica correspondiente al par (N,d) en formato radix-64
VOLUMEN	Aumentara a manera que aumenten los usuarios de la aplicación, y de la actualización que le de, el usuario de la misma. Soportara dependiendo de la capacidad del disco.

ACCESO	Puede ser accesada por cualquier persona.

Almacen de datos 3

ALMACEN DE DATOS	**TRAMA FIRMADA**
DESCRIPCION	Contiene el mensaje inicial firmado después de ser comprimido
FLUJOS DE DATOS RECIBIDOS	- Datos Concatenados
FLUJOS DE DATOS PROPORCIONADOS	- Archivo firmado comprimido.
DESCRIPCION DE DATOS	Contiene el resumen cifrado con Lucas y el mensaje original todo esto comprimido
VOLUMEN	Aumentara a manera que aumente el tamaño del mensaje.
ACCESO	Puede ser accesada por cualquier persona.

almacen de datos 4

ALMACEN DE DATOS	**TRAMA FIRMADA Y CIFRADA**
DESCRIPCION	Contiene el mensaje inicial firmado Y cifrado después de ser comprimido
FLUJOS DE DATOS RECIBIDOS	- Datos Concatenados 2
FLUJOS DE DATOS PROPORCIONADOS	- Archivo cifrado comprimido.
DESCRIPCION DE DATOS	Contiene el resumen cifrado con Lucas y el mensaje original todo esto cifrado con la clave de sesión mas la clave de sesión cifrada con Lucas, todo esto comprimido
VOLUMEN	Aumentara a manera que aumente el tamaño del mensaje.
ACCESO	Puede ser accesada por cualquier persona.

2.7.3.3 Estructuras De Datos

Estructura 1

ESTRUCTURA DE DATOS	**ARCHIVO CON CLAVE PRIVADA CIFRADA**
DESCRIPCION	Clave privada generada con el algoritmo Lucas, y que debe ser cifrada para mantenerse en secreto, contiene el par (N,e), pero con motivos del algoritmo .
CONTENIDO	**p** - Numero primo grande **q** - Numero primo grande . **e1** - Clave privada 1, Numero primo grande **e2** - Clave privada 2, Numero primo grande **e3** - Clave privada 3, Numero primo grande **e4** - Clave privada 4, Numero primo grande **Encriptados con el cifrador IDEA.**

Estructura 2

ESTRUCTURA DE DATOS	**DATOS PÚBLICOS.**
DESCRIPCION	Contiene los datos de los usuarios públicos

	con los que se pueden intercambiar mensajes sin presentar errores.
CONTENIDO	Tabla Clave Publica - Nombre - Identificador - Clave publica

Estructura 3

ESTRUCTURA DE DATOS	**DATOS PRIVADOS.**
DESCRIPCION	Contiene los datos de los usuarios privados con los que se pueden intercambiar mensajes sin presentar errores.
CONTENIDO	Tabla Clave Privada - Nombre - Identificador - Clave publica - Clave Privada

Estructura 4

ESTRUCTURA DE DATOS	**CLAVE LUCAS**
DESCRIPCION	Clave privada sin cifrar, en su formato numérico.
CONTENIDO	**p** - Numero primo grande **q** - Numero primo grande **e1** - Clave privada 1, Numero primo grande. **e2** - Clave privada 2, Numero primo grande **e3** - Clave privada 3, Numero primo grande

	e4 - Clave privada 4, Numero primo grande

Estructura 5

ESTRUCTURA DE DATOS	**CLAVE PUBLICA LUCAS**
DESCRIPCION	Clave publica Lucas, en su formato de radix-64, excepto por el punto que es el separador de (N,d).
CONTENIDO	**N -** Numero primo grande, resultado de pxq. **d -** Clave publica; Numero primo grande.

Estructura 6

ESTRUCTURA DE DATOS	**ARCHIVO FIRMADO**
DESCRIPCION	Resumen cifrado y mensaje original donde: • H(m) = Resumen. • EKRa = Encripción con la clave privada Lucas. • M =Mensaje Original
CONTENIDO	- EKRa [H(m)] - M - Identificador de Usuario, emisor.

Estructura 7

ESTRUCTURA DE DATOS	**ARCHIVO FIRMADO Y CIFRADO**
DESCRIPCION	Contiene la estructura ARCHIVO FIRMADO, cifrado con la clave de sesión Ks; La clave de sesión cifrada; y el identificador de usuario del receptor. Donde KUa simboliza la clave publica del emisor

CONTENIDO	- Eks(ARCHIVO FIRMADO) - EKUa(Ks) – Identificador del receptor

2.7.3.4 Descripción De Procesos

Proceso 1

PROCESO	**VERIFICAR TIPO DE SERVICIO**
DESCRIPCION	Verifica el tipo de servicio solicitado por el usuario.
ENTRADA	TIPO DE SERVICIO
SALIDA	- EDICION - FIRMADO - FIRMADO Y CIFRADO - CHEQUEO FIRMA - DESCIFRADO Y COMPARACION DE FIRMA - GESTIÓN DE CLAVES PUBLICAS - GESTIÓN DE CLAVES PRIVADAS - GESTIÓN DE AYUDA
RESUMEN DE LA LOGICA	Verificar que las entradas de servicio introducidas por el usuario sean validos y avisarle a cada proceso de iniciar su

	ejecución.

Proceso 2

PROCESO	**EDITOR DE ARCHIVO**
DESCRIPCION	Manejar los procesos de edición.
ENTRADA	Nombre de archivo - Datos para la edición.
SALIDA	MENSAJE
RESUMEN DE LA LOGICA	Encargado de mostrarle al usuario los datos que se encuentran guardados en un archivo, así como recibir nuevos datos para modificar este, también permite crear y guardar un archivo nuevo.

Proceso 3

PROCESO	**FIRMAR MENSAJE**
DESCRIPCION	Controla todo el proceso de firma de un mensaje.
ENTRADA	- FIRMADO - CLAVE IDEA - IDENTIFICADOR DE USUARIO - NOMBRE DE ARCHIVO

SALIDA	DATOS CONCATENADOS
RESUMEN DE LA LOGICA	Se encarga de velar por que el proceso de firma cumpla con cada uno de sus partes como son** recibir nombre de archivo y verificar su existencia; generar resumen de este; y cifrar este con la clave privada Lucas luego de desencriptarla con la clave IDEA; concatenar el resultado.

Proceso 4

PROCESO	**FIRMAR Y CIFRAR.**
DESCRIPCION	Controla todo el proceso de firmado y cifrado de un mensaje.
ENTRADA	- IDENTIFICADOR DE USUARIO INICIAL. - NOMBRE DE ARCHIVO - CLAVE IDEA - IDENTIFICADOR DE CLAVE DEL RECEPTOR
SALIDA	DATOS CONCATENADOS 2
RESUMEN DE LA LOGICA	Encargado de cifrar un documento firmado , generando la clave de sesión para cifrar el archivo con IDEA y esta clave, luego firma con la clave publica Lucas por medio del cifrador Lucas esta clave de sesión, y generar el archivo final que es el resultado de estos procesos mas el identificador del receptor.

Proceso 5

PROCESO	**REVISAR FIRMA**
DESCRIPCION	Verifica si un documento no ha sido violado desde que se firmó.
ENTRADA	- CHEQUEO DE FIRMA. - ARCHIVO FIRMADO - NOMBRE DE ARCHIVO
SALIDA	- DATOS DEL ARCHIVO ORIGINAL - MENSAJE RESULTADO - MENSAJE DE ID NO ENCONTRADO
RESUMEN DE LA LOGICA	Recibe el archivo firmado y separa el resumen cifrado para descifrarlo del mensaje, luego del mensaje genera un nuevo resumen para emitir un resultado de si ha sido violado o no un archivo firmado.

Proceso 6

PROCESO	**ACLARAR DOCUMENTO FIRMADO Y CIFRADO.**
DESCRIPCION	Se encarga de descifrar y revisar la autenticidad de un archivo.
ENTRADA	DESCIFRADO Y COMPARACION DE FIRMA NOMBRE DE ARCHIVO CLAVE PRIVADA IDEA
SALIDA	- DATOS DEL ARCHIVO ORIGINAL - MENSAJE DE USUARIO NO ENCONTRADO - MENSAJE RESULTADO - MENSAJE DE ID NO ENCONTRADO
RESUMEN DE LA LOGICA	Revisa la autenticidad de un archivo, primero descifra la clave de sesión con la clave privada Lucas, la cual es desencriptada con la clave IDEA, luego con la clave de sesión se encarga de descifrar el resto del paquete que se encuentra firmado y llama a revisar firma para generar el resultado final.

Proceso 7

PROCESO	**ADMINISTRADOR DE CLAVES PUBLICAS.**
DESCRIPCION	Gestiona los cambios que se desean realizar sobre el contenido de la tabla claves publicas que contiene la información de diferentes usuarios.
ENTRADA	GESTIÓN DE CLAVES PUBLICAS TIPO DE SERVICIO
SALIDA	DATOS DEL USUARIO
RESUMEN DE LA LOGICA	Recibe cada de los tipos de servicio que desea realizar un usuario sobre la base de datos en la tabla de datos públicos, como son modificar, eliminar, adicionar, y permite al usuario seleccionar un usuario.

Proceso 8

PROCESO	**ADMINISTRADOR DE CLAVES PRIVADAS.**
DESCRIPCION	Gestiona los cambios que se desean realizar sobre el contenido de la tabla claves privadas que contiene la información de diferentes usuarios.
ENTRADA	- GESTIÓN DE CLAVES PRIVADAS - TIPO DE SERVICIO
SALIDA	DATOS PRIVADOS DE USUARIO
RESUMEN DE LA LOGICA	Recibe cada de los tipos de servicio que desea realizar un usuario sobre la base de datos en la tabla de datos privados , como son modificar, eliminar, adicionar, y permite al usuario seleccionar un usuario.

Proceso 9

PROCESO	**VERIFICAR EXISTENCIA DE ARCHIVO.**
DESCRIPCION	Recibe el nombre de un archivo y verifica si

	este existe en el sistema.
ENTRADA	NOMBRE DE ARCHIVO.
SALIDA	ID archivo existente.
RESUMEN DE LA LOGICA	Busca dentro del sistema la existencia de un archivo y devuelve las respuesta si existe devuelve un identificador de este.

Proceso 10

PROCESO	**GENERAR RESUMEN.**
DESCRIPCION	Genera un resumen del archivo recibido.
ENTRADA	ID archivo existente ARCHIVO A FIRMAR.
SALIDA	RESUMEN.
RESUMEN DE LA LOGICA	Utiliza a RIPEMD-160 para generar un resumen de 160 bits de un archivo existente en el sistema

Proceso 11

PROCESO	**RECUPERAR ARCHIVO QUE CONTIENE LA CLAVE PRIVADA LUCAS**
DESCRIPCION	Recupera el archivo con la clave privada Lucas.
ENTRADA	Datos del emisor recuperados ARCHIVO CON CLAVE LUCAS.
SALIDA	ARCHIVO CON CLAVE LUCAS RECUPERADO.
RESUMEN DE LA LOGICA	Realiza un llamado al administrador de

	claves quien le envía los datos del receptor escogidos por el emisor, de estos datos escoge nombre de archivo para abrir el archivo donde se encuentra la clave privada cifrada, abre el archivo y envía su contenido.

Proceso 12

PROCESO	**RECUPERAR CLAVE CON IDEA.**
DESCRIPCION	Recibe la clave idea y un archivo para descifrar lo que fue encriptado con este.
ENTRADA	CLAVE IDEA. ARCHIVO CON CLAVE LUCAS RECUPERADO.
SALIDA	CLAVE LUCAS RECUPERADA.
RESUMEN DE LA LOGICA	Recibe el conjunto de caracteres en radix-64 para convertirlo a un formato numérico de 128 bits que son la clave IDEA, la cual aplicara al archivo recibido para desencriptarlo y obtiene como resultado la clave privada Lucas en su formato numérico.

Proceso 13

PROCESO	**CIFRAR CON LUCAS**
DESCRIPCION	Cifra con la clave privada Lucas el resumen del archivo.
ENTRADA	RESUMEN o CLAVE DE SESION. ARCHIVO ORIGINAL.

SALIDA	- RESUMEN CIFRADO. - - CLAVE DE SESIÓN CIFRADA
RESUMEN DE LA LOGICA	Cifra por medio de la secuencia de Lucas, el resumen si es el caso o la clave de sesión en caso contrario con la clave privada recibida.

Proceso 14

PROCESO	**CONCATENAR.**
DESCRIPCION	Concatena los datos recibidos.
ENTRADA	RESUMEN CIFRADO. ARCHIVO ORIGINAL.
SALIDA	- DATOS CONCATENADOS 1. - DATOS CONCATENADOS 2.
RESUMEN DE LA LOGICA	Recibe varios datos los cuales los concatena y los guarda en un archivo.

Proceso 15

PROCESO	**GENERA CLAVE DE SESIÓN.**
DESCRIPCION	Genera una clave de 128 bits.
SALIDA	CLAVE DE SESION
RESUMEN DE LA LOGICA	Revisa la hora en milisegundos del sistema y de los dígitos menos significativos le aplica una formula matemática para generar la clave de sesión.

Proceso 16

PROCESO	**CIFRAR CON IDEA EL ARCHIVO**

	FIRMADO.
DESCRIPCION	Cifra utilizando el algoritmo IDEA, con la clave de sesión un archivo
ENTRADA	CLAVE DE SESION. RESUMEN Y DATOS DE ARCHIVO
SALIDA	DATOS CIFRADOS CON IDEA
RESUMEN DE LA LOGICA	Utilizando un desarrollo del cifrador de bloques IDEA, subdivide el archivo en bloques y por medio del cifrador y con la clave de sesión cifra el archivo firmado.

Proceso 17

PROCESO	**BUSCAR DATOS USUARIO.**
DESCRIPCION	Recibe los datos de usuario guardados en la base de datos.
ENTRADA	- CLAVE PUBLICA. - ID DE USUARIO.
SALIDA	- MENSAJE DE ID NO ENCONTRADO. - CLAVE PUBLICA LUCAS o CLAVE PRIVADA
RESUMEN DE LA LOGICA	Accesa la base de datos dependiendo de la tabla que requiera accesar, luego lee los datos y busca en esta el identificador para luego sacar la información del usuario, si no existe manda mensaje y detiene el proceso.

Proceso 18

PROCESO	**DESCONCATENAR.**
DESCRIPCION	Separa las diferentes partes de que esta compuesto un archivo.
ENTRADA	- ARCHIVO FIRMADO. - ID DE ARCHIVO - ARCHIVO FIRMADO Y CIFRADO
SALIDA	- DATOS DEL ARCHIVO ORIGINAL - ID DE USUARIO - RESUMEN CIFRADO - CLAVE DE SESIÓN CIFRADA CON LUCAS - ARCHIVO CIFRADO
RESUMEN DE LA LOGICA	Se encarga de separa los archivos concatenados, dependiendo del tipo de archivo,

	la información concatenada la separa de la misma manera como fue unida.

Proceso 19

PROCESO	**DESCIFRAR CON LUCAS**
DESCRIPCION	Halla el valor correspondiente según la secuencia de Lucas.
ENTRADA	- CLAVE PUBLICA LUCAS. - CLAVE PRIVADA LUCAS RESUMEN - ENCRIPTADO. CLAVE DE SESIÓN CIFRADA CON LUCAS
SALIDA	- RESUMEN RECIBIDO. - RESUMEN RECIBIDO
RESUMEN DE LA LOGICA	Por medio de la secuencia de Lucas, halla el valor correspondiente del valor cifrado o sea el valor descifrado, utiliza las mismas variables del cifrador pero con diferentes valores, dependiendo sea el caso.

Proceso 20

PROCESO	**COMPARAR RESUMENES**
DESCRIPCION	Compara dos valores para saber si son

	iguales
ENTRADA	RESUMEN RECIBIDO. RESUMEN GENERADO.
SALIDA	MENSAJE RESULTADO
RESUMEN DE LA LOGICA	Convierte los dos de 160 bits valores a formatos numéricos y compara silos dos son iguales, de lo contrario envía mensaje de mensaje violado.

Proceso 21

PROCESO	**DESCIFRAR CON IDEA**
DESCRIPCION	Descifra utilizando el algoritmo IDEA, y una clave de 128 bits
ENTRADA	- CLAVE PRIVADA LUCAS CIFRADA. - CLAVE IDEA - ARCHIVO CIFRADO - CLAVE DE SESIÓN
SALIDA	- DATOS DESCIFRADOS - CLAVE PRIVADA LUCAS
RESUMEN DE LA LOGICA	Con una clave de 128 bits que son la clave IDEA, se le aplicara al archivo recibido para desencriptarlo y obtiene como resultado el valor que fue encriptado con este.

Proceso 22

PROCESO	**DESCIFRAR CON IDEA**
DESCRIPCION	Descifra utilizando el algoritmo IDEA, y una clave de 128 bits

ENTRADA	- CLAVE PRIVADA LUCAS CIFRADA. - CLAVE IDEA - ARCHIVO CIFRADO - CLAVE DE SESIÓN
SALIDA	- DATOS DESCIFRADOS - CLAVE PRIVADA LUCAS
RESUMEN DE LA LOGICA	Con una clave de 128 bits que son la clave IDEA, se le aplicara al archivo recibido para desencriptarlo y obtiene como resultado el valor que fue encriptado con este.

Proceso 23

PROCESO	**SELECCIÓN DE DATOS PÚBLICOS**
DESCRIPCION	Accesa la base de datos y en la tabla claves publicas o privadas, lee los datos del usuario pertenecientes al identificador recibido.
ENTRADA	- ID DE USUARIO PARA DATOS PUBLICOS. - ID DE USUARIO PARA DATOS PRIVADOS. - DATOS PÚBLICOS. - DATOS PRIVADOS
SALIDA	DATOS PÚBLICOS DE USUARIO SELECCIONADO. DATOS PRIVADOS DE USUARIO SELECCIONADO.
RESUMEN DE LA LOGICA	Por medio del identificador de usuario recibido accesa la base de datos y

	dependiendo de la tabla a la que deba accesar lee los datos que necesitara, correspondientes al ID de usuario.

Proceso 24

PROCESO	**VERIFICAR TIPO DE SERVICIO**
DESCRIPCION	Verifica el tipo de servicio solicitado por el usuario.
ENTRADA	TIPO DE SERVICIO 2
SALIDA	- ADICION DE REGISTRO - ELIMINADO DE REGISTROS - MODIFICADO - ELECCION DE DATOS DE USUARIO
RESUMEN DE LA LOGICA	Verificar que las entradas de servicio introducidas por el usuario sean validos y avisarle a cada proceso de iniciar su ejecución, dependiendo de quien lo solicite puede consultar la tabla publica o la privada.

Proceso 25

PROCESO	**ADICIONAR REGISTRO**
DESCRIPCION	Adiciona un nuevo registro a la base de

	datos.
ENTRADA	- DATOS DE USUARIO PUBLICO - TIPO DE SERVICIO - DATOS DE USUARIO PUBLICO
SALIDA	DATOS DEL NUEVO REGISTRO
RESUMEN DE LA LOGICA	Recibe los datos de cada uno de los campos de la tabla a la que se le agregara un registro, abre la tabla y agrega un nuevo registro.

Proceso 26

PROCESO	**MODIFICAR DATOS DEL REGISTRO**
DESCRIPCION	Modifica los cambios en el registro y actualiza la base de datos.
ENTRADA	- MODIFICADO - DATOS MODIFICADOS
SALIDA	DATOS MODIFICADOS A GUARDAR.
RESUMEN DE LA LOGICA	Permite que el usuario realice cambios sobre los datos almacenados, luego los recoge para actualizar la Base de datos.

Proceso 27

PROCESO	**ELIMINAR REGISTRO**
DESCRIPCION	Elimina un registro de la base de datos
ENTRADA	- DATOS DE USUARIO SELECCIONADO

SALIDA	- DATOS NUEVOS SIN DATOS SELECCIONADO.
RESUMEN DE LA LOGICA	Permite que el usuario seleccione un usuario al cual lo elimina de la base de datos.

Proceso 28

PROCESO	**ENVIAR DATOS**
DESCRIPCION	Envía los datos del usuario a quien lo requiera.
ENTRADA	DATOS DE USUARIO SELECCIONADO
SALIDA	DATOS DE USUARIO.
RESUMEN DE LA LOGICA	Permite que el usuario seleccione un usuario, y luego envía sus datos al proceso que lo requiera.

proceso 29

PROCESO	**COMPRIMIR**
DESCRIPCION	Comprime los datos recibidos y los guarda en un archivo utilizando el algoritmo de compresión Zip.
ENTRADA	- Datos concatenados - Datos concatenados 2
SALIDA	- Datos concatenados comprimidos

	- Datos concatenados 2 comprimidos
RESUMEN DE LA LOGICA	Recibe un archivo al cual comprime en uun nuevo arcvo, en un formato comprimido.

3. DESARROLLO DE LA INVESTIGACION

3.1 PRELIMINARES

La fase de la investigación se dividió en cuatro bloques principales. El primero se centro en la búsqueda de material criptográfico sobre firma digital, la materia prima del proyecto. En esta parte se encontraron artículos en Internet, publicaciones, y libros donde hicieran referencia al tema. Se analizaron varios diseños de firma digital que nos sirvieron como base para nuestra aplicación. Este bloque se inicio en Noviembre del 1998 y culmino en marzo de 1999 pero sin embargo, hasta la finalización del proyecto observaron otros diseños.

3.2 SELECCIÓN DE ALGORITMOS

El segundo bloque se centra en los algoritmos que se hubieran podido utilizar en la aplicación, en esta etapa se recolecto información sobre estos, la cual fue encontrada en artículos de Internet y en los libros, de aquí se dedujo que los algoritmos mas utilizados por aplicaciones como RSA, DES y otros presentan problemas de patente al ser utilizados fuera de los Estados Unidos , por lo que seleccionamos aquellos que no presentaran este problema y que presentaran una igual o mayor seguridad. Esta etapa se realizó durante marzo a mayo de 1999.

3.3 INVESTIGACION DEL LENGUAJE DE ALTO NIVEL

En esta etapa se investiga todo lo referente a la programación en Java, se opto por este lenguaje por ser independiente de la plataforma y presentar muchas utilidades para trabajar y desarrollar algoritmos criptigraficos utilizando matemática modular para números grandes. Este lenguaje es interpretado por lo que la compilación se hace un poco más lenta.

También se utilizó, visual j++, que presenta algunos conflictos al trabajar con el lenguaje Java original, sobre todo en el momento de refrescar las ventanas. Sin embargo el 90% de la aplicación esta desarrollada en Java de Sun original, y su código puede ser reutilizable.

Esta etapa se inicia prácticamente con el inicio del proyecto y culmina con el mismo, puesto que es un lenguaje muy extenso y conlleva a que se investiga según medida que se necesite, por lo que podríamos decir que siempre estaremos aprendiendo algo nuevo de este lenguaje.

3.4 DISEÑO DE LA APLICACIÓN

Se determina cada uno de los procesos de la aplicación y su interconexión, para esto se utilizan los diagramas de flujo de datos, descripción del diccionario de datos, se tiene como objetivo lo primordial de la aplicación y como se desea realizar, se inicia en junio y culmina en agosto.

3.5 DESARROLLO DE LA APLICACION

Se implementan cada uno de los objetos de la aplicación y se interconectan entre ellos, teniendo como base el diseño de flujo de datos, y cada uno de los objetivos iniciales del diseño.

Se inicia a mitad de agosto y culmina a mitad de octubre el bloque principal, pero en realidad sufre cambios en la etapa de depuración, por lo que se extiende y culmina junto con el proyecto.

3.6 DEPURACION Y PRUEBAS

Se somete el código a un examen para poder determinar los posibles errores que se podrían presentar, también se ejecuta la aplicación firmando y cifrando documentos y utilizando claves falsas o violando documentos para observar los errores y la respuesta de la aplicación ante ellos.

También se investiga sobre herramientas que permitan crear instaladores de programas realizados en el lenguaje Java y, para generar una aplicación de ayuda.

3.7 DOCUMENTACIÓN

Inicia y termina con el proyecto, y consiste en documentar cada una de los pasos que se siguieron, y realizaron. Como resultado tenemos manuales de la aplicación y éste documento.

GLOSARIO

AUTORIDAD CERTIFICANTE LICENCIADA:

Organo administrativo que emite CERTIFICADOS DE CLAVE PUBLICA.

CLAVE PRIVADA:

En un CRIPTOSISTEMA ASIMETRICO, es aquella que se utiliza para firmar digitalmente.

CLAVE PUBLICA:

En un CRIPTOSISTEMA ASIMETRICO, es aquella que se utiliza para verificar una FIRMA DIGITAL.

COMPUTACIONALMENTE NO FACTIBLE:

Dícese de aquellos cálculos matemáticos asistidos por computadora que para ser llevados a cabo requieren de tiempo y recursos informáticos que superan ampliamente a los disponibles en la actualidad.

CORRESPONDER:

Con referencia a un cierto PAR DE CLAVES, significa pertenecer a dicho par.

CRIPTOSISTEMA ASIMETRICO:

Algoritmo que utiliza un PAR DE CLAVES, una CLAVE PRIVADA para firmar digitalmente y su correspondiente CLAVE PUBLICA para verificar esa FIRMA DIGITAL. A efectos de este Decreto, se entiende que el CRIPTOSISTEMA ASIMETRICO deberá ser TECNICAMENTE CONFIABLE.

RESUMEN SEGURO (Hash Result):

La secuencia de bits de longitud fija producida por una FUNCION DE RESUMEN SEGURO luego de procesar un DOCUMENTO DIGITAL.

DOCUMENTO DIGITAL:

Representación digital de actos, hechos o datos relevantes.

DOCUMENTO DIGITAL FIRMADO:

DOCUMENTO DIGITAL al cual se le ha aplicado una FIRMA DIGITAL.

EMISION DE UN CERTIFICADO:

La creación de un CERTIFICADO por parte de una AUTORIDAD **CERTIFICANTE LICENCIADA.**

FIRMA DIGITAL:

Resultado de una transformación de un DOCUMENTO DIGITAL empleando un CRIPTOSISTEMA ASIMETRICO y un RESUMEN SEGURO, de forma tal que una persona que posea el DOCUMENTO DIGITAL inicial y la CLAVE PUBLICA del firmante pueda determinar con certeza:

1. Si la transformación se llevó a cabo utilizando la CLAVE PRIVADA que corresponde a la CLAVE PUBLICA del firmante,

2. Si el DOCUMENTO DIGITAL ha sido modificado desde que se efectuó la transformación.

La conjunción de los dos requisitos anteriores garantiza su NO REPUDIO y su INTEGRIDAD.

FUNCION DE RESUMEN SEGURO:

Es una función matemática que transforma un DOCUMENTO DIGITAL en una secuencia de bits de longitud fija, llamada RESUMEN SEGURO, de forma tal que:

1.Se obtiene la misma secuencia de bits de longitud fija cada vez que se calcula esta función respecto del mismo DOCUMENTO DIGITAL;

2. Es COMPUTACIONALMENTE NO FACTIBLE inferir o reconstituir un DOCUMENTO DIGITAL a partir de su RESUMEN SEGURO;

3. Es COMPUTACIONALMENTE NO FACTIBLE encontrar dos DOCUMENTOS DIGITALES diferentes que produzcan el mismo RESUMEN SEGURO.

INTEGRIDAD:

Condición de no-alteración de un DOCUMENTO DIGITAL.

LISTA DE CERTIFICADOS REVOCADOS:

Es la lista publicada por la AUTORIDAD CERTIFICANTE LICENCIADA, de los CERTIFICADOS DE CLAVE PUBLICA por ella emitidos cuya vigencia ha cesado antes de su fecha de vencimiento, por acto revocatorio.

NO REPUDIO:

Cualidad de la FIRMA DIGITAL, por la cual su autor no puede desconocer un DOCUMENTO DIGITAL que él ha firmado digitalmente.

PAR DE CLAVES:

CLAVE PRIVADA y su correspondiente CLAVE PUBLICA en un CRIPTOSISTEMA ASIMETRICO, tal que la CLAVE PUBLICA puede verificar una FIRMA DIGITAL creada por la CLAVE PRIVADA.

PERIODO DE VIGENCIA (de un CERTIFICADO):

Período durante el cual el SUSCRIPTOR puede firmar DOCUMENTOS DIGITALES utilizando la CLAVE PRIVADA correspondiente a la CLAVE PUBLICA contenida en el CERTIFICADO, de modo tal que la FIRMA DIGITAL no sea repudiable.

El PERIODO DE VIGENCIA de un CERTIFICADO comienza en la fecha y hora en que fue emitido por la AUTORIDAD CERTIFICANTE LICENCIADA, o en una fecha y hora posterior si así lo especifica el CERTIFICADO, y termina en la fecha y hora de su vencimiento o revocación.

REVOCACION DE UN CERTIFICADO:

Acción de dejar sin efecto en forma permanente un CERTIFICADO a partir de una fecha cierta, incluyéndolo en la LISTA DE CERTIFICADOS REVOCADOS.

SISTEMA CONFIABLE:

Equipos de computación, software y procedimientos relacionados que:

1. Sean razonablemente confiables para resguardar contra la posibilidad de intrusión o de uso indebido;

2. Brinden un grado razonable de disponibilidad, confiabilidad, confidencialidad y correcto funcionamiento;

3. Sean razonablemente aptos para el desempeño de sus funciones específicas;

4. Cumplan con los requisitos de seguridad generalmente aceptados.

SUSCRIPTOR:

Persona:

1.A cuyo nombre se emite un CERTIFICADO, y

2.Que es titular de la CLAVE PRIVADA correspondiente a la CLAVE PUBLICA incluida en dicho CERTIFICADO.

VERIFICACION DE UNA FIRMA DIGITAL:

Con relación a un DOCUMENTO DIGITAL, su FIRMA DIGITAL, el correspondiente CERTIFICADO DE CLAVE PUBLICA y la LISTA DE CERTIFICADOS REVOCADOS, es la determinación fehaciente de que:

El DOCUMENTO DIGITAL fue firmado digitalmente con la CLAVE PRIVADA correspondiente a la CLAVE PUBLICA incluida en el CERTIFICADO;

El DOCUMENTO DIGITAL no fue alterado desde que fue firmado digitalmente.

Para aquel documento cuya naturaleza pudiera exigir la necesidad de certificación de fecha cierta, o bien ésta fuere conveniente dado sus efectos, deberá determinarse adicionalmente que el mismo fue firmado digitalmente durante el PERIODO DE VIGENCIA del correspondiente CERTIFICADO.

Whitfield Diffie es actualmente un distinguido ingeniero de *Sun Microsystems*, pero es mejor conocido por su descubrimiento en 1975 del concepto de criptografía de clave pública, por el que obtuvo un doctorado Honoris Causa en ciencias técnicas por el *Swiss Federal Institute of Technology*.

Diffie realizó una licenciatura en ciencias matemáticas por el prestigioso *Massachusetts Institute of Technology* (MIT) en 1965

4. CONCLUSIONES

“ Los pequeños detalles son la base de la perfección”

® En el mundo moderno donde se realizan millones de transacciones a diario, es necesario crear herramientas que garanticen la seguridad de los documentos digitales ya sean guardados en medios magnéticos o se envíen a través de una red. Nuestra herramienta brinda el servicio de autenticidad y privacidad.

® Los algoritmos criptográficos son la clave para garantizar la seguridad de la información, funcionan de diferente manera según el tipo y/o modo. Los cifradores reciben un valor a cifrar y una clave con la cual se hallara un resultado ilegible para un tercero. Los generadores de claves crean claves que son casi imposible de ser repetidas. Los generadores de resúmenes, toman un texto del cual sacaran un resultado que sea único para este texto y que ningún otro texto pueda generar el mismo resumen.

® La relación costo - beneficio es incalculable, tomemos un valor imaginario y digamos que en una investigación de estas se pueden invertir cien millones de unidades, ahora miremos un beneficio básico, - Un profesor no puede asistir a una clase y envía una mensaje a todos los alumnos de la clase, firmando este con la aplicación ARMASEG para garantizar la autenticidad del documento, digamos que son 30 alumnos, los cuales se ahorrarían el pasaje cada uno (mil unidades), por cada uno serian 30.000 unidades, tomando una población de un millón de personas y que en un año a la mitad de la población se le brindo este servicio básico, serian quince mil millones de unidades, ahora adicionándole otros servicios como negocios en red, contratos, transacciones bancarias etc., ¿cuanto sumarían los beneficios?.

® De acuerdo al análisis de datos referentes a la construcción y uso de firmas digitales, uno de los mejores algoritmos existentes es el RIPEMD-160; por ser un algoritmo seguro que garantiza la autenticidad y confiabilidad del diseño por tales razones fue tenido en cuenta para la realización de la aplicación de firma digital [1].

[1] Applied Criptography, pag 429, 605.

® Con respecto al proceso de cifrado y descifrado de la firma, notamos que con el algoritmo Lucas se presentan inconvenientes por el manejo de los números grandes y la generación de la secuencia de Luc. Haciendo esto un poco lento todo este proceso. Sin embargo esto no le quita méritos al algoritmo de Lucas [2].

® Se pudo diseñar e implementar un esquema de firma digital con el cual se reduce al mínimo la probabilidad de fraguar las transacciones digitales. Permitiéndonos esto una mayor seguridad en los sistemas computacionales y de redes.

® Se logró probar la aplicación de la firma digital utilizando herramientas de alto nivel como el lenguaje de programación Java, encriptar archivos de tipo texto y firmarlos, luego comprobar estos resultados haciéndoles pequeñas modificaciones y chequear los resultados para garantizar el funcionamiento correcto de la aplicación.

® La firma digital comparada con la firma manual es más segura, confiable y garantizada. Ya que cualquier más mínima modificación del texto firmado digitalmente nos es informada por la aplicación.

[2] Network and Internetwork security, principles and practice, pag 300

® La realización de una aplicación de este tipo sería muy costosa pero los servicios que en ella se presta ahorraría la cantidad de dinero o el valor de la información cada vez que sea usada la aplicación.

5. RECOMENDACIONES

❖ El proceso de actualización de datos en la base de datos Local, en la cual se guardan las claves publicas, nombre del de usuario y su identificador de clave, se puede realizar desarrollando una aplicación cliente servidor.

❖ Para mejorar la seguridad, se podrían utilizar algunos de los modos de operación con el algoritmo Idea (OFB, CFB, CBC y ECB). contemplar la

inclusión de otros algoritmos para realizar el resumen y la firma digital. Tales como el SHA, El Gamal, RSA, etc.

- No cifrar archivos cuyo tamaño sea mayor de 1 mega, pues el tiempo en realizarse esta operación puede ser demasiado grande.

- Mantenerse en contacto con la autoridad certificante para saber que claves publicas están activas y cuales han dejado de existir.

- Al publicar una clave publica tener en cuenta que los datos de la base de datos local sean los mismos a los de la base de datos publicados en el servidor de claves.

BIBLIOGRAFIA

Manual de la materia, Seguridad y Criptografía. Cartagena 1998.

MICROSOFT Corporation. Visual j++ Manual del programador. Madrid: McGraw-Hill, 1998.

SCHENEIER, Bruce. APPLIED CRYPTOGRAPHY Segunda Edición. Protocols, Algorithms, and source in c, New York: John Wiley & Sons, Inc, 1996. Cap 3, 7, 8 , 9 , 10, 13, 18, 19.

SEAN, James A. ANALISIS Y DISEÑO DE SISTEMAS DE INFORMACIÓN. México: Mac Graw Hill, 1996. Cap 3,4,5,6,7.

SHAN Mark C, GRIFFITH Steven W. y LASI Antony F. 1001 TIPS PARA PROGRAMAR CON JAVA, México: Mac Graw Hill 1998.

WEHLING Jason, bHARAT Vidya Y Otros. Aproveche las noches con JAVA, México: Prentice-Hall Hispanoamericana, S.A, 1998.

WILLIAN Stallings. NETWORK AND INTERNETWORK. SECURITY PRINCIPLES AND PRACTICE, New Jersey: Prentice-Hall, 1993.Cap 5,7.

Anexo A. Manual Del Usuario

Introducción

Este manual es una guía integrada, pero el usuario debe estar familiarizado con las funcione básicas de Windows, tales como ventanas, menús, cajas de diálogo y el Mouse. Si el usuario no esta familiarizado con este sistema se le recomienda leer la guía de manejo de ventanas en Windows.

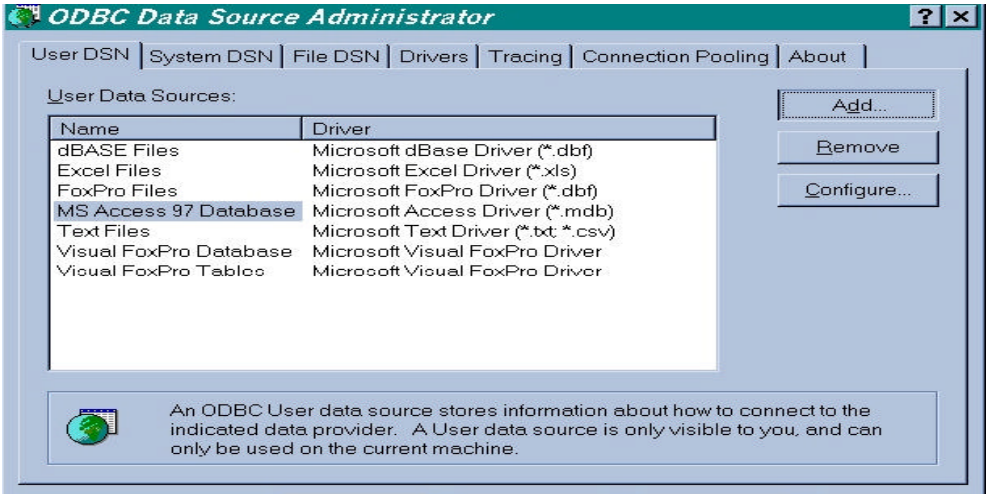
Instalación.

Para la instalación de la aplicación, ejecute el setup.exe de la aplicación y siga las instrucciones.

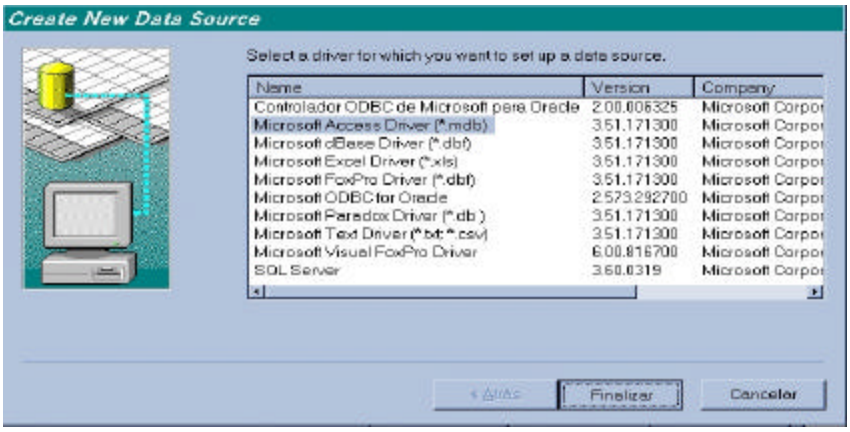
Instalar La Base De Datos.

1. En el panel de control ejecute la aplicación ODBC32.

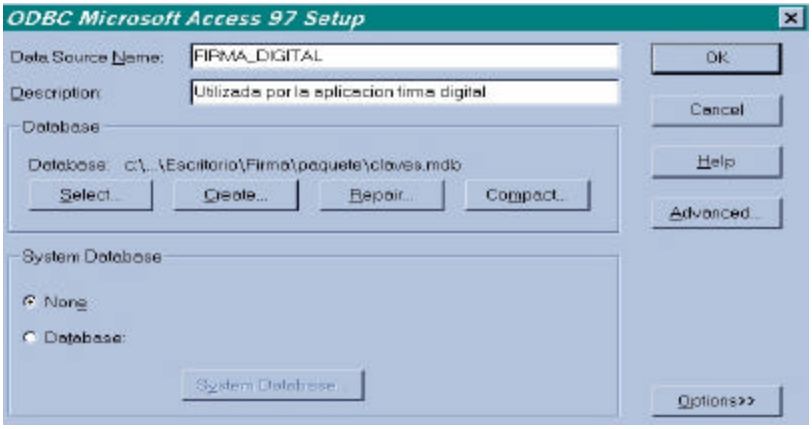
2. Seleccione Add.



3. Elija Drivers de Miscrosoft Access y elija aceptar.

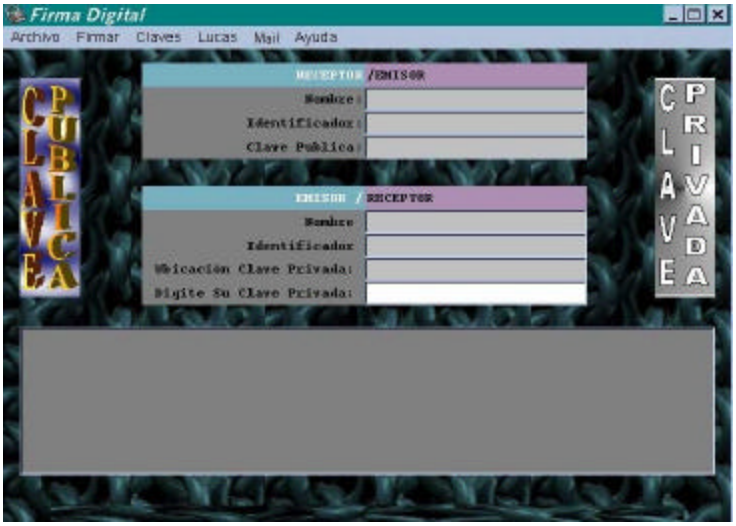


4. En nombre de la base de datos llámela **" FIRMA_DIGITAL ",** y haga un link a donde se encuentra la base de datos **"claves"**



UTILIZACION DE LA HERRAMIENTA

Al iniciar la aplicación le aparece la siguiente pantalla:



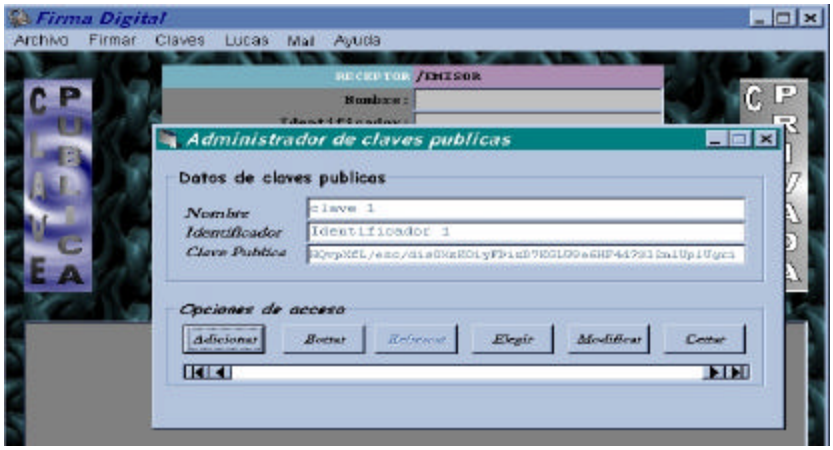
En el cuadro de la pantalla principal el usuario puede hacer click sobre cualquier objeto que se encuentra en ella.

La barra de menú de la pantalla se compone de los siguientes ítems:

- **Archivo**
- **Firmar**
- **Claves**

- **Lucas**
- **Mail**
- **Ayuda**

La pantalla también cuenta con dos botones llamados “Clave Publica” y “Clave Privada” Un click sobre el botón “clave publica“ le mostrará al usuario una nueva ventana para acceder a las claves publicas. En esa ventana se observa el administrador de clave publica para que el usuario acceda a ellas.



Los campos en este cuadro de dialogo son los siguientes:

Nombre:

En este campo va el nombre del usuario al cual le pertenece la clave publica.

Identificador:

En este campo va el identificador único de usuario al cual le pertenece la clave pública.

Clave Publica:

En este campo como su nombre lo indica va la clave publica del usuario.

El Botón Adicionar:

Este botón le permite al usuario adicionar a la base de datos más registros que contenga información sobre la clave publica.

El Botón Borrar:

Este botón le permite al usuario eliminar de la base de datos los registros que él halla seleccionado.

El Botón Elegir:

Este botón le permite al usuario elegir la clave publica que se encuentre en los registros de la base de datos la información será mostrada en la pantalla original.

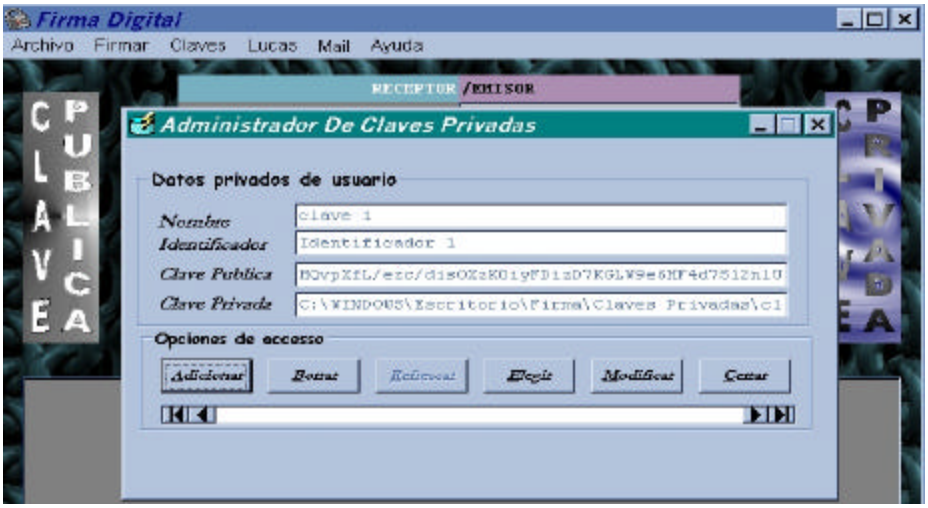
El Botón Modificar:

Este botón le permite al usuario modificar los campos de los registros de la base de datos.

El Botón Cerrar:

Este botón cierra la ventana de dialogo del administrador de clave publica.

Un click sobre le botón “Clave Privada”



Se le mostrará al usuario una nueva ventana para acceder a las claves privadas. En esa ventana se observa el administrador de clave privada para que el usuario acceda a ellas.

Los campos en este cuadro de dialogo son los siguientes:

Nombre:

En este campo va el nombre del usuario al cual le pertenece la clave privada.

Identificador:

En este campo va el identificador único de usuario al cual le pertenece la clave privada.

Clave Publica:

En este campo como su nombre lo indica va la clave publica del usuario.

Clave Privada:

En este campo como su nombre lo indica se muestra la clave privada del usuario pero encriptada con el algoritmo IDEA.

El Botón Adicionar:

Este botón le permite al usuario adicionar a la base de datos más registros que contenga información sobre la clave publica y la clave privada del nuevo usuario.

El Botón Borrar:

Este botón le permite al usuario eliminar de la base de datos los registros que él halla seleccionado.

El Botón Elegir:

Este botón le permite al usuario elegir la clave privada que se encuentre en los registros de la base de datos, la información será mostrada en la pantalla original.

El Botón Modificar:

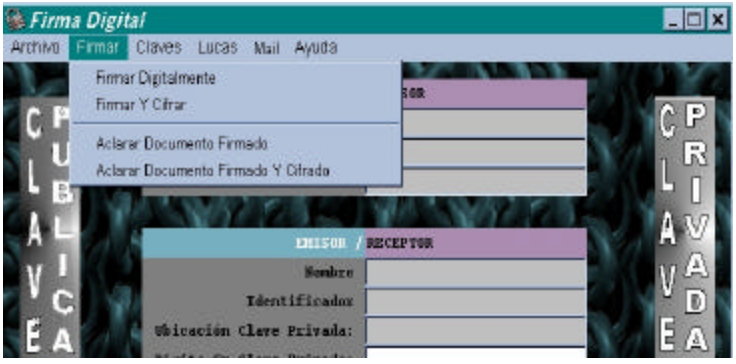
Este botón le permite al usuario modificar los campos de los registros de la base de datos.

El Botón Cerrar:

Este botón cierra la ventana de dialogo del administrador de clave privada.

Menú Firmar:

Al hacer click en este elemento aparece el siguiente menú desplegable:



Comando Firmar Digitalmente (Menú Firmar):

1. Abre un cuadro de dialogo para que el usuario elija el archivo que desea firmar;
2. Si los campos de clave privada se encuentran vacíos pide al emisor que elija con cual clave desea firmar el documento y le pide la clave privada.
3. Si la locación de clave privada esta vacía, pide que digite la clave privada.
4. Luego generar un archivo. Dso(Digital Signature Only).

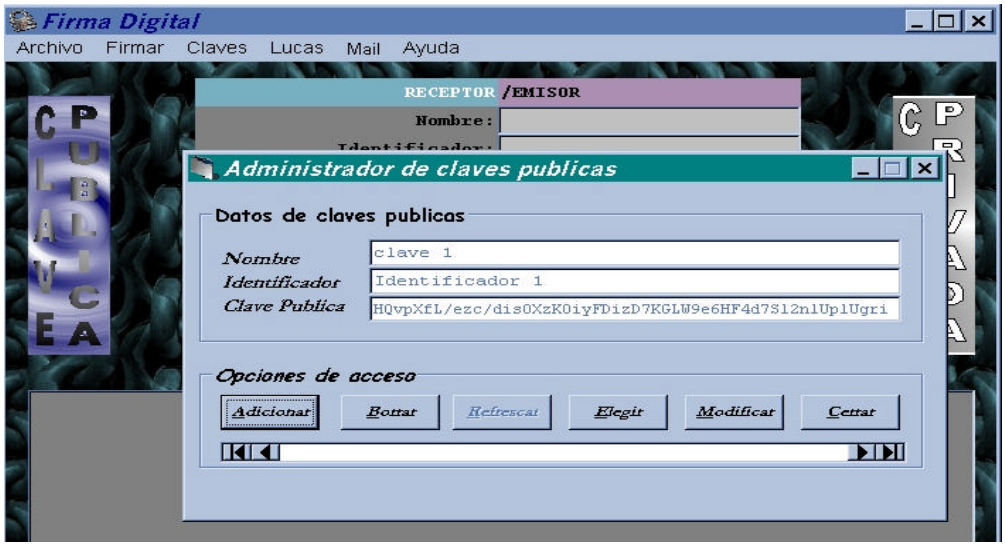
Comando Firmar Y cifrar

1. Abre un cuadro de dialogo para que el usuario elija archivo a firmar, y lo muestra.
2. Pide la clave con la que se desea firmar el documento
3. Muestra un cuadro de dialogo de la cual se escogerá la clave publica del receptor.
4. Genera un archivo con extensión. Mss(Maxime Signature Secure),

Comando Revisar Firma.

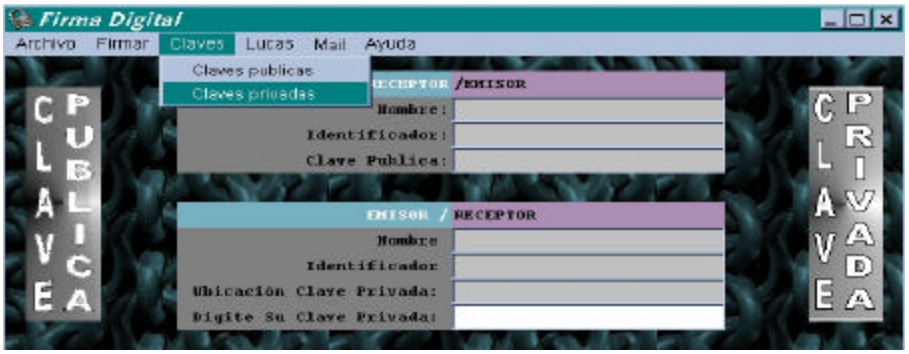
1. Abre un cuadro de dialogo de la cual el usuario selecciona el nombre del archivo.
2. Busca él la Base de datos y si se encuentra el identificador del usuario se revisa la firma, de lo contrario se enviara mensaje de usuario no encontrado.
3. Revisa la firma y envía mensaje, dependiendo si el archivo es correcto o no.

Aclarar Documento Firmado y Cifrado.



1. Despliega un cuadro de dialogo del cual el usuario selecciona el archivo.
2. Se abre el archivo y se busca en la base de datos si se encuentra el Identificador del receptor, con el cual se cifro el documento.
3. Pide al usuario digitar a clave privada correspondiente para el Identificador encontrado.
4. Busca el Id del emisor para revisar la firma y envía un mensaje si el documento fue violado o no.

Menú Claves



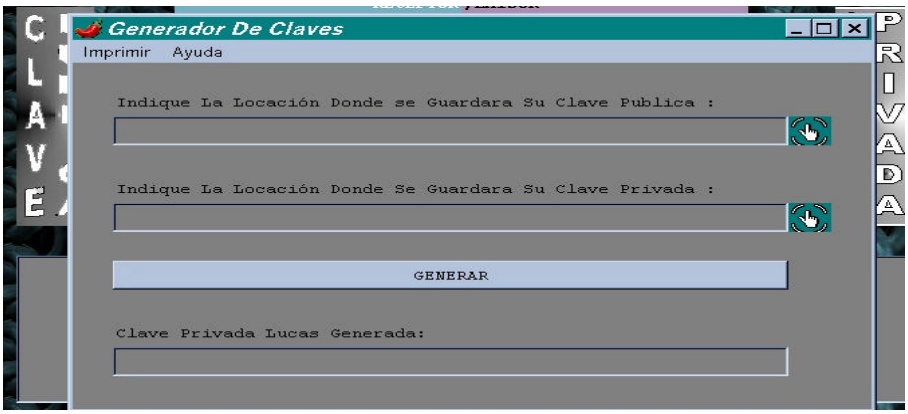
Administrador de claves publicas

Despliega la ventana de administrar claves publicas.

Administrador de claves privadas

Despliega la ventana de administrar claves privadas.

Menú Lucas

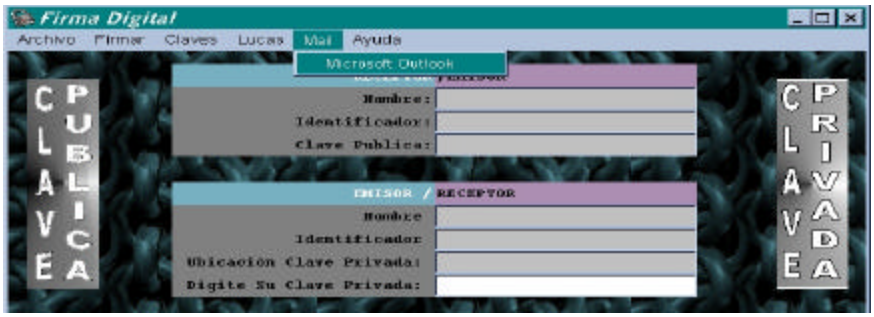


Utilizado únicamente por la autoridad certificante.

Mail

Microsoft Outlook

Activa la aplicación Microsoft Outlook.



Menú Ayuda

Despliega el proceso de ayuda de la aplicación.

II. DESCRIPCIÓN DETALLADA DE CLASES

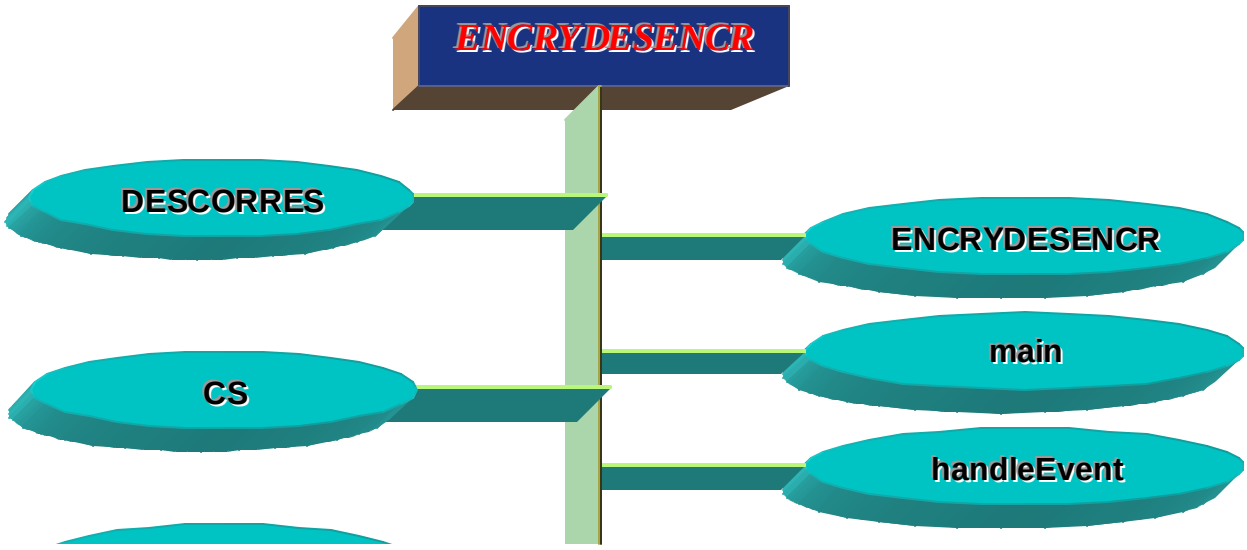


DIAGRAMA.1

II.1 class ENCRYDESENCR extends Frame

Clase principal, que se compone de métodos propios con los cuales realiza llamados a métodos que se encuentran en otros objetos, se encarga de la administración general de todas las partes que compone la aplicación.

Define las siguientes variables que serán conocidas por los métodos de esta clase:

II.1.1 Variables:

- **t3.** Es de tipo área de texto e imprime en pantalla la información del archivo a encriptar o desencriptar, serán legibles únicamente los caracteres ASCII.

- **t10.** Utilizada para imprimir la clave publica en pantalla.

- **t11.** Guardará dirección del archivo donde se encuentra la clave privada que fue generada por Lucas, y que se encuentra encriptada con la clave privada IDEA.
- **t12.** Contendrá la clave Privada IDEA digitada por el usuario.
- **Lin.** Contendrá la clave publica que se leerá del archivo de clave publica

- **dir1,dir2 ,arch1, arch2.** Contendrán los nombres de archivos y directorios donde están guardadas las claves publicas y privadas LUCAS.

- **Original.** Guardará la información del archivo que se va a encriptar.

- **CPR.** Contendrá la clave privada como un valor numérico

- **VAL[].** Guardará los valores que se recuperan de la base de datos de la tabla donde se encuentran los datos de la clave privada:

  [0] nombre .
  [1] Identificador
  [2] Dirección de clave privada
  [3]Clave publica correspondiente.

- **VAL1[].** Guardará los valores que se recuperan de la base de datos, de la tabla donde se guardan las claves publicas.

  [0] nombre
  [1] Identificador
  [2] Clave publica

- **bp, bp1.** Botones que utilizan las imágenes arriba, abajo.

- **Icono, Imagarriba, Imagabajo.** Imágenes utilizadas por la interfaz gráfica.

II.1.2 METODOS

II.1.2.1 METODO ENCRYDESENCR

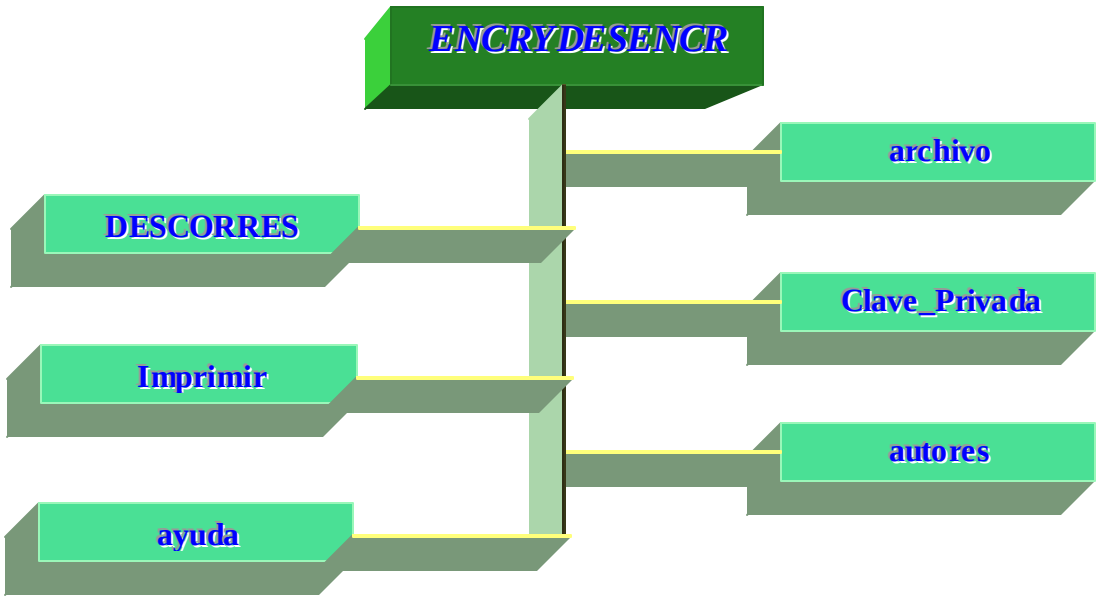


DIAGRAMA.2

Descripción : Se encarga del diseño de la interfaz gráfica de la ventana inicial.

Definición : ENCRYDESENCR(String nombre)

Parámetros de entrada : nombre – Su valor será él titulo de la ventana Inicial

Valor retornado : ninguno.

Invoca a **: -** errores_archivo(this," ¡ Error ! 537","!Falta Botón Gráfico!");

- Clave_Privada(this, VAL);

- autores();

- Imprimir(TextArea parra)

- archivo(Frame padre, String titulo, String mensaje_error)

- ayuda();

Descripción de la lógica **:** En él se declaran las variables del menú principal, de los objetos que serán ubicados dentro de la pantalla principal, y de los que contendrán las localizaciones de las claves publica y privada. Llama al método ENCRYDESENCR para crear el diseño inicial de la pantalla. Llama a errores_archivo() en caso de que se presente un error, al tratar de abrir los archivos de imágenes o cualquier otro archivo que la aplicación requiera abrir. Llama a Clave_Privada() con el fin de obtener la clave secreta IDEA para desencriptar la clave privada. El

método autores() muestra información acerca de los autores.

II.1.2.2 main(String argv[ ])

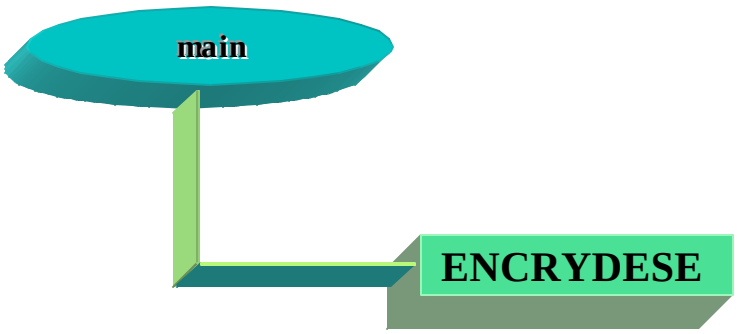


DIAGRAMA.3

Descripción : Se encarga de crear un objeto de ENCRYDESENCR y mostrarlo en pantalla.

Definición : public static void main(String argv[ ])

Parámetros de entrada : argv[ ] – en esta caso será null

Valor retornado : ninguno.

Invoca a **:** ENCRYDESENCR(String)

Descripción de la lógica **:** Se declara el tipo de fuente con el cual se invoca al objeto ENCRYDESENCR; y se le coloca color de fondo.

II.1.2.3 handleEvent(Event evt)

Descripción **:** Cuando ocurre un evento revisa si este es el de cerrar la ventana, si es así cierra la aplicación.

Definición **:** public boolean handleEvent(Event evt)

Parámetros de entrada **:** Evt nombre – evento realizado.

Valor retornado **:** Retorna el evento si no fue procesado.

Descripción de la lógica **:** Se pregunta por el evento el cual tiene relación con la destrucción de la ventana y se realiza su acción correspondiente; si no se devuelve el evento que se ha realizado.

II.1.2.4 action (Event evt, Object arg)

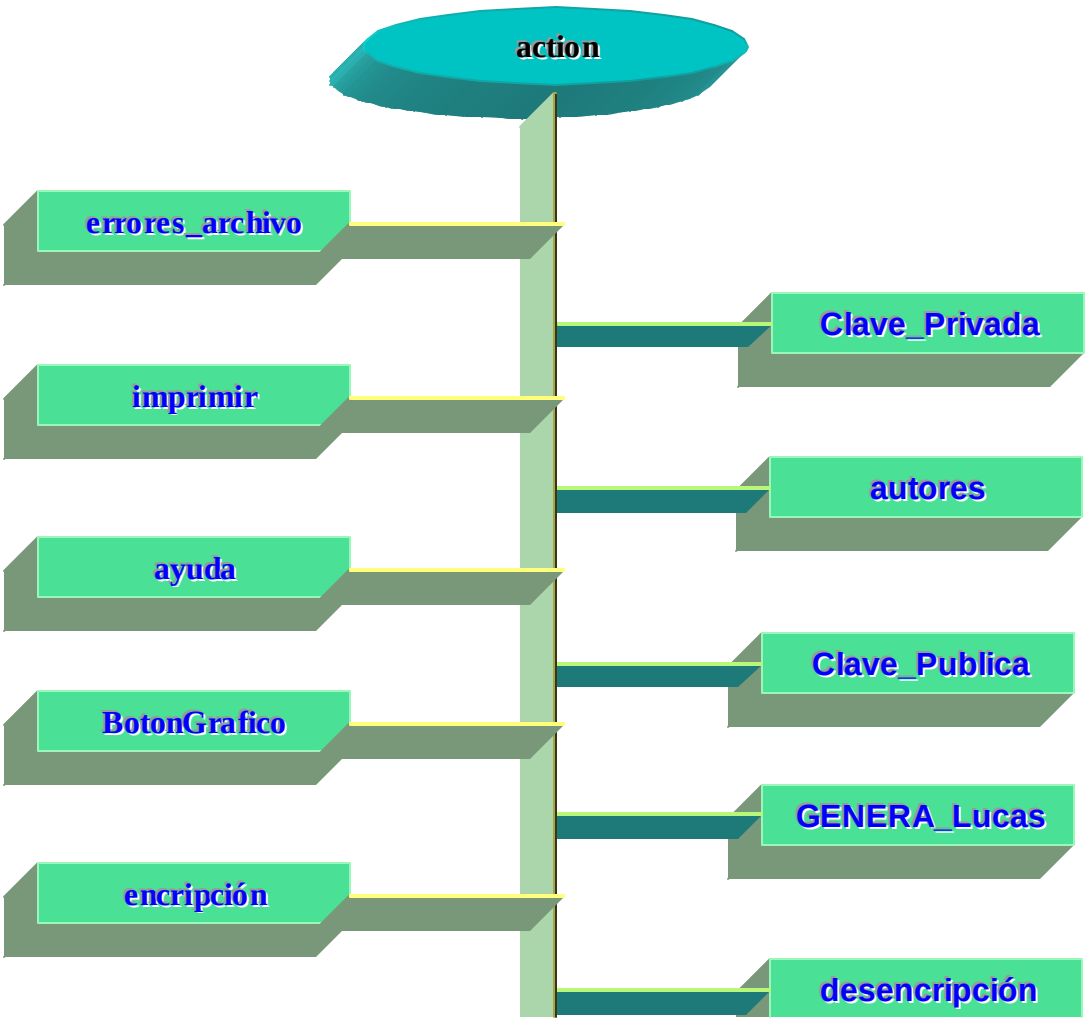


DIAGRAMA.4

Descripción	**:** Monitorea las acciones realizadas sobre cada objeto mostrado en la pantalla.
Definición	**:** public boolean action(Event evt, Object arg)
Parámetros de entrada	**: evt** evento realizado **arg** sobre el objeto que se realizo
Valor retornado	**:** true si se revisó el evento y se tomo acción, false si no se revisó.
Invoca a	**:** BotonGrafico.ponerBotón(BotónGrafico.ARRIBA) **-** archivo() **-** errores_archivo(Frame, String, String) **-** Clave_Privada(Frame, java.math.BigInteger[ ])

- Clave_Pública(Frame , java.math.BigInteger[ ])
- GENERA_Lucas (String)
- encripción()
- errores_archivo(Frame,String,String)
- desencripción();
- Imprimir(TextArea) ;
- autores();
- ayuda();
- Guardarcomo();
- Abrir();

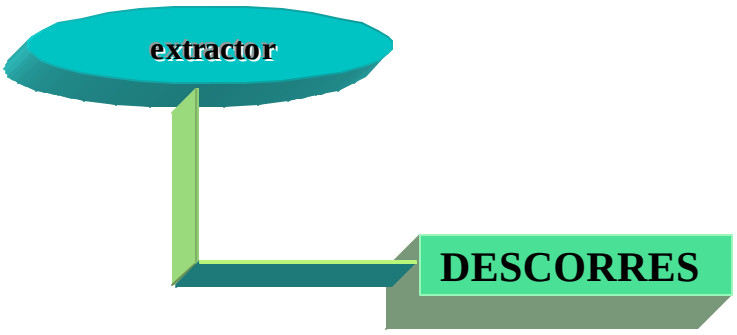
Descripción de la lógica : Sé monitorea la acción sobre uno de los botones gráficos; los cuales se le dan efectos. Dependiendo del botón seleccionado se realiza su respectiva tarea; la cual es la de llamar a la base de datos para actualizar las áreas de textos con las respectivas claves.

Se pregunta por los eventos realizados sobre le menú; llamando a genera_lucas para generar las claves en caso de que se requieran. Cuando el

evento es para encriptar, se verifica que haya correspondencia en sus respectivas áreas de texto. Si esa verificación no es correcta se invoca a errores_archivo() para que se envíe el mensaje de error. Alas acciones del menú se verifican de igual forma cuando hay una instancia en el menú, se verifica cual es la opción y se realiza la acción correspondiente; en la opción guardar como se verifica que halla un archivo abierto o creado; si no es así se envía un mensaje de error en caso contrario, se llama al método Guardarcomo(); el cual realiza la interfaz con el usuario.

II.1.2.5 extractor(String M)

DIAGRAMA.5



Descripción **:** Convierte la cadena recibida en 2 valores numéricos correspondientes a la clave publica Lucas (N, e)

Definición **:** public Vector extractor(String M)

Parámetros de entrada **:** String M - Clave publica

Valor retornado **:** Vector con los valores(N, e)

Invoca a **:** DESCORRES(char)

Descripción de la lógica **:** recibe una variable M la cual contiene a la clave publica; para convertirla en un número de formato legible. Toma los primeros seis bits para convertirlos en su respectivo valor el cual es definido por DESCORRES(). Desplaza seis posiciones a la derecha al valor de la variable M y la actualiza para escoger los siguientes seis bits de la cadena.

II.1.2.6 DESCORRES

Descripción **:** Convierte un char a un valor numérico, basado en los caracteres que se utilizaron para representar de una forma más sencilla las claves publica y privada.

Definición : int DESCORRES(char)

Parámetros de entrada : valor

Valor retornado : retorna un valor numérico de 0 a 63

Descripción de la lógica : Recibe como parámetro una variable de tipo char; toma éste carácter conocido y le asigna un número respectivo; retornando éste valor.

II.1.2.7 escoger_clave

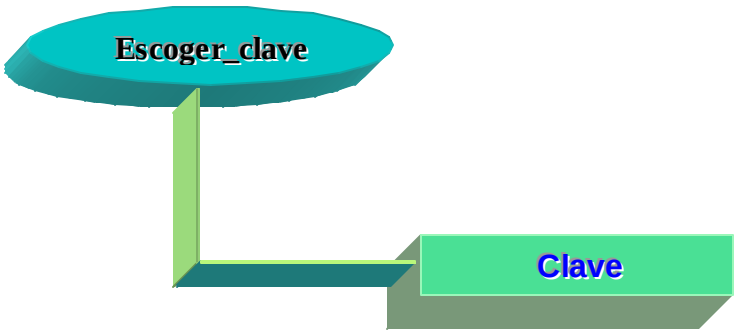


DIAGRAMA.6

Descripción **: Calcula los valores de Legendre para un valor numérico**

Valores de Legendre:

D/p	D/q	Clave Privada
1	1	**1**
1	-1	**2**
-1	1	**3**
-1	-1	**4**

Definición **:** public int escoger_clave(java.math.BigInteger valor, java.math.BigInteger Secret5, java.math.BigInteger Secret6)

Parámetros de entrada **:** - valor a encriptar o desencriptar

Secret5 - Equivale a p

Secret6 - Equivale a q

Valor retornado **:** retorna un valor numérico de 1 a 4

Invoca a **:**clave.jacobi(java.math.BigInteger, java.math.BigInteger).

Descripción de la lógica **:** Calculamos el valor del determinante en una variable auxiliar llamada aux. declaramos la variable ENCIFRA la cual contendrá los valores de la función de Jacobi para las diferentes claves, llamando al método clave(). Se pregunta por los diferentes valores respectivos para la función Jacobi y se devuelve el valor a la cual pertenece la clave para desencriptar.

II.1.2.8 encripción

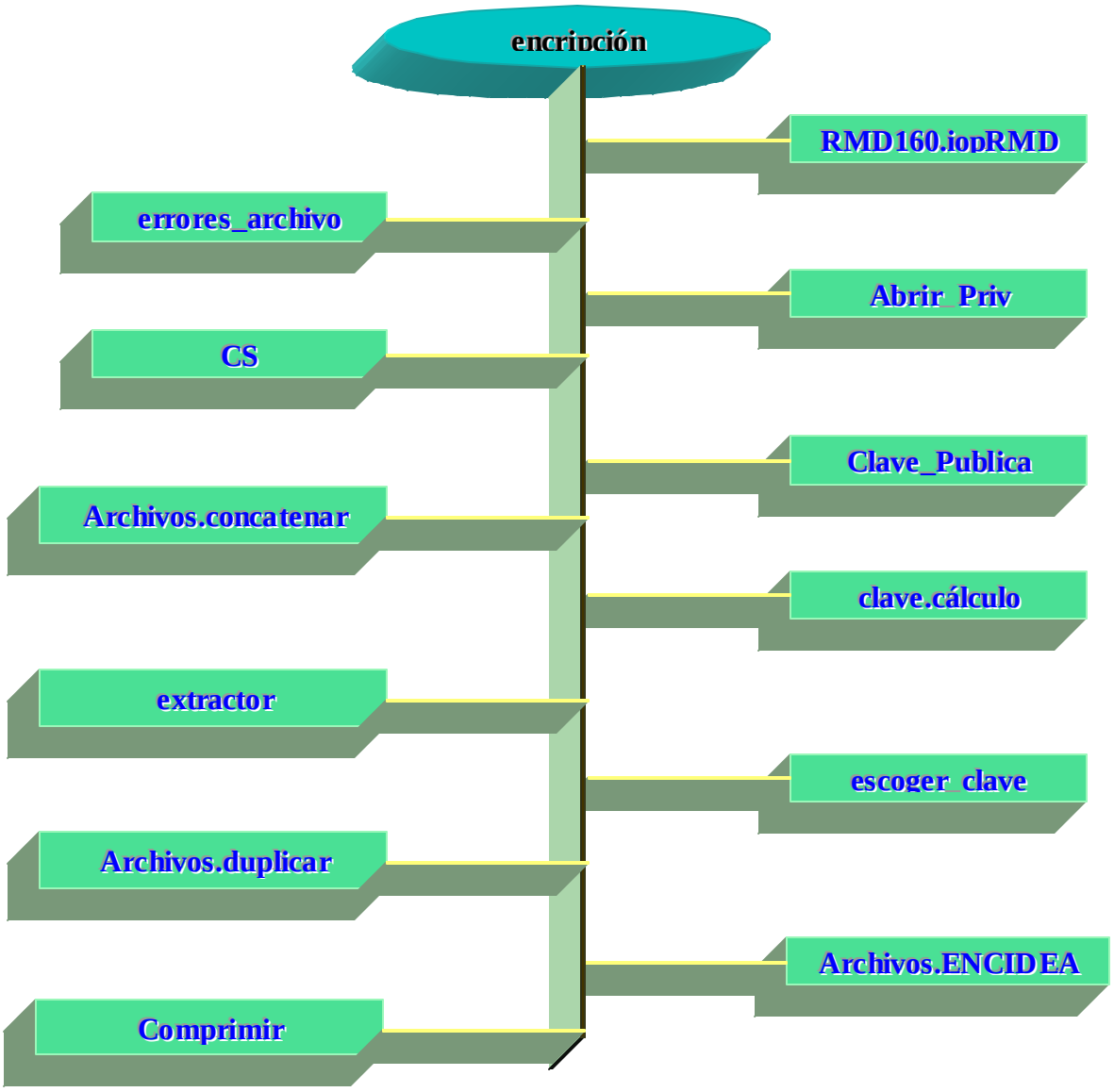

encripción
RMD160.iopRMD
errores_archivo
Abrir_Priv
CS
Clave_Publica
Archivos.concatenar
clave.cálculo
extractor
escoger_clave
Archivos.duplicar
Archivos.ENCIDEA
Comprimir

DIAGRAMA.7

Descripción **:** Coordinará el proceso de encripción.

Definición **:** public void encripción()

Parámetros de entrada **:** Directamente ninguno, pero necesita la dirección del archivo a encriptar, la clave privada Lucas, La clave publica Lucas, la clave de sesión.

Valor retornado **:** Ninguno directamente, pero crea un archivo de extensión Dso u Mss, que contendrá el archivo Firmado o encriptado o cifrado respectivamente.

Métodos Invocados **:**

- RMD160.iopRMD(String)
- clave.cálculo (java.math.BigInteger, java.math.BigInteger, java.math.BigInteger)
- archivos.concatenar(String, java.math.BigInteger, Frame)
- Abrir_Priv (java.math.BigInteger)
- extractor(String)
- errores_archivo (Frame, String, String)
- escoger_clave(java.math.BigInteger, java.math.BigInteger, java.math.BigInteger);
- archivos.duplicar(String, String , Frame);

- CS();
- archivos.ENCIDEA(java.math.BigInteger, String, Frame, int);

Descripción de la lógica : Se declaran variables que contendrán a la función Hash, la clave privada y publica. Sé desencripta la clave privada con el método Abrir_Priv(), y se extracta la clave publica convirtiéndola a su forma original con el método extractor(); ambos procesos se hacen por separados, si existe un error se llama a errores_archivo().

Se halla la función Hash al archivo con el método iopRMD(), se escoge la clave privada con la cual va a ser encriptada mediante el método escoger_clave(). Después sé encripta la función Hash haciendo una instancia al método clave.cálculo() y se concatena con el archivo al cual se le da la extensión ".Dso"(Digital Signature Only) mediante le método concatenar(), para luego comprimirla.

Se duplica el archivo para trabajar con el duplicado; se genera una clave de sesión para encriptar todo el archivo con el algoritmo IDEA. Sé encripta la clave de sesión con la clave publica y se concatena al archivo y se comprime en un archivo de extensión ".Mss"(Maxime Secure Signature).

II.1.2.9 desencripción( )

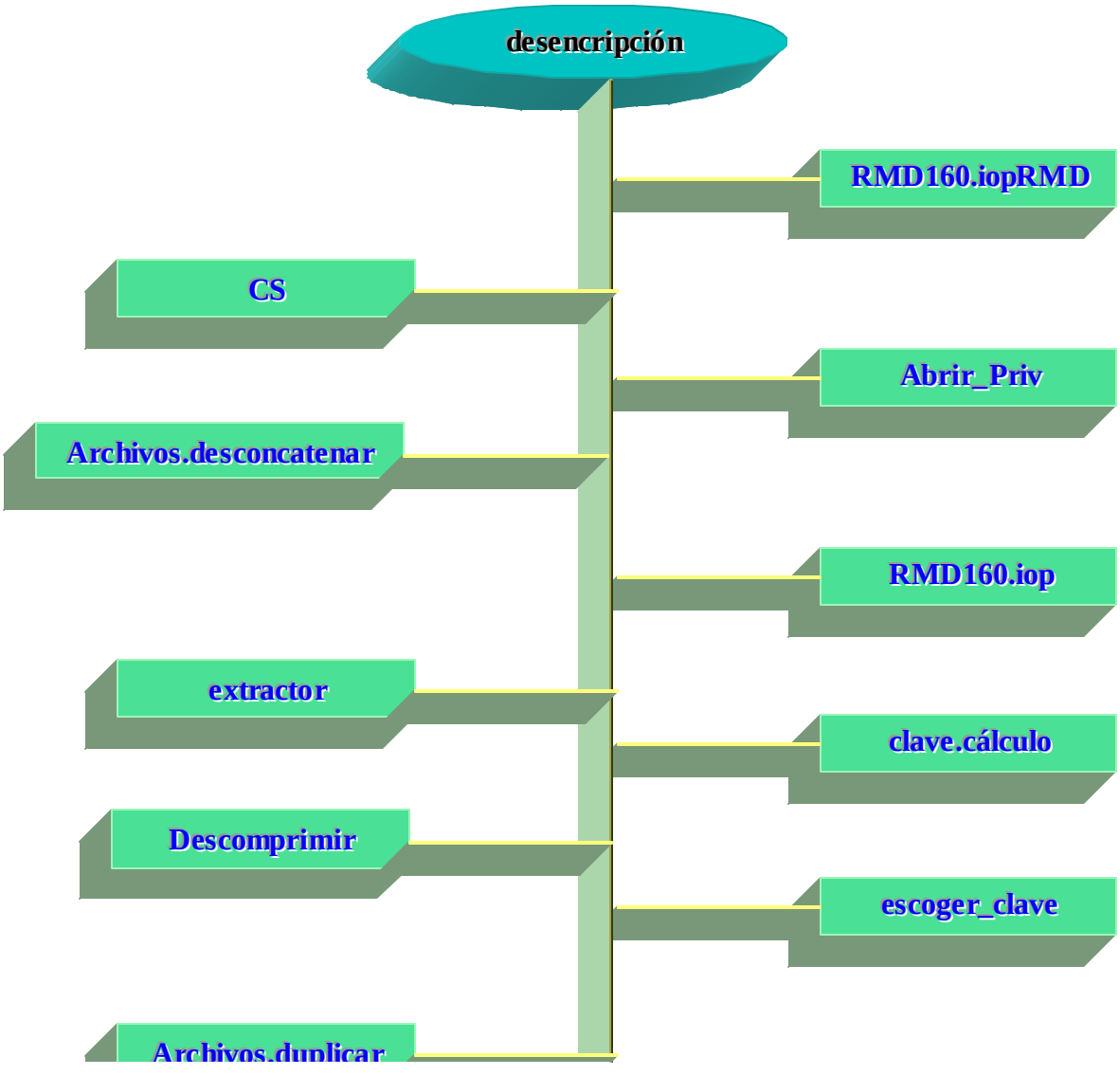


DIAGRAMA.8

Descripción **:** Se encarga de coordinar el proceso de desencripción.

Definición **:** public void desencripción()

Parámetros de entrada **:** Directamente ninguno, pero necesita la dirección del archivo a encriptar, la clave privada Lucas, La clave publica Lucas, la clave de sesión.

Valor retornado **:** Directamente no retorna ningún valor. Pero crea un archivo al que le quita la extensión ".Dso" u ".Mss", y lo convierte a la extensión antes de que lo encripten.

Métodos Invocados **:-** **R**MD160.iop(java.math.BigInteger [] , java.math.BigInteger)

- RMD160.iopRMD(String)
- clave.cálculo(java.math.BigInteger, java.math.BigInteger, java.math.BigInteger);
- archivos.ENCIDEA(java.math.BigInteger, String, Frame, int)

- archivos.desconcatenar(String, Frame, int)
- Abrir_Priv(java.math.BigInteger)
- extractor(String)
- archivos.duplicar(String, String, Frame);
- escoger_clave(java.math.BigInteger, java.math.BigInteger , java.math.BigInteger).

Descripción de la lógica : Se crea un objeto de RMD160 para generar la función Hash de comparación; también se crea un objeto de clave() para poder desencriptar; se toman los datos del emisor y del receptor; sé desencripta la clave privada del receptor, se separa del archivo la clave de sesión y el texto del documento mediante el método desconcatenar(); sé desencripta la clave de sesión con la clave privada con el método calculo(); sé desencripta el archivo con la clave de sesión; se separa la función Hash encriptada del texto; sé desencripta la función Hash y se genera una nueva función Hash al texto desencriptado;

después se comparan las dos funciones Hash si hay una desigualdad se envía un mensaje de error.

II.1.2.10 Abrir_Priv(java.math.BigInteger)

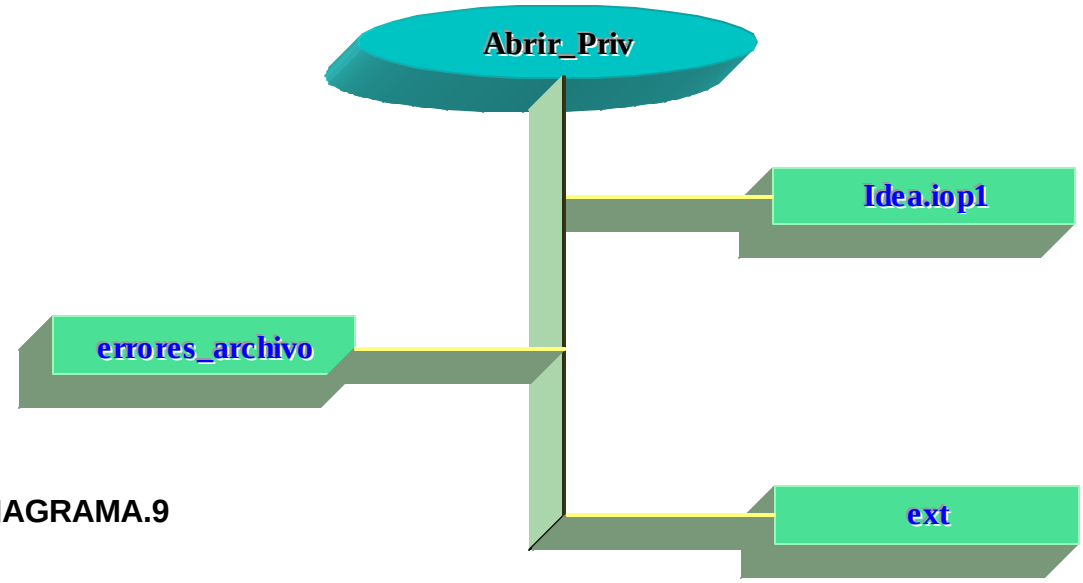


DIAGRAMA.9

Descripción **:** Desencripta la clave privada Lucas que fue encriptada con una clave idea.

Definición : public void Abrir_Priv(java.math.BigInteger secret[])

Parámetros de entrada : recibe un arreglo donde se encuentran las claves privadas Lucas, p, q.

Valor retornado : Directamente ninguno, pero recordemos que en Java cuando se recibe un arreglo y este es modificado en un método, el cambio es permanente y el método que envía el arreglo lo recibe modificado.

Métodos Invocados :

- Idea.iop1(java.math.BigInteger[ ] , java.math.BigInteger)
- CPR = ext(String)
- errores_archivo(Frame, String, String);

Descripción de la lógica: Recibe la clave privada IDEA para desencriptar y devolver la clave privada de LUCAS, se crea un archivo temporal para guardar en él la clave privada de LUCAS desencriptada; obtenemos la longitud de la clave privada; convertimos la clave digitada por el usuario a un número por el método ext(); convertimos los 8 bits leídos a los 16 bits para que sean reconocidos por IDEA.

Esto se realiza mediante desplazamientos de 16 bits a una variable auxiliar; hacemos un 'and' entre la clave y el valor de 63, luego a este valor arrojado le

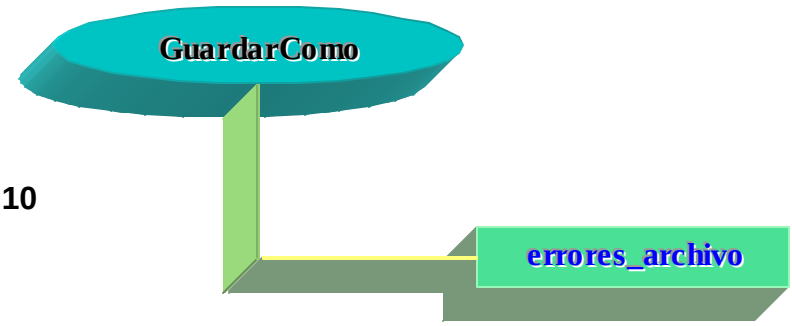
hacemos un 'or' con la variable auxiliar la cual contiene sólo cero.

Luego desplazamos la variable que contiene la clave 8 bits a la derecha y repetimos esto en un ciclo 'for' 8 veces. En un ciclo 'while' leemos carácter por carácter del archivo de la clavé privada y luego se envían a desencriptar en paquetes de cuatro al ciclo IDEA.

Del archivo temporal se leen los datos que se quieren; se lee el tamaño en bytes de N, luego se extrae y se convierte a un entero grande; de igual forma se hace para los demás valores; si existe algún error se envía un mensaje.

II.1.2.11 GuardarComo

DIAGRAMA.10



Descripción : Genera el cuadro de dialogo de guardar como...

Definición : public void Guardarcomo()

Parámetros de entrada : Directamente ninguno.

Valor retornado : Crea un archivo y guarda el contenido del área de texto.

Métodos Invocados : errores_archivo(Frame, String, String)

Descripción de la lógica : Se genera la interfaz de usuario para el cuadro de dialogo de Guardar Como; se le da por defecto un nombre con la extensión ".txt"; el usuario tecleara el nombre y se guardará en la variable "original"; una vez obtenido el nombre se guarda el contenido del

área de texto cerrando el archivo al cual se ha dado nombre; en caso de error se envía un mensaje con el método errores_archivo().

II.1.2.12 guardar

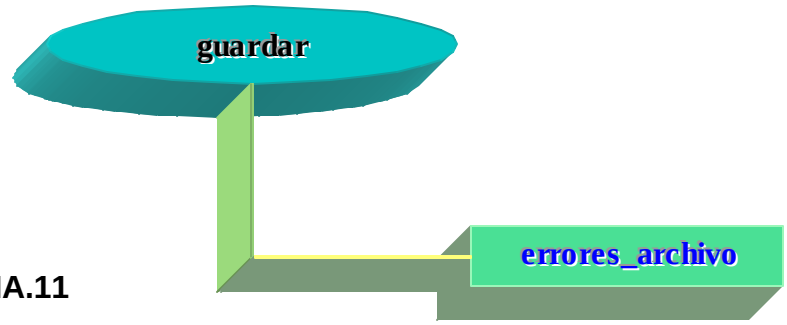


DIAGRAMA.11

Descripción : Actualiza el archivo con la nueva información

Definición : public void guardar(String)

Parámetros de entrada : Cadena que es el nombre del archivo a actualizar.

Valor retornado : Actualiza el archivo

Métodos Invocados : errores_archivo(Frame, String, String)

Descripción de la lógica : Recibe el nombre del archivo; y guarda el archivo que se encuentra abierto; se abre en forma de escritura y se actualiza con lo que haya en el área de texto.

II.1.2.13 Abrir

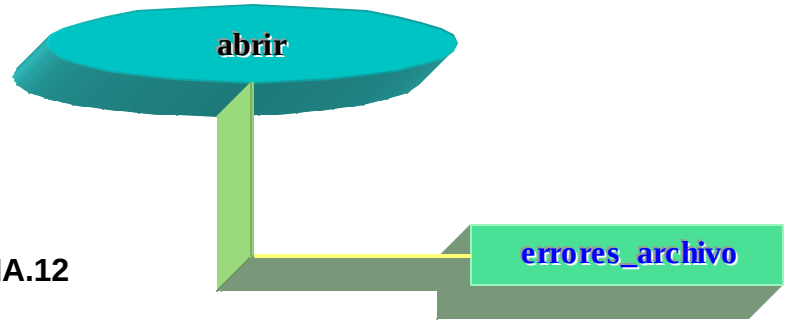


DIAGRAMA.12

Descripción : Abre un archivo y lee línea por línea.

Definición : public void Abrir()

Parámetros de entrada : Ninguno directamente.

Valor retornado : Actualiza el área de texto de la ventana principal.

Métodos Invocados : errores_archivo(Frame, String, String)

Descripción de la lógica : Se invoca la interfaz de usuario para que él elija el archivo que desea abrir; por defecto se muestran los de extensión ".Mss"; el contenido de este archivo se

muestra en el área de texto, si hay algún error se envía un mensaje con el método errores_archivo().

II.1.2.14 recibirinfo

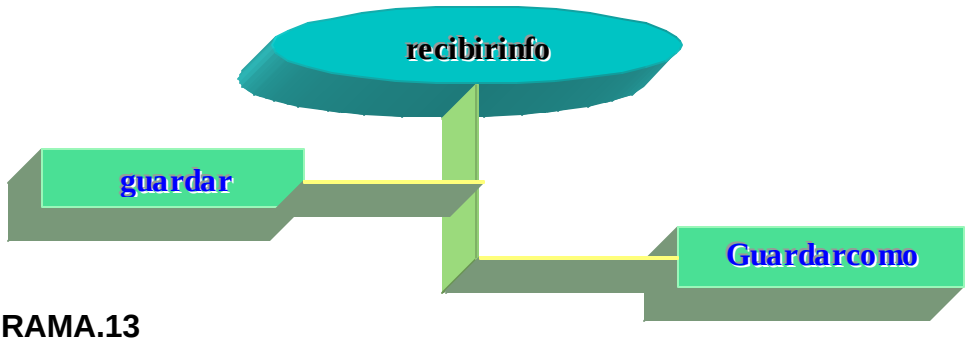


DIAGRAMA.13

Descripción : Establece una comunicación con la clase archivo y recibe la información que él devuelve.

Definición : public void recibirinfo(boolean)

Parámetros de entrada : boolean true si se desea realizar la operación.

Valor retornado : Actualiza el área de texto de la ventana principal.

Métodos Invocados : Guardarcomo();

- guardar(original);

Descripción de la lógica : Este método recibe la información del usuario; si este quiere guardar un archivo se cerciora de que tenga un nombre sino lo tiene llama a Guardarcomo(); si lo tiene llama a guardar().

II.1.2.15 CS

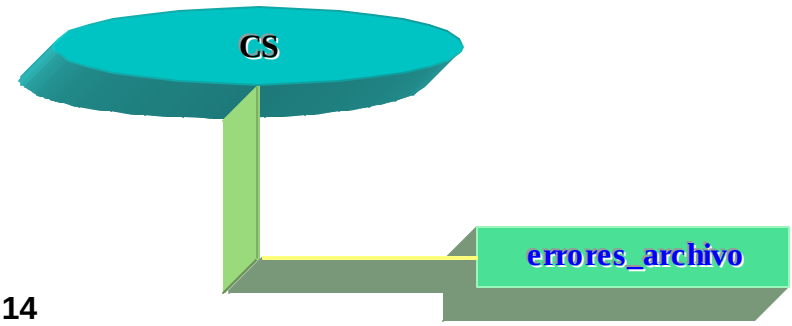


DIAGRAMA.14

Descripción : Genera la clave de sesión.

Definición : public java.math.BigInteger CS()

Parámetros de entrada : Ninguno directamente.

Valor retornado : Una clave de sesión de 128 bits

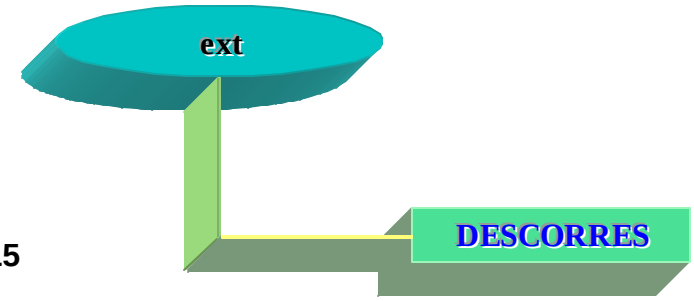
Métodos Invocados : errores_archivo(Frame, String, String)

Descripción de la lógica : Se captura la hora del sistema y se realiza una sencilla operación éste resultado se escogerá como semilla para la generación del número; escogemos

los primeros 128 bits del número generado y se retorna el valor.

II.1.2.16 ext

DIAGRAMA.15



Descripción : Convierte la clave privada digitada por el usuario, en un valor numérico.

Definición : java.math.BigInteger ext(String M)

Parámetros de entrada : Clave digitada por el usuario.

Valor retornado : java.math.BigInteger

Métodos Invocados : DESCORRES(char)

Descripción de la lógica : Capturamos la longitud de la cadena recibida, verificamos en que posición se encuentra el carácter

punto (.); convertimos en caracteres los valores contenidos en la cadena recibida. Hacemos un ciclo “for” iniciado en la posición donde se encuentra el carácter punto menos una, hasta cero; con el fin de obtener los esos caracteres que se encuentran en esas posiciones; de igual manera se hace otro ciclo “for” para obtener el resto de los caracteres. Y se retorna el valor final obtenido en el último ciclo “for”.

II.2 CLASE Clave_Privada extends Dialog

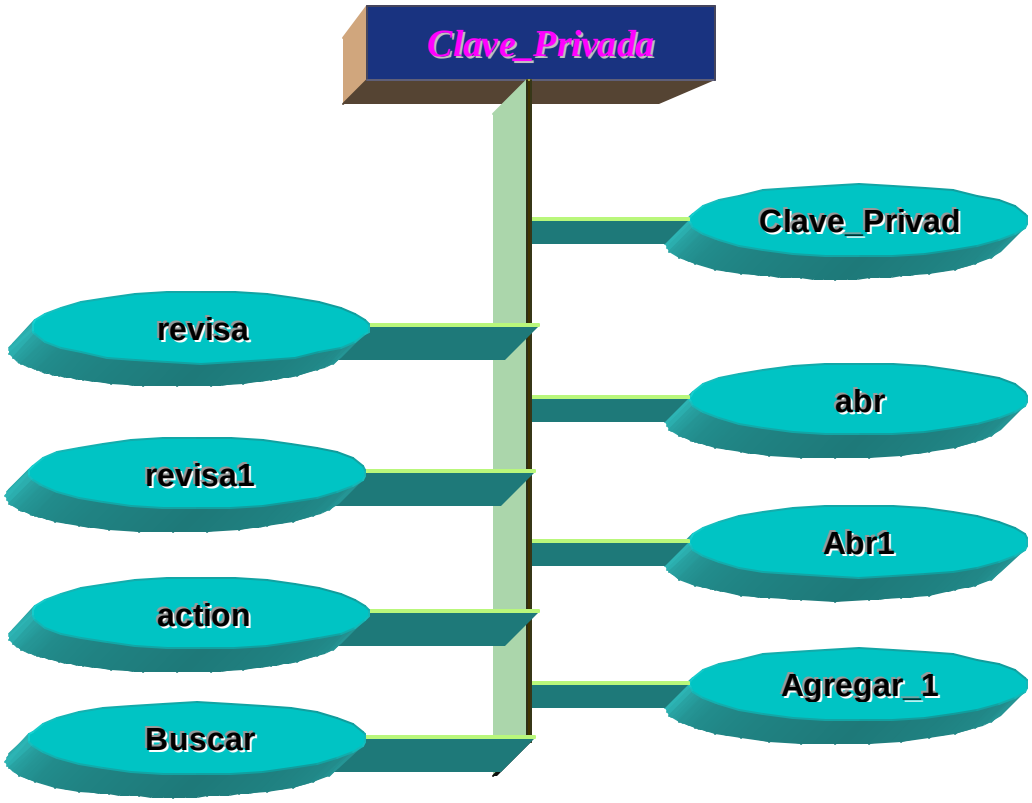


DIAGRAMA.16

Es utilizada para manejar la base de datos, crea un objeto de tipo dialogo donde monitorea las diferentes ordenes del usuario y cuida de que estas se realicen.

II.2.1 VARIABLES

- **archivo1, archivo2.** Contendrán nombres de los archivos
- **mensa.** Guarda el mensaje que le muestra al usuario, sobre la actividad a realizar.
- **d2.** Lista los Identificadores que son leídos de la base de datos.
- **Lista.** Contiene la información leída de la base de datos.
- **Name.** Guarda el nombre, que actualiza la BD.
- **ID.** Guarda el identificador, que actualiza la BD.
- **Cpu.** Guarda clave publica, que actualiza la BD.
- **CPr.** Guarda la clave privada , que actualiza la BD.
- **DEV1[].** Guarda los datos devueltos por la BD.

II.2.2 METODOS

II.2.2.1 Clave_Privada

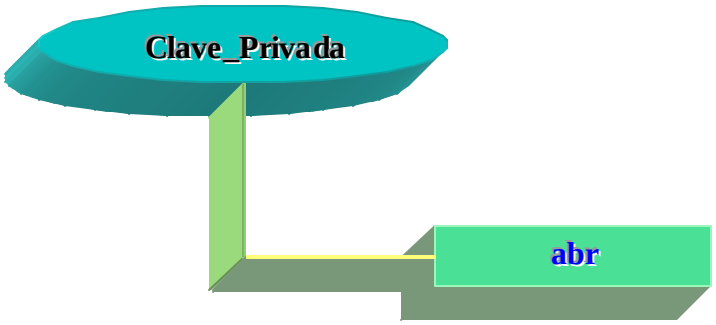


DIAGRAMA.17

Descripcion **:** Inicializa las variables, y llama a la primer pantalla que muestra el cuadrto de dialogo.

Definicion **:** Clave_Privada(Frame ,String [ ]).

Parámetros de entrada : Frame que lo llama, Arreglo que sera modificado.

Valor retornado **:** Implicitamente DEV [ ] con la nueva informacion.

Métodos Invocados **:** Abr()

Descripción de la lógica **:** Se obtiene el nombre del padre que lo llamo; el titulo de la ventana, en este caso, "Acceso A Claves Privadas", todos los cambios que se hagan en este método serán visto por el padre que los llamo. Se fijan todos los campos en 'null' y se invoca al método Abr() con el fin de realizar los diseños iniciales de los paneles; después se le agrega un nuevo panel al diseño original.

II.2.2.2 Abr( )

Descripcion **:** Despliega el diseño inicial del cuadro de dialogo, este intercambia informacion con el usuario sobre lo que desea hacer en la BD.

Definición **:** public void Abr()

Parámetros de entrada : Ninguno

Valor retornado : Ninguno

Métodos Invocados : Ninguno

Descripción de la lógica : Fija todos los campos que se van a mostrar en la pantalla en nulo; adiciona las etiquetas y los botones respectivamente.

II.2.2.3 Abr1( )

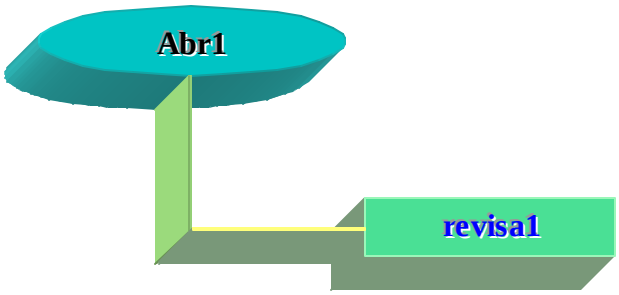


DIAGRAMA.18

Descripcion **:** Despliega un nuevo acomodo del diseño del cuadro de dialogo cuando el usuario hace clic en “abrir”.

Definición **:** public void Abr1()

Parámetros de entrada : Ninguno

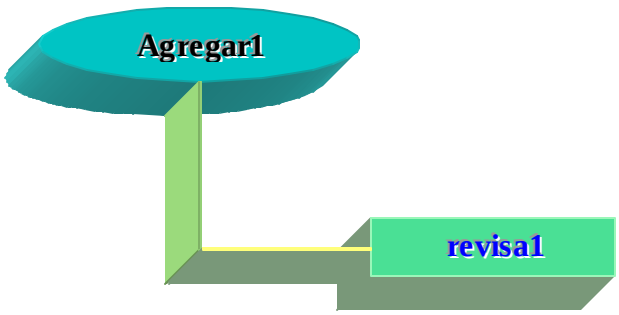
Valor retornado **:** Ninguno

Métodos Invocados **:** revisa1()

Descripción de la lógica **:** Fija todos los campos en nulo y agrega nuevas etiquetas de información se fijan los campos de textos en No-Editables y se le agrega la información respectiva; después se llama al método revisa1() para que verifique los nombres de los botones para luego agregar su respectiva etiqueta.

II.2.2.4 Agregar_1( )

DIAGRAMA.19



Descripcion **:** Despliega un nuevo acomodo del diseño del cuadro de dialogo cuando el usuario hace clic en “agregar”.

Definición **:** public void Agregar()

Parámetros de entrada : Ninguno

Valor retornado **:** Ninguno

Métodos Invocados **:** revisa1()

Descripción de la lógica **:** Actualiza el diseño de la ventana cuando se va a agregar una nueva clave; fija inicialmente todos los campos en nulos; después permiten que sean editables para que el usuario digite los nombres y los identificadores para los nuevos usuarios. Invoca a revisa1() para que revise las etiquetas de los botones y agrega dos nuevos botones para permitir el acceso a la búsqueda de las claves.

II.2.2.5 revisa( )

Descripcion **:** Revisa la etiqueta de los botones después de un nuevo acomodo del diseño del cuadro de dialogo.

Definición **:** public void revisa()

Parámetros de entrada : Ninguno

Valor retornado **:** Ninguno

Métodos Invocados **:** Ninguno

Descripción de la lógica **:** Revisa el nombre de los botones 1, 2, o 3 preguntando siempre primero que si su etiqueta es “Aceptar” se cambie por la respectiva acción tomada ya sea “Abrir”, Agregar”, o “Eliminar”.

II.2.2.6 revisa1( )

Descripcion	**:** Imprime en pantalla un mensaje al usuario indicándole la operación a realizar.
Definición	**:** public void revisa1()
Parámetros de entrada	: Ninguno
Valor retornado	**:** Ninguno
Métodos Invocados	**:** Ninguno
Descripción de la lógica	**:** De igual manera que en revisa(); se pregunta por la etiqueta que tiene el botón 1, 2 o 3 y se actualiza dependiendo de la acción tomada por el usuario.

II.2.2.7 public boolean action

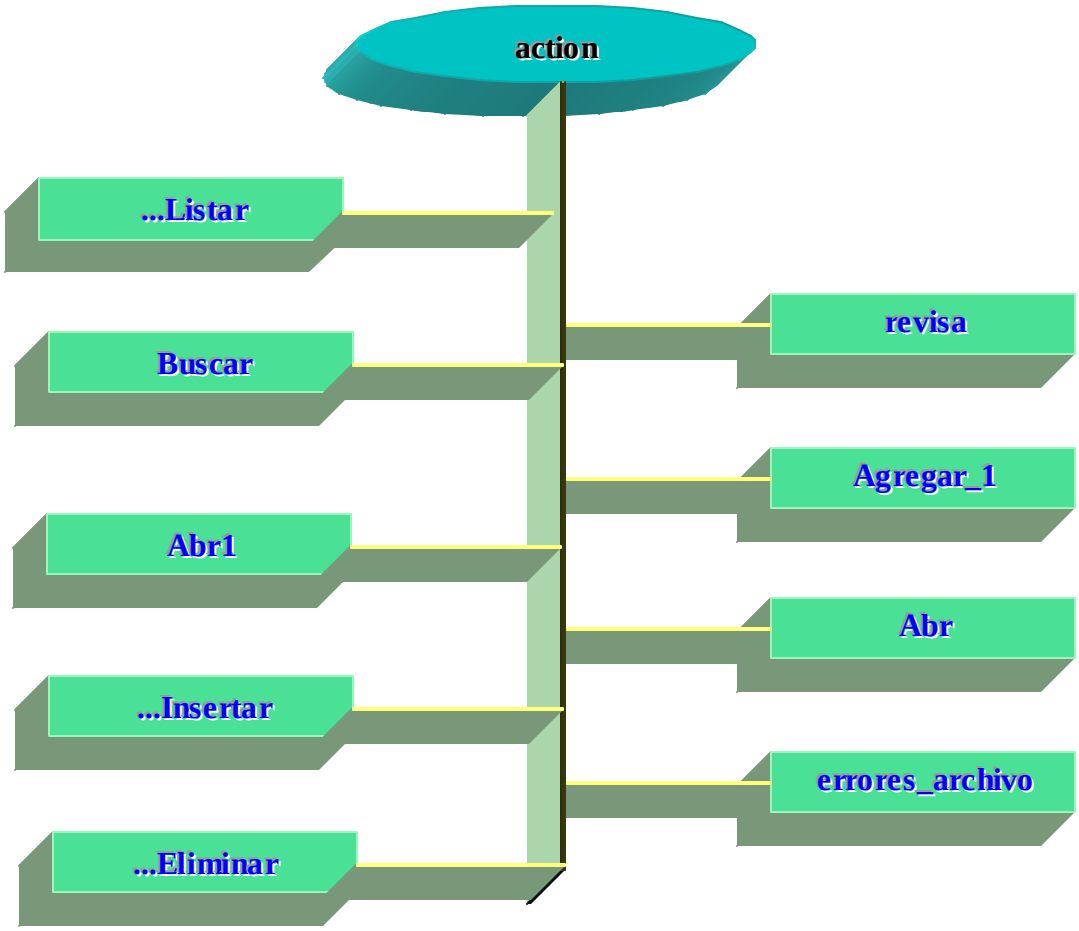


DIAGRAMA.20

Descripcion **:** Captura los diferentes eventos sobre los objetos de la ventana actual (Button,Choice. . .) , y monitorea la realización de la acción a ejecutar.

Definición **:** public boolean action(Event evt, Object arg)

Parámetros de entrada : El evento y el objeto sobre el que se debe realizar la acción.

Valor retornado **:** Directamente true si el evento fue procesado o false en caso contrario, Indirectamente actualiza la variable de tipo arreglo recibida.

Métodos Invocados **:**

- BaseDeDatos.Listar()
- revisa();
- Abr1()
- Agregar_1()
- Buscar(int);
- Abr();

- BaseDeDatos.insertar(String [ ]);
- errores_archivo(Frame, String, String)
- BaseDeDatos.Eliminar(String);

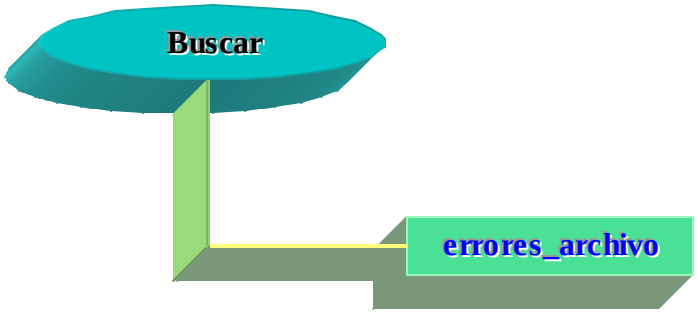
Descripción de la lógica : Se pregunta por el evento que se ha realizado si es en el “choice” se le muestran los nombres al usuario para que él elija el que desea y poder mostrarle en la ventana con su respectivo identificador y clave; si el evento es en un botón se verifica preguntando en cual botón fue el evento para tomar la acción correspondiente; si es el botón abrir, se declara un objeto para manipular la base de datos; se guardan los elementos que hay en la base de datos y los registros que contiene. Se agregan los elementos al “Choice”, se revisan la s etiquetas invocando a revisa(), se le cambia la etiqueta al botón 1.

Se remueven todos los elementos de la ventana actual y se invoca Abr1() para que actualice el diseño de la ventana. Si es el botón ‘Agregar’, se verifica las etiquetas con revisa() se remueven los elementos de la ventana y se actualiza el diseño llamando a Agregar_1() y se muestra el nuevo diseño. Si es le botón ‘Buscar’ ya sea el 1 ó 2, se llama el método Buscar(int) con su respectivo entero que lo diferencia.

Si es el botón ‘Eliminar’ se muestran los elementos actuales de la base de datos con el método Listar(); si el usuario acepta eliminar la clave; entonces, al método Eliminar() y se actualizan nuevamente los datos. Sino hay un elemento seleccionado se envía un mensaje al usuario mediante el método errores_archivo() al usuario indicándole que debe seleccionar uno.

II.2.2.8 Buscar

DIAGRAMA.21



Descripción **:** Despliega un cuadro de dialogo buscar, cuando se selecciona el archivo donde se debe encontrar la clave publica o privada , si es publica lo abre lo lee y actualiza la variable donde se almacena la clave publica, si es privada actualiza la variable donde se almacena locación del archivo donde se encuentra la clave privada.

Definición **:** public void Buscar(int)

Parámetros de entrada **:** Entero que define la opción si busca archivos donde se encuentra la clave publica , o privada.

Valor retornado **:** Directamente ninguno.

Métodos Invocados **:** errores_archivo(Frame, String, String) .

Descripción de la lógica **:** Se crea la interfaz con el sistema operativo para desplegar el cuadro de dialogo de Abrir ya sean los de clave publica o clave privada dependiendo del parámetro recibido; si es clave publica se buscan los de extensión ".Lcc" y se muestra en el cuadro

de dialogo en caso de haber error se envía un mensaje al usuario por medio de errores_archivo(); si no son los de clave publica simplemente se obtiene la dirección de la clave privada y se actualiza la variable.

II.3 class archivo

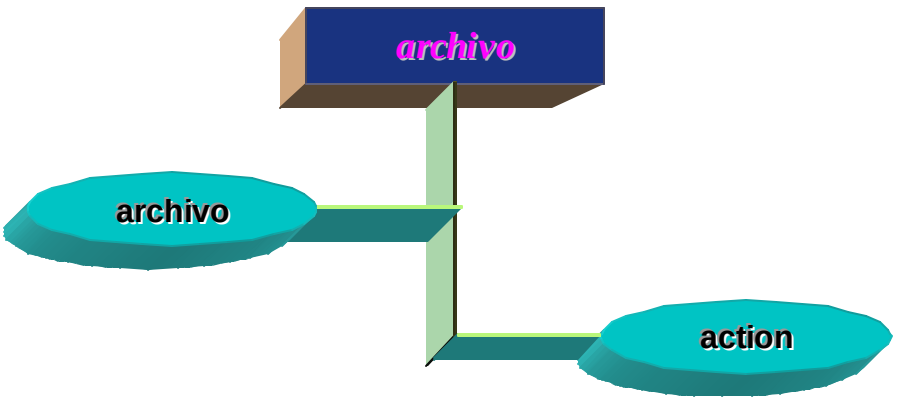


DIAGRAMA.22

Descripcion **:**Esta clase tiene la tarea de desplegar un cuadro de dialogo, donde el usuario puede elegir la opciones si no cancelar, y luego este establece una conexión con el objeto que lo llamo y le devuelve la respuesta del usuario, estableciendo una comunicación con un método de este.

II.3.1 VARIABLES

II.3.2 MÉTODOS

II.3.2.1 archivo

Descripción **:** Diseño de la interfaz gráfica de este cuadro de dialogo.

Definición **:** public archivo(Frame,String ,String)

Parámetros de entrada **:** Frame del método que lo invoca, titulo de la ventana, mensaje que desplegara.

Valor retornado **:** Ninguno

Métodos Invocados **:** Ninguno

II.3.2.2 action

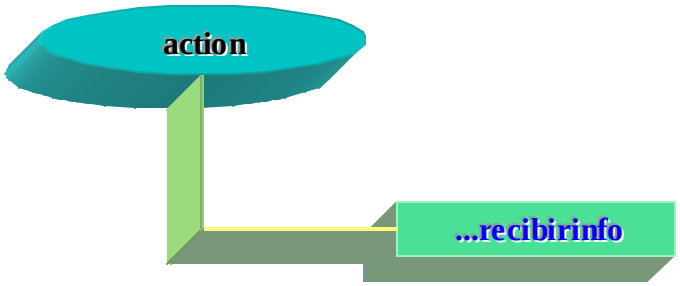


DIAGRAMA.23

Descripción	**:** Encargado de realizar una acción cuando se realiza un evento sobre algún objeto de este cuadro de dialogo.
Definición	**:** public boolean action(Event evt, Object arg)
Parámetros de entrada	**:** Evento realizado , objeto sobre el que se realizo el evento.
Valor retornado	**:** Directamente true o false si se realiza o no la acción, indirectamente envía true o false a un método que invoca dependiendo de la respuesta del usuario.
Métodos Invocados	**:** ENCRYDESENCR.recibirinfo(boolean)

Descripción de la lógica : Se recibe el nombre del padre que lo llamo, él titulo con el cual lo llama y el mensaje que será mostrado por el cuadro de dialogo que se crea; si el usuario pulsa el botón de “si” se llama a recibirinfo() para que se procese la información ; si pulsa el botón de “no”, se le devuelve la información para que el usuario la corrija.

II.4 class errores_archivo extends Dialog

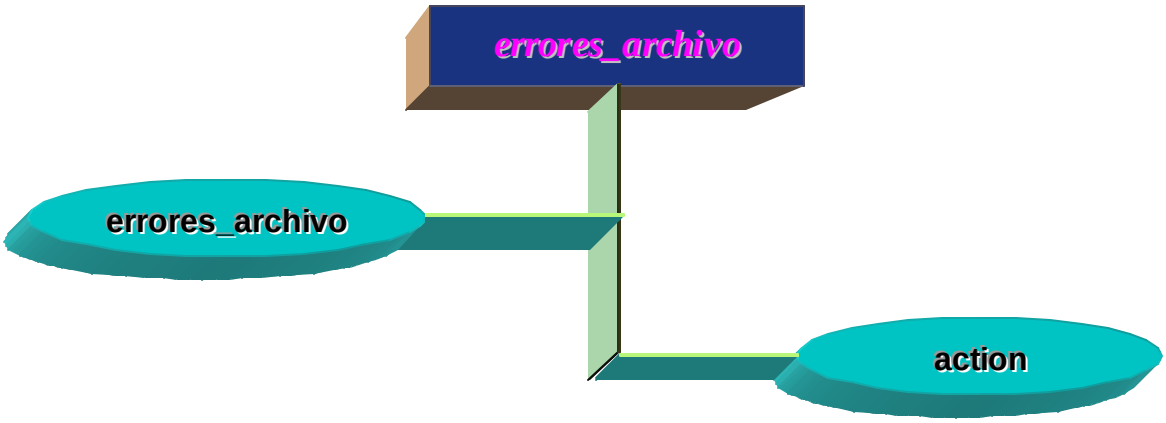


DIAGRAMA.24

Descripción: Despliega un cuadro de dialogo de los errores que se cometen.

II.4.1 VARIABLES COMUNES

II.4.2 METODOS

I.4.2.1 errores_archivo

Descripción	**:** Encargado de la interfaz gráfica de este cuadro de dialogo.
Definición	**:** public errores_archivo(Frame, String , String)
Parámetros de entrada	**:** Frame que lo invoca, titulo del cuadro de dialogo, mensaje de error que imprimirá.
Valor retornado	**:** Ninguno.
Métodos Invocados	**:** Ninguno.

II.4.2.2 action

Descripción	**:** Encargado de realizar una acción cuando se realiza un evento sobre algún objeto de este cuadro de dialogo.
Definición	**:** public boolean action(Event evt, Object arg)

Parámetros de entrada **:** Evento realizado , objeto sobre el que se realizo el evento.

Valor retornado **:** true o false, dependiendo si se realizo o no, la acción.

Métodos Invocados **:** Ninguno.

Descripción de la lógica **:** Muestra un cuadro de dialogo donde se informa al usuario que ha ocurrido un error y una vez haga clic en “aceptar” el cuadro de dialogo se destruirá.

II.5 class Imprimir extends Frame

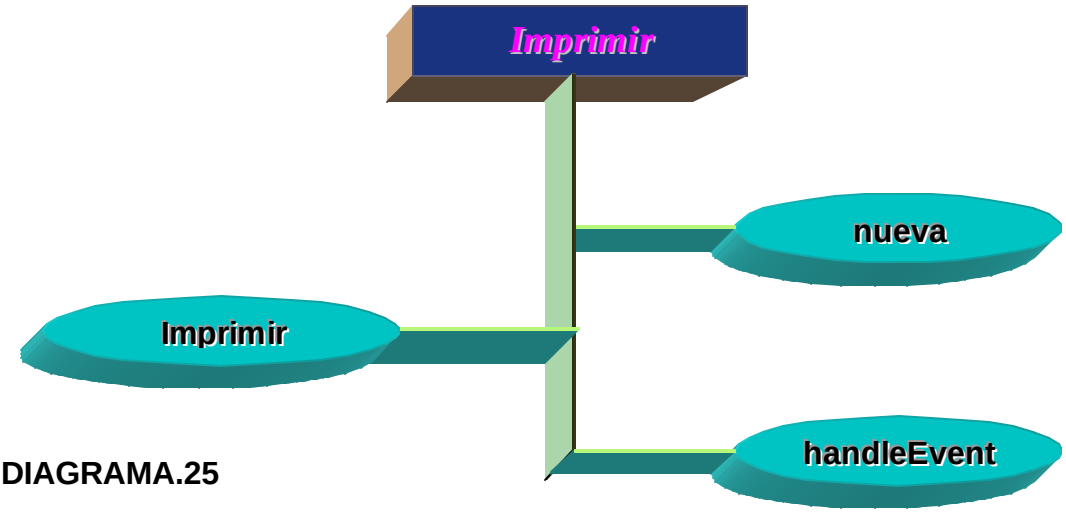


DIAGRAMA.25

Objeto utilizado para imprimir un documento.

Nota: **La versión de Java 1.1, que utilizamos no traía consigo métodos para imprimir cadenas de caracteres, solo imprimía un área de la pantalla, por lo que**

recurrimos a imprimir un objeto en pantalla y luego esta imagen mandarla a la impresora, en la versión de Microsoft puede que este método no funcione.

II.5.1 VARIABLES

- **texto** guarda la información a imprimir
- **texto1** auxiliar.

I.5.2 METODOS

I.5.2.1 Imprimir

Descripción **:** Imprime en pantalla el área de texto que ira a la impresora.

Definición **:** public Imprimir (TextArea)

Parámetros de entrada **:** Area de texto a imprimir

Valor retornado : **Ninguno.**

Métodos Invocados **:** Ninguno.

Descripción de la lógica **:** Le asigna el nombre al método, se agrega un área de texto con un tipo de letra por defecto y se le fija el fondo de color blanco; se fija en no editable y se centra en la pantalla.

II.5.2.2 nueva

Descripción **:** Controla todo el proceso para imprimir el documento, el arrea de texto lo guarda en un archivo temporal, lo lee linea por linea, imprime en el nuevo arrea de texto, claro esta solo las líneas que se pueden imprimir, y repite el proceso hasta la ultima linea.

Definición **:** public void nueva()

Parámetros de entrada **:** Ninguno Directamente.

Valor retornado : **Ninguno Directamente, pero automáticamente cierra la ventana actual.**

Métodos Invocados **:** Ninguno.

Descripción de la lógica **:** Crea un archivo temporal para escribir en el todo lo que contiene el área de texto, se abre el archivo

temporal en forma de lectura; se realiza un ciclo ‘while’ para leer el archivo línea por línea, luego imprime lo que ha leído en una hoja y el resto lo imprime en otras. Una vez termina se destruye el archivo temporal.

II.5.2.3 handleEvent

Descripción	**:** Captura cualquier evento ocurrido, si es así destruye la ventana .
Definición	**:** public boolean handleEvent(Event evt)
Parámetros de entrada	**:** Evt El tipo de evento.
Valor retornado	: **Ninguno.**
Métodos Invocados	**:** Ninguno.

II.6 class autores extends Frame

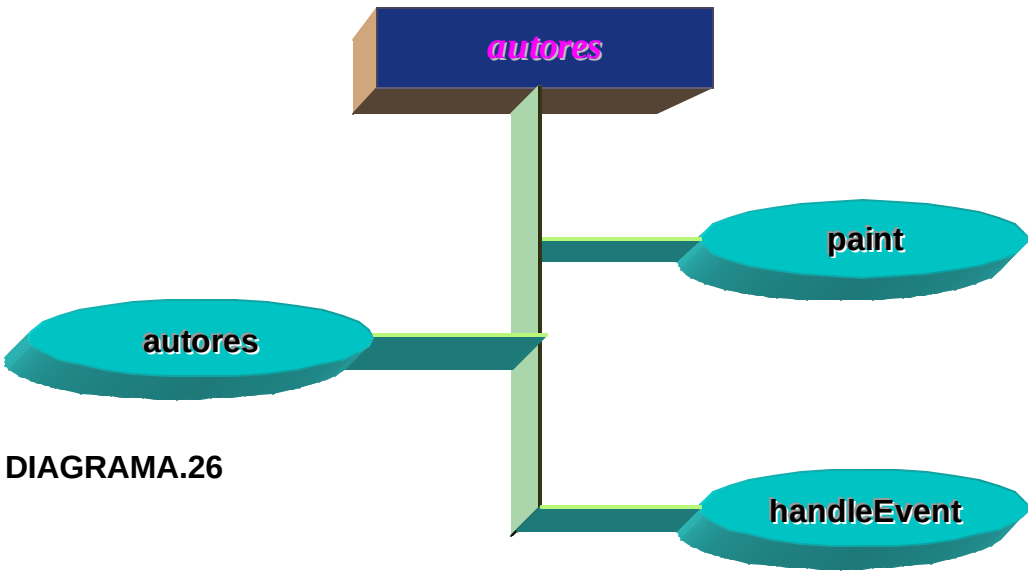


DIAGRAMA.26

Muestra el cuadro de dialogo versión llama una imagen donde se encuentra la información sobre la versión.

II.6.1 METODOS

II.6.1.1 public autores()

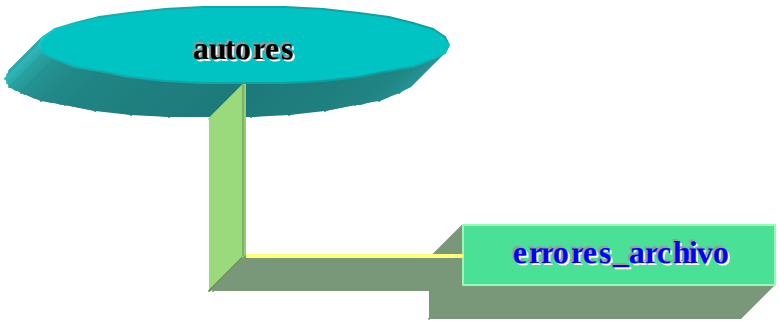


DIAGRAMA.27

Descripción : **Diseño de la interfaz gráfica de este cuadro de dialogo, carga e imprime las imágenes en pantalla.**

Definición : **public autores()**

Parámetros de entrada : **Ninguno Directamente.**

Valor retornado : **Ninguno.**

Métodos Invocados : **errores_archivo(Frame, String, String)**

Descripción de la lógica : Crea un cuadro de dialogo con el titulo de "versión". Carga las imágenes y espera hasta que todas estén listas para mostrarlas en pantalla, si ocurre un error llama al método errores_archivo para informarle al usuario.

II.6.1.2 handleEvent

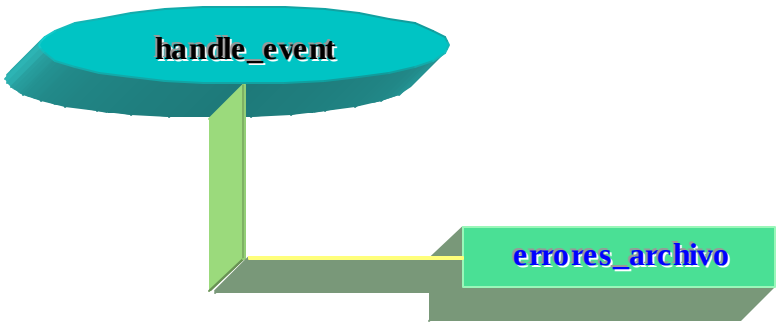


DIAGRAMA.28

Descripción : **Monitorea cuando ocurre un evento, destruye la ventana actual.**

Definición **:** public boolean handleEvent(Event evt)

Parámetros de entrada : **El evento ocurrido**

Valor retornado : **true o false dependiendo si se proceso el evento.**

Métodos Invocados : **errores_archivo(Frame, String, String)**

Descripción de la lógica : **Preguntamos por el evento de destruir la ventana y realizamos esa acción.**

II.6.1.3 paint

Descripción : **Dibuja una imagen en la ventana actual.**

Definición **:** public void paint(Graphics)

Parámetros de entrada : **El evento ocurrido**

Valor retornado : **Ninguno.**

Métodos Invocados : **Ninguno.**

Descripción de la lógica : Captura el tamaño del monitor y dibuja las imágenes.

II.7 class ayuda extends Frame

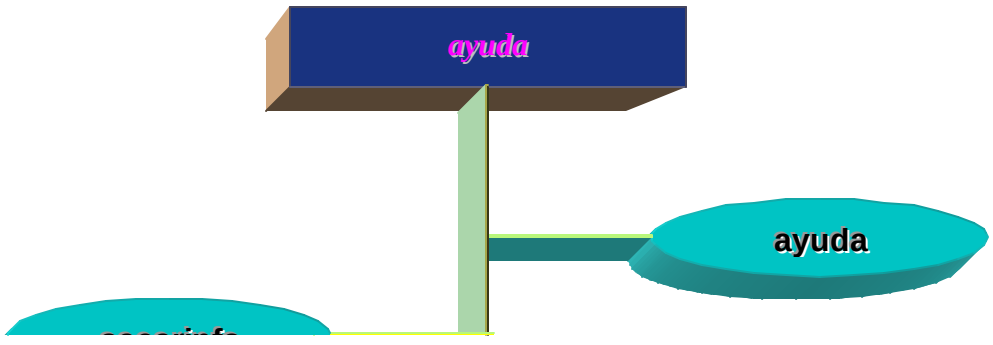


DIAGRAMA.29

Despliega una ventana con la ayuda de la aplicación, crea un área de texto dentro de la cual muestra un manual de la aplicación.

II.7.1 MÉTODOS

I.7.1.1 ayuda

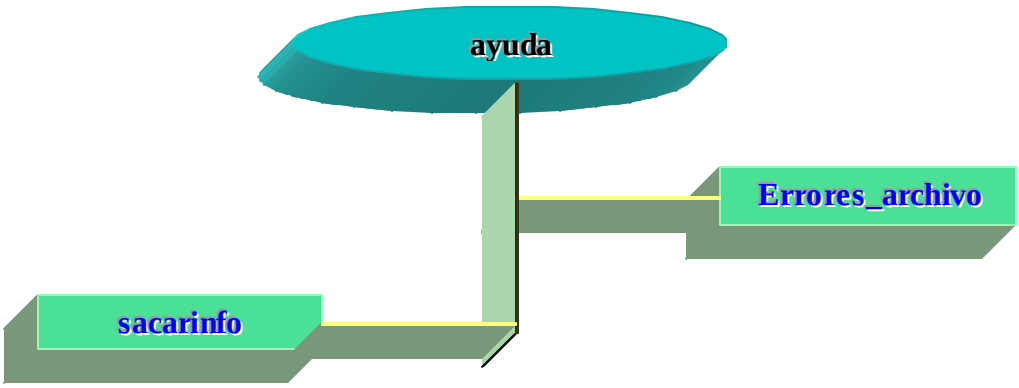


DIAGRAMA.30

Descripción **:** Interfaz gráfica de este contenedor.

Definición : **public ayuda()**

Parámetros de entrada **:** Ninguno.

Parámetros de salida : **Ninguno.**

Métodos Invocados : - **errores_archivo(Frame , String, String)**

- sacarInfo()

II.7.1.2 sacarinfo

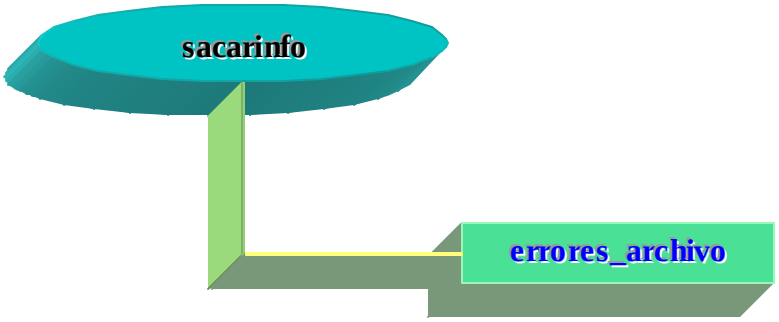


DIAGRAMA.31

Descripción **:** Lee el texto del archivo de ayuda y actualiza el arrea de texto.

Definición **:** public void sacarInfo()

Parámetros de entrada **:** Ninguno.

Parámetros de salida : **Ninguno.**

Métodos Invocados : - **errores_archivo(Frame , String, String)**

Descripción de la lógica : Crea un objeto para accesar al archivo MANUAL_ENCRYDESENCR de tipo texto; lo muestra en pantalla para que el usuario lea la información que necesite. Si ocurre un error se invoca al método errores_archivo() para informarle al usuario.

II.7.1.3 handleEvent

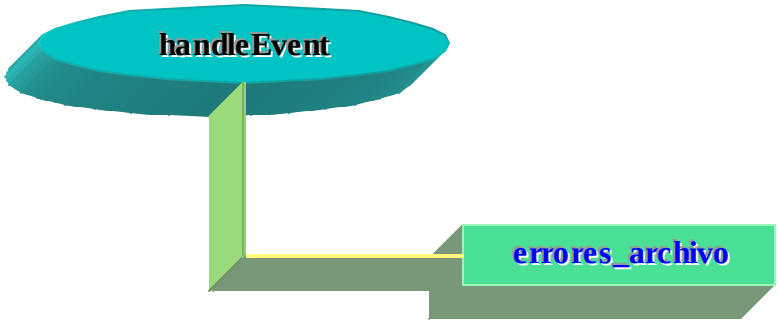


DIAGRAMA.32

Descripción : **Monitorea cuando ocurre un evento, destruye la ventana actual.**

Definición **:** public boolean handleEvent(Event evt)

Parámetros de entrada : **El evento ocurrido**

Valor retornado : **true o false dependiendo si se proceso el evento.**

Métodos Invocados : **errores_archivo(Frame, String, String)**

II.8 CLASE Clave_Pub extends Dialog

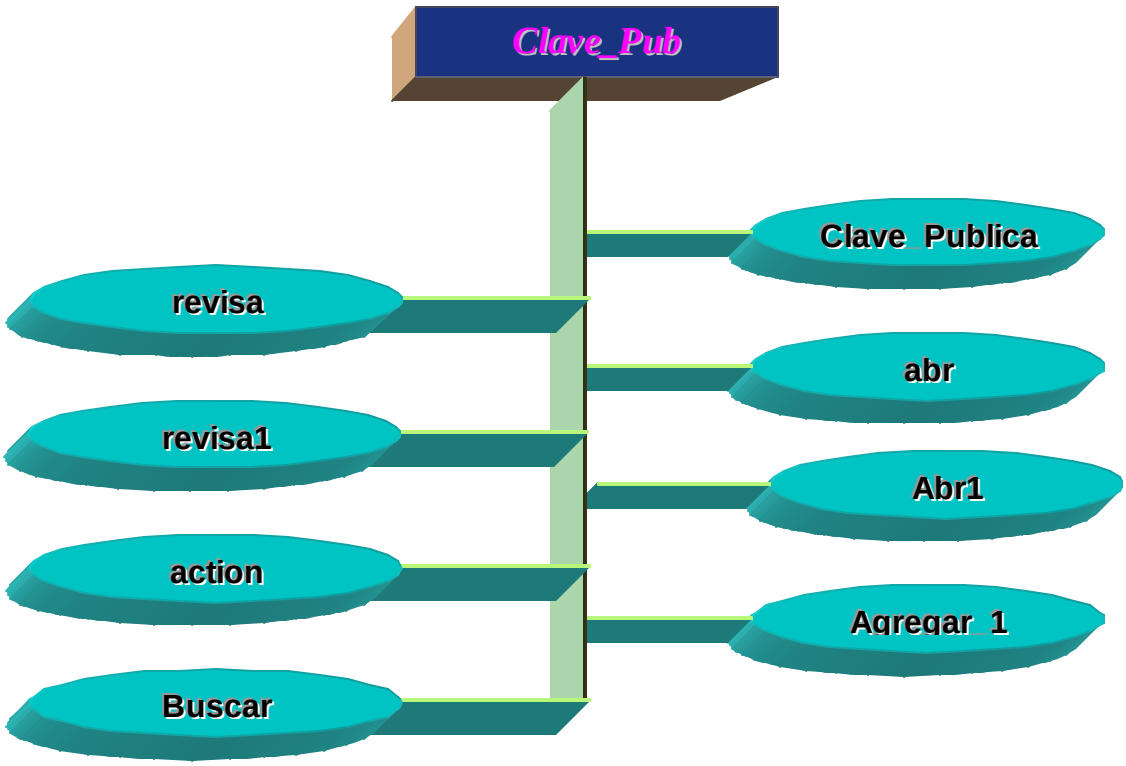


DIAGRAMA.33

Es utilizada para manejar la base de datos, crea un objeto de tipo dialogo donde monitorea las diferentes ordenes del usuario y cuida de que estas se realicen.

Tabla Clave_Publica

II.8.1 VARIABLES

- **archivo1, archivo2.** Contendrán nombres de los archivos
- **mensa.** Guarda el mensaje que le muestra al usuario, sobre la actividad a realizar.
- **d2.** Lista los Identificadores que son leídos de la base de datos.
- **Lista.** Contiene la informacion leída de la base de datos.
- **Name.** Guarda el nombre, que actualiza la BD.
- **ID.** Guarda el identificador, que actualiza la BD.
- **Cpu.** Guarda clave publica, que actualiza la BD.
- **DEV1[]. Guarda los datos devueltos por la BD.**

II.8.2 METODOS

I.8.2.1 Clave_Publica

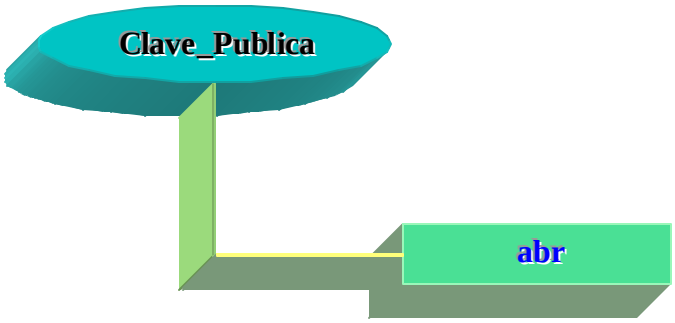


DIAGRAMA.34

Descripcion **:** Inicializa las variables, y llama a la primer pantalla que muestra el cuadrto de dialogo.

Definicion **:** Clave_Publica(Frame ,String [ ]).

Parámetros de entrada : Frame que lo llama, Arreglo que sera modificado.

Valor retornado **:** Implicitamente DEV [ ] con la nueva informacion.

Métodos Invocados **:** Abr()

Descripción de la lógica **:** Se fijan las áreas de texto en nulo y se invoca al método Abr() para que actualice la pantalla.

II.8.2.2 abr( )

Descripcion **:** Despliega el diseño inicial del cuadro de dialogo, este intercambia informacion con el usuario sobre lo que desea hacer en la BD.

Definición **:** public void Abr()

Parámetros de entrada : Ninguno

Valor retornado **:** Ninguno

Métodos Invocados **:** Ninguno

Descripción de la lógica : Fija todos los campos en nulos y agrega todas las etiquetas que se necesita para actualizar la pantalla para el usuario; junto con los botones.

II.8.2.3 abr1( )

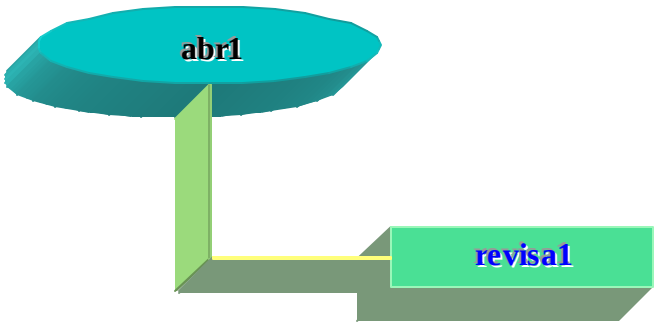


DIAGRAMA.35

Descripcion **:** Despliega un nuevo acomodo del diseño del cuadro de dialogo cuando el usuario hace clic en “abrir”.

Definición **:** public void Abr1()

Parámetros de entrada : Ninguno

Valor retornado **:** Ninguno

Métodos Invocados **:** revisa1()

Descripción de la lógica **:** Fija todos los campos en nulo y agrega las etiquetas correspondientes; se llama a revisa1() para que verifique las etiquetas que correspondan a la información que será mostrada.

II.8.2.4 Agregar_1( )

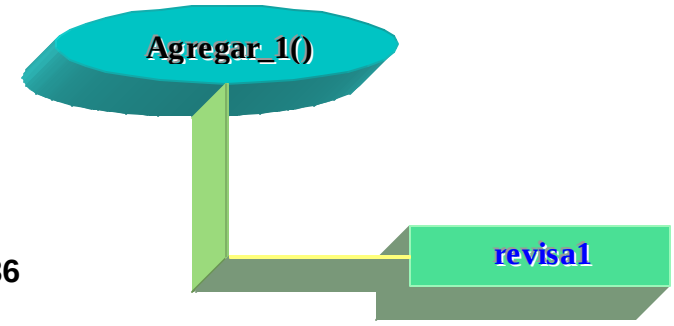


DIAGRAMA.36

Descripción **:** Despliega un nuevo acomodo del diseño del cuadro de dialogo cuando el usuario hace clic en “agregar”.

Definición **:** public void Agregar_1()

Parámetros de entrada : Ninguno

Valor retornado **:** Ninguno

Métodos Invocados **:** revisa1()

Descripción de la lógica **:** Se fijan las áreas de texto en nulo, se agregan las etiquetas y se fijan las áreas de texto de forma editable; se invoca el método revisa1() para que verifique las áreas de texto; se adiciona la información y el botón con la etiqueta “buscar”.

II.8.2.5 revisa( )

Descripción **:** Revisa la etiqueta de los botones después de un nuevo acomodo del diseño del cuadro de dialogo.

Definición **:** public void revisa()

Parámetros de entrada : Ninguno

Valor retornado **:** Ninguno

Métodos Invocados **:** Ninguno

Descripción de la lógica **:** Pregunta que si la etiqueta que tienen los botones 1, 2, ó 3 es “Aceptar” y la cambia dependiendo de la acción que halla tomado; ya sea Abrir, Eliminar, o Agregar.

II.8.2.6 revisa1( )

Descripción	**:** Imprime en pantalla un mensaje al usuario indicándole la operación a realizar.
Definición	**:** public void revisa1()
Parámetros de entrada	: Ninguno
Valor retornado	**:** Ninguno
Métodos Invocados	**:** Ninguno
Descripción de la lógica	**:** Pregunta por la etiqueta de los botones 1, 2 ó 3 y las cambia dependiendo de la acción que halla tomado el usuario.

II.8.2.7 action

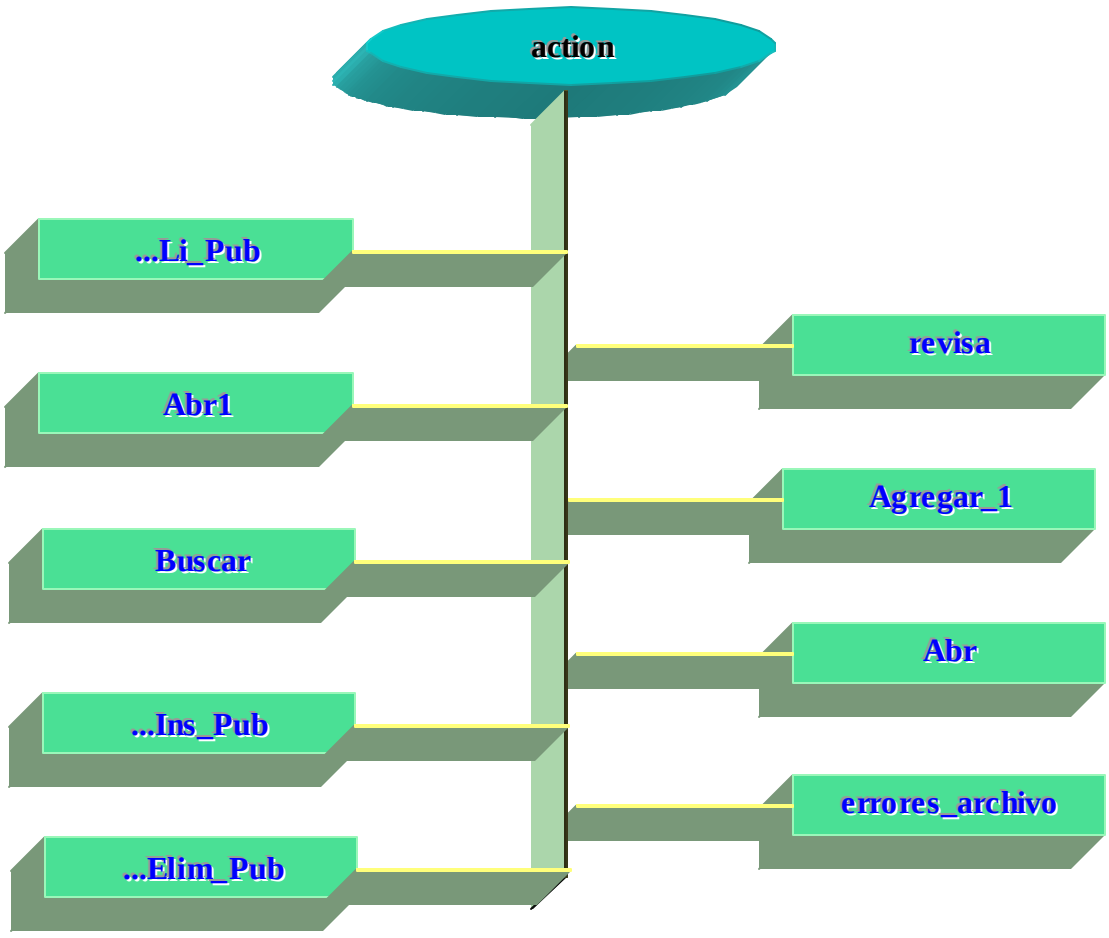


DIAGRAMA.37

Descripción **:** Captura bs diferentes eventos sobre los objetos de la ventana actual (Button,Choice. . .) , y monitorea la realización de la acción a ejecutar.

Definición **:** public boolean action(Event evt, Object arg)

Parámetros de entrada : El evento y el objeto sobre el que se debe realizar la acción.

Valor retornado **:** Directamente true si el evento fue procesado o false en caso contrario, Indirectamente actualiza la variable de tipo arreglo recibida.

Métodos Invocados **:** - BaseDeDatos.Li_Pub()

- revisa();
- Abri1()
- Agregar_1()
- Buscar(int);
- Abr();
- BaseDeDatos.ins_Pub(String [ ]);
- errores_archivo(Frame, String, String)
- BaseDeDatos.Elim_Pub(String);

Descripción de la lógica : Se pregunta por el evento que si es ‘Choice’ y se realiza la acción correspondiente; se muestra la lista en el área de texto para que el usuario seleccione un nombre. Si el evento se registro sobre un botón se pregunta cual de ellos si es “Abrir” se crea un objeto de tipo “BaseDeDatos”, se listan las claves publicas y se agregan al elemento ‘choice’ , se revisan las etiquetas de los botones con el método revisa() y se limpian las áreas de textos; se invoca el método Abr1() ya que el usuario hizo clic en el botón “Abrir”. Si es “Agregar”, se cambian las etiquetas a los botones; al botón 2 se le asigna la etiqueta “Aceptar”, se remueve toda la información, se capturan los datos del usuario y se agregan a la base de datos. Si es “Buscar” se invoca el método Buscar() para que realice la operación. Si es

“Eliminar”, se crea un objeto de tipo BaseDeDatos y se listan todos los elementos en ‘Choice’; se revisan las etiquetas con el método revisa() y se le cambia la etiqueta al botón 3 por “Aceptar”; se remueve la información que ha sido seleccionada por el usuario y se invoca el método Abr1(). Si es “Cancelar” se revisan las etiquetas de los botones para saber cual de ellos tiene la etiqueta, se llama el método revisa(), se remueve la información y se invoca el método abr() para que el usuario sepa lo que va a hacer; si el botón es “Aceptar” se verifica cual de ellos tiene esa etiqueta, se declara un objeto tipo ‘String’, se revisan las áreas de texto que estén en blanco o vacías; se obtienen los datos teclados por el usuario y se asignan a la base de datos con ins_Pub() se abre los archivos de claves que van a ser asignados por el usuario y se limpian todos los campos nuevamente. Si los campos no están llenos se envía un mensaje de error.

II.8.2.8 Buscar

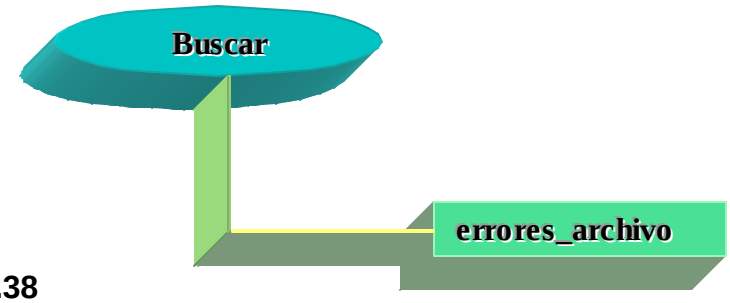


DIAGRAMA.38

Descripción **:** Despliega un cuadro de dialogo buscar, cuando se selecciona el archivo donde se debe encontrar la clave publica o privada , si es publica lo abre lo lee y actualiza la variable donde se almacena la clave publica, si es privada actualiza la variable donde se

almacena locación del archivo donde se encuentra la clave privada.

Definición **:** public void Buscar(int)

Parámetros de entrada **:** Entero que define la opción si busca archivos donde se encuentra la clave publica , o privada.

Valor retornado **:** Directamente ninguno.

Métodos Invocados **:** errores_archivo(Frame, String, String)

Descripción de la lógica **:** Crea un cuadro de dialogo que dependiendo del parámetro recibido muestra los archivos de claves de extensión ".Lcc" o ".Lic"; si son los de clave publica se obtiene la dirección, se crea un objeto de tipo archivo para leer el contenido del archivo y mostrarlo en el área de texto.

II.9 GENERADOR_DE_CLAVE_PUBLICA

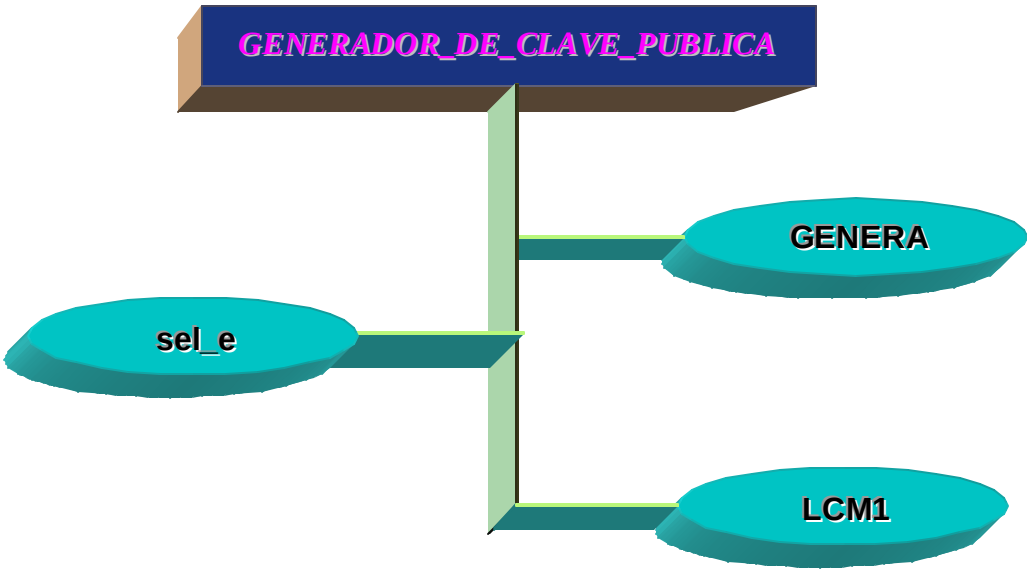


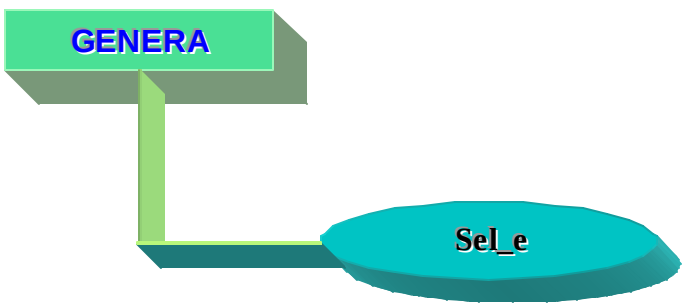
DIAGRAMA.39

Contiene Métodos que son utilizados para el desarrollo del algoritmo de Lucas, el método principal es GENERA.

II.9.1 METODOS

II.9.1.1 GENERA

DIAGRAMA.40



Descripción : Genera p, q y e , También calcula las claves privadas Lucas

Definición **:** public Vector GENERA()

Parámetros de entrada **:** Ninguno directamente pero necesita la hora actual en milisegundos.

Valor Retornado **:** Vector de 8 posiciones con:

0 - N

1 - e

2 - clave privada 1

3 - clave privada 2

4 - clave privada 3

5 - clave privada 4

6 - p

7 - q

Métodos Invocados : sel_e(Z,Y)

Descripción de la lógica **:** Obtenemos la fecha del sistema y convertimos este entero largo en una cadena. Preguntamos que si su longitud es mayor de 7 o si es menor con el fin de poder generar la números aleatorios; una vez se tiene la longitud deseada convertimos nuevamente

la cadena a entero largo y enviamos este valor a la función ‘Random()’ creamos 3 objetos de tipo BigIntegrer de los cuales 2 tienen longitud entre 300 y 210 dígitos; con estos números invocamos el método Sel_e() el cual nos devolverá los valores de N, la clave publica y la clave privada.

II.9.1.2 sel_e(Z,Y)

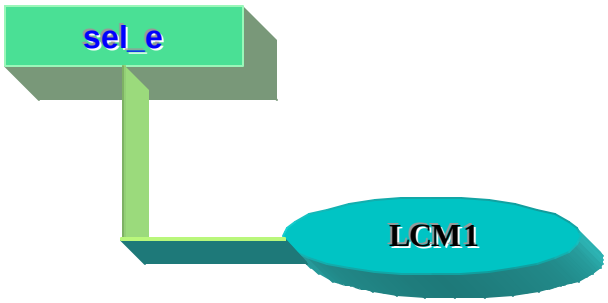


DIAGRAMA.41

Descripción **:** A partir de p y q y genera las diferentes clave privadas.

Definición : **public Vector sel_e(java.math.BigInteger, java.math.BigInteger)**

Parámetros de entrada **:** Los números primos p ,q

Valor retornado : **Un vector con: N,e, K1,K2,K3,K4**

Métodos Invocados **:** LCM1(java.math.BigInteger, java.math.BigInteger)

Descripción de la lógica **:** Se crean 7 variables de tipo BigInteger, se calcula al número N, se calcula la multiplicación de (p+1)(p-1)(q+1)(q-1), se genera un nuevo número para hallar otro número que sea primo relativo del generado por la multiplicación anterior. Se adicionan a un vector todos los elementos hallados.

II.9.1.3 LCM1

Descripción **:** Método utilizado para calcular el mínimo común múltiplo de 2 números.

Definición : **public java.math.BigInteger LCM1(java.math.BigInteger , java.math.BigInteger)**

Parámetros de entrada : **Dos valores numéricos cualquiera.**

I. **Valor retornado** **:** El mínimo común múltiplo de los 2 valores de entrada.

Métodos Invocados **:** Ninguno.

Descripción de la lógica : Recibe como parámetro dos números de tipo BigInteger para hallar entre ellos el mínimo común múltiplo, primero halla el gran común divisor luego, este valor lo divide entre el segundo número, finalmente el resultado lo multiplica por el primer número y retorna este valor.

II.10 public class clave

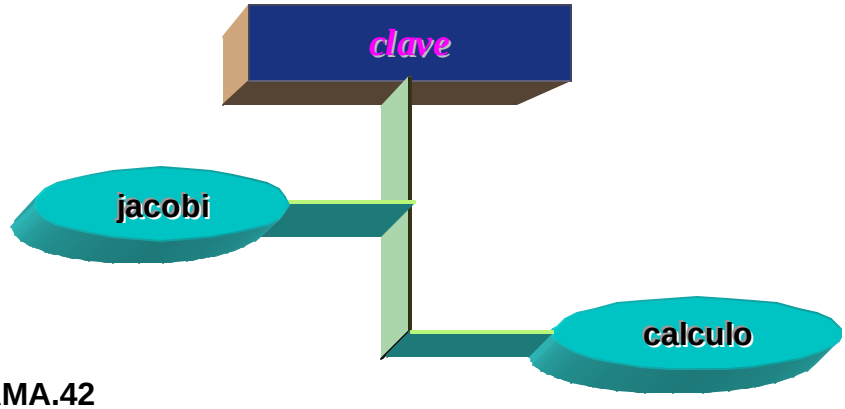


DIAGRAMA.42

Contiene métodos para calcular la secuencia de Lucas.

I.10.1 METODOS

II.10.1.1 calculo

Descripción : Calcula el valor correspondiente según la secuencia de Lucas a un valor recibido.

Definición : public java.math.BigInteger calculo (java.math.BigInteger , java.math.BigInteger e,java.math.BigInteger N)

Parámetros de entrada **:** Par clave publica (e,N) y otro valor numérico al que se le calculara la secuencia de Lucas.

Valor Retornado : **Numero calculado según la secuencia de Lucas. (Vt)**

Métodos Invocados **:** Ninguno.

Descripción de la lógica **:** Se calcula el determinante del texto plano recibido; se procede a hallar el valor de Ve(P, 1); se pregunta si la clave publica es par, si es así se le asigna el valor de 0 y 2 a dos variables diferentes respectivamente, sino se divide entre dos y se escoge la parte entera; y hasta que este valor sea cero se hace en un ciclo lo siguiente: a una variable se le asigna el texto plano mientras que a otra se le asigna el texto plano al cuadrado. Se pregunta que si el texto al cuadrado modulo N es diferente de tres se le agrega el valor de N y se le resta 2; sino se pregunta que si la clave publica modulo 2 es igual a 1 entonces, se hacen operaciones para hallar Ve. Al final se retorna el valor que obtenga la variable v.

II.10.1.2 Jacobi

Descripción **:** Calcula el valor Jacobi de 2 números, 1 o –1. Calcula si es un residuo cuadratico.

Definición **:** public int Jacobi(java.math.BigInteger , java.math.BigInteger)

Parámetros de Entrada **:** Dos valores numéricos cualquiera.

Valor Retornado : **1 o –1, dependiendo si el residuo es un residuo cuadratico, según la aritmética modular.**

Métodos Invocados **:** Ninguno.

Descripción de la lógica **:** Se reciben dos enteros grandes a y b; se pregunta en un ciclo ‘while’ que si el valor de a es igual a , entonces se hace un modulo 2 con el valor de a y se le asigna el resultado a otra variable. Se pregunta que si el resultado es igual a cero entonces que el valor de b lo eleve al cuadrado y le reste 1, y con el resultado realice un modulo 2, el resultado }se le asigna a la variable test . todo esto con el fin de saber si un número es residuo cuadratico de otro. Luego el valor de a se divide entre dos. Y se

pregunta que si el valor de test no es igual a cero entonces a la variable sing que la multiplique por –1. Sino que a la variable test le asigne (a-1)*(b-1) dividido entre 4 modulo 2, a una variable temporal llamada temp se le asigna el valor de a; luego a se le asigna el valor de b modulo temp y a b se le asigna temp. Preguntamos si test no es igual a cero entonces a la variable sing se le multiplica por –1 y se repite todo el ciclo ‘while nuevamente. Preguntamos que si a es cero le asignamos 0 a la variable sing. retornamos el valor de sing.

II.11 public class Idea

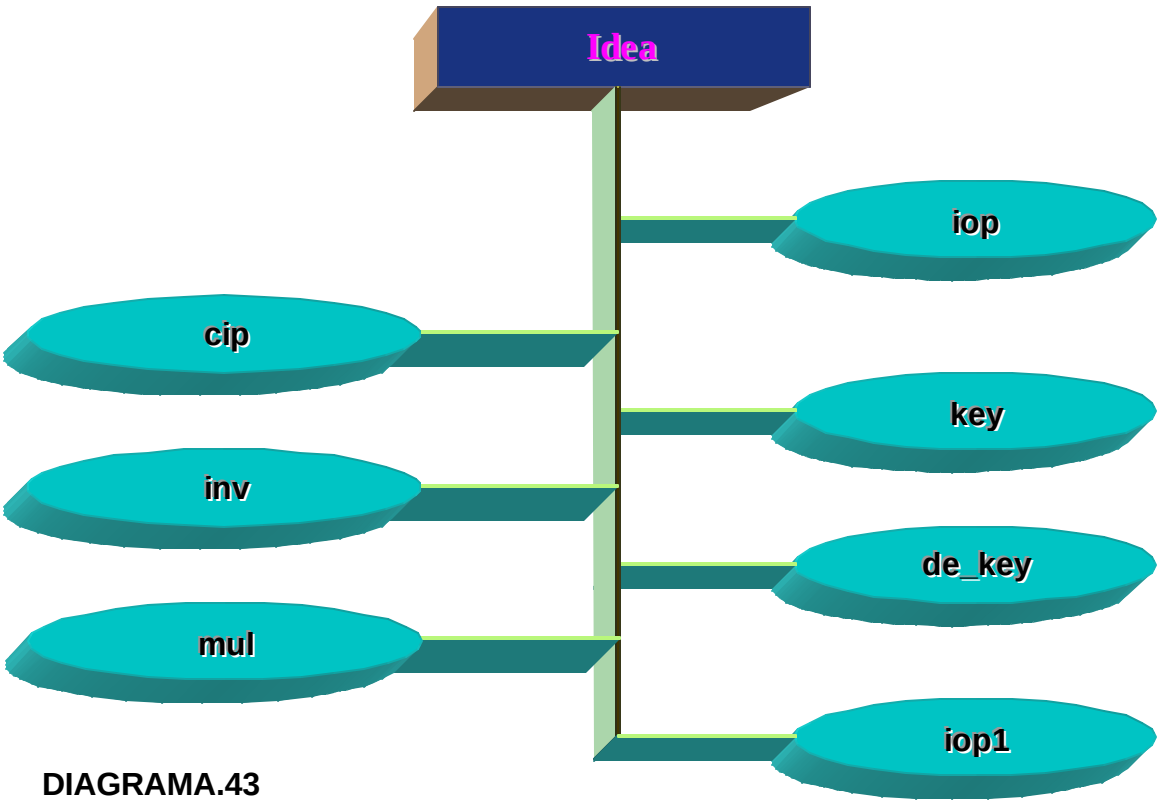


DIAGRAMA.43

Se encuentran los procesos de encripción y desencripción con el algoritmo idea

recibe bloques de 64 bits y una clave de 128 bits

II.11.1 VARIABLES

- **maxim, fuyi , one.** para convertir a un valor UNSIGNED.

II.11.2 METODOS

II.11.2.1 iop

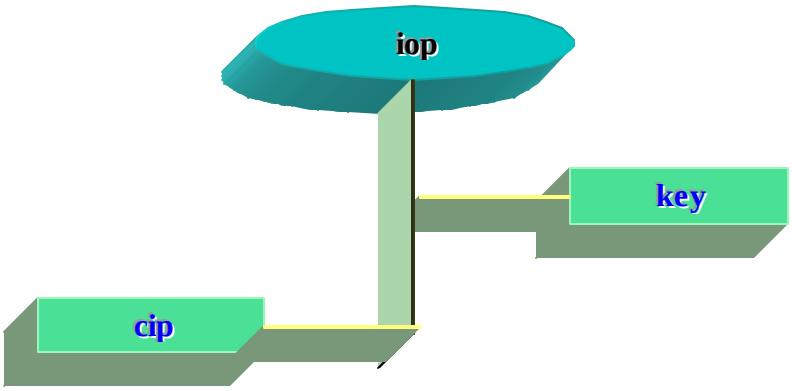


DIAGRAMA.44

Descripción : **Controla el proceso de encripción utilizando el cifrador IDEA.**

Definición **:** public Vector iop(java.math.BigInteger ,java.math.BigInteger)

Parámetros De Entrada : **Clave de 128 bits y un Bloque 64 bits(4 palabras de 16 bits) que será encriptado.**

Valor Retornado **:** Un bloque de 64 bits encriptado, en un vector de 4 posiciones cada una con 160 bits.

Métodos Invocados :- **key(uskey,Z)**

- cip(XX,YY,Z)

Descripción de la lógica : Se recibe el texto plano y la clave como parámetros; se una variable para guardar las subclaves, y otra variable para retornar el valor encriptado. Se inicializan las variables con valor de cero. Hacemos un ciclo 'for' para tomar los valores de la clave convirtiéndolos en valores numéricos. Invocamos el método key() para obtener las subclaves. Asignamos los bloques de texto plano que van a ser encriptados a sus respectivas variables; invocamos el método cip() que ser encarga de encriptar los subbloques de texto con sus respectivas subclaves; adicionales el texto

encriptado a su respectiva variable y retornamos el valor encriptado.

II.11.2.2 cip

Descripción	: **Proceso de encripción IDEA.**
Definición	**:** public void cip(java.math.BigInteger [ ] , java.math.BigInteger[] , java.math.BigInteger [ ][ ])
Parámetros De Entrada	**:** Texto a encriptar o desencriptar, variable que modificara(retornara los nuevos valores hallados), Subclaves de encripción o desencripción.
Valor Retornado	**:** Directamente ninguno, pero modifica el arreglo que recibió como parámetro(El nuevo valor del

arreglo recibido es la encripción o desencripción del otro)

Métodos Invocados : **Ninguno.**

Descripción de la lógica : Recibimos como parámetros el texto plano y las subclaves; obtenemos los valores del texto plano en variables temporales. Se hacen las operaciones básicas de Idea con las primeras subclaves. Hacemos la estructura de operación de la función MA (ver descripción del Algoritmo IDEA). Realizamos la permutación involuntaria PI y la transformación a la salida para retornar el bloque de texto cifrado.

II.11.2.3 key

Descripción : **Genera las subclaves de encripción recibe la clave del usuario**

Definición : public void key (java.math.BigInteger [ ], java.math.BigInteger[ ][ ])

Parámetros De Entrada : Clave del usuario (valor numérico), y una variable de tipo arreglo que modificara.

Valor Retornado **:** Directamente ninguno, pero modifica el arreglo que recibió como parámetro(El nuevo valor, del arreglo recibido, contendrá las subclaves de encripción).

Métodos Invocados : **Ninguno.**

Descripción de la lógica : Recibe la clave del usuario y generamos las subclaves de encripción. Inicializamos la variable que contendrá los diferentes desplazamientos que se realizan a la subclave. En las primeras ocho posiciones estarán los bits en forma original. Al resto se le realizan los desplazamientos que ordena IDEA. Después se obtienen las subclaves para cada iteración y se retorna éste valor.

II.11.2.4 inv

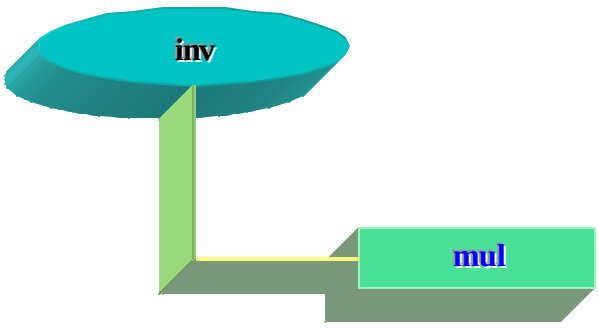


DIAGRAMA.45

Descripción : **Calcula en inverso de un numero, claro esta modularmente.**

Definición **:**public java.math.BigInteger inv(java.math.BigInteger)

Parámetros De Entrada **:** Valor numérico a la que se hallara la inversa modular.

Valor Retornado **:** Valor numérico que será la correspondiente inversa del numero.

Métodos Invocados : **mul.**

Descripción de la lógica **:** Recibe como parámetro la subclave de desencripción. Se pregunta que si este valor recibido es cero. Si es así se retorna el valor de cero. Sino se realiza operaciones que hace que el

valor recibido se haga uno. En un ciclo ‘do-While’ obtenemos el residuo y el cociente modular entre un valor constante (que en realidad no es constante sino que se parte de una base) y la subclave recibida. Preguntamos que si el residuo es cero entonces modificamos el valor base. Sino modificamos el residuo, el valor base y el valor que será retornado, cerramos el ciclo ‘do-while’cuando el residuo sea igual a 1.

II.11.2.5 de_key

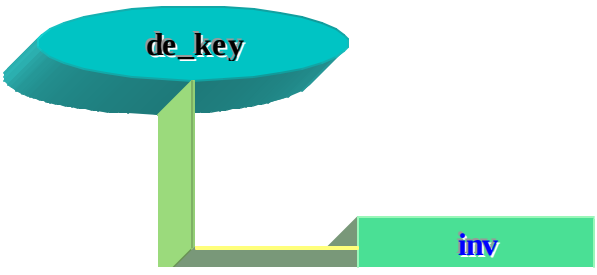


DIAGRAMA.46

Descripción : **Genera las subclaves de desencripción.**

Definición **:** public void de_key(java.math.BigInteger, java.math.BigInteger)

Parámetros De Entrada **:** Claves de encripción y una variable la cual modificara.

Valor Retornado **:** Directamente ninguno, pero en la variable recibida la modificara guardando en ella las subclaves de encripción.

Métodos Invocados : - **inv(java.math.BigInteger)**

Descripción de la lógica **:** Se reciben las subclaves de encripción y se devuelven las subclaves de desencripción. Generamos las subclaves que se necesitan para cada iteración; invocamos el método inv(). Para obtener la subclave de desencripción.

II.11.2.6 mul

Descripción **:** Implementación del algoritmo conocido como Low-High y que multiplica 2 números sin tener en cuenta el bit de signo, y teniendo en cuenta un modulo que en este caso es 65535.

Definición : **public java.math.BigInteger mul (java.math.BigInteger, java.math.BigInteger)**

Parámetros De Entrada **:** Valores que se multiplicaran.

Valor Retornado **:** El resultado de la multiplicación.

Métodos Invocados : **Ninguno**

Descripción de la lógica **:** Recibe dos enteros a y b los cuales se van a multiplicar.

Se crean dos variables p y q las cuales van a ayudar en el proceso. Preguntamos que si el valor de a es igual a cero si es así a la variable p le asignamos maxim (65537) menos el valor de b. Sino preguntamos que si el valor de b es cero y hacemos la misma operación pero con a. Sino a q le asignamos la multiplicación de a por b y, a p le asignamos el resultado de hacer la operación 'and' entre q y el valor de 65535 y al resultado le restamos q desplazado 16 bits a la derecha. Luego preguntamos que si p comparado con cero es igual a –1 ó 0, entonces que a p le asigne p más maxim (65537) y retorne p; sino que retorne p con una operación 'and' entre p y 65535.

II.11.2.7 iop1

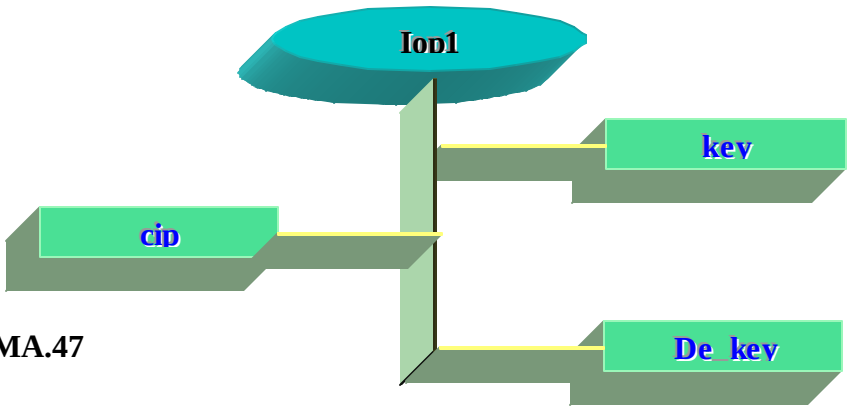


DIAGRAMA.47

Descripción : **Controla el proceso de desencripción utilizando el decifrador IDEA.**

Definición : public Vector iop1(java.math.BigInteger, java.math.BigInteger)

Parámetros De Entrada : **Clave de 128 bits y un Bloque 64 bits(4 palabras de 16 bits) que será desencriptado.**

Valor Retornado **:** Un bloque de 64 bits desencriptado, en un vector de 4 posiciones cada una con 160 bits.

Métodos Invocados :- **key(uskey, Z)**

- cip(YY, TT, DK)
- de_key(Z, DK)

Descripción de la lógica : Recibe el texto encriptado y la clave. Obtiene la clave digitada por el usuario y genera las subclaves de encripción para poder generar después las subclaves de desencripción. Invocando los métodos key() y De_key() respectivamente. Obtiene el texto encriptado e invoca el método cip() para que desencripte el texto cifrado. Retorna el texto desencriptado.

II.12 public class animador extends Frame implements Runnable

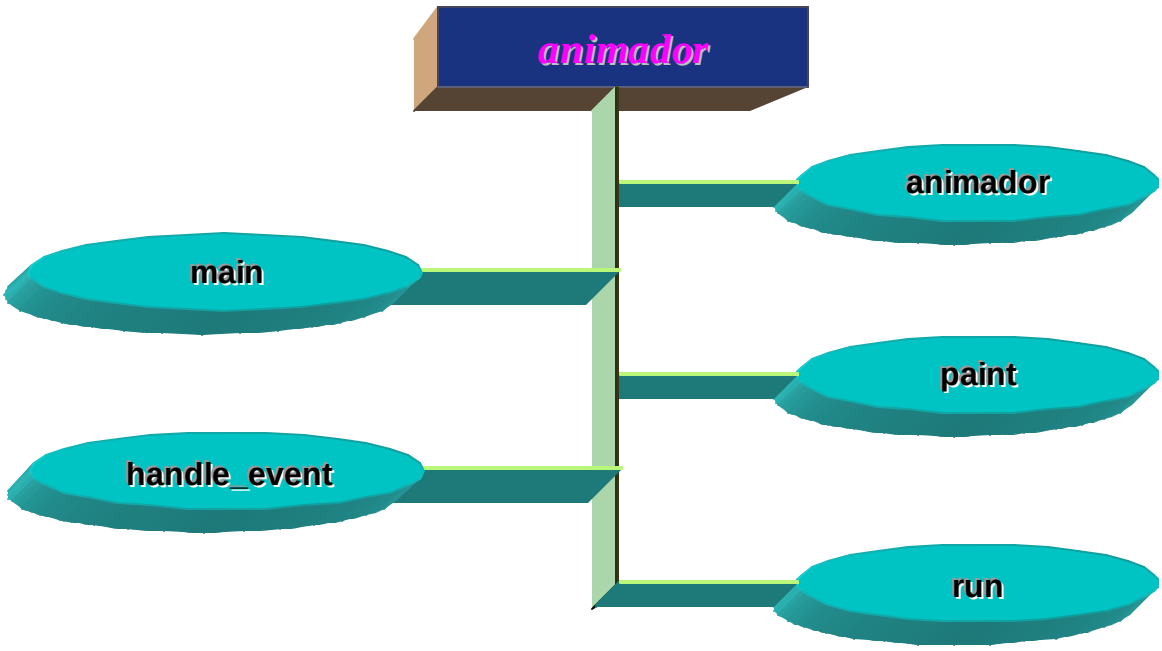


DIAGRAMA.48

Crea un hilo que carga una secuencia de imágenes y las muestra como una animación

II.12.1

- **anima.** nombre del hilo el cual se encargara de hacer la animación.

I.12.2 METODOS

I.12.2.1 animador

Descripción : **Crea el hilo para encargarse de la animación.**

Definición **:** animador(Thread)

Parámetros De Entrada : **Nombre del hilo padre.**

Valor Retornado **:** Ninguno.

Métodos Invocados : **Ninguno.**

Descripción de la lógica : Crea un hilo, se hace un ciclo ‘for’ del tamaño del grupo de imágenes (6) y las carga a un vector de imágenes.

II.12.2.2 main

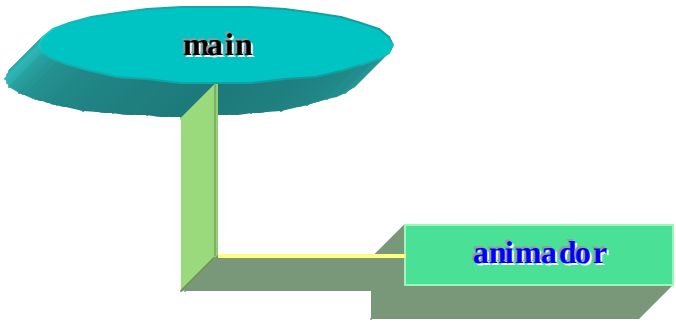


DIAGRAMA.49

Descripción : **Crea un objeto de tipo Thread, lo muestra e inicializa el hilo.**

Definición : public static void main(String argv [])

Parámetros De Entrada : **Ninguno.**

Valor Retornado : Ninguno.

Métodos Invocados :- **animador(j)**

Descripción de la lógica : Crea un objeto tipo hilo y otro de tipo animador; se muestra el objeto animador para que cargue el hilo padre se crea otro objeto tipo hilo para que inicialice el hilo hijo.

I.12.2.3 paint

Descripción : **Dibuja una imagen en el lugar especificado.**

Definición : public void paint(Graphics g)

Parámetros De Entrada : **El objeto gráfico g.**

Valor Retornado **:** Ninguno.

Métodos Invocados : **Ninguno.**

Descripción de la lógica : Obtiene las gráficas contenidas en el vector de imágenes y las dibuja.

I.12.2.4 handleEvent

Descripción : **Detecta el evento realizado sobre la ventana.**

Definición **:** public boolean handleEvent(Event evt)

Parámetros De Entrada : evt **es el evento realizado por el usuario.**

Valor Retornado **:** retorna el evento realizado por el usuario si este no es el de cerrar o iconizar la ventana.

Métodos Invocados : **Ninguno.**

Descripción de la lógica : Se pregunta que si el evento realizado es el de destruir la ventana. Si es así se destruye. Sino se pregunta que si el evento es el de iconizar la ventana entonces, se agrega una etiqueta y se retorna el evento.

II.12.2.5 run

Descripción : **Método que se encarga de poner a ejecutar un hilo.**

Definición : public void run()

Parámetros De Entrada : **Ninguno.**

Valor Retornado : Ninguno.

Métodos Invocados : **Ninguno.**

Descripción de la lógica : Se ejecuta un ciclo 'while' infinito y pinte las imágenes y retrase el hilo durante 5 milisegundos. Si hay una interrupción se sale del ciclo.

II.13 public class BaseDeDatos

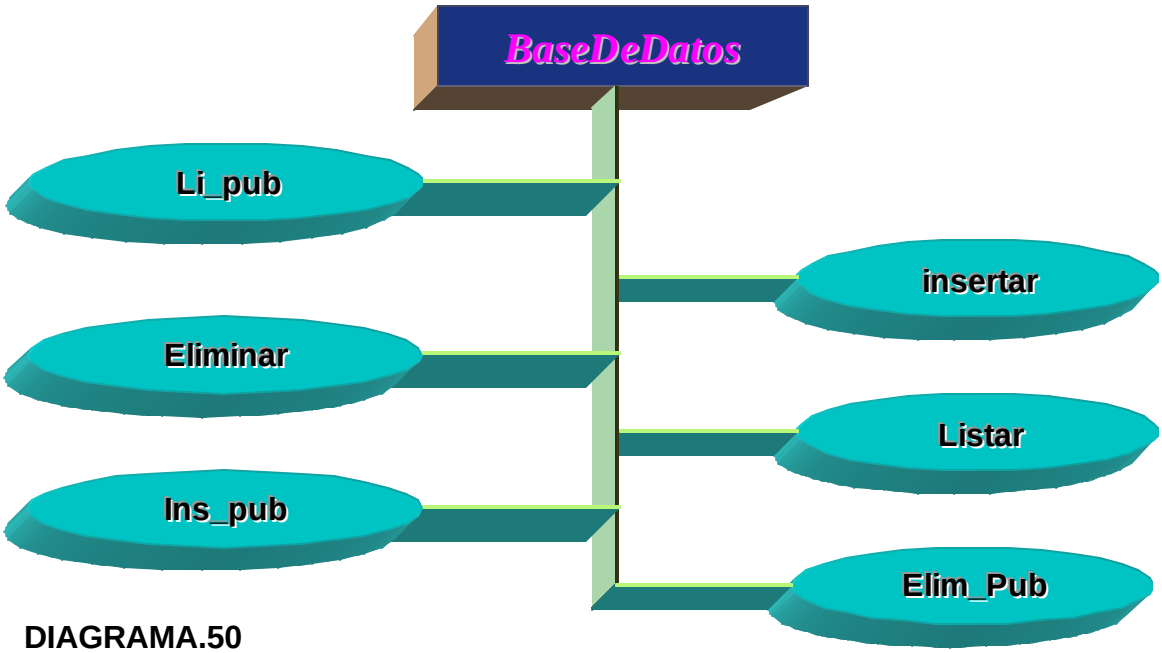


DIAGRAMA.50

Dentro de este objeto se encontraran métodos para manipular la Base de Datos tanto como para las claves privadas como para las claves publicas.

II.13.1 VARIABLES

Ninguna.

II.13.2 METODOS

II.13.2.1 insertar

Descripción **:** Recibe un arreglo de tipo String dentro del cual se encuentran los datos con los que se actualizara la base de datos, carga los controladores de la Base de Datos y establece la conexión con ella.

Definición : **public void insertar(String Ins[])**

Parámetros De Entrada : Ins **Contendrá los datos entrados por el usuario.**

Valor Retornado **:** Ninguno.

Métodos Invocados : **Ninguno.**

Descripción de la lógica **:** Recibe un objeto de tipo String que contiene los datos que se van a insertar en la base de datos. Se obtiene los datos recibidos con variables temporales. Se asignan a unas variables los manejadores de la base de datos (el url, el administrador y su password). Se carga el controlador de la base de datos de Sun; se establece una conexión con la base de datos con el método DriveManager.getconnection(). Des pues se declaran sentencias de SQL y se ejecutan con el

método prepareStatement(); inserta las cadenas y se cierra la conexión con la base de datos.

II.13.2.2 Eliminar

Descripción **:** Elimina el elemento correspondiente al elemento recibido que se encuentra en la Base de Datos.

Definición : **public void Eliminar(String Nombre)**

Parámetros De Entrada : Nombre **Contendrá los datos que el usuario quiere eliminar.**

Valor Retornado **:** Ninguno.

Métodos Invocados : **Ninguno.**

Descripción de la lógica : Recibe como parámetro la cadena que se desea eliminar; crea los objetos manejadores de la base de datos y establece la comunicación. Ejecuta sentencias del lenguaje SQL para borrar los elementos de la base de datos y cierra la conexión; si hay algún error envía una excepción.

II.13.2.3 Listar

Descripción **:** Devuelve un vector con todos los elementos encontrados en la Base de Datos.

Definición : **public Vector Listar()**

Parámetros De Entrada : **Ninguno.**

Valor Retornado **: Cadena**. Contiene los datos extraídos de la Base de Datos

Métodos Invocados : **Ninguno.**

Descripción de la lógica : Se crea un vector donde van a estar todos los elementos de la base de datos que se le van a mostrar al usuario. Se establecen los manejadores, la comunicación se hacen sentencias del lenguaje SQL y en un ciclo ‘while’ se van agregando las cadenas hasta que no encuentre el fin de la base de

datos o no haya por leer otro elemento. Se cierra la conexión y si ocurre un error se envía un mensaje. Al final se retorna la cadena del vector.

NOTA. Los métodos siguientes son análogos a insertar(), Eliminar() y Listar. Con la diferencia de que sólo actúan sobre la tabla de clave publica.

II.13.2.4 ins_pub() análogo a **insertar()**

II.13.2.5 Elim_Pub() análogo a **Eliminar()**

II.13.2.6 Li_Pub() **análogo a** listar ()

II.14 public class Archivos

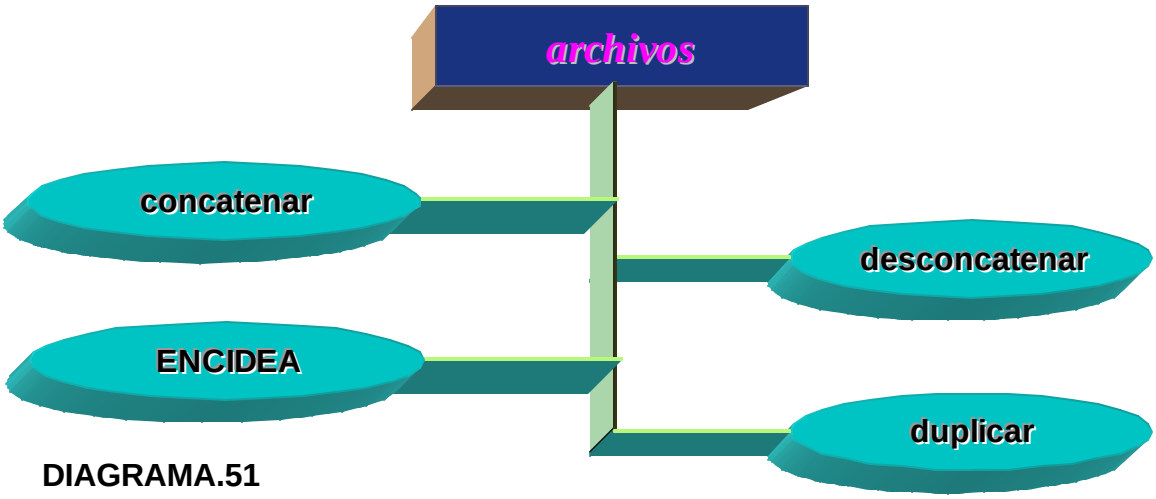


DIAGRAMA.51

Contiene los métodos para realizar modificaciones en archivos. Adicionalmente también contiene un método que monitorea el proceso de encriptar un archivo con el cifrador IDEA.

II.14.1 VARIABLES

Ninguna.

II.14.2 METODOS

II.14.2.1 concatenar

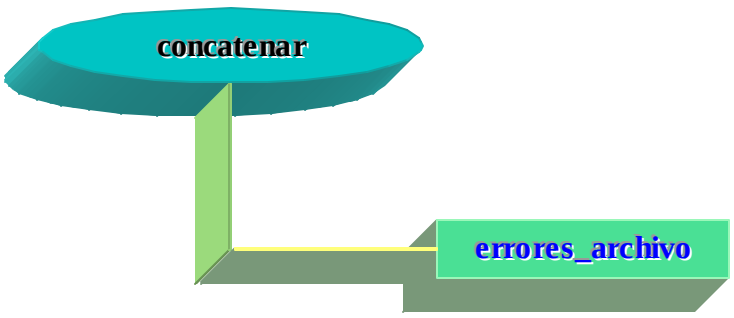


DIAGRAMA.52

Descripción **:** Concatena un archivo y un valor numérico en un nuevo archivo, la estructura del nuevo archivo es: número de bytes que componen al número, el número, el número de bytes que ocupaba el antiguo archivo y el archivo antiguo.

Definición : **public void concatenar(String, java.math.BigInteger, Frame)**

Parámetros De Entrada : **Nombre del archivo a concatenar, el valor a concatenar y el Frame que lo invoca.**

Valor Retornado **:** Ninguno.

Métodos Invocados : - **errores_archivo(padre,"Error de Archivo Invalido!","Tenga cuidado con los archivos")**

Descripción de la lógica **:** Recibe como parámetro el nombre del archivo, el valor numérico y el nombre del padre que lo llamo. Convierte el valor numérico a una cadena de bytes, se obtiene la longitud en bytes, se crea un objeto de tipo archivo y se obtiene la longitud del archivo original. Se abre el archivo de forma de lectura y se crea otro temporal que se pueda leer y escribir en él. Se escribe en el archivo temporal el número de

bytes que contiene el valor que se va a concatenar, el tamaño original del archivo, el valor numérico y el contenido del archivo. Se cierran los archivos y se renombra el archivo temporal y se elimina el que se actualizó. Si existe algún error se invoca el método errores_archivo() para informarle al usuario.

II.14.2.2 desconcatenar

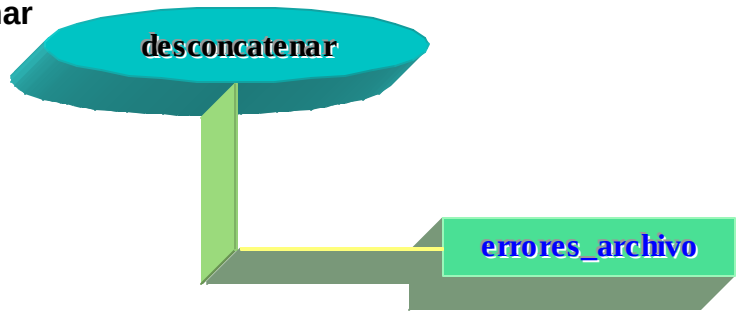


DIAGRAMA.53

Descripción	**:** Método utilizado para desconcatenar el archivo y el valor numérico, tiene en cuenta como es concatenado por el método concatenar, su funcionamiento es similar a concatenar.
Definición	: **public java.math.BigInteger desconcatenar(String, Frame)**
Parámetros De Entrada	: **Nombre del archivo a desconcatenar y el Frame que lo invoca.**
Valor Retornado	**:** directamente retorna el valor que desconcatenó e implícitamente crea un nuevo archivo sin el valor desconcatenado.
Métodos Invocados	: - **errores_archivo(padre,"Error de Archivo Invalido!","Tenga cuidado con los archivos")**

Descripción de la lógica : Funciona casi igual a concatenar; tiene en cuenta como fue escrito en concatenar para realizar su operación. Se lee el tamaño y si ha sido alterado se envía un mensaje de error. Lee los valores respectivos y escribe el contenido necesario en un nuevo archivo.

II.14.2.3 duplicar

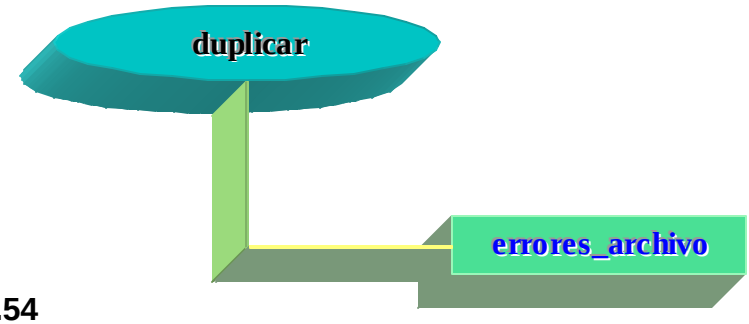


DIAGRAMA.54

Descripción **:** Utilizado para duplicar un archivo.

Definición : **public void duplicar(String archivo,String archivo,Frame padre).**

Parámetros De Entrada : **Nombre del archivo que se va a duplicar, Nombre del archivo en el que se duplicara, Frame que lo llama.**

Valor Retornado **:** Directamente ninguno, implícitamente crea el nuevo archivo.

Métodos Invocados :- **errores_archivo(Frame,String, String)**

Descripción de la lógica : Recibe como parámetro el archivo que se va a duplicar, el nombre del padre que lo llamo y el nombre del archivo en el cual se duplicará. Se verifica que si ya existe el archivo duplicado y se elimina. Se abren los archivos, uno en forma de lectura y el otro, en forma de lectura y escritura; en un ciclo ‘while’ se leen los datos del que se abrió en forma de lectura y se escribe en el otro archivo. Se cierran los archivos si el proceso ocurrió normalmente. Si hubo alguna excepción se envía un mensaje por el método errores_archivo().

II.14.2.4 ENCIDEA

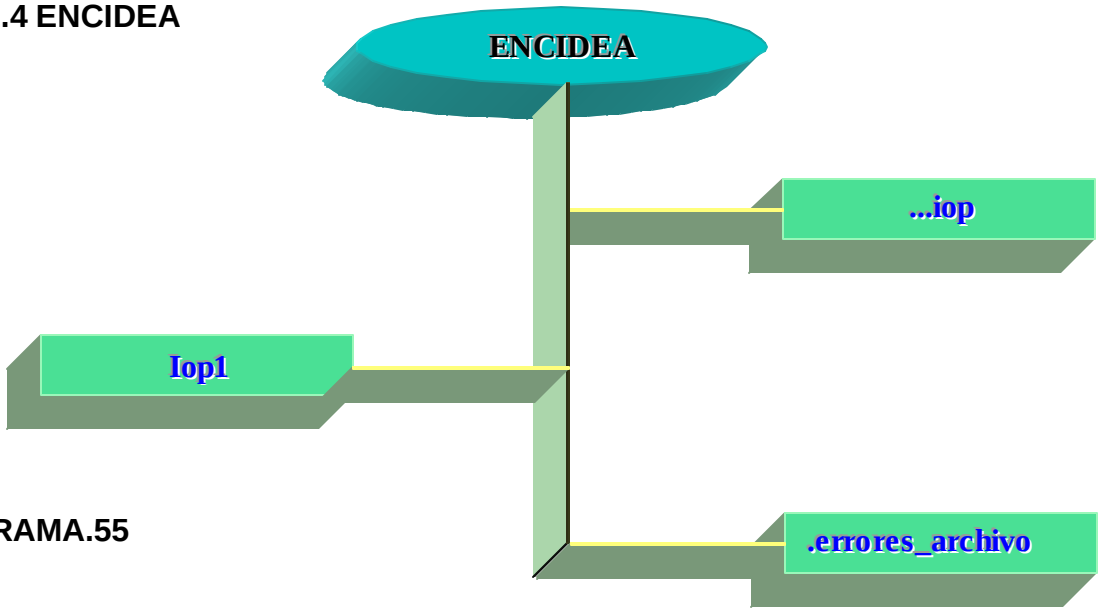


DIAGRAMA.55

Descripción **:** Encripta o Desencripta un archivo con la clave de sesión idea.

Definición : **public void ENCIDEA(java.math.BigInteger CS,String archivo,Frame padre,int fa)**

Parámetros De Entrada : **Clave de sesión, Nombre del archivo, Frame que lo llama, modo(1- Encripción, 2- desencripción).**

Valor Retornado **:** Directamente ninguno, implícitamente crea el nuevo archivo encriptado o desencriptado.

Métodos Invocados : - **errores_archivo(Frame,String, String)**

- **Idea.iop(String, java.math.BigInteger)**
- **Idea.iop1(String, java.math.BigInteger)**

Descripción de la lógica : Recibe como parámetro la clave de sesión, el archivo a encriptar o desencriptar, el nombre del padre que lo llamo y una bandera que le dice si va a encriptar o desencriptar. Crea un objeto del archivo recibido para abrirlo en forma de lectura; y se crea un archivo temporal para accesárlo de forma tanto de escritura como de lectura.

Dentro de un ciclo 'while' se lee del archivo los primeros 64 bits que se le enviarán al método que sea invocado; si es encripción se llama el método iop() y el resultado se le asigna a una variable llamada REC; si ocurre un error se cierran los archivos y se destruye el archivo temporal. Enviándole un mensaje al usuario. Si es desencripción se llama el método iop1() para que desencripte los bits enviados. Después en un cilco 'for' se escriben los datos en el archivo temporal y se inicializa nuevamente con la operación cerrando el ciclo 'while'.

Si que dan bits por encriptar se repite el proceso nuevamente con los bits que sobraron que no

completaron el bloque de 64 bits. Se escriben en el archivo temporal. Se renombra éste archivo, si ocurre un error se envía un mensaje llamando el método errores_archivo(). Se retorna el archivo encriptado o desencriptado.

II.15 public class RMD160H

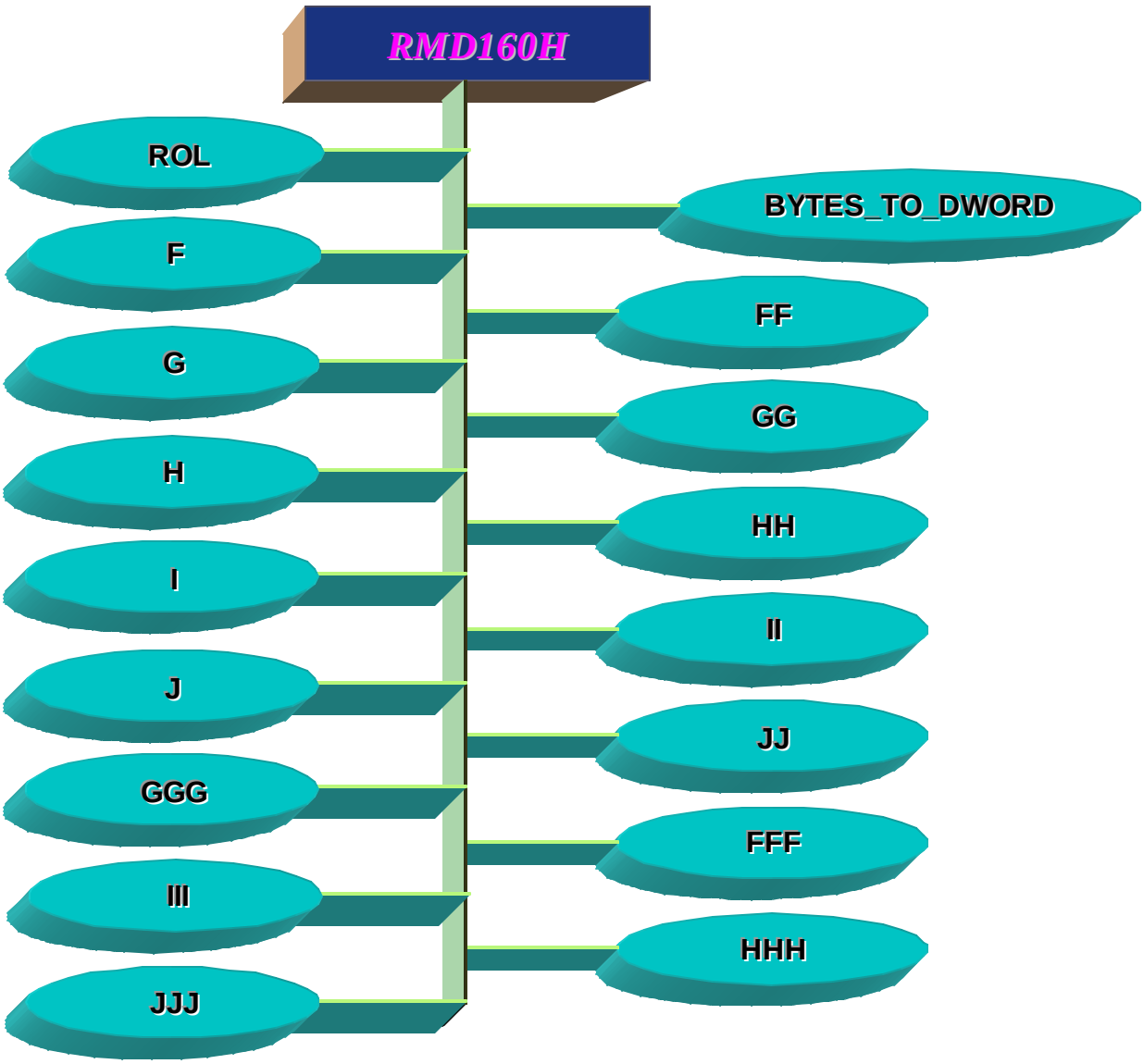


DIAGRAMA.56

Contiene Métodos que hacen parte de la clase RMD160, y son utilizados para la implementaron del algoritmo RIPEMD-160.

II.15.1 VARIABLES COMUNES A LOS MÉTODOS DE ESTA CLASE

- **Rem.** Utilizada para corregir el bit de signo y así poder trabajar con un numero especifico de bits sin signo.

II.15.2 MÉTODOS

II.15.2.1 BYTES_TO_DWORD

Descripción **:** De un arreglo de bytes, toma 4 bytes y los convierte en una palabra de 32 bits.

Definición **:** public long BYTES_TO_DWORD(byte[], int , int)

Parámetros De Entrada **:** Arreglo de bytes, la posición inicial del arreglo la dan la suma de los dos valores enteros recibidos.

Valor Retornado **:** Un numero de 32 bits sin signo.

Métodos Invocados **:** Ninguno.

Descripción de la lógica **:** Recibe un arreglo de 4 bytes, y el valor donde estos inician, que se obtiene de los dos enteros recibidos. Hace unos corrimientos a la izquierda a los bits y el resultado realiza un or con los otros bits de posiciones menores, para devolver el mismo arreglo de 4 bytes pero con los bits entrelazados.

II.15.2.2 ROL

Descripción **:** realiza una rotación de n bits de la palabra x a la izquierda.

Definición **:** public long ROL(long x, long n)

Parámetros De Entrada **:** palabra de que se va a rotar y el numero de bits que se rotará.

Valor Retornado **:** La palabra de 32 bits recibida rotada n posiciones.

Métodos Invocados **:** Ninguno.

Descripción de la lógica **:** Recibe como parámetro la palabra que se desea rotar y las posiciones que se va a rotar; se rota primero a la izquierda las posiciones y se hace un or

con la palabra rotada 32 – n bits a la derecha. Y retorne el resultado.

II.15.2.3 F

Descripción	**:** realiza la primera función básica de RIPEMD-160.
Definición	**:** public long F(long x, long y, long z)
Parámetros De Entrada	**:** Los valores iniciales de la función Hash.
Valor Retornado	**:** Una palabra de 16 bits con los cambios realizados en la función.
Métodos Invocados	**:** Ninguno.
Descripción de la lógica	**:** recibe tres palabras de 32 bits y realiza un or-exclusivo entre las tres palabras para retornar ese resultado. Estas son operaciones ya definidas por RIPEMD-160.

De igual manera se describen los siguientes cuatro métodos, con la diferencia en las operaciones realizadas dentro de ellas (ver descripción de RIPEMD-160).

public long G(long x, long y, long z)

public long H(long x, long y, long z)

public long I(long x, long y, long z)

public long J(long x, long y, long z)

II.15.2.4 FF

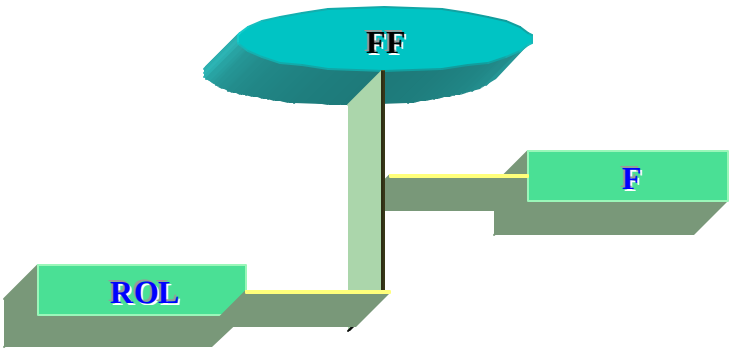


DIAGRAMA.57

Descripción **:**Realiza la suma del valor de la constante con los debidos desplazamientos según sea la iteración.

Definición **:** public void FF(Long MDaux[], int a, int b, int c, int d, int e, long x, long s)

Parámetros De Entrada **:** Los bloques de 16 bits, las posiciones y, los valores de las constantes.

Valor Retornado **:** Directamente no retorna ningún valor pero realiza modificaciones a los bits de palabras escogidos del mensaje.

Métodos Invocados **:-** F(MDaux[b].longValue(),MDaux[c].longValue(), MDaux[d].longValue())

- ROL(MDaux[a].longValue(), s)

Descripción de la lógica **:** Recibe como parámetro los bloques de 16 bits, las posiciones que se van a accesar para los bloques y las constantes que se van a sumar. Invoca al método F() para realizar operaciones básicas requeridas por RIPEMD-160, e invoca el método

ROL() para que realice las rotaciones en las palabras que lo requieran (según sea la iteración).

NOTA. De igual manera para los otros métodos, con la diferencia de llamar al método correspondiente a su iteración, realizar los desplazamientos y, el valor de la constante.

public void GG(Long MDaux[],int a, int b, int c, int d, int e, Long x, long s)

public void HH(Long MDaux[],int a, int b, int c, int d, int e, Long x, long s)

public void II(Long MDaux[],int a, int b, int c, int d, int e, Long x, long s)

public void JJ(Long MDaux[],int a, int b, int c, int d, int e, Long x, long s)

II.15.2.5 FFF

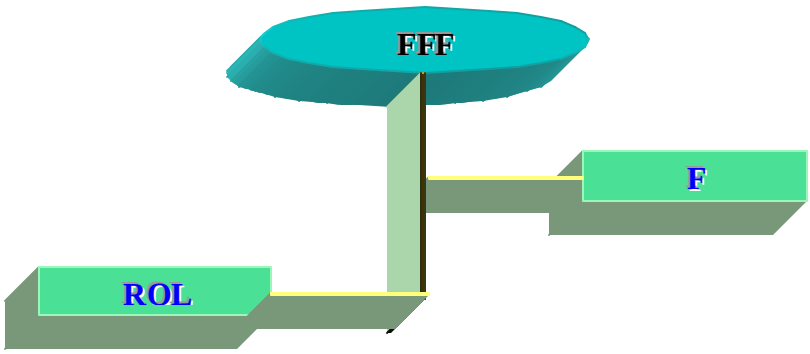


DIAGRAMA.58

Descripción **:**Realiza la suma del valor de la constante con los debidos desplazamientos según sea la iteración.

Definición **:** public void FFF(Long MDaux[], int a, int b, int c, int d, int e, long x, long s)

Parámetros De Entrada **:** Los bloques de 16 bits, las posiciones y, los valores de las constantes.

Valor Retornado **:** Directamente no retorna ningún valor pero realiza modificaciones a los bits de palabras escogidos del mensaje.

Métodos Invocados **: -** F(MDaux[b] .longValue(), MDaux[c].longValue(), MDaux[d].longValue())

- ROL(MDaux[a].longValue(), s)
-

Descripción de la lógica : Recibe como parámetro los bloques de 16 bits las posiciones que se van a accesar y las constantes que serán sumadas. Invoca F() para que realice las operaciones básicas pero ya en las iteraciones necesaria, e invoca a rol para que desplace los bits requeridos.

NOTA. De igual manera para los otros métodos, con la diferencia de llamar al método correspondiente a su iteración, realizar los desplazamientos y, el valor de la constante.

public void GGG(Long MDaux[],int a, int b, int c, int d, int e, Long x, long s)

public void HHH(Long MDaux[],int a, int b, int c, int d, int e, Long x, long s)

public void III(Long MDaux[],int a, int b, int c, int d, int e, Long x, long s)

public void JJJ(Long MDaux[],int a, int b, int c, int d, int e, Long x, long s)

II.16 public class excepciones extends Exception

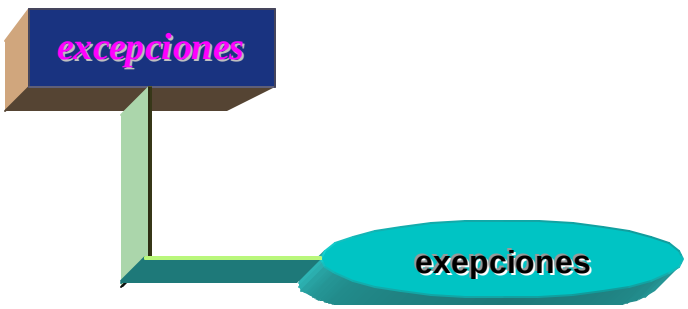


DIAGRAMA.59

Contiene excepciones propias que podemos lanzar en caso de ser necesario.

II.16.1 VARIABLES

Ninguna.

II.16.2 MÉTODOS

II.16.2.1 excepciones

Descripción **:** Especifica esta excepción.

Definición **:** public excepciones(String)

Parámetros de entrada **:** La cadena como que enviara la excepción al ser lanzada.

Valor Retornado **:** Directamente ninguno, implícitamente envía la excepción.

Métodos Invocados **:**Ninguno.

Descripción de la lógica **:** Recibe como parámetro la cadena que contiene el error; invoca la instancia super() para mostrar el problema originado por pantalla.

II.17 public class BotonGrafico extends Canvas implements ImageObserver

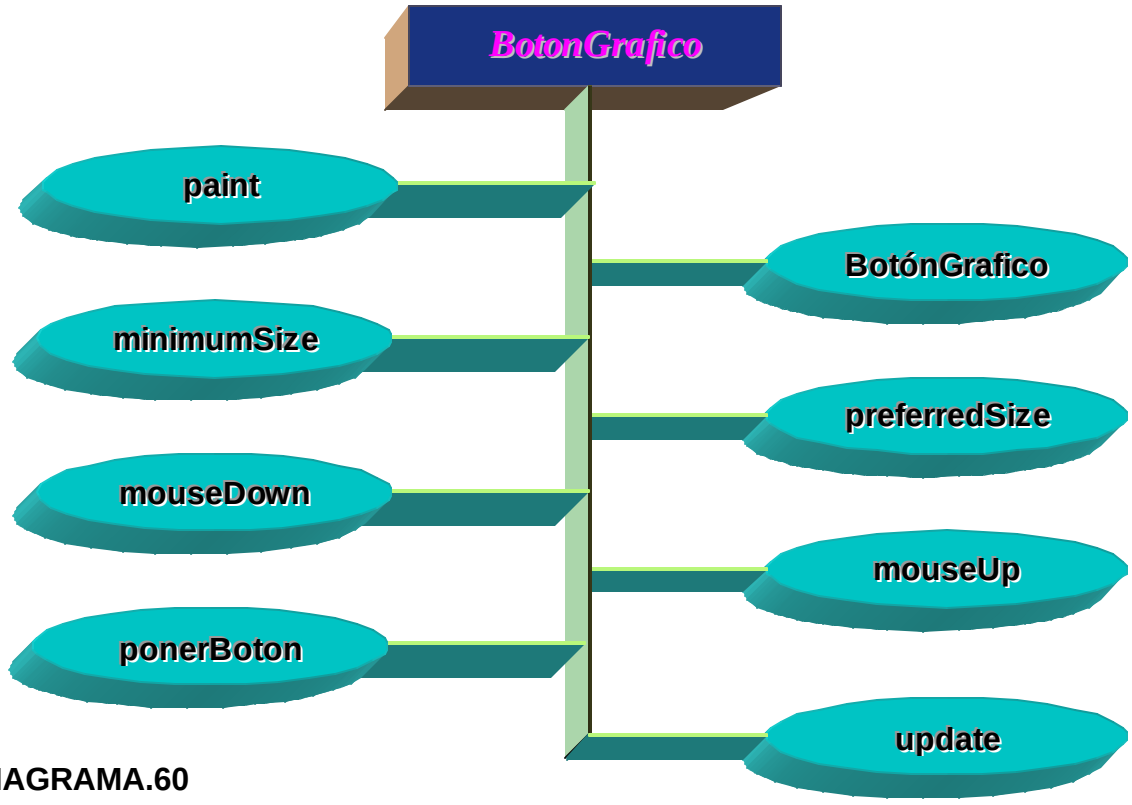


DIAGRAMA.60

Crea un botón gráfico recibe 2 imágenes que serán a su vez arriba y abajo para dar el efecto del botón.

II.17.1 VARIABLES

- **img_arriba.** Imagen arriba
- **img_abajo.** Imagen abajo
- **ABAJO.** termino booleano para saber en que posición esta actualmente el botón
- **ARRIBA.** termino booleano para saber en que posición esta actualmente el botón.

II.17.2 MÉTODOS

I.17.2.1 BotonGrafico

Descripción **:** Actualiza las variables conocida a los métodos locales con la información recibida.

Definición **:** public BotonGrafico(String n, Image up, Image down)

Parámetros de entrada **:** Nombre de la ventana, Imagen del botón cuando esta arriba y la imagen del botón cuando este abajo.

Valor Retornado **:** Ninguno

Métodos Invocados **:** Ninguno

Descripción de la lógica **:** Recibe como parámetro el nombre de la ventana y, las imágenes, estos parámetros son asignados a nuevas variables.

II.17.2.2 minimumSize

Descripción	**:** Actualiza las variables propias del botón actualizando el tamaño que ocupa.
Definición	**:** public Dimensión minimumSize()
Parámetros de entrada	**:** Nombre de la ventana, Imagen del botón cuando esta arriba y la imagen del botón cuando este abajo.
Valor Retornado	**:** Actualiza las variables propias de la dimensión del botón.
Métodos Invocados	**:** Ninguno.
Descripción de la lógica	**:** Retorna el tamaño original de la imagen. Obtiene el valor de altura y ancho, de la imagen que esta en la variable img_arriba.

II.17.2.3 preferredSize()

Descripción **:** Actualiza las variables propias del botón actualizando el
tamaño mínimo que ocupa.

Definición **:** public Dimensión preferredSize()

Parámetros de entrada **:** Ninguno.

Valor Retornado **:** Actualiza las variables propias de la dimensión del botón.

Métodos Invocados **:** Ninguno.

Descripción de la lógica **:** invoca la propiedad minimumSize() y retorna el valor obtenido.

II.17.2.4 mouseDown

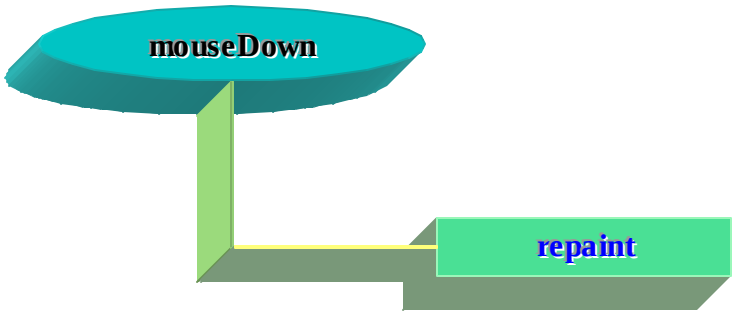


DIAGRAMA.61

Descripción **:** Si ocurre un evento y este ocurre sobre el, ejecuta una acción que en este caso es el cambio la imagen.

Definición **:** public boolean mouseDown(Event evt, int x, int y)

Parámetros de entrada : Si ocurre en evento mouse Down que esta en las coordenadas x,y.

Valor Retornado : true, evento procesado

Métodos Invocados : repaint().

Descripción de la lógica : Recibe como parámetro el evento que se detecto y las coordenadas donde ocurrió el evento. A la variable esta_arriba se le asigna el estado botón (ABAJO), se pinta este elemento en pantalla invocando al método repaint() y se retorna verdadero.

II.17.2.5 mouseUp

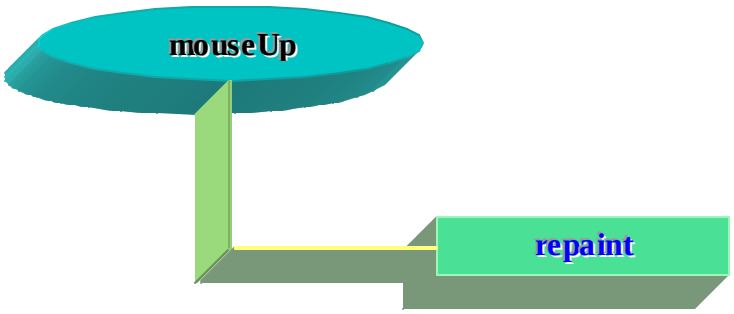


DIAGRAMA.62

Descripción : Si cuando sueltan el botón aun están sobre el, envía uno diciendo que ocurrió sobre el , si no simplemente actualiza la imagen.

Definición **:** public boolean mouseUp(Event evt, int x, int y)

Parámetros de entrada **:** Si ocurre en evento mouse Up que esta en las coordenadas x,y.

Valor Retornado **:** true, evento procesado

Métodos Invocados **:** repaint().

Descripción de la lógica **:** Recibe como parámetro el evento ocurrido y las coordenada donde este ocurrió. Se pregunta que si cuando se suelta el botón del mouse aún están dentro de la imagen se confirma el evento. Sino simplemente se actualiza la imagen asignándole a la variable esta_arriba el estado imagen ARRIBA, e invocando el método repaint() dibujarla y, se retorna verdadero.

II.17.2.6 ponerBoton

Descripción **:** Actualiza la variable que guarda el estado del botón, arriba o abajo.

Definición **:** public void ponerBoton()

Parámetros de entrada **:** El objeto gráfico sobre la que pintara.

Valor Retornado **:** Ninguno

Métodos Invocados **:** Ninguno.

Descripción de la lógica : Recibe como parámetro el estado del botón, actualiza la variable esta_arriba con el estado y dibuja el botón con el método repaint().

II.17.2.7 update

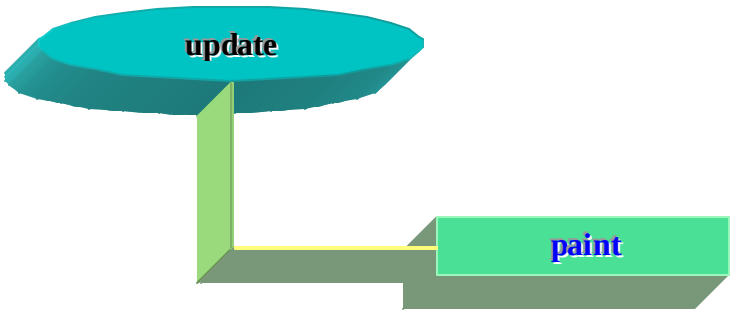


DIAGRAMA.63

Descripción **:** Actualiza la imagen pintando la nueva.

Definición **:** public void update(g)

Parámetros de entrada **:** El objeto gráfico sobre la que pintara.

Valor Retornado **:** Ninguno

Métodos Invocados **:** paint(graphics)

Descripción de la lógica **:** Recibe como parámetro la imagen que va a pintar e invoca al método paint() para pintar la imagen.

II.17.2.8 paint

Descripción **:** Pinta por primer vez sobre el objeto Gráfico.

Definición **:** public void paint(Graphics g)

Parámetros de entrada **:** El objeto gráfico sobre el que pintara.

Valor Retornado **:** Ninguno

Métodos Invocados **:** Ninguno.

Descripción de la lógica **:** Recibe como parámetro la imagen que va a dibujar, pregunta por el estado que tiene la variable esta_arriba y dependiendo de este dibuja la imagen deseada.

II.18 public class BotonGrafico1 extends Canvas implements ImageObserver

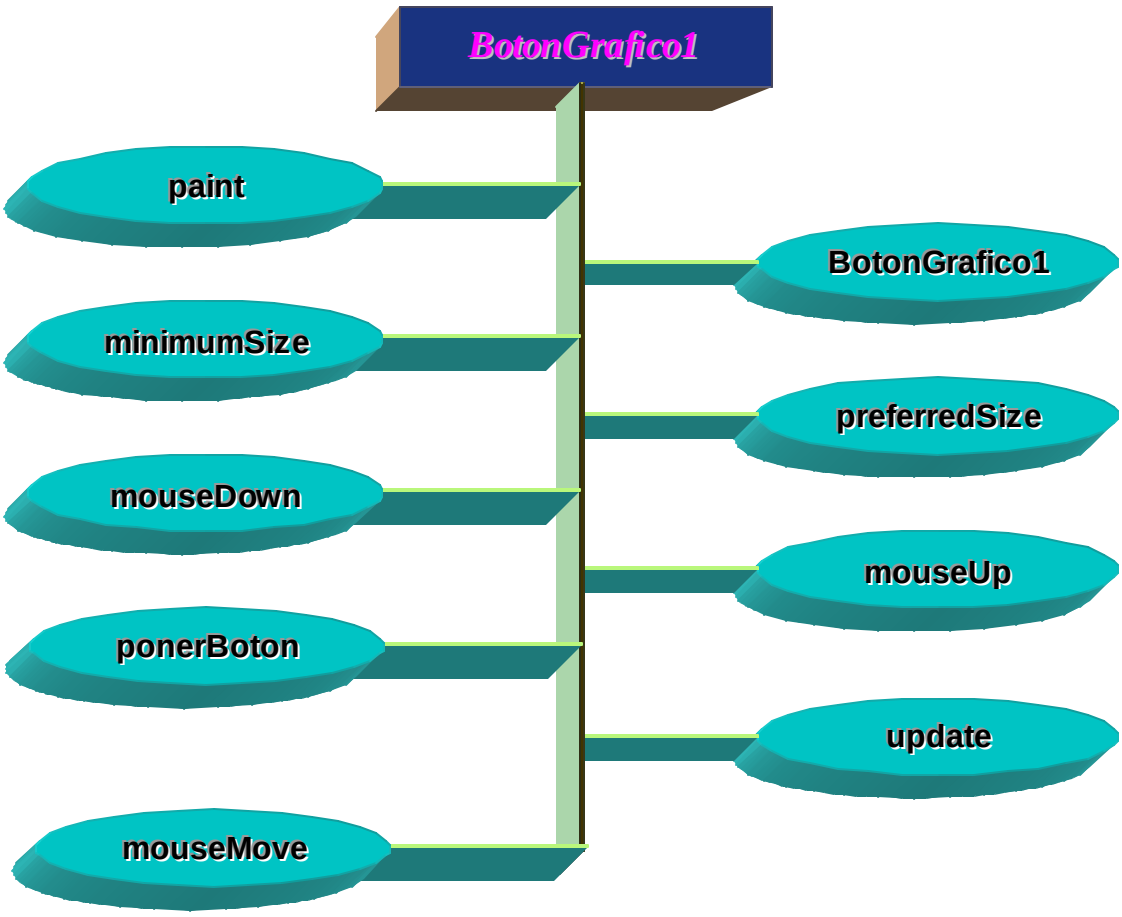


DIAGRAMA.64

Crea un botón gráfico recibe 3 imágenes que serán a su vez arriba, intermedia y abajo para dar efecto al botón.

II.18.1 VARIABLES

- **img_arriba.** Imagen arriba
- **img_inter.** imagen intermedia
- **img_abajo.** Imagen abajo
- **ABAJO.** valor numérico de 3 para saber en que posición esta actualmente el botón
- **INTERMEDIO.** valor numérico de 2 para saber en que posición esta actualmente el botón
- **ARRIBA.** valor numérico de 1 para saber en que posición esta actualmente el botón
- **esta_arriba.** inicializada en 1
- **padre.** para saber el método que lo invoco

II.18.2 MÉTODOS

I.18.2.1 BotonGrafico1

Descripción **:** Actualiza las variables conocida a los métodos locales con la información recibida.

Definición **:** public BotonGrafico1(String n, Image up, Image Med, Image down,Frame padre1)

Parámetros de entrada **:** Nombre de la ventana, Imagen del botón cuando esta arriba, imagen del botón cuando esta intermedio, imagen del botón cuando este abajo y el nombre del padre que lo invoca.

Valor Retornado **:** Ninguno

Métodos Invocados **:** Ninguno

Descripción de la lógica **:** Recibe como parámetro el nombre de la ventana y, las imágenes, estos parámetros son asignados a nuevas variables.

II.18.2.2 minimumSize

Descripción **:** Actualiza las variables propias del botón actualizando el tamaño que ocupa.

Definición **:** public Dimensión minimumSize()

Parámetros de entrada **:** Ninguno.

Valor Retornado **:** Actualiza las variables propias de la dimensión del botón.

Métodos Invocados **:** Ninguno.

Descripción de la lógica : Retorna el tamaño original de la imagen. Obtiene el valor de altura y ancho, de la imagen que esta en la variable img_arriba.

II.18.2.3 preferredSize()

Descripción **:** Actualiza las variables propias del botón actualizando el
tamaño mínimo que ocupa.

Definición **:** public Dimensión preferredSize()

Parámetros de entrada **:** Ninguno.

Valor Retornado **:** Actualiza las variables propias de la dimensión del botón.

Métodos Invocados **:** Ninguno.

Descripción de la lógica : invoca la propiedad minimumSize() y retorna el valor obtenido.

II.18.2.4 mouseDown

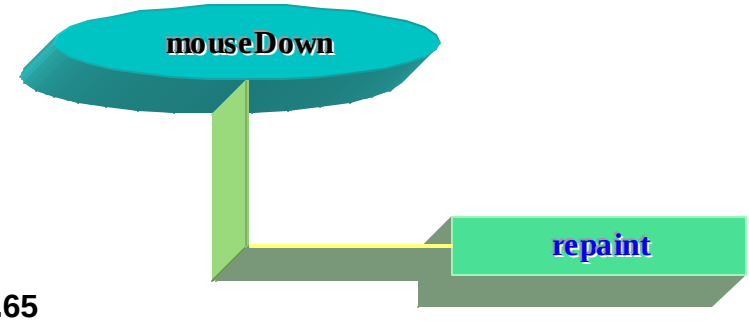


DIAGRAMA.65

Descripción	**:** Si ocurre un evento y este ocurre sobre el botón, ejecuta una acción que en este caso es el cambio la imagen.
Definición	**:** public boolean mouseDown(Event evt, int x, int y)
Parámetros de entrada	**:** Si ocurre en evento mouse Down que esta en las coordenadas x,y.
Valor Retornado	**:** true, evento procesado
Métodos Invocados	**:** repaint().
Descripción de la lógica	**:** Recibe como parámetro el evento que se detecto y las coordenadas donde ocurrió el evento. A la variable esta_arriba se le asigna el estado BotónGrafico1.ABAJO, se pinta este elemento en pantalla invocando al método repaint() y se retorna verdadero.

II.18.2.5 mouseMove

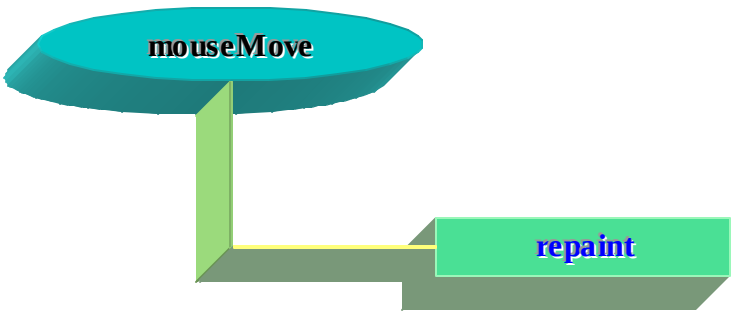


DIAGRAMA.66

Descripción	**:** Si ocurre un evento y este ocurre sobre el botón, ejecuta una acción que en este caso es el cambio de la imagen.
Definición	**:** public boolean mouseMove(Event evt, int x, int y)
Parámetros de entrada	**:** Si ocurre en evento mouse Move que esta en las coordenadas x,y.
Valor Retornado	**:** true, evento procesado
Métodos Invocados	**:** repaint().
Descripción de la lógica	**:** Recibe como parámetro el evento que se detecto y las coordenadas donde ocurrió el evento. A la variable esta_arriba se le asigna el estado BotónGrafico1.INTERMEDIO, se pinta este elemento en pantalla invocando al método repaint() y se retorna verdadero.

II.18.2.6 mouseUp

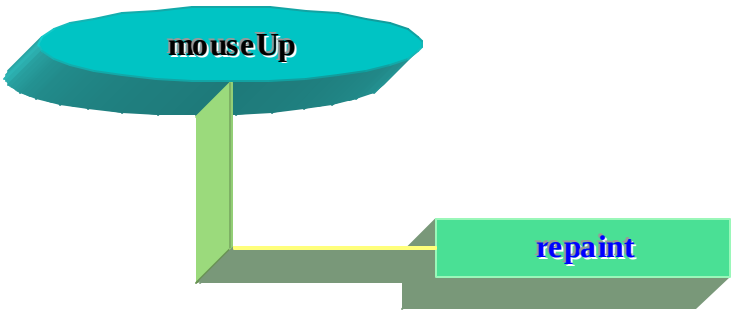


DIAGRAMA.67

Descripción **:** Si cuando sueltan el botón aún están sobre él, envía uno diciendo que ocurrió sobre el , si no simplemente actualiza la imagen.

Definición **:** public boolean mouseUp(Event evt, int x, int y)

Parámetros de entrada **:** Si ocurre en evento mouse Up que esta en las coordenadas x,y.

Valor Retornado **:** true, evento procesado

Métodos Invocados **:** repaint().

Descripción de la lógica **:** Recibe como parámetro el evento ocurrido y las coordenada donde este ocurrió. Se pregunta que si cuando se suelta el botón del mouse aún están dentro de la imagen se confirma el evento. Sino simplemente se actualiza la imagen asignándole a la variable esta_arriba el estado imagen ARRIBA, e invocando el método repaint() dibujarla y, se retorna verdadero.

II.18.2.7 ponerBoton

Descripción	**:** Actualiza la variable que guarda el estado del botón, arriba o abajo.
Definición	**:** public void ponerBoton()
Parámetros de entrada	**:** El objeto gráfico sobre la que pintara.
Valor Retornado	**:** Ninguno
Métodos Invocados	**:** Ninguno.
Descripción de la lógica	**:** Recibe como parámetro el estado del botón, actualiza la variable esta_arriba con el estado y dibuja el botón con el método repaint().

II.18.2.8 update

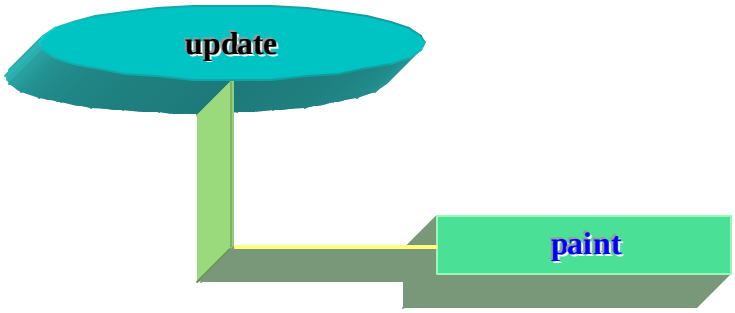


DIAGRAMA.68

Descripción : Actualiza la imagen pintando la nueva.

Definición : public void update(g)

Parámetros de entrada : El objeto gráfico sobre la que pintara.

Valor Retornado : Ninguno

Métodos Invocados : paint(graphics)

Descripción de la lógica : Recibe como parámetro la imagen que va a pintar e invoca al método paint() para pintar la imagen.

II.18.2.9 paint

Descripción **:** Pinta por primer vez sobre el objeto Gráfico.

Definición **:** public void paint(Graphics g)

Parámetros de entrada **:** El objeto gráfico sobre el que pintara.

Valor Retornado **:** Ninguno

Métodos Invocados **:** Ninguno.

Descripción de la lógica **:** Recibe como parámetro la imagen que va a dibujar, pregunta por el estado que tiene la variable esta_arriba y dependiendo de este dibuja la imagen deseada.

II.19 class RMD160 extends RMD160H

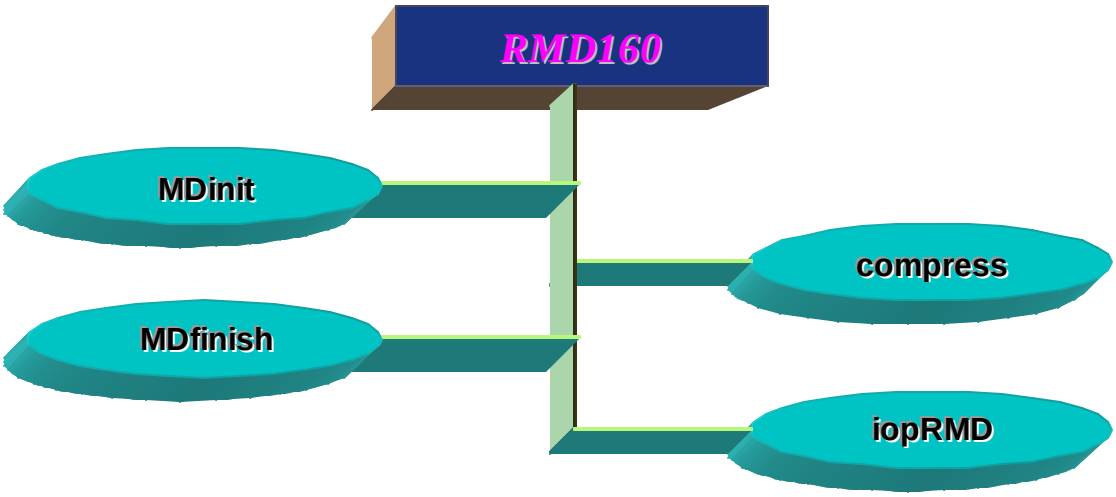


DIAGRAMA.69

Es una extensión de RMD160H, de donde llama sus métodos para poder hallar la función de un archivo.

II.19.1 VARIABLES

Ninguna.

II.19.2 MÉTODOS

II.19.2.1 MDinit

Descripción **:** Inicializa un arreglo con las cuatro constantes iniciales (constantes mágicas).

Definición **:** public void MDinit(Long [])

Parámetros de entrada **:** Arreglo al que modificara.

Valor Retornado **:** Directamente ninguno, implícitamente: el arreglo recibido, modificado.

Métodos Invocados **:** Ninguno.

Descripción de la lógica **:** Inicializa la variables de los registros que se necesitan para operar sobre los bits del mensaje.

II.19.2.2 compress

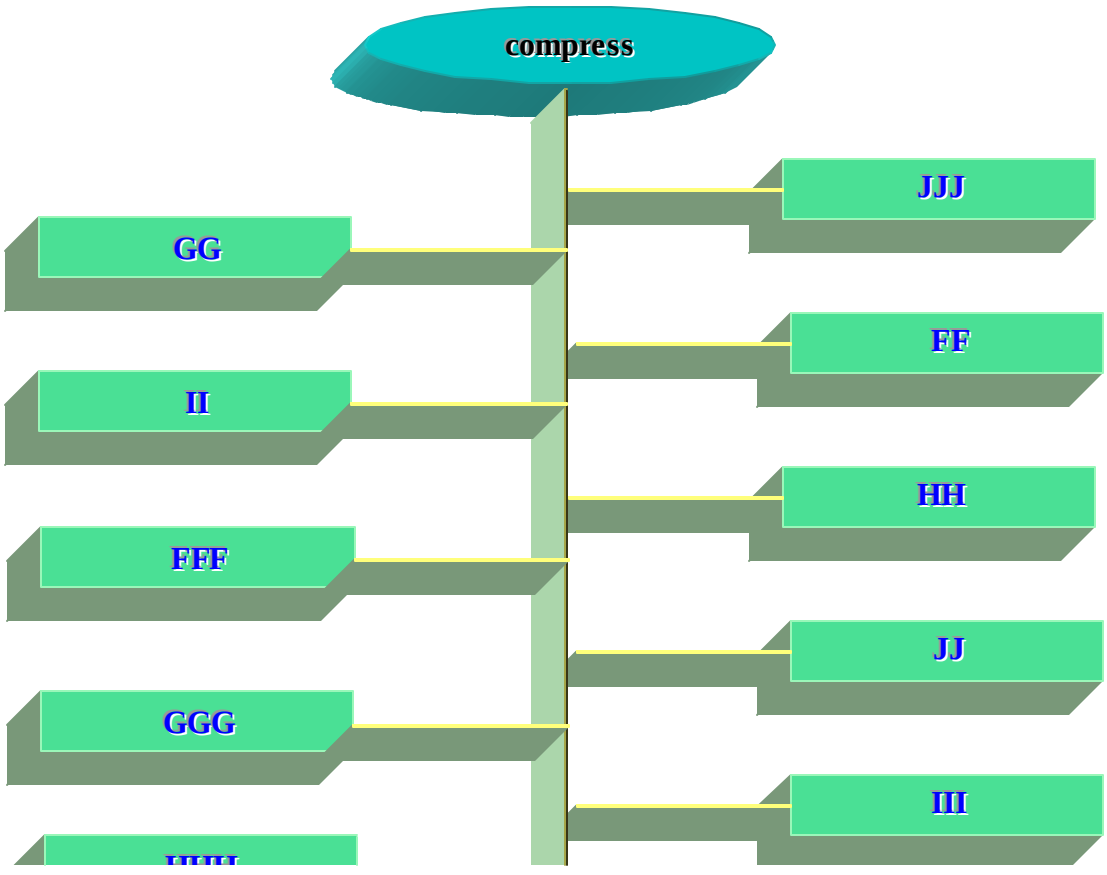


DIAGRAMA.70

Descripción **:** Monitorea las iteraciones que se deban realizar.

Definición **:** public void compress(Long [], Long [])

Parámetros de entrada **:** El valor después del padding que son 10 bloques de 32 bits, y las variables iniciales(constantes mágicas)

Valor Retornado **:** Implícitamente ninguno, pero modifica los arreglos recibidos.

Métodos Invocados:

- FF(Long [] , int ,int , int , int , int ,Long , long)
- GG (Long [] , int ,int , int , int , int ,Long , long)
- HH (Long [] , int ,int , int , int , int ,Long , long)
- HH (Long [] , int ,int , int , int , int ,Long , long)

- II (Long [] , int ,int , int , int , int ,Long , long)
- JJ (Long [] , int ,int , int , int , int ,Long , long)
- JJJ (Long [] , int ,int , int , int , int ,Long , long)
- III (Long [] , int ,int , int , int , int ,Long , long)
- HHH (Long [] , int ,int , int , int , int ,Long , long)
- GGG (Long [] , int ,int , int , int , int ,Long , long)
- FFF (Long [] , int ,int , int , int , int ,Long , long)

Descripción de la lógica : En un ciclo ‘for’ se inicializa los arreglos que ayudaran a contener la información final del digesto requerido. Se realizan las operaciones de la primera iteración, invocando el método FF(), con los parámetros requeridos (la información del mensaje,

las posiciones que se van a rotar y los desplazamientos que se realizaran). Esto es para cada iteración. Después de realizar las iteraciones, se combinan los resultados para los arreglos que contendrán la información final.

II.19.2.3 MDfinish

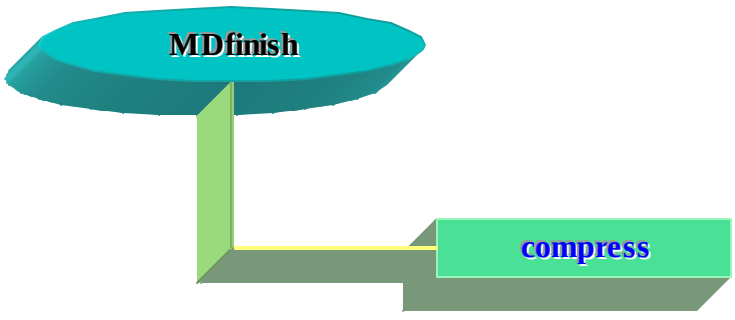


DIAGRAMA.71

Descripción **:** Obtendrá la función Hash final, recibiendo el ultimo bloque al que le debe realizar las diferentes iteraciones, si el bloque esta descompleto realiza el padding.

Definición **:** public void MDfinish(Long MDbuf[], byte strptr[],int offset, int lswlen, long mswlen)

Parámetros de entrada **:** Recibe un arreglo que contiene la función Hash parcial hallada hasta el momento. Además recibe un arreglo de bytes una variable que le indique cual bloque de 64 bytes quedo sin transformar, otra que le indique a partir de que posición de ese bloque debe operar, otra indica el numero de bloques transformado.

Valor Retornado **:** Directamente ninguno, implícitamente en la variable recibida devuelve la función Hash.

Métodos Invocados **:** compress(Long [], Long [])

Descripción de la lógica **:** En un ciclo ‘for’ se crean palabras de 32 bits; se realizan operaciones de or_exclusivo para las palabras creadas. Si el vector de bytes no esta lleno se completa al final con ceros. Se guardan los

últimos valores para completar el relleno y se hacen las ultima iteraciones invocando el método compress().

II.19.2.4 iopRMD

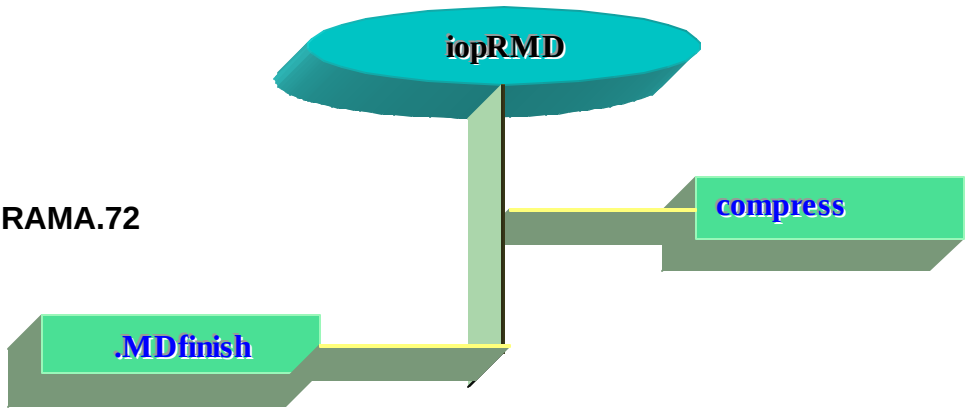


DIAGRAMA.72

Descripción : Monitorea que se halle completamente la función Hash a un archivo.

Definición : public java.math.BigInteger iopRMD(String Archivo)

Parámetros de entrada : El nombre del archivo a hallar el resumen.

Valor Retornado : Un vector que de 5 posiciones cada uno de 32 bits = 160 bits equivalente a la función Hash.

Métodos Invocados : - compress(Long [], Long [])

- MDfinish (Long Mdbuf [], byte strptr[], int offset, int lswlen , long mswlen)

Descripción de la lógica : Se abre el archivo que se le va a obtener la función Hash, se inicializa la variable que contendrá la función Hash con las constantes mágicas invocando el método MDinit();

II.20 class GENERA_LUCAS extends Frame

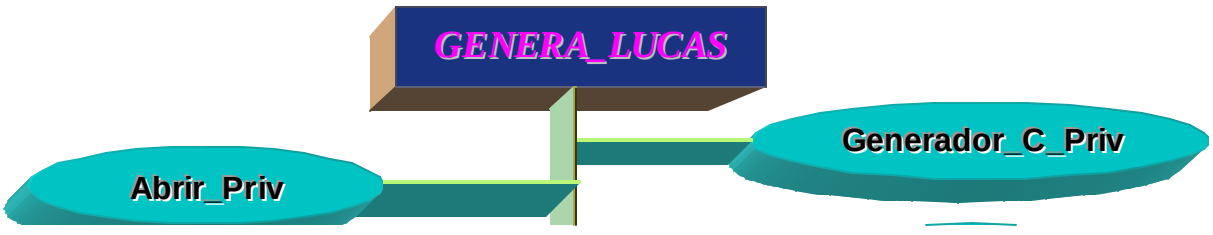


DIAGRAMA.73

Clase que se encarga de generar las claves publicas y privadas utilizando sus propios métodos los cuales se encuentran en otros objetos.

Define las siguientes variables que serán conocidas por los métodos de esta clase:

II.20.1 Variables:

- **t1** Es de tipo área de texto e imprime en pantalla la información del archivo de clave publica, mostrando sólo caracteres ASCII.
- **t2** Utilizada para imprimir la clave privada en pantalla.
- **t3** Muestra la clave IDEA con la cual fue encriptada la clave privada el usuario debe guardar esta clave para su respectiva identificación.
- **t4** Contendrá la clave Publica para que ésta sea impresa.
- **dir1,dir2 ,arch1, arch2** Contendrán los nombres de archivos y directorios donde están guardadas las claves publicas y privadas LUCAS.
- **original** Guardará la información del archivo que se va a encriptar.
- **band** Verifica que un área de texto contenga la dirección o el nombre de algún archivo.
- **CPR** Contendrá la clave privada como un valor numérico
- **números** Vector que contiene
- **bp, bp1** Botones que utilizan las imágenes arriba, abajo
- **Icono, Imagarriba, Imagabajo** Imágenes utilizadas por la interfaz gráfica.

I.20.2 Métodos

II.20.2.1 METODO GENERA_LUCAS

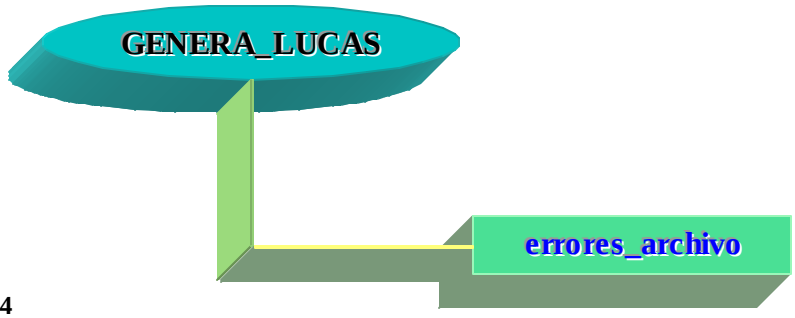


DIAGRAMA.74

Descripción : Se encarga del diseño de la interfaz gráfica de la ventana inicial.

Definición : GENERA_LUCAS(String nombre)

Parámetros de entrada : nombre – Su valor será el titulo de la ventana Inicial

Valor retornado : ninguno.

Invoca a : errores_archivo(this," ¡ Error 537 !","¡ Falta Botón Gráfico!");

II.20.2.2 main(String argv[ ])

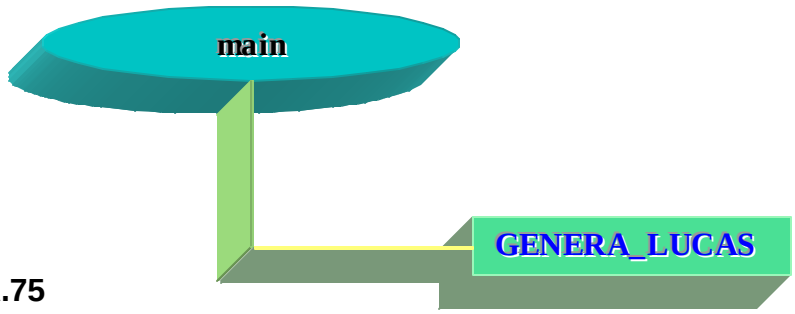


DIAGRAMA.75

Descripción : Se encarga de crear un objeto de GENERA_LUCAS y mostrarlo en pantalla.

Definición : public static void main(String argv[ ])

Parámetros de entrada : argv[ ] – en esta caso será null

Valor retornado : ninguno.

Invoca a : GENERA_LUCAS(String)

II.20.2.3 handleEvent(Event evt)

Descripción **:** Cuando ocurre un evento revisa si éste es el de cerrar la ventana, si es así cierra la ventana de generación de claves.

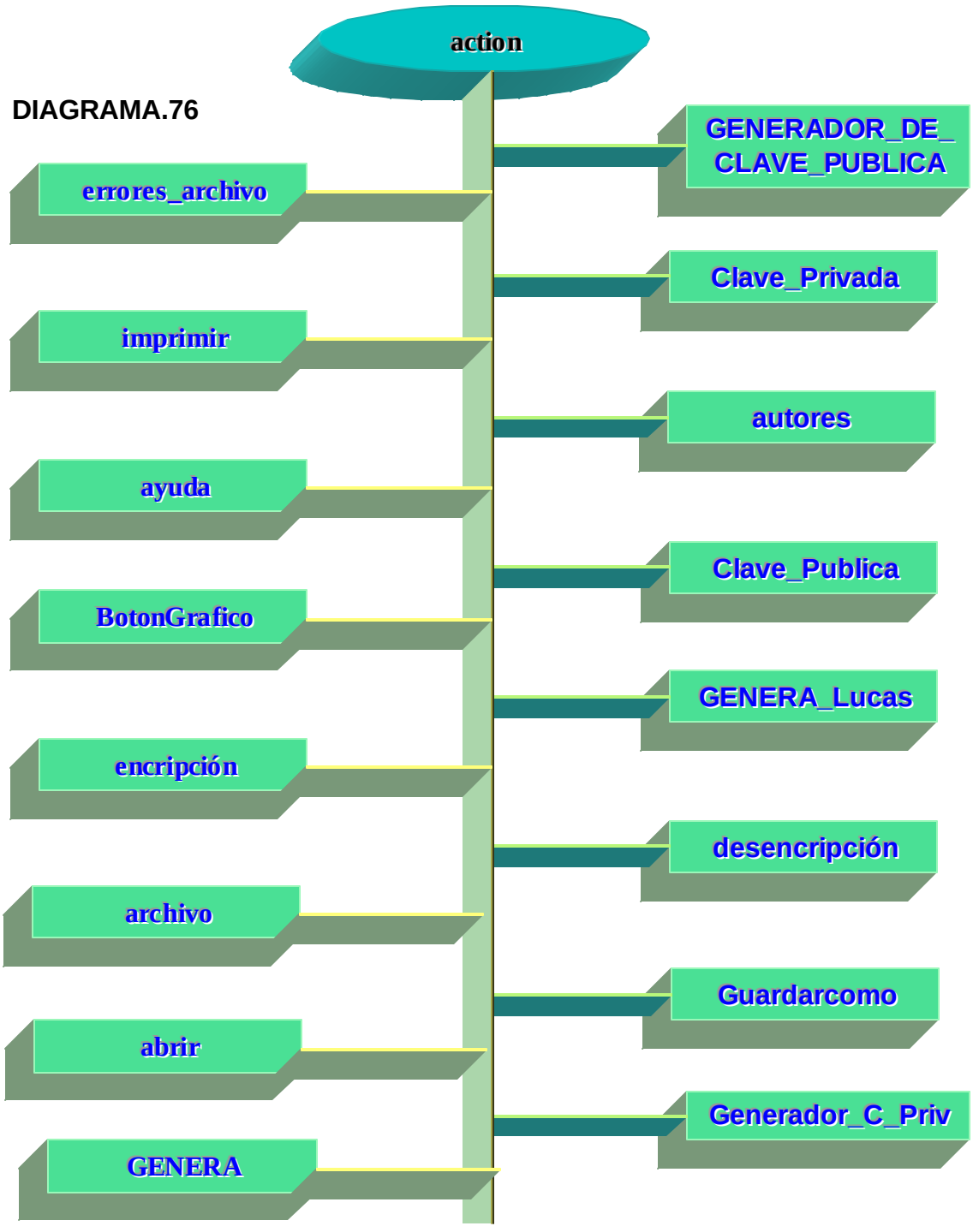
Definición **:** public boolean handleEvent(Event evt)

Parámetros de entrada **:** Evt nombre – evento realizado.

Valor retornado **:** Retorna el evento si no fue procesado.

II.20.2.4 action (Event evt, Object arg)

DIAGRAMA.76



Descripción **:** Monitorea las acciones realizadas sobre cada objeto mostrado en la pantalla.

Definición **:** public boolean action(Event evt, Object arg)

Parámetros de entrada **: evt,** evento realizado, y

arg, sobre el objeto que se realizo

Valor retornado **:** true si se revisó el evento y se tomo acción,

false si no se revisó.

Invoca a **: -** Imprimir (t3,30);

- archivo()

- errores_archivo(Frame, String, String)

- GENERADOR_DE_CLAVE_PUBLICA()

- Clave_Privada(Frame, java.math.BigInteger[ ])

- Clave_Publica(Frame , java.math.BigInteger[ ])

- GENERA_Lucas (String)

- encripción()

- errores_archivo(Frame,String,String)
- desencripción();
- Imprimir(TextArea) ;
- autores();
- ayuda();
- Guardarcomo();
- Abrir();
- GENERA()
- generador_C_Priv()
- crear_archivo(java.math.BigInteger, java.math.BigInteger)
- Privado (java.math.BigInteger , java.math.BigInteger , java.math.BigInteger , java.math.BigInteger, java.math.BigInteger, java.math.BigInteger, java.math.BigInteger, java.math.BigInteger)

II.20.2.5 abrir_arch (String ruta, String archivo, int i)

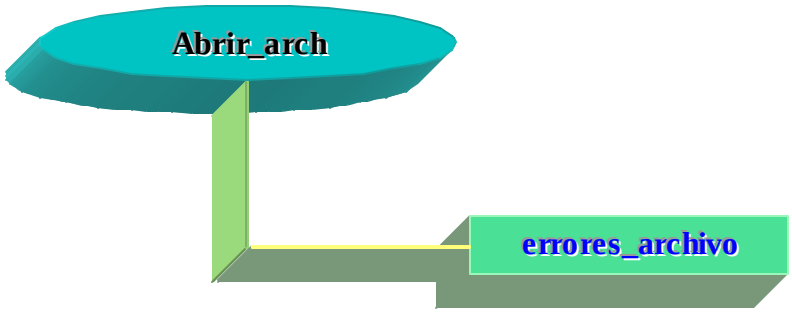


DIAGRAMA.77

Descripción **:** Es utilizado para verificar en el momento en que se abra un archivo este sea correcto.

Definición : public boolean abrir_arch(String ruta, String archivo, int i)

Parámetros de entrada : **ruta,** contiene la ruta del archivo especificada por el usuario

Archivo, nombre del archivo.

Valor retornado : true si la ruta es correcta,

false y con mensaje de error, si la ruta es equivocada.

Invoca a :- errores_archivo(Frame, String, String)

II.20.2.6 crear_archivo(java.math.BigInteger N, java.math.BigInteger Ku)

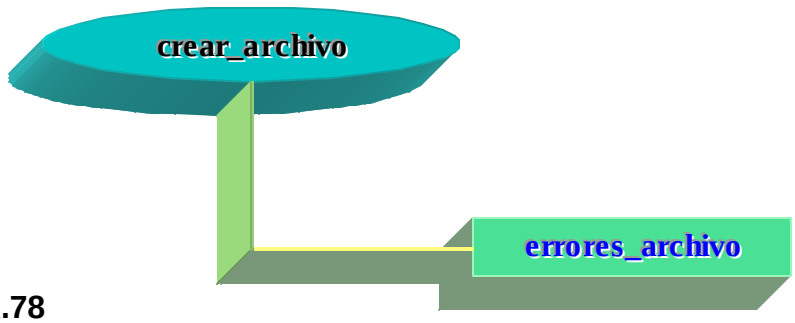


DIAGRAMA.78

Descripción : Crea un archivo y copia en él los parámetros que se le pasan.

Definición : public void crear_archivo(java.math.BigInteger N, java.math.BigInteger Ku)

Parámetros de entrada : **N,** contiene el número N que es generado por p y q

Ku, contiene la clave publica

Valor retornado : Ninguno.

Invoca a :- errores_archivo(Frame, String, String)

II.20.2.7 Privado(java.math.BigInteger N,java.math.BigInteger K1, java.math.BigInteger K2, java.math.BigInteger K3 , java.math.BigInteger K4, java.math.BigInteger P, java.math.BigInteger q)

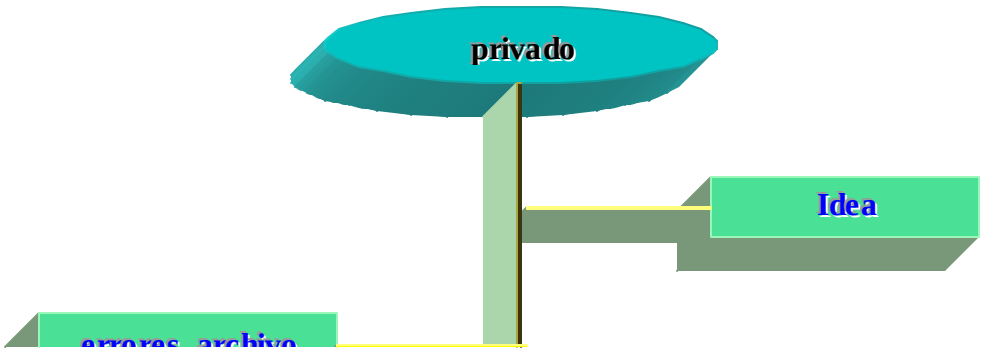


DIAGRAMA.79

Descripción **:** Se utiliza para encriptar la clave privada, mediante una clave Idea en un archivo

Definición **:** public void Privado(java.math.BigInteger N, java.math.BigInteger K1, java.math.BigInteger K2, java.math.BigInteger K3, java.math.BigInteger K4, java.math.BigInteger P, java.math.BigInteger q)

Parámetros de entrada **: N,** contiene el número N que es generado por p y q,

K1, contiene la clave privada nímero1,

K2, contiene la clave privada número 2,

K3, contiene la clave privada número 3,

K4, contiene la clave privada número 4,

P, contiene el número primo p, y

q, contiene el número primo q

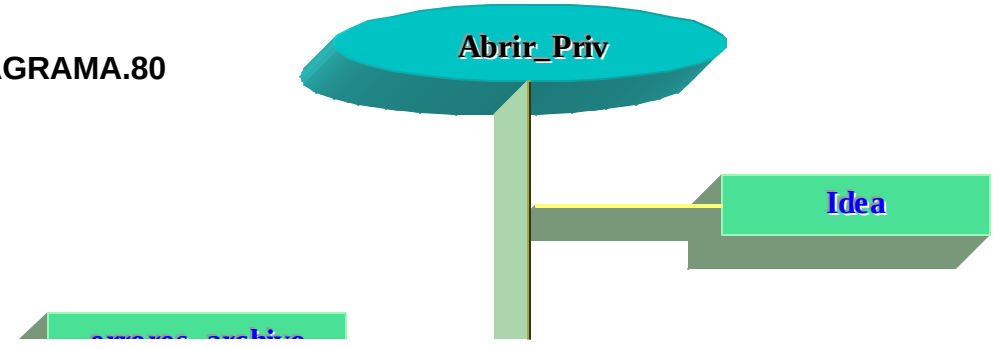
Valor retornado **:** Ninguno.

Invoca a **:**- errores_archivo(Frame, String, String)

- Idea()

II.20.2.8 Abrir_Priv()

DIAGRAMA.80



Descripción	**:** Este método es utilizado para desencriptar la clave privada Lucas que fue encriptada con una clave idea
Definición	**:** public void Abrir_Priv()
Parámetros de entrada	**:** Ninguno.
Valor retornado	**:** Ninguno.
Invoca a	**:**- errores_archivo(Frame, String, String) - Idea()

II.20.2.9 nombre(java.math.BigInteger numero)

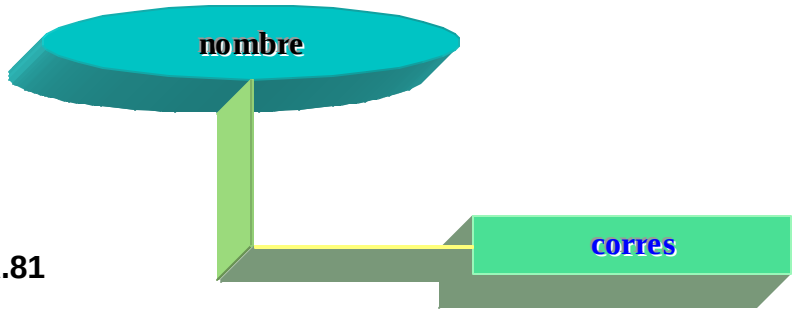


DIAGRAMA.81

Descripción : Convierte la clave privada Lucas a sus valores originales

Definición : String nombre(java.math.BigInteger numero)

Parámetros de entrada : numero.

Valor retornado : true si todo el valor fue convertido correctamente.

Invoca a :- corres(val)

II.20.2.10 corres(int valor)

Descripción	**:** pregunta por los valores para pasarlos a caracteres conocidos para hacerlas claves más legibles
Definición	**:** char corres(int valor)
Parámetros de entrada	**:** valor.
Valor retornado	**:** el carácter correspondiente a la variable valor, si se pasa del rango retorna el carácter '$'.
Invoca a	**:**Ninguno

II.20.2.11 generador_C_Priv()

Descripción	**:** Genera una clave Idea de 128 bits dependiendo de la hora y una operación matemática
Definición	**:** public java.math.BigInteger generador_C_Priv()
Parámetros de entrada	**:** Ninguno.
Valor retornado	**:** ren el cual contendrá la clave privada.

Invoca a **:**Ninguno.

BASE DE DATOS “claves”

Tabla Privada

Nombre del campo	Tipos de Datos	**Descripción**
Nombre	Texto	Guarda el nombre del usuario.
Identificador	Texto	Nombre con el cual el usuario diferenciara sus claves privadas, debe ser único en la columna para cada registro.
Clave Pública	Texto	Contiene la clave publica del usuario.
Clave Privada	Hipervinculo	Contiene la direccion donde se encuentra la clave privada Lucas cifrada.

Tabla : Publica

Nombre del Campo	Tipo de datos	Descripción
Nombre	Texto	Contiene nombre que identifica los usuarios a los que les puede enviar mensajes.
Identificador	Texto	Nombre Unico que identifica al usuario de todos los demas usuarios.
Clave Publica	Texto	Contiene la clave publica de cada usuario.

III. Codigo De La Aplicación.

```
**
Desde este archivo se coordinan las diferentes actividades de la aplicación, se realizan llamados a los objetos y métodos de estos,
también se encuentra la implementación  de la ventana principal se podría decir que esta es la  esencia de la aplicación.
 @ Autores
 @ Milton Hernandez ZakZuk
 @ Fabio Duran Ortiz
 */
```

```
import java.util.*;
import java.applet.*;
import java.awt.*;
import java.net.*;
import java.io.*;
import paquete.*;
import com.ms.wfc.core.*;
import com.ms.wfc.ui.*;
import com.ms.lang.Delegate;
                                        /* ***********************************
                                         *      ENCRIPTA Y DESENCRYPTA      *
                                        * ***** ********************************/
/**
 * @com.register ( clsid=6C90B10E-7976-11D3-9623-444553540001, typelib=6C90B105-7976-11D3-9623-444553540001 )
 */
class ENCRYDESENCR extends Frame
{
/**
Declaración de las variables que van a ser conocidas por los métodos de esta clase aquí se definen las variables /
//Se crea un objeto  barra menú
MenuBar barra_menu = new MenuBar();
//Elementos que hacen parte del objeto barra menú
java.awt.Menu Archivo12     = new java.awt.Menu("Archivo",true);
java.awt.Menu Criptografia  = new java.awt.Menu("Firmar",true);
java.awt.Menu Claves        = new java.awt.Menu("Claves"  ,true);
java.awt.Menu Ayuda         = new java.awt.Menu("Ayuda"  ,true);
java.awt.Menu Lucas         = new java.awt.Menu("Lucas"  ,true);
java.awt.Menu Correo        = new java.awt.Menu("Mail"  ,true);
//Contendrá la información del archivo a encriptar o desencriptar
TextArea      t3 = new TextArea(8,60);
//contendrá el nombre del dueño de la clave publica
TextField     t8 = new TextField(20);
//contendrá en identificador de la clave publica
TextField     t9 = new TextField(20);
//Contendrá la Clave Publica Lucas
TextField     t10 = new TextField(20);
//recibe si se desea reemplazar un archivo
boolean recibe;
//contendrá la dirección de la clave privada
TextField     t11 = new TextField(20);
//Contendrá la clave Privada
TextField     t12 = new TextField(20);
//Contendrá el Nombre de la clave privada
TextField     t21 = new TextField(20);
//Contendrá el Identificador de la clave privada
TextField     t22 = new TextField(20);
//contendrá la clave publica que se leerá del archivo de clave publica
String lin;
//variables que contendrán los archivos y directorios donde están
//guardadas las claves publicas y privadas
String dir1     = new String();
String dir2     = new String();
String arch1    = new String();
String arch2    = new String();
//Contendrá la dirección del archivo a encriptar
String original = null;
//varible para establecer el tipo de letra
java.awt.Font F2 = new java.awt.Font("Helvetica",java.awt.Font.PLAIN,14);
//contendrá la clave privada como un valor numérico
java.math.BigInteger CPR;
```

```
//recibe los valores que se recuperan de la base de datos de la tabla donde se encuentran los datos de la //clave privada
//[0] nombre .
//[1] Identificador
//[2] Dirección de clave privada
//[3]Clave publica correspondiente
String VAL[]  = new String[4];
//recibe los valores que se recuperan de la base de datos de la
//tabla donde se guardan las claves publicas.
//[0] nombre
//[1] Identificador
//[2] Clave publica
String VAL1[] = new String[3];
//botones que utilizan las imágenes arriba, abajo
BotonGrafico1 bp  = null;
BotonGrafico1 bp1 = null;
//imágenes que se usan para el icono de la aplicación y los botones
java.awt.Image Icono;
java.awt.Image Fondo;
java.awt.Image publica1;
java.awt.Image publica2;
java.awt.Image publica3;
java.awt.Image privada1;
java.awt.Image privada2;
java.awt.Image privada3;
java.awt.Font f = new java.awt.Font("Courier",java.awt.Font.BOLD,14);
//especial
Dimensión Ven;
Dimensión Img;
//fin de especial
GridBagLayout       gb = new GridBagLayout();
java.awt.Panel  Centro = new java.awt.Panel();
java.awt.Panel Centro1 = new java.awt.Panel();
medidorAvance medidor;
GridBagConstraints nueva;
/**
En el siguiente método esta contenido el diseño de la presentación
inicial.
*/

ENCRYDESENCR( String nombre )
{
   //nombre de la ventana
   super(nombre);
   recibe=true;
      //Creación del icono y el Fondo de la aplicación
   Icono          = Toolkit.getDefaultToolkit().getImage("icono1.gif");
   Fondo          = Toolkit.getDefaultToolkit().getImage("eslabones.gif");
publica1      = Toolkit.getDefaultToolkit().getImage("publica_1.gif");
   publica2      = Toolkit.getDefaultToolkit().getImage("publica_2.gif");
   publica3      = Toolkit.getDefaultToolkit().getImage("publica_3.gif");
     privada1      = Toolkit.getDefaultToolkit().getImage("privada_1.gif");
   privada2      = Toolkit.getDefaultToolkit().getImage("privada_2.gif");
   privada3      = Toolkit.getDefaultToolkit().getImage("privada_3.gif");
//Inicio de la carga de las imágenes
//creo un objeto Media tracker que contiene manejadores de Imágenes
   MediaTracker tracker = new MediaTracker(this);
   //Adiciono las imágenes al objeto creado
   tracker.addImage(Icono,0);
   tracker.addImage(Fondo,0);
   tracker.addImage(publica1,0);
   tracker.addImage(publica2,0);
   tracker.addImage(publica3,0);
   tracker.addImage(privada1,0);
   tracker.addImage(privada2,0);
   tracker.addImage(privada3,0);
     try{
   //espera a que se carguen todas las imágenes para poder mostrarlas
    tracker.waitForAll();
   }
   catch(InterruptedException e)
   {
                errores_archivo err = new errores_archivo(this,"Error!! 100","! Error cargando Imágenes !");
```

```
                err.pack();
                err.show();
               return;
    }
     //*final de carga de las imágenes, esto evita el parpadeo
  //************************
  // A esta ventana se le define el icono que mostrara
   this.setIconImage(Icono);
  //************************
  medidor = new medidorAvance(200,Fondo);

t8.setEditable(false);t8.setBackground(java.awt.Color.lightGray);
t9.setEditable(false);t9.setBackground(java.awt.Color.lightGray);
t10.setEditable(false);t10.setBackground(java.awt.Color.lightGray);

t21.setEditable(false);t21.setBackground(java.awt.Color.lightGray);
t22.setEditable(false);t22.setBackground(java.awt.Color.lightGray);
t11.setEditable(false);t11.setBackground(java.awt.Color.lightGray);
t12.setEditable(true);t12.setBackground(java.awt.Color.white);
t12.setFont(f);
t12.setEchoCharacter('*');

//Se establece la barra menú de esta ventana
   setMenuBar(barra_menu);
   //Se establece el tipo de letra de este menú
   barra_menu.setFont(F2);

   //Se le adicionan los componentes al menú se le adiciona el objeto archivo12 al menú
   barra_menu.add(Archivo12);
   //Al objeto archivo12 se le adicionan las siguientes opciones
   Archivo12.setFont(F2);
   Archivo12.add(" Nuevo... ");
   Archivo12.add(" Abrir... ");
   Archivo12.add(" Cerrar ");
   Archivo12.add("-");
   Archivo12.add(" Guardar ");
   Archivo12.add(" Guardar Como... ");
   Archivo12.add("-");
   Archivo12.add(" Salir ");

//se le adiciona el objeto Criptografia al menú
   barra_menu.add(Criptografia);

//Al objeto criptografia se le define el tipo de letra y se le adicionan las siguientes opciones
   Criptografia.setFont(F2);
   Criptografia.add(" Firmar Digitalmente ");
   Criptografia.add(" Firmar Y Cifrar ");
   Criptografia.add("-");
   Criptografia.add(" Aclarar Documento Firmado ");
   Criptografia.add(" Aclarar Documento Firmado Y Cifrado ");
barra_menu.add(Claves);
   Claves.setFont(F2);
   Claves.add(" Claves publicas ");
   Claves.add(" Claves privadas ");

 //se le adiciona el objeto Lucas al menú
   barra_menu.add(Lucas);
   Lucas.setFont(F2);
   Lucas.add(" Generador De Claves ");
   //*****
   barra_menu.add(Correo);
   Correo.add(" Microsoft Outlook ");
  //se le adiciona el objeto Ayuda al menú
   barra_menu.add(Ayuda);
   Ayuda.add(" A cerca Del Generador ");
   Ayuda.add(" Versión ");
   Ayuda.setFont(F2);
//empezamos desde la línea 167 en los comentarios :
//Se definen los paneles que se dividirá la pantalla

  //imagen de fondo
  //Dimensión Im  = Icono.size();
  //Se le ordena  la ventana sea de tamaño estándar
   //setResizable(false) ;
   t3.setEditable(false);
   setLayout(gb);
   nueva = new GridBagConstraints();
   bp = new BotonGrafico1(" Buscar A_C_Pu ",publica1,publica2,publica3,this);
  // this.setCursor(Frame.WAIT_CURSOR);
   nueva.gridx = 0;
   nueva.gridy = 0;
```

```
nueva.gridwidth  = 1;
nueva.gridheight = 2;
nueva.insets.top    = 30;
nueva.insets.left = 50;
nueva.fill          = GridBagConstraints.NONE;
nueva.anchor        = GridBagConstraints.CENTER;
gb.setConstraints(bp,nueva);
add(bp);
//adición del área de clave publica
publica();
nueva.gridx = 1;
nueva.gridy = 0;
nueva.gridwidth  = 1;
nueva.gridheight = 1;
nueva.insets.top = 25;
nueva.insets.right = 60;
nueva.insets.left  = 60;
nueva.fill          = GridBagConstraints.BOTH;
nueva.anchor        = GridBagConstraints.CENTER;
gb.setConstraints(Centro,nueva);
add(Centro);
//Adicion del área de clave privada
privada();
nueva.gridx = 1;
nueva.gridy = 1;
nueva.gridwidth  = 1;
nueva.gridheight = 1;
nueva.fill          = GridBagConstraints.NONE;
nueva.anchor        = GridBagConstraints.CENTER;
gb.setConstraints(Centro1,nueva);
add(Centro1);

bp1 = new BotonGrafico1(" Buscar A_C_Pv ",privada1,privada2,privada3,this);
nueva.gridx = 2;
nueva.gridy = 0;
nueva.gridwidth     = 1;
nueva.gridheight    = 2;
nueva.insets.top    = 40;
nueva.insets.bottom = 10;
nueva.insets.right  = 50;
nueva.insets.left   = 5;
nueva.fill          = GridBagConstraints.NONE;
nueva.anchor        = GridBagConstraints.NORTHEAST;
gb.setConstraints(bp1,nueva);
add(bp1);



nueva.gridx = 0;
nueva.gridy = 2;
nueva.gridwidth  = 3;
nueva.gridheight = 1;
nueva.insets.top    = 20;
nueva.insets.bottom = 20;
nueva.insets.left   = 50;
nueva.insets.right  = 50;
nueva.fill          = GridBagConstraints.BOTH;
nueva.anchor        = GridBagConstraints.CENTER;
gb.setConstraints(t3,nueva);
add(t3);


nueva.gridx = 0;
nueva.gridy = 3;
nueva.gridwidth  = 3;
nueva.gridheight = 1;
nueva.insets.top    = 10;
nueva.insets.bottom = 10;
nueva.insets.left   = 80;
nueva.insets.right  = 80;
nueva.fill          = GridBagConstraints.BOTH;
nueva.anchor        = GridBagConstraints.CENTER;
gb.setConstraints(medidor,nueva);
add(medidor);
}


public void paint(java.awt.Graphics g)//dibuja la imagen del boton
   {
```

```
    Ven = this.getSize();
    int x ,y ,x1 ,y1;
                //averiguo el tamaño de la ventana
                 x = Ven.width;
                 y = Ven.height;
                 //averigo el tamaño de la imagen
                x1 = Fondo.getWidth(this);
                y1 = Fondo.getHeight(this);
               int i=0,j;
for(j=0;((j*y1)<(y));j++)
     {
      for(i=0;((i*x1)<(x));i++)
                              {//se dibuja la imagen del fondo
                              g.drawImage(Fondo,(x1*i),(y1*j),this);
                              }
     }//for
   //area_publica(g);
   System.gc();
} //paint

//******** crea un contenedor que muestra la información de la clave publica
public void publica()
{
java.awt.Label A    = new java.awt.Label("RECEPTOR",2);A.setFont(f);
java.awt.Label AA    = new java.awt.Label("/EMISOR",0);AA.setFont(f);
A.setForeground(java.awt.Color.white);
A.setBackground( java.awt.Color.getHSBColor( ((float)0.4),((float)1.2),((float)2.7) )  );
AA.setForeground(java.awt.Color.black);
AA.setBackground( java.awt.Color.getHSBColor( ((float)0.8),((float)0.2),((float)0.7) )  );


java.awt.Label aa    = new java.awt.Label("Nombre:",2);aa.setFont(f);
java.awt.Label bb    = new java.awt.Label("Identificador:",2);bb.setFont(f);
java.awt.Label cc    = new java.awt.Label("Clave Publica:",2);cc.setFont(f);

Centro.setLayout(new GridLayout(4,2));
Centro.add(A);
Centro.add(AA);
Centro.add(aa);
Centro.add(t8);
Centro.add(bb);
Centro.add(t9);
Centro.add(cc);
Centro.add(t10);
return;
}
//**
/**comprime un archivo y lo guarda en otro con el mismo nombre
 * agregándole una z al nombre del archivo nuevo.
*/

public void comprimir(String archivo,int lij)
{
                        try{
                                File nombre = new File(archivo);

                          char sep = File.separatorChar;
//se obtiene el separador del sistema
                          int ext = archivo.lastIndexOf(sep);
                          String a = archivo.substring(ext,archivo.length());
                          String b = archivo.substring(0,archivo.length()-8);
                          java.util.zip.ZipOutputStream hh;
                          java.io.FileInputStream jj = new java.io.FileInputStream(archivo);
                          if(lij==1){
                            hh = new java.util.zip.ZipOutputStream(new java.io.FileOutputStream(b+".Dso"));
                          }else{
                             hh = new java.util.zip.ZipOutputStream(new java.io.FileOutputStream(b+".Mss"));
                            }

                          java.util.zip.ZipEntry nuevo = new java.util.zip.ZipEntry(a);
                          //se comprime en un nuevo objeto
                          hh.putNextEntry(nuevo);
                          int lo;
                          byte cad[] = new byte[1024];
                          while( (lo=jj.read(cad))>0)
                           {
                                hh.write(cad,0,lo);
                            }
              hh.finish();
              hh.close();
              jj.close();
```

```
                nombre.delete();
                }catch(java.io.IOException ge){}
return;
}

//**
/**comprime un archivo y lo guarda en otro con el mismo nombre
 * agregándole una z al nombre del archivo nuevo.
*/
public String descomprimir(String archivo)
{
                String nombre1 ="";
                                try{
                  java.io.File arc = new java.io.File(archivo);
                  java.util.zip.ZipInputStream    oo = new java.util.zip.ZipInputStream(new java.io.FileInputStream(archivo));
                  java.util.zip.ZipEntry      nombre = oo.getNextEntry();
                  nombre1 = arc.getPath().substring(0,arc.getPath().length()-arc.getName().length()) +
nombre.getName().substring(1,nombre.getName().length()-4);
                  File nom = new File(nombre1);
                  if(nom.exists())
                   {
                   recibir_desi nue = new recibir_desi(this,"Hola"  );
                   nue.pack();
                   nue.show();
                   if(recibe==true){nom.delete();}else{oo.close();return(null);}
                   }
                  java.io.FileOutputStream  ss = new java.io.FileOutputStream(nombre1);
                  byte a[] = new byte[1024];
                  int lo = 1;
                  while((lo=oo.read(a))>0)
                  {
                    ss.write(a,0,lo);
                                }
                                oo.close();
                                ss.close();
                }catch(java.io.IOException ge){}
return nombre1;
}



//*** crea un contenedor que muestra la informacion de la clave  privada
public void privada()
{
java.awt.Label A    = new java.awt.Label("EMISOR /",2);A.setFont(f);
java.awt.Label AA    = new java.awt.Label("RECEPTOR",0);AA.setFont(f);
java.awt.Label aa    = new java.awt.Label("Nombre ",2);aa.setFont(f);
java.awt.Label bb    = new java.awt.Label("Identificador ",2);bb.setFont(f);
java.awt.Label cc    = new java.awt.Label("Ubicación Clave Privada: ",2);cc.setFont(f);
java.awt.Label PP    = new java.awt.Label("Digite Su Clave Privada: ",2);PP.setFont(f);
//colores del cuadro de dialogo
A.setForeground(java.awt.Color.white);
A.setBackground( java.awt.Color.getHSBColor( ((float)0.4),((float)1.2),((float)2.7) )  );
AA.setForeground(java.awt.Color.black);
AA.setBackground( java.awt.Color.getHSBColor( ((float)0.8),((float)0.2),((float)0.7) )  );
Centro1.setLayout(new GridLayout(5,2));
Centro1.add(A);
Centro1.add(AA);
Centro1.add(aa);
Centro1.add(t21);
Centro1.add(bb);
Centro1.add(t22);
Centro1.add(cc);
Centro1.add(t11);
Centro1.add(PP);
Centro1.add(t12);
return;}


/************ Este es main el principal**************************/

 public static void main(String argv[])

 {
   //se declara un tipo de fuente
   java.awt.Font F1            = new java.awt.Font("Helvetica",java.awt.Font.PLAIN,14);
   //se crea un objeto de la ventana principal
   ENCRYDESENCR firma = new ENCRYDESENCR("Firma Digital");
   //se le define el tipo de fuente
   firma.setFont(F1);
```

```
   //Se  le coloca el color del fondo
   firma.setBackground(java.awt.Color.gray);
   //firma.pack();
   firma.resize(790,560);
   //se muestra la ventana
   firma.show();
 }
//******
//********** detecta en donde esta el mouse en cualquier momento
public boolean mouseMove( java.awt.Event evt,int x,int y )
 {
                //calculo en que posición y el tamaño del boton
                float a,b;
                int c,d;
                a = bp.TOP_ALIGNMENT;
                b = bp.LEFT_ALIGNMENT;
                c = bp.HEIGHT;
                d = bp.WIDTH;
                 //si el cursor esta dentro de estas coordenadas...
                if(!((x >= a)&&(x<=(a+d))&&(y>=b)&&(y<=(b+c))))
                {
                bp.ponerBoton(BotonGrafico1.ARRIBA);
                }
                //calculo en que posición y el tamaño del boton
                float a1,b1;
                int c1,d1;
                a1 = bp1.TOP_ALIGNMENT;
                b1 = bp1.LEFT_ALIGNMENT;
                c1 = bp1.HEIGHT;
                d1 = bp1.WIDTH;
                //si el cursor esta dentro de estas coordenadas...
                if(!((x >= a1)&&(x<=(a1+d1))&&(y>=b1)&&(y<=(b1+c1))))
                {
                //cambio la posición del boton
                bp1.ponerBoton(BotonGrafico1.ARRIBA);
                }
                setCursor(Frame.DEFAULT_CURSOR);//coloco la imagen del mouse por defecto
                return( true );
 }
  /**  Este método detecta los eventos que se realizan sobre la ventana  **/
  public boolean handleEvent(java.awt.Event evt)
  {
   //si se detecta cerrar ventana salga de la aplicación
   if(evt.id == java.awt.Event.WINDOW_DESTROY)
    {
    System.exit(0);
    }
    return super.handleEvent(evt);
                //sino devuelve el evento
   }
/*************   acciones sobre  cada objeto de la ventana ***************/
public boolean action(java.awt.Event evt, Object arg)
 {
 //**********  Manejo de los eventos de los Botones  *********************/
 //si se hace click sobre un objeto de tipo boton gráfico o Button
 if ((evt.target instanceof java.awt.Button)||(evt.target instanceof BotonGrafico1))
  {
                if (arg.equals(" Buscar A_C_Pu "))
    {
                //pone el boton arriba
                 bp1.ponerBoton(BotonGrafico1.ARRIBA);

                if(t8.getText().length()<1)
                {
                claves_publicas();
                }else{
                 t8.setText(null);
                 t9.setText(null);
                 t10.setText(null);
                 lin = null;
                  }
                }/** fin del if */
//es análogo a   *****  if (arg.equals(" Buscar A_C_Pu ") ) **********/
    if (arg.equals(" Buscar A_C_Pv "))
    {
                bp.ponerBoton(BotonGrafico1.ARRIBA);
                if(t21.getText().length()<1)
                {
                claves_privadas();
                }else        {
                                        t21.setText(null);
```

```
                                        t22.setText(null);
                                        t11.setText(null);
                                        t12.setText(null);
                                        }
                }
  return true;//evento procesado
  } // (evttarget instanceof Button)
// Manipulación de los eventos del Menu //
 if(evt.target instanceof java.awt.MenuItem)
 {
  //objeto utilizado para mostrar un dialogo
  archivo arr;
  if(arg.equals(" Microsoft Outlook "))
  {
                try{
                        java.lang.Runtime de = java.lang.Runtime.getRuntime();
                        de.exec("OUTLOOK.EXE");
                        }catch(Exception j){
                                errores_archivo err = new errores_archivo(this,"Error 002","Imposible ejecutar Aplicación");
                                                                        err.pack();
                                                                        err.show();
                                                                        return true;
                                        }
 }

  else if (arg.equals(" Generador De Claves "))
    {
   Dimension tam,tam_di;
   java.awt.Font F1        = new java.awt.Font("Courier",java.awt.Font.PLAIN,14);
                //llama al objeto GENERA_LUCAS el cual cumple su proceso
    GENERA_LUCAS p = new GENERA_LUCAS("Generador De Claves ");
   p.setBackground(java.awt.Color.gray);
   p.setFont(F1);
   p.pack();
   //fija que la ventana se abra en el centro
                tam    = Toolkit.getDefaultToolkit().getScreenSize();
   tam_di =             p.size();
   p.move((tam.width-tam_di.width)/2,(tam.height-tam_di.height)/2);
   p.show();
    }
  else if(arg.equals(" Firmar Digitalmente "))
  {
                boolean col = true;
                t3.setText(null);
                t3.setEditable(false);
                this.setTitle("Firma Digital");
                original = null;

                if(original == null){ Abrir("Abrir Archivo A Firmar . . .",false);
                                                                        if(original ==null ){ col=false;}
                                                                         }
    if( ((t11.getText().trim()).length() == 0)&&(col==true) ){ claves_privadas();

                  if(( (t11.getText().trim()).length() == 0 )){col=false;}

                        }
                if(( (t12.getText().trim()).length() == 0)&&(col==true))
                  {
                        String datos[] = {t11.getText(),t21.getText(),t22.getText()};
                        recibir_privados(datos,true);
                    if(t12.getText().trim().length()== 0 ){col = false;}
                  }
                  if(col==true)
                   {
                    try{
                                encripcion1();
                        }catch(Exception re){
                                                                        errores_archivo err =  new
errores_archivo(this," ¡ Error 120 ! ","! Clave Errónea !");
                                                                        err.pack();
                                                                        err.show();
                                                                        return true;
                                                                }
                  }//if(col==true)
  }
else if(arg.equals( " Aclarar Documento Firmado "))
{
                t3.setText(null);
                t3.setEditable(false);
                this.setTitle("Firma Digital");
                original = null;
```

```
            boolean col = true;
            if(original == null){ Abrir(" Seleccione Archivo A Revisar Firma (*.Dso)",true);
                                                                        if(original == null){ col=false;}
                                                                                }

             if(col==true)
              {
               try{
                        desencripcion1();
                   }catch(Exception re){
                                                                                errores_archivo err =  new
errores_archivo(this," ¡ Error 120 ! ","! Clave Errónea !");
                                                                                err.pack();
                                                                                err.show();
                                                                                return true;
                                                                        }
              }//if(col==true)
}
else if (arg.equals(" Firmar Y Cifrar "))
            {
             boolean col = true;
             t3.setText(null);
             t3.setEditable(false);
             original = null;
             this.setTitle("Firma Digital");

             if(original == null){ Abrir("Archivo A firmar y cifrar",false);
                                                                        if(original == null){ col=false;}
                                                                                }
             if( ((t10.getText().trim()).length() == 0)&&(col==true) ){ claves_publicas();
             if(( (t10.getText().trim()).length() == 0 )){col=false;}
                    }
             if( ((t11.getText().trim()).length() == 0)&&(col==true) ){ claves_privadas();
             if(( (t11.getText().trim()).length() == 0 )){col=false;}
                   }
            if(( (t12.getText().trim()).length() == 0)&&(col==true))
             {
                        String datos[] = {t11.getText(),t21.getText(),t22.getText()};
                        recibir_privados(datos,true);
              if(t12.getText().trim().length()== 0 ){col = false;}
             }
             if(col==true)
              {
               try{
                        encripcion();
                   }catch(Exception re){
                                                                                errores_archivo err =  new
errores_archivo(this," ¡ Error 120 ! ","! Clave Errónea !");
                                                                                err.pack();
                                                                                err.show();
                                                                                return true;
                                                                        }
              }//if(col==true)
   }//cierra el if de (arg.equals(" Encripción "))
 /**
sino pregunto si es para Desencripción
  funciona análogamente a if(arg.equals(" Encripción "))
 */
   else if (arg.equals(" Aclarar Documento Firmado Y Cifrado "))
            {
             boolean col = true;
             t3.setText(null);
             t3.setEditable(false);
             this.setTitle("Firma Digital");
             original = null;
             if(original == null){ Abrir("Seleccione Archivo Firmado y Cifrado(*.Mss)",true);
                                                                        if(original == null){ col=false;}
                                                                                }

              if(col==true)
               {
                try{
                        desencripcion();
                   }catch(Exception re){errores_archivo err =  new errores_archivo(this," ¡ Error 120 ! ","! Clave Errónea !");
                                        err.pack();
                                         err.show();
                                          return true;
                                          }
              }//if(col==true)

   }//(arg.equals(" desencripcion "))
```

```
   else if(arg.equals(" Guardar Como... "))
     {
                if(t3.isEditable()==false)//si esta desactivado no hay un archivo abierto
                {
                        errores_archivo err = new errores_archivo(this," ¡ Error 160 !","! Archivo Sin Abrir!");
                        err.pack();
                        err.show();
                 }
                else{//si no si pregunte si desea guardar y guarde el archivo
                                arr = new archivo(this,original,"¿ Desea Guardar Archivo Actual?");
                                arr.pack();
                                arr.show();
                                //llama al método guardar como
                                Guardarcomo();
                   }
     }


   else
              if(arg.equals(" Nuevo... "))
     {
                        if( (original == null)&&(t3.isEditable()==false) )
                        {
                                //si es nuevo active el  área de texto
                                t3.setEditable(true);
                   }
                    else {//Sin no es nuevo guarde el archivo y cree uno nuevo
                                                arr = new archivo(this,original,"Desea Guardar El Archivo");
                                                arr.pack();
                                                arr.show();
                                                original = null;
                                                t3.setText(null);
                          }
}// else if
   else
              if(arg.equals(" Abrir... "))
     {
                 //si no esta activado no hay archivo abierto,abralo
                 if(t3.isEditable()== false)
                        {
                        //
                 //t3.setEditable(true);
                        //llama al método abrir
                        Abrir("Abrir archivo. . .",false);
                 }else{//sino pregunte, ciérrelo y abra uno nuevo
                                arr = new archivo(this,original,"Desea Guardar El Archivo");
                                arr.pack();
                                arr.show();
                                t3.setText(null);
                                t3.setEditable(false);
                                //llama al método abrir
                                Abrir("Abrir archivo",false);
                                }
              }// else if
 else
              if(arg.equals(" Guardar "))
     {
              if(t3.isEditable()==false)//si no hay archivo abierto...
              {
              errores_archivo err = new errores_archivo(this," ¡ Error 160 ! ","! Archivo Sin Abrir!");
              err.pack();
              err.show();
              }
              else if(original != null)//sino guardelo si tiene nombre
              {guardar(original);}
              else//si no tiene nombre llame a guardar como
              {
              Guardarcomo();
              }
    }// else if
   else
              if(arg.equals(" Cerrar "))
     {
              if(t3.isEditable()==false)//si hay archivo abierto cierrelo sino mande error
              {
              errores_archivo err = new errores_archivo(this,"¡ Error 170 !","¡ Archivo Sin Abrir !");
              err.pack();
              err.show();
              }
              else
              {
```

```
                arr = new archivo(this,original,"Desea Guardar El Archivo");
                arr.pack();
                arr.show();
                t3.setText(null);
                t3.setEditable(false);
                this.setTitle("Firma Digital");
                original=null;
                }
    }// else if
else if(arg.equals(" Claves privadas "))//para accesar a las base de datos
    {
                claves_privadas();
    }// else if
  else//analogo a if(arg.equals(" Claves privadas "))
               if(arg.equals(" Claves publicas "))
               {
                 claves_publicas();
               }
else if (arg.equals(" Salir "))//sale de la aplicación
    {
                System.exit(0);
    }
 else if(arg.equals(" Versión "))
    {
                //crea un objeto donde se encuentra la descripción de los autores y versión
     autores  Aplicación = new autores();
     Aplicacion.resize(300,300);
                //lo muestra
     Aplicacion.show();
     return true;//evento procesado
    }
 else if(arg.equals(" A cerca Del Generador ")              )
  {
  //crea un objeto de ayuda que despliega la ayuda de la aplicación
    try{
                        java.lang.Runtime de = java.lang.Runtime.getRuntime();
                        de.exec("MANUAL.html");
                        }catch(Exception j){
                            errores_archivo err = new errores_archivo(this,"Error 030"," Imposible Abrir Explorador ");
                                                                        err.pack();
                                                                        err.show();
                            }
   return true;
  }
    //*********************
   return true;
 }//(evt.target instanceof MenuItem)
//************************************************************
  return false;//evento no procesado
 }//boolean action
//***************Recibe los datos de usuario de la tabla privada , escogidos por el usuario
public void recibir_privados(String datos[],boolean band)
{
               //existen datos insertados pregunte si desea cambiar datos
                if (band ==true)
                 {
                        t11.setText(datos[0]);
                        t21.setText(datos[1]);
                        t22.setText(datos[2]);
                        String dat[] = { t21.getText(),t22.getText() };
                        Vector clave_traida = new Vector();
                        recupera_clave cap = new recupera_clave(dat,clave_traida,this);
                        cap.inicio(cap);
                        if(clave_traida.size()>0)
                 {t12.setText((String)clave_traida.elementAt(0));}
                 }else{
                            t11.setText(datos[0]);
                            t21.setText(datos[1]);
                            t22.setText(datos[2]);
                             }

return;
}
public void rec(boolean respuesta)
{
              if(respuesta==true){recibe = true;}else{recibe = false;}


}

//***************Recibe los datos de usuario de la tabla publico , escogidos por el usuario
```

```
public void recibir_publicos(String datos[])
{
  t8.setText(datos[0]);
  t9.setText(datos[1]);
  t10.setText(null);
  t10.setText(datos[2].trim());

   String lan = t10.getText().trim();
   lin = lan;
}



//************************************************************/
/**
- El siguiente método recibe la variable M la cual es la que contiene la clave
  publica .
- El método  extractor toma este valor y la convierte a un Numero, el cual
  será la clave Publica.
- Utiliza a descorres
- retorna la clave publica
public Vector extractor(String M)
{

 int k = M.length();//toma el tamaño de la cadena recibida
 int y = M.indexOf('.');//se toma en que posición de la cadena esta el carácter "."
 Vector Pub = new Vector(2,1);//Vector en el que e guardara (N,e) = Clave Publica
 //inicializacion ellas variables
 char m[] = M.toCharArray();//se convierte la cadena a un arreglo de caracteres

 java.math.BigInteger N = new java.math.BigInteger("0");//se guardara provisionalmente N
 java.math.BigInteger L = new java.math.BigInteger("0");//se guardara provisionalmente e


int i=0;
for(i=y -1;i>=0;i--)
 {
  // " 125 "    a N se le realiza un corrimiento de 6 bits a la izquierda
   // y se actualiza N
N = N.shiftLeft(6);
  //a N se le realiza un or con el valor que devuelve DESCORRES(m[i]) y se actualiza N
  //como descorres devuelve un carácter que en nuestros caracteres corresponde a un valor
  //de 0-63 ; 2^6=64(caracteres) que fue como se guardo la clave
N = N.or(java.math.BigInteger.valueOf(DESCORRES(m[i])) );
 }
 for(i=k -1; i>y ;i--)
 {//análoga al "125", anterior
  L = L.shiftLeft(6);
  L = L.or(java.math.BigInteger.valueOf(DESCORRES(m[i])) );
 }
 //adición de elementos al vector
 Pub.addElement(N);//Es el valor N resultado de la multiplicación P*Q
 Pub.addElement(L);//Es Un  valor aleatorio Utilizado por Lucas que combinado con
                // N conforman la clave publica
return Pub;
}


//*****************************************/
/**
Descorres contiene la representación de los caracteres ASCII utilizados
para representar la clave publica y privada, las cuales son valores numéricos
por lo que era necesario representarlos de una manera mas sencilla
  Del valor char recibido se devuelve su valor numérico menor de 64
  2^6 = 64;
Basada en la representación R64

/*/
int DESCORRES(char valor)
{
 //al carácter 'a' le corresponde el numero 0 que seria igual a 000000 = 6 bits...
 if (valor == 'A'){return 0;}
 if (valor == 'B'){return 1;}
 if (valor == 'C'){return 2;}
 if (valor == 'D'){return 3;}
 if (valor == 'E'){return 4;}
 if (valor == 'F'){return 5;}
 if (valor == 'G'){return 6;}
 if (valor == 'H'){return 7;}
 if (valor == 'I'){return 8;}
 if (valor == 'J'){return 9;}
```

```
 if (valor == 'K'){return 10;}
 if (valor == 'L'){return 11;}
 if (valor == 'M'){return 12;}
 if (valor == 'N'){return 13;}
 if (valor == 'O'){return 14;}
 if (valor == 'P'){return 15;}
 if (valor == 'Q'){ret urn 16;}
 if (valor == 'R'){return 17;}
 if (valor == 'S'){return 18;}
 if (valor == 'T'){return 19;}
 if (valor == 'U'){return 20;}
 if (valor == 'V'){return 21;}
 if (valor == 'W'){return 22;}
 if (valor == 'X'){return 23;}
 if (valor == 'Y'){return 24;}
 if (valor == 'Z'){return 25;}
 if (valor == 'a'){return 26;}
 if (valor == 'b'){return 27;}
 if (valor == 'c'){return 28;}
 if (valor == 'd'){return 29;}
 if (valor == 'e'){return 30;}
 if (valor == 'f'){return 31;}
 if (valor == 'g'){return 32;}
 if (va lor == 'h'){return 33;}
 if (valor == 'i'){return 34;}
 if (valor == 'j'){return 35;}
 if (valor == 'k'){return 36;}
 if (valor == 'l'){return 37;}
 if (valor == 'm'){return 38;}
 if (valor == 'n'){return 39;}
 if (valor == 'o'){return 40;}
 if (valor == 'p'){return 41;}
 if (valor == 'q'){return 42;}
 if (valor == 'r'){return 43;}
 if (valor == 's'){return 44;}
 if (valor == 't'){return 45;}
 if (valor == 'u'){return 46;}
 if (valor == 'v'){return 47;}
 if (valor == 'w'){return 48;}
 if (valor == 'x'){return 49;}
 if (valor == 'y'){return 50;}
 if (valor == 'z'){return 51;}
 if (valor == '0'){return 52;}
 if (valor == '1'){return 53;}
 if (valor == '2'){return 54;}
 if (valor == '3'){return 55;}
 if (valor == '4'){return 56;}
 if (valor == '5'){return 57;}
 if (valor == '6'){return 58;}
 if (valor == '7'){return 59;}
 if (valor == '8'){return 60;}
 if (valor == '9'){return 61;}
 if (valor == '+'){return 62;}
 if (valor == '/'){return 63;}
//si retorna el menos quiere decir que la clave es una clave falsa
return 64;
}
/*************************************************



int void escoger_clave()
devuelve el valor correspondiente a la clave
privada que se necesita para encriptar o desencriptar
un valor
este valor se le calcula (V*V) - 4
Esta escogencia  es un  valor estándar

SE UTILIZA LA FUNCION DE Jacobi PARA DETERMINAR LOS SIMBOLOS LEGENDRE

  símbolos LEGENDRE
 D/p              D/q      Clave Privada
 1                                                  1                    1
 1                                                  -1                   2
-1                                                          1                    3
-1                                                  -1                   4
/************************************************/
public int escoger_clave(java.math.BigInteger valor,java.math.BigInteger Secret5,java.math.BigInteger Secret6)throws Exception
{
// Se declara la variable auxiliar aux la cual contendrá el valor del determinante D
// o sea que aux = (D*D) - 4
```

```
java.math.BigInteger aux = (valor.multiply(valor)).subtract(java.math.BigInteger.valueOf(4));
//CIFRA contendrá los valores de Jacobi
clave ENCIFRA = new clave();
//se pregunta por los diferentes valores que contendrá ENCIFRA para la respectiva clave
if((ENCIFRA.jacobi(aux,Secret5)== 1)&&(ENCIFRA.jacobi(aux,Secret6)== 1))
 {
  return(1);
 }
else
if((ENCIFRA.jacobi(aux,Secret5)== 1)&&(ENCIFRA.jacobi(aux,Secret6)== -1))
 {
return(2);
 }
else
if((ENCIFRA.jacobi(aux,Secret5)== -1)&&(ENCIFRA.jacobi(aux,Secret6)== 1))
 {
return(3);
 }
else
if((ENCIFRA.jacobi(aux,Secret5)== -1)&&(ENCIFRA.jacobi(aux,Secret6)== -1))
 {
return(4);
 }
// retorna cero en caso de que ninguno de los valores sea el correcto
// podemos pensar que la clave es errónea
 return 0;
}
//***********************************************
```

```
public void encripcion()throws Exception
{
//contendrá la función has encriptada
 java.math.BigInteger ENDES = new java.math.BigInteger("0");
  Vector Publica = new Vector(2,1);//contendrá los valores de la clave publica
//se encuentra el proceso para hallar la función has
  RMD160 P = new RMD160();
//Se guardaran los valores de la clave privada
  java.math.BigInteger Secret[] = new java.math.BigInteger[7];
//para determinar la función Jacobi
  clave ENCIFRA = new clave();
// se declara un objeto tipo archivo para luego utilizar sus métodos
  archivos arch = new archivos();

  int  Long1;
  byte HAS[];
  java.math.BigInteger has//contendra la  función has
  java.math.BigInteger hash1;//
try{
                medidor.ponerAvance(0.0f);
                Abrir_Priv(Secret);//Desencripta el archivo de clave privada
                //y guarda en secret las claves secretas Lucas
    }catch(Exception e)//si existe un problema al desencriptar la
                //clave privada envíe mensaje de error
    {
      errores_archivo err = new errores_archivo(this,"¡ Error 190 !","¡ Clave Privada Invalida !");
                 err.pack();
                 err.show();
                 return;
      }
try{//si la clave  publica es invalida envíe mensaje de error
              medidor.ponerAvance(0.01f);
    Publica = extractor(lin);//Extracta el archivo de clave publica
    }
 catch(Exception e)
    {
                //se crea un objeto de dialogo para desplegar un mensaje de error
                 errores_archivo err = new errores_archivo(this,"¡ Error 200 ! ","¡ Clave publica invalida !");
                 err.pack();
                 err.show();
                 return;
    }
 //Se abren los archivos d3e clave publica y privada respectivamente
```

```
 medidor.ponerAvance(0.02f);
 has = (java.math.BigInteger)P.iopRMD(original);
 //Se halla la función has
 //Se encripta la función has con la clave privada correspondiente
 int i = escoger_clave(hash,Secret[5],Secret[6]);
 medidor.ponerAvance(0.03f);
 ENDES = ENCIFRA.calculo(hash,Secret[i],Secret[0]);//encifra con la clave correspondiente a has
 String nuevo = original".Mss";//le agrega la extensión al archivo

 //duplica al archivo para trabajar sobre el duplicado
 medidor.ponerAvance(0.04f);
 arch.duplicar(original,nuevo,this);
 //concatena arch+hash(E)
 medidor.ponerAvance(0.05f);
 arch.concatenar(nuevo,ENDES,this);

//se concatena con la identificación del usuario del emisor
String nombre_id = new String(t22.getText().trim());
byte nombre[] = nombre_id.getBytes();
java.math.BigInteger Ident = new java.math.BigInteger(nombre);
arch.concatenar(nuevo,Ident,this);
////////////////////////////////////////////////

 //generar clave de sesión
 java.math.BigInteger KSe = new java.math.BigInteger("0");
 medidor.ponerAvance(0.06f);
 KSe = CS();
 //encriptar con idea y la clave de sesió n
 try{
 medidor.ponerAvance(0.7f);
 arch.ENCIDEA(KSe,nuevo,this,1);
  }catch(Exception e){
                        errores_archivo err = new errores_archivo(this,"¡ Error 210 ! ","¡ Error al Desencriptar !");
                        err.pack();
                        err.show();
                        return;
      }

 //encriptar CS con la clave publica
 medidor.ponerAvance(0.08f);
 ENDES = ENCIFRA.calculo(KSe,(java.math.BigInteger)Publica.elementAt(1),(java.math.BigInteger)Publica.elementAt(0));//enc
a has
 //y concatenar
 medidor.ponerAvance(0.09f);
 arch.concatenar(nuevo,ENDES,this);//concatena arch+hash(E)

//concatenar el Identificador de  usuario receptor
String nombre_id1 = new String(t9.getText().trim());
byte nombre1[] = nombre_id1.getBytes();
java.math.BigInteger Ident1 = new java.math.BigInteger(nombre1);
arch.concatenar(nuevo,Ident1,this);
//**************************************
medidor.ponerAvance(1f);
comprimir(nuevo,2);
medidor.inicio();
return;
}

/**Solo firma un documento
 */
public void encripcion1()throws Exception
{
//contendrá la función has encriptada
 java.math.BigInteger ENDES = new java.math.BigInteger("0");

  RMD160 P = new RMD160();
//Se guardaran los valores de la clave privada
  java.math.BigInteger Secret[] = new java.math.BigInteger[7];
//para determinar la función Jacobi
  clave ENCIFRA = new clave();
// se declara un objeto tipo archivo para luego utilizar sus métodos
  archivos arch = new archivos();
int  Long1;
  byte HAS[];
  java.math.BigInteger has//contendra la función has
  java.math.BigInteger hash1;//

  try{
                medidor.ponerAvance(0.0f);
                Abrir_Priv(Secret);//Desencripta el archivo de clave privada
                //y guarda en secret las claves secretas Lucas
```

```
   }catch(Exception e)//si existe un problema al desencriptar la
                //clave privada envíe mensaje de error
   {
     errores_archivo err = new errores_archivo(this,"¡ Error 190 !","¡ Clave Privada Invalida !");
                err.pack();
                err.show();
                return;
      }
 medidor.ponerAvance(0.02f);
 has = (java.math.BigInteger)P.iopRMD(original);
 //Se halla la función has
 //Se encripta la función has con la clave privada correspondiente
 int i = escoger_clave(hash,Secret[5],Secret[6]);
 medidor.ponerAvance(0.03f);
 ENDES = ENCIFRA.calculo(hash,Secret[i],Secret[0]);//encifra con la clave correspondiente a has
 String nuevo = original+" .Dso";//le agrega la extensión al archivo

 //duplica al archivo para trabajar sobre el duplicado
 medidor.ponerAvance(0.04f);

 arch.duplicar(original,nuevo,this);
 //concatena arch+hash(E)
medidor.ponerAvance(0.05f);
arch.concatenar(nuevo,ENDES,this);
medidor.ponerAvance(0.09f);
//se concatena el identificador y la archivo firmado y la firma cifrada
String nombre_id = new String(t22.getText().trim());
byte nombre[] = nombre_id.getBytes();
java.math.BigInteger Ident = new java.math.BigInteger(nombre);
arch.concatenar(nuevo,Ident,this);
medidor.ponerAvance(1f);
comprimir(nuevo,1);
medidor.inicio();
return;
}


/**
proceso de desencripcion DE la firma
**/
public void desencripcion1()throws Exception
{
 //se crea un objeto para generar la función has de comparación
 RMD160 P = new RMD160();
 //se crea un objeto donde se encuentran los métodos de desencripcion
 clave ENCIFRA = new clave();
 //Objeto para manejar los archivos
 archivos arch = new archivos();
 java.math.BigInteger ENHASH;//se guardara la función has encriptada
 java.math.BigInteger has//se guardara la función has desencriptada
//Se guardaran los valores numéricos de la clave privada del receptor
 Vector Publica = new Vector(2,1);
//captura de la locación del archivo a revisar

 String nuevo  = descomprimir(original);
 if(nuevo==null){return;}
 //String nuevo1 = descomprimir(original);


 //crea una copia la cual es la que voy a manipular
// arch.duplicar(original,nuevo,this);

 //separa la clave de sesión encriptada y el texto encriptado con esta
  try{

 //              separa el identificador del archivo
 java.math.BigInteger j = new java.math.BigInteger("0");
 j = arch.desconcatenar(nuevo,this);
 byte Ident2[] = j.toByteArray();
 String nom = new String();
 int cont = 0,num = Ident2.length;
 while(cont<num)
 {
 nom = nom + String.valueOf((char)Ident2[cont]);
 cont++;
 }
 Buscar_Datos(nom.trim(),1);
 if(t12.getText().trim().length()==0){return;}
 Publica = extractor(lin);//Extracta de la clave publica los valores numéricos de esta
 //separa la función hash encriptada y el texto
 ENHASH = arch.desconcatenar(nuevo,this);
```

```
//desencripción de la función hash
 ENHASH =
ENCIFRA.calculo(ENHASH,(java.math.BigInteger)Publica.elementAt(1),(java.math.BigInteger)Publica.elementAt(0));//enc a
hash
 hash = (java.math.BigInteger)P.iopRMD(nuevo);//genero la nueva función hash del resumen obtenido

 //comparacion de los 2 resúmenes hash
 if(ENHASH.equals(hash))
 {//mensaje
 errores_archivo err = new errores_archivo(this," Firma digital correcta  ","¡ Archivo sin vio lar, FIRMA EXACTA !");
 try
  {
              //Se abre el archivo de forma lectura
                        FileInputStream publo  = new FileInputStream(nuevo);
                        DataInputStream ipu    = new DataInputStream(publo);
                        t3.setText(null);
                        t3.setEditable(true);
                        String lin;//se leerá por líneas
                        this.setTitle("Firma Digital - "+nuevo);

                         while ((lin = ipu.readLine()) != null)//mientras hayan líneas por leer
                    {
                        t3.appendText(lin+"\n");//actualice el área de texto
                    }
                    ipu.close();//cierre el archivo
                        }
                        catch (IOException e)//si no se pudo abrir envíe mensaje de error
                         {
                        errores_archivo err5 = new errores_archivo(this," ¡ Error 280 Archivo Invalido !","Tenga cuidado con los
archivos");
                        err5.pack();
                        err5.show();
                        t3.setText(null);
                        t3.setEditable(false);
                        return;
                        }
  err.pack();
  err.show();
 }else{//mensaje
                errores_archivo err = new errores_archivo(this,"! Función Hash ! ","¡ Archivo Violado, Firma Incorrecta !");
                err.pack();
                err.show();
                return;
                }
 }catch(Exception e)
   {
              errores_archivo err = new errores_archivo(this,"!   OJO   Función Hash ! ","¡ Archivo Violado !");
              err.pack();
              err.show();
   }

return;
}
/**
 * Busca en la base de datos y dependieron el tipo de tabla a accesar
 *  1 -Publica
 *  2 - privada
 * Llama a recibir datos con los datos actualizados de la tabla
 */


public void Buscar_Datos(String Ident,int tipo)
{
 try{

              //se asigna a una variable la base de datos a utilizar manejador:protocolo:base de datos
              String url1 = "jdbc:odbc:FIRMA_DIGITAL";
              //se carga el controlador de base de datos de sun
              Class.forName("com.ms.jdbc.odbc.JdbcOdbcDriver");
              //establece la conexión con la Base de datos
              java.sql.Connection  conexión = java.sql.DriverManager.getConnection(url1);
              //declara las sentencias sal
              java.sql.Statement instrucción = conexion.createStatement();
              String sql;

              if(tipo==1)
              { sql = "SELECT * FROM Publica WHERE Identificador = '"+Ident+"'";}
              else
               { sql = "SELECT * FROM Privada WHERE Identificador = '"+Ident+"'";}
```

```
            java.sql.ResultSet rs = instruccio n.executeQuery(sql);

            if(rs.next())
             {
              String datos[] = new String[3];
                if(tipo==1){
                datos[0] = rs.getString(1);
                datos[1] = rs.getString(2);
                datos[2] = rs.getString(3);
                          conexion.close();
                recibir_publicos(datos);
                 }else{
                                                datos[1] = rs.getString(1);
                                                datos[2] = rs.getString(2);
                                                datos[0] = rs.getString(4);
                                                conexion.close();
                                                recibir_privados(datos,true);
                    }

            }else{
                          errores_archivo err = new errores_archivo(this,"¡ CUIDADO !","! No existe Identificador '"+Ident+"' en la
En el acceso a la base de datos !");
                  err.pack();
                  err.show();
                          rs.close();
                          conexion.close();
                          return;
                    }
            conexion.close();

            }catch(Exception e){
                errores_archivo err = new errores_archivo(this,"¡ Error Al accesar Base de datos!","! IMPOSIBLE ACCESAR  LA
BASE DE DATOS FIRMA_DIGITAL!");
                err.pack();
                err.show();
                        return;

                              }
return;
}

/*
proceso de desencripción
**/
public void desencripcion()throws Exception

{

 //se crea un objeto para generar la función hash de comparación
 RMD160 P = new RMD160();
//se crea un objeto donde se encuentran los métodos de desencripción
 clave ENCIFRA = new clave();
 //Objeto para manejar los archivos
 archivos arch = new archivos();
 java.math.BigInteger KSE;//se guardara la clave de sesión encriptada
 java.math.BigInteger KS;//se guardara la clave de sesión desencriptada
 java.math.BigInteger ENHASH;//se guardara la función hash encriptada
 java.math.BigInteger hash;//se guardara la función hash desencriptada
//Se guardaran los valores numéricos de la clave privada del receptor
java.math.BigInteger Secret[] = new java.math.BigInteger[7];
//Se guardaran los valores numéricos  de la clave privada del emisor
 Vector Publica = new Vector(2,1);

 String nuevo  = descomprimir(original);
 if(nuevo==null){return;}

//Se duplica el archivo para trabajar sobre el duplicado
 //String nuevo = original.substring(0,original.length()-4);
 //arch.duplicar(original,nuevo,this);

 //Se separa el identificador del archivo
 java.math.BigInteger j = new java.math.BigInteger("0");
 j = arch.desconcatenar(nuevo,this);
 byte Ident2[] = j.toByteArray();
 String nom = new String();
 int cont = 0,num = Ident2.length;
 while(cont<num)
 {
 nom = nom + String.valueOf((char)Ident2[cont]);
 cont++;
 }
```

```
 Buscar_Datos(nom.trim(),2);//
 if(t12.getText().trim().length()==0){return;}

  //Se abren los datos del receptor
 //****************

 try{
    Abrir_Priv(Secret);//Desencripta el archivo de clave privada
  }catch(Exception e)
   {
    errores_archivo err = new errores_archivo(this,"¡ Error 220 !","! Clave Privada Invalida !");
                err.pack();
                err.show();
                return;
    }


 //Se abren los archivos de clave publica y privada respectivamente
 //separa la clave de sesión encriptada y el texto encriptado con esta
 try{
 KSE = arch.desconcatenar(nuevo,this);

//**Sé desencripta la KS con la clave privada correspondiente
  //escoge con cual clave privada sé desencripta
 int i = escoger_clave(KSE,Secret[5],Secret[6]);
 //Sé desencripta la clave de sesión
 KS = ENCIFRA.calculo(KSE,Secret[i],Secret[0]);
 //desencripta archivo con la KS//
 arch.ENCIDEA(KS,nuevo,this,2);

//prueba de duplicado
 arch.duplicar(nuevo,"resultado.txt",this);
 //*/*/*/*/*/*//*/*/

 //separa la función hash encriptada y el texto del id de usuario
 //              separa el identificador del archivo
 java.math.BigInteger j1 = new java.math.BigInteger("0");
 j1 = arch.desconcatenar(nuevo,this);
 System.out.println(j1);
 byte Ident5[] = j1.toByteArray();
 String nom5 = new String();
 int cont5 = 0,num5 = Ident5.length;
 while(cont5<num5)
 {
 nom5 = nom5 + String.valueOf((char)Ident5[cont5]);
 cont5++;
 }
 Buscar_Datos(nom5.trim(),1);
if(t12.getText().trim().length()==0){return;}
 //se han separado el identificador de datos públicos
 Publica = extractor(lin);//Extracta de la clave publica los valores numéricos de esta
 //separa la función hash encriptada y el texto
 ENHASH = arch.desconcatenar(nuevo,this);
//desencripción de la función hash
ENHASH =
ENCIFRA.calculo(ENHASH,(java.math.BigInteger)Publica.elementAt(1),(java.math.BigInteger)Publica.elementAt(0));//enc a
hash
hash = (java.math.BigInteger)P.iopRMD(nuevo);//genero la nueva función hash del resumen obtenido
//comparacion de los 2 resúmenes hash
 if(ENHASH.equals(hash))
 {//mensaje
 errores_archivo err = new errores_archivo(this," Firma Digital Correcta  ","¡ Archivo segura, MENSAJE CONFIABLE !");
 try
  {
            //Se abre el archivo de forma lectura
                        FileInputStream publo  = new FileInputStream(nuevo);
                        DataInputStream ipu    = new DataInputStream(publo);
                        t3.setEditable(true);
                        t3.setText(null);
                        String lin;//se leerá por líneas
                        this.setTitle("Firma Digital - "+nuevo);
                         while ((lin = ipu.readLine()) != null)//mientras hayan líneas por leer
                {
                        t3.appendText(lin+"\n");//actualice el área de texto
                }
                ipu.close();//cierre el archivo
                        }
                        catch (IOException e)//si no se pudo abrir envíe mensaje de error
                         {
                        errores_archivo err5 = new errores_archivo(this," ¡ Error 280 Archivo Invalido !","Tenga cuidado con los
archivos");
```

```
                        err5.pack();
                        err5.show();
                        t3.setText(null);
                        t3.setEditable(false);
                        }
err.pack();
 err.show();
 }else{//mensaje
          errores_archivo err = new errores_archivo(this,"! Función Hash ! ","¡ Función Hash Incorrecta !");
          err.pack();
          err.show();
          return;
          }
 }catch(Exception e)
   {
          errores_archivo err = new errores_archivo(this,"!   OJO   Función Hash ! ","¡ Archivo Violado !");
          err.pack();
          err.show();
   }

return;
}
/*********************************************************
método utilizado para desencriptar las claves privadas Lucas
que fue encriptada con una clave idea,
recibe la clave privada Idea y envía las claves secretas Lucas
*************************************************************/
public void Abrir_Priv(java.math.BigInteger secret[])throws Exception
{
File privad;
FileInputStream p;
DataInputStream z;
FileOutputStream ar;
DataOutputStream SS;
FileInputStream privada;
DataInputStream os;
try
{
 privad       = new File(t11.getText().trim());
 p = new FileInputStream(t11.getText().trim());
 z =          new DataInputStream(p);
 Vector                    PO  = new Vector();
 java.math.BigInteger  enviar[] = new java.math.BigInteger[4];
for(int g=0;g<4;g++){enviar[g] = new java.math.BigInteger("0");}
//crea un objeto de la clase Idea para luego usar sus métodos
Idea Z= new Idea();
int l;
int R=0;
//creamos un archivo temporal donde se guardara momentáneamente
//la clave Lucas desencriptada
ar = new FileOutputStream("temporal4.FDO");
SS = new DataOutputStream(ar);
// se captura la longitud de la clave privada
long F = privad.length();
long count = 0;
//toma la clave privada digitada y la convierte a numero
CPR = ext(t12.getText().trim());
// se le asigna a una variable auxiliar al valor obtenido
java.math.BigInteger CPRAUX = CPR;
java.math.BigInteger CPRAUX1 = new java.math.BigInteger("0");
//contiene la clave privada idea a ser enviada
CPRAUX1 = (java.math.BigInteger)CPRAUX;
while(count < (F/2))
   {
          //lee un carácter de la clave privada
          l = z.readChar();
          //se almacena en enviar[] en la posición R con su respectivo valor numérico
          enviar[R] = java.math.BigInteger.valueOf(l);
          R++;
          if(R==4)
            {
                    try{
                        PO = Z.iop1(enviar,CPRAUX1);//sé desencripta la clave privada
                    }catch(Exception e)
                    {
                    SS.close();
                    z.close();
                    throw  e;
                    }
            // se escribe en un archivo temporal
            SS.writeChar(((java.math.BigInteger)PO.elementAt(0)).intValue());
```

```
                SS.writeChar(((java.math.BigInteger)PO.elementAt(1)).intValue());
                SS.writeChar(((java.math.BigInteger)PO.elementAt(2)).intValue());
                SS.writeChar(((java.math.BigInteger)PO.elementAt(3)).intValue());
                R=0;
               }
              count++;
   }
SS.close();
z.close();
//del archivo temporal obtenemos los valores que se necesitan N,K1;K2,K3 y K4
//donde Ki i=1..4 son las claves privadas
privada = new FileInputStream("temporal4.FDO");
          os = new DataInputStream(privada);
int j = 0;
// lee los bytes de N
j = os.readInt();
if(j<=0||j>1000){os.close();throw new Exception();
}//evito Error del compilador, si la clave es invalida

//y lee un data erróneo
byte Arr[]= new byte[j];
//lee el valor de N desde la posición 0 hasta la j
os.read(Arr,0,j);
//convierte el valor de N a un entero grande
secret[0]= new java.math.BigInteger(Arr);

//de igual forma como en N se hacen con los demás...
j=os.readInt();
if(j<=0||j>1000){os.close();throw new Exception();}
byte Arr1[]= new byte[j];
os.read(Arr1,0,j);
secret[1]= new java.math.BigInteger(Arr1);

j = os.readInt();
if(j<=0||j>1000){os.close();throw new Exception();}
byte Arr2[]= new byte[j];
os.read(Arr2,0,j);
secret[2]= new java.math.BigInteger(Arr2);

j=os.readInt();
if(j<=0||j>1000){os.close();throw new Exception();
}
byte Arr3[]= new byte[j];
os.read(Arr3,0,j);
secret[3]= new java.math.BigInteger(Arr3);

j = os.readInt();
if(j<=0||j>1000){os.close();throw new Exception();}
byte Arr4[]= new byte[j];
os.read(Arr4,0,j);
secret[4]= new java.math.BigInteger(Arr4);

j = os.readInt();
if(j<=0||j>1000){os.close();throw new Exception();}
byte Arr5[]= new byte[j];
os.read(Arr5,0,j);
secret[5]= new java.math.BigInteger(Arr5);

j = os.readInt();
if(j<=0||j>1000){os.close();throw new Exception();}
byte Arr6[]= new byte[j];
os.read(Arr6,0,j);
secret[6]= new java.math.BigInteger(Arr6);
os.close();

// son siete valores por que aquí se encuentran los valores de p y q.
}
catch(IOException e)//si ocurre un error de archivo envíe una excepción
  {
  errores_archivo err = new errores_archivo(this," ¡ Error 250 Archivo Invalido !","Tenga cuidado con los archivos");
  err.pack();
  err.show();
  return;
  }
 finally {
                              File borra = new File("temporal4.FDO");
                              borra.delete();
                             }

}//public void crear_archivo(java.math.BigInteger N ,java.math.BigInteger valor Ku)
```

```
/***********************************************/
public void Guardarcomo()
{
 //método que genera el cuadro de dialogo de guardar como...
 Dimension tam, tam_di;              //dimensión de la pantalla indep del sistema
 java.awt.FileDialog fd = new java.awt.FileDialog(this,"Guardar Como ",1);//análogo a abrir como 0 - 1
 fd.setFile("Firma.txt");//por defecto
 tam    =    Toolkit.getDefaultToolkit().getScreenSize();
 tam_di =  fd.size();
 fd.move((tam.width-tam_di.width)/5,(tam.height-tam_di.height)/3);
 fd.show();
 //contendrá el nombre del archivo con el cual se va a guardar
 original = fd.getDirectory();
 if(original != null) // si es diferente de nulo se envía mensaje de error
 {
  //los valores obtenidos de la dirección del archivo lo guardo en original
  original = (String)(fd.getDirectory())+(String)(fd.getFile());
  try
   {
  // guarde lo que hay el área de texto en el archivo
   PrintStream QQ = new PrintStream(new FileOutputStream(original));
   QQ.println(t3.getText());
   QQ.close();
  }
  catch (IOException e)
                {
     errores_archivo err = new errores_archivo(this,"¡ Error 260  Archivo Invalido !"," no se puede guardar el archivo");
      err.pack();
      err.show();
                 return;
                }
 }
return;
}
//*********** Guardar  ****************************/
public void guardar(String Nom_Arc)
{ //guarda el archivo abierto
  try
   {
    //se abre en forma de escritura y se actualiza con lo nuevo del área de texto
    PrintStream QQ = new PrintStream(new FileOutputStream(Nom_Arc));
    QQ.println(t3.getText());
    QQ.close();
   }
  catch (IOException e)
              {//si no lo puede guardar envía error
  errores_archivo err = new errores_archivo(this,"Error 270, Archivo Invalido!","No se puede guardar");
  err.pack();
  err.show();
  return;
                }
return ;
}



/***********************************************/
public void claves_publicas()
{
  acceso_publicas  hjo = new acceso_publicas(this);
  hjo.inicio(hjo);
  return;
 }
/***********************************************/
public void claves_privadas()
{
                 accesar_privada hh = new accesar_privada();
                hh.inicio(hh,this);
return;
}
/***********************************************/
public void Abrir(String mensaje,boolean re)
{//abre un archivo y su contenido lo guarda en el textarea
 //se guardaran coordenadas de la pantalla y de la nueva ventana
 Dimension tam, tam_di;
 //cree un dialogo abrir
 java.awt.FileDialog fd = new java.awt.FileDialog(this,mensaje,0);
 fd.setFile("*.Mss;*.Dso;*.txt;*.Doc");//muestre por defecto los de extensión ...
 tam    =    Toolkit.getDefaultToolkit().getScreenSize();//resolución del monitor
 tam_di =  fd.size();//tamaño de la ventana
 fd.move((tam.width-tam_di.width)/5,(tam.height-tam_di.height)/3);//centre la nueva ventana
```

```
 fd.show();//muestre la ventana
 //análogo a guardar como . . .
 original = fd.getDirectory();
 //si se devolvió algún directorio
 if(original != null)
 {
  original = (String)(fd.getDirectory())+(String)(fd.getFile());
  //se guarda en original la dirección del archivo
  if(re==false)
  {
  try
  {
                //Se abre el archivo de forma lectura
    FileInputStream publo  = new FileInputStream(original);
    DataInputStream ipu    = new DataInputStream(publo);
                t3.setEditable(true);
    String lin;//se leerá por líneas
                this.setTitle("Firma Digital - "+original);

                 while ((lin = ipu.readLine()) != null)//mientras hayan líneas por leer
                    {
                                t3.appendText(lin+"\n");//actualice el área de texto
                    }

                    ipu.close();//cierre el archivo
  }
  catch (IOException e)//si no se pudo abrir envíe mensaje de error
                {
     errores_archivo err = new errores_archivo(this," ¡ Error 280 Archivo Invalido !","Tenga cuidado con los archivos");
      err.pack();
      err.show();
                 t3.setText(null);
                 t3.setEditable(false);
                 return;
                }
  }
 }
return ;
}
```

```
//aquí termina la pagina anterior
//************************************************
// método que recibe información del cuadro  de dialogo que despliega la clase "archivo"
//el cual le manda la información que el usuario describe

public void recibirinfo(boolean K)
{
 if(K==true)
 {
    if (original==null)
    {
     Guardarcomo();
    }else {
          guardar(original);
          }
 }
 return;
 }
//******************calcula la clave de sesion
//Importante:  se tomo como semilla la hora en milisegundos para generar numeros aleatorios utilizando
//el estandar ANSI x9.17 para crear numetros aleatorios pasando  tres veces por el cifrador IDEA.
public java.math.BigInteger CS()
{

//la variable ahora guardara la hora actual del sistema
Date ahora = new Date(); // luego se convierte en milisegundos
long  mil = ahora.getTime(); //se realiza esta operación para variar el resultado
mil = (mil*mil)/3; // el resultado es tomado como semilla para la función aleatoria Random para generar números aleatorios
//Random Hor = new Random(mil); java.math.BigInteger CL1 =new java .math.BigInteger(128,0,Hor); //toma 128 bits a part ir de
//la posición 0 de la variable Hor

java.math.BigInteger CL = new java.math.BigInteger("0");
java.math.BigInteger CL5 = new java .math.BigInteger(64,0,Hor);
```

```
java.math.BigInteger a1 = CL5.and(java.math.BigInteger.valueOf(65535));
java.math.BigInteger a2 = (CL5.shiftRight(16)).and(java.math.BigInteger.valueOf(65535));
java.math.BigInteger a3 = (CL5.shiftRight(32)).and(java.math.BigInteger.valueOf(65535));
java.math.BigInteger a4 = (CL5.shiftRight(48)).and(java.math.BigInteger.valueOf(65535));
java.math.BigInteger H[] = { a1,a2,a3,a4 };
Idea gene = new Idea();

try{


Vector clave = gene.iop(H,CL1);
java.math.BigInteger  a = (java.math.BigInteger)clave.elementAt(0);
java.math.BigInteger  b = (java.math.BigInteger)clave.elementAt(1);
java.math.BigInteger  c = (java.math.BigInteger)clave.elementAt(2);
java.math.BigInteger  d = (java.math.BigInteger)clave.elementAt(3);

java.math.BigInteger nueva = a;
nueva = nueva.shiftLeft(16).or(b);
nueva = nueva.shiftLeft(16).or(c);
nueva = nueva.shiftLeft(16).or(d);
ahora =new Date();

Hor = new Random(ahora.getTime());
java.math.BigInteger CL6 = new java .math.BigInteger(64,0,Hor);
java.math.BigInteger ult = nueva.xor(CL6);
a1 = ult.and(java.math.BigInteger.valueOf(65535));
a2 = (ult.shiftRight(16)).and(java.math.BigInteger.valueOf(65535));
a3 = (ult.shiftRight(32)).and(java.math.BigInteger.valueOf(65535));
a4 = (ult.shiftRight(48)).and(java.math.BigInteger.valueOf(65535));

java.math.BigInteger H12[] = { a1,a2,a3,a4 };
java.util.Vector clave1 = new java.util.Vector();
clave1 = gene.iop(H12,CL1);
a = (java.math.BigInteger)clave1.elementAt(0);
b = (java.math.BigInteger)clave1.elementAt(1);
c = (java.math.BigInteger)clave1.elementAt(2);
d = (java.math.BigInteger)clave1.elementAt(3);




java.math.BigInteger nueva1 = a;
nueva1 = nueva1.shiftLeft(16).or(b);
nueva1 = nueva1.shiftLeft(16).or(c);
nueva1 = nueva1.shiftLeft(16).or(d);
nueva = nueva1.xor(nueva);



a1 = nueva.and(java.math.BigInteger.valueOf(65535));
a2 = (nueva.shiftRight(16)).and(java.math.BigInteger.valueOf(65535));
a3 = (nueva.shiftRight(32)).and(java.math.BigInteger.valueOf(65535));
a4 = (nueva.shiftRight(48)).and(java.math.BigInteger.valueOf(65535));

java.math.BigInteger H13[] = { a1,a2,a3,a4 };

Vector salida = gene.iop(H13,CL1);

a = (java.math.BigInteger)salida.elementAt(0);
b = (java.math.BigInteger)salida.elementAt(1);
c = (java.math.BigInteger)salida.elementAt(2);
d = (java.math.BigInteger)salida.elementAt(3);

java.math.BigInteger numsal = a;
numsal = numsal.shiftLeft(16).or(b);
numsal = numsal.shiftLeft(16).or(c);
numsal = numsal.shiftLeft(16).or(d);

            CL = nueva1;
            CL =CL.shiftLeft(64).or(numsal);
            return CL;

}catch(Exception e){}

return CL;
}


java.math.BigInteger ext(String M)
{
```

```
 int k = M.length();//toma el tamaño de la cadena recibida
 int y = M.indexOf('.');//se toma en que posición de la cadena esta el carácter "."

 char m[] = M.toCharArray();//se convierten en caracteres los valores de M
 java.math.BigInteger N = new java.math.BigInteger("0");//se guardara provisionalmente N
 java.math.BigInteger L = new java.math.BigInteger("0");//se guardara provisionalmente L

//ojo

int i=0;
for(i=y -1;i>=0;i--)
 {
  // a N se le realiza un corrimiento a la izquierda y se actualiza N
  N = N.shiftLeft(6);
  // se la hace un or con el valor existente en DESCORRES(m[i])
  N = N.or(java.math.BigInteger.valueOf(DESCORRES(m[i])) );
 }
 for(i=k -1; i>y ;i--)
 {
//a L se la hace un corrimiento a la izquierda de 6 posiciones
  L = L.shiftLeft(6);
  // se la hace un or exclusivo con el valor en DESCORRES(M[i]()
  L = L.or(java.math.BigInteger.valueOf(DESCORRES(m[i])) );
 }

return L;//retorna la clave privada en un número grande
}

                public static class ClassInfo extends com.ms.wfc.core.ClassInfo
                {
                                public void getEvents(IEvents events)
                                {
                                                super.getEvents(events) ;
                                }
//aqui empieza la siguiente pagina

                                public void getProperties(IProperties props)
                                {
                                                super.getProperties(props);
                                }
                }
}//final de la clase inicial

//clase       */
//**************** nuevas clase diálogos ************************
/*
Esta clase tiene la tarea de desplegar un cuadro de dialogo, donde el usuario
puede elegir la opciones si,no,cancelar, y luego este establece una conexión con
el objeto que lo llamo  y le devuelve la respuesta del usuario
*/


/**
 * @com.register ( clsid=6C90B107-7976-11D3-9623-444553540001, typelib=6C90B105-7976-11D3-9623-444553540001 )
 */
class archivo extends Dialog
{
 public archivo(Frame padre,String titulo,String mensaje_error)
 {

  super(padre,titulo,true);
  java.awt.Panel info    = new java.awt.Panel();
  java.awt.Panel aceptar = new java.awt.Panel();

  info.setLayout(new GridLayout(3,1));
  info.add(new java.awt.Label("Firma Digital . . .",java.awt.Label.CENTER ));
  info.add(new java.awt.Label(mensaje_error,java.awt.Label.CENTER ));
  info.add(new java.awt.Label(" -LUCAS- 1999 ",java.awt.Label.CENTER ));
  add("North",info);

  aceptar.setLayout(new GridLayout(1,2));
  aceptar.add(new java.awt.Button(" Sí "));
  aceptar.add(new java.awt.Button(" No "));
  add("South",aceptar);
  }             //err

 public boolean action(java.awt.Event evt, Object arg)
 {
   if (evt.target instanceof java.awt.Button)
   {
```

```
                if(arg.equals(" Sí "))
                {
                 ((ENCRYDESENCR)getParent()).recibirinfo(true);
                 this.dispose();
                 return true;
                }
                else//(if(arg.equals(" No ")))
                {
               //llama al objeto que lo llamo y le devuelve la información que tecleo el usuario
               ((ENCRYDESENCR)getParent()).recibirinfo(false);
                this.dispose();
                return true;
                }
               }
  return false;
 } //boolean
}//clase     */


 //**************** nuevas clase diálogos  ************************

/**
Despliega un cuadro de dialogo de los errores que se cometen
recibe
Frame                frame que lo llama
String               titulo
mensaje_error        mensaje que se desplegara

@com.register ( clsid=6C90B10F-7976-11D3-9623-444553540001, typelib=6C90B105-7976-11D3-9623-444553540001 )
*/
class errores_archivo extends Dialog
{
 public errores_archivo(Frame padre,String titulo,String mensaje_error)
 {
  super(padre,titulo,true);
  java.awt.Panel info    = new java.awt.Panel();
  java.awt.Panel aceptar = new java.awt.Panel();

  info.setLayout(new GridLayout(3,1));
  info.add(new java.awt.Label("Firma Digital . . .",java.awt.Label.CENTER ));
  info.add(new java.awt.Label(mensaje_error,java.awt.Label.CENTER ));
  info.add(new java.awt.Label(" LUCAS 1999 ",java.awt.Label.CENTER ));
  add("North",info);

  aceptar.add(new java.awt.Button("Aceptar"));
  add("South",aceptar);
  }            //err
//*******************************************************
 public boolean action(java.awt.Event evt, Object arg)
 {
   if (evt.target instanceof java.awt.Button)
   {
               this.dispose();
               return true;
   }
   return false;
 } //boolean

}//clase     */




//**************** nuevas clase diálogos ************************
/**
Muestra el cuadro de dialogo versión
llama una imagen donde esta información
@com.register ( clsid=6C90B108-7976-11D3-9623-444553540001, typelib=6C90B105-7976-11D3-9623-444553540001 )
*/
class autores extends Frame
 {

  boolean seterror;
  java.awt.Image img ;
  java.awt.Graphics h;
  Dimension tam;
  public autores()
```

```
   {
   super(" Versión ");

   img =  Toolkit.getDefaultToolkit().getImage("version.gif");
   MediaTracker tracker = new MediaTracker(this);
   tracker.addImage(img,0);
   try{
   tracker.waitForAll();
      }
   catch(InterruptedException e)
      {
                errores_archivo err = new errores_archivo(this,"¡ Error 290 !","!Falta Boton Gráfico!");
                err.pack();
                err.show();
                return;
      }
   setResizable(false);
   repaint();
   }

//**********************************************
public boolean handleEvent(java.awt.Event evt)
 {
  if(evt.id == java.awt.Event.WINDOW_DESTROY)
  {
   this.dispose();
   return true;
  }

  return super.handleEvent(evt);
 }
//*********************************************

public void paint(java.awt.Graphics g)
{
 tam = this.size();
   int x = tam.width;
   int y = tam.height;
 g.drawImage(img,0,0,x+1,y+1,this);
}

}//class
/**
Despliega una ventana con la ayuda de la aplicación
crea un text área y lee la información del archivo para
mostrarla en el text área
@com.register ( clsid=6C90B10A-7976-11D3-9623-444553540001, typelib=6C90B105-7976-11D3-9623-444553540001 )
*/
class ayuda                extends Frame
{
  TextArea J = new TextArea(30,60);
  java.awt.Image Icono;


public ayuda()
 {
  super(" Ayuda ");
  Icono      = Toolkit.getDefaultToolkit().getImage("icono.gif");
  MediaTracker tracker = new MediaTracker(this);
  tracker.addImage(Icono,0);
  try{
  tracker.waitForAll();
  }
  catch(InterruptedException e)
  {
  errores_archivo err = new errores_archivo(this,"¡ Error 300 ¡","! Falta Boton Gráfico !");
  err.pack();
  err.show();
  return;
  }
   //***********************
  this.setIconImage(Icono);
  //***********************

  setLayout(  new BorderLayout() );
  add("North", new java.awt.Label("  ")    );
  add("South", new java.awt.Label("  ")    );
  add("East",  new java.awt.Label("  ")    );
  add("West",  new java.awt.Label("  ")    );

  java.awt.Panel centro = new java.awt.Panel();
```

```
 java.awt.Panel centro1 = new java.awt.Panel();

 centro1.add(J);
 centro.setLayout(new BorderLayout());
 centro.add("Center",centro1);
 add("Center",centro);
 sacarInfo();
 J.setEditable(false);
}



public void sacarInfo()
{
            try
            {
              FileInputStream publo  = new FileInputStream("MANUAL_ENCRYDESENCR.txt");
    DataInputStream ipu    = new DataInputStream(publo);
    String lan;
             while ((lan = ipu.readLine()) != null)
               {
                        J.append("\n"+lan);
                        }

               ipu.close();
             }
             catch (IOException e)
                        {
                         errores_archivo err = new errores_archivo(this,"¡ Error 310 !","! No se encuentra
MANUAL_ENCRYDESENCR.txt !");
                          err.pack();
                          err.show();
                          return;

                        }
}

public boolean handleEvent(java.awt.Event evt)
 {
   if(evt.id == java.awt.Event.WINDOW_DESTROY)
   {
            this.dispose();
            return true;
   }
  return super.handleEvent(evt);
 }

}//class
/**
 * @com.register ( clsid=B6BD3FB2-96D6-11D3-9630-A07750C10000, typelib=6C90B105-7976-11D3-9623-444553540001 )
 */
class recibir_desi extends Dialog
{
 String clave;
TextField xx = new TextField(30);
 ENCRYDESENCR padre1;
recibir_desi(ENCRYDESENCR padre,String a)
 {
  super(padre,"Error Encontrado",true);
   clave=a;
  padre1=padre;
  java.awt.Button a1 = new java.awt.Button(" Aceptar ");
  java.awt.Button a2 = new java.awt.Button(" Cancelar ");
  setLayout(new GridLayout(2,2));
  add(new java.awt.Label("¿ DESEA BORRAR ",2));
  add(new java.awt.Label(" EL EXISTENTE ?",0));
  add(a1);
  add(a2);
  }

public boolean action(java.awt.Event evt, Object arg)
 {
  if (evt.target instanceof java.awt.Button)
  {
            if(arg.equals(" Aceptar "))
             {
             padre1.rec(true);
             this.dispose();
             return true;
             }
```

```
                else if(arg.equals(" Cancelar "))
                 {
                        padre1.rec(false);
                        this.dispose();
                        return true;
  }
 }//if
  return false;
 }//public boolean
public boolean handleEvent(java.awt.Event evt)
  {
    if(evt.id == java.awt.Event.WINDOW_DESTROY)
    {
              this.dispose();
              return true;
    }
   return super.handleEvent(evt);
  }

}//class



/**
 * @com.register ( clsid=6C90B114-7976-11D3-9623-444553540001, typelib=6C90B105-7976-11D3-9623-444553540001 )
 */
public class archivos
{
public void concatenar(String  archivo,java.math.BigInteger Valor,Frame padre)
 {
//realiza la función de concatenación de un archivo con un valor numérico

  byte NUM[] = Valor.toByteArray();//convierte el valor a un arreglo de bytes
      int y = NUM.length;//toma él numero de bytes
    byte n[] = new byte[1];           //crea una variable byte cuyo  tamaño es 1
             //
  try{
  File A = new File(archivo);//crea una variable archivo
int  x = (int)A.length();//guarda el tamaño del archivo que se va a concatenar
  //abre un archivo de solo lectura
  RandomAccessFile P = new RandomAccessFile(archivo,"r");
  //crea un archivo temporal de lectura escritura
  RandomAccessFile N = new RandomAccessFile("Temp.tem","rw");
  /**
  IMPORTANTE
  ESTE METODO CONCATENA:
  1. EL NUMERO DE BYTES DEL VALOR NUMERICO QUE SE VA A CONCATENAR
  2.EL TAMAÑO DE LA LONGITUD DEL ARCHIVO A CONCATENAR
  3.EL VALOR NUMERICO
  4.EL ARCHIVO QUE SE VA A CONCATENAR
  */
  N.writeInt(y);//
  N.writeInt(x);
  N.write(NUM);
  int tam = 0;
  while( tam < x )
     {
    P.read(n);
     N.write(n);
    tam++;
     }
  //cierra los archivos y elimina al temporal
  N.close();
  P.close();

  File K = new File("Temp.tem");
  A.delete();
  K.renameTo(A);
     }
  catch(IOException e)
  {
   errores_archivo1 err = new errores_archivo1(padre,"¡ Error 430, Archivo Invalido !","¡ Tenga cuidado con los archivos !");
    err.pack();
    err.show();
  }
  return;
 }

 //**********************************************
/**
```

```
 Método utilizado para desconcatenar el archivo y el valor numérico  ,tiene en cuenta como es concatenado por el método
concatenar  - su funcionamiento es inverso a concatenar,    Devuelve el primer valor numérico concatenado
   y el archivo sin el valor que estaba concatenado
 */
 public java.math.BigInteger desconcatenar(String archivo,Frame padre)throws Exception
 {
  int val,val1;
  java.math.BigInteger Valor = new java.math.BigInteger("0");
  File K = new File("Temp.tem");
  File A = new File(archivo);

  try{

  RandomAccessFile P = new RandomAccessFile(archivo,"r");
  RandomAccessFile N = new RandomAccessFile("Temp.tem","rw");
  val  = P.readInt();
  val1 = P.readInt();

  if( val<0 || val>500000)
  {
  N.close();
 P.close();
  K.delete();
 throw new Exception();
  }

  if( val1<0 || val1>500000)
  {
  N.close();
 P.close();
  K.delete();
 throw new Exception();
  }

  byte L[] = new byte[val];
 P.read(L);
  Valor = new java.math.BigInteger(1,L);
  byte n[] = new byte[1];
  int tam = 0;
  while( tam < val1 )
     {
    P.read(n);
     N.write(n);
    tam++;
     }
  N.close();
 P.close();
  A.delete();
  K.renameTo(A);
     }
  catch(IOException e)
  {
   errores_archivo1 err = new errores_archivo1(padre,"¡ Error  440, Archivo Invalido !","¡ Tenga cuidado con los archivos !");
   err.pack();
   err.show();
  }
  return Valor;
 }

//********************** duplica un archivo1 en 2
/**
 Este método recibe 2 archivos
 - archivo1 es el archivo que se va a duplicar
 - archivo2 es el nombre del archivo en el cual se duplicará.
 */

 public void duplicar(String  archivo1,String archivo2,Frame padre)
 {
  //revisa si archivo2  existe y lo elimina
  File ar = new File(archivo2);
  ar.delete();
  //duplica el archivo
  try{

  RandomAccessFile P = new RandomAccessFile(archivo1,"r");
  RandomAccessFile N = new RandomAccessFile(archivo2,"rw");
  byte n[] = new byte[1];
  while( P.read(n) != -1)
     {
    N.write(n);
     }
```

```
 N.close();
 P.close();
    }
 catch(IOException e)
 {
  errores_archivo1 err = new errores_archivo1(padre,"¡ Error 450, Archivo Invalido !","¡ Tenga cuidado con los archivos !");
  err.pack();
  err.show();
 }
 return;
}
/**
Encripta o Desencripta un archivo con la clave de sesión idea.
*/


//********************** Encripta un archivo con idea*****************/
public void ENCIDEA(java.math.BigInteger CS,String archivo,Frame padre,int fa)throws Exception
 {
//la variable fa Define : 1 encripción    2 desencripción
  try{
        Vector  REC   = new Vector(4,1);
  //archivo a encriptar
  RandomAccessFile z    = new RandomAccessFile(archivo,"r");
  //archivo temporal
  RandomAccessFile tem = new RandomAccessFile("Temp.tem","rw");
  byte n[] = new byte[2];//de2 posiciones para leer 2 bytes
  java.math.BigInteger enviar[] = new java.math.BigInteger[4];
  //guarda la información que enviara a encriptar en un momento determinado
  int J=0;
              //objeto idea donde se encuentra los métodos para encriptar o desencriptar
     Idea Z = new Idea();

 while(z.read(n)!=-1)//mientras no sea fin de archivo lea 2 bytes
  {
   enviar[J] = new java.math.BigInteger(1,n);//genera un valor positivo sin signo y prepárelo a enviarlo
   J++;
   if(J==4)//si se completo el arreglo  entonces tendrá 64 bits con los que trabaja IDEA
    {
if(fa==1)//si es encripción
    {
                        try {
                          REC = Z.iop(enviar,CS);//encriptelo
                                }catch(Exception e){
                                                                        z.close();
                                                                        tem.close();
                                                                        File K = new File("Temp.tem");
                                                                        K.delet e();
                                                                        throw new Exception();
                                                                    }
    }else{
                        try{
                           REC = Z.iop1(enviar,CS);
                           }catch(Exception e){
                                                                z.close();
                                                                tem.close();
                                                                File K = new File("Temp.tem");
                                                                K.delete();
                                                                throw new Exception();
                                                                }
              }//sino desencriptelo
    for(int g=0;g<4;g++){//coloca lo encriptado en un archivo temporal
    tem.writeChar( ((java.math.BigInteger)REC.elementAt(g)).intValue() );
                }
    for(int g=0;g<4;g++){enviar[g]=java.math.BigInteger.valueOf(0);}//coloca enviar a cero
    J=0;
    }
   }//while
 if(J!=0)//si quedan bytes sin encriptar repita el proceso por ultima vez
  {
    if(fa==1)
        {
                        try {
                            REC = Z.iop(enviar,CS);
                                }catch(Exception e){
                                                                        z.close();
                                                                        tem.close();
                                                                        File K = new File("Temp.tem");
                                                                        K.delete();
                                                                        throw new Exception();
                                                                    }
```

```
                    } else {
                                                try{
                                    REC = Z.iop1(enviar,CS);
                                                        }catch(Exception e){
                                                                                                                z.close();
                                                                                                                tem.close();
                                                                                                                File K = new File("Temp.tem");
                                                                                                                K.delete();
                                                                                                                throw new Exception();
                                                                                                    }

                    }
   for(int g=0;g<4;g++)
    {
    tem.writeChar(((java.math.BigInteger)REC.elementAt(g)).intValue() );
    }
  }
 //renombra el archivo y borra el anterior
  z.close();
  tem.close();

     }
  catch(IOException e)
  {
   errores_archivo1 err = new errores_archivo1(padre,"¡ Error 460, Archivo Invalido !","¡ Error de archivo, al encriptar !");
   err.pack();
   err.show();
  }
  finally
  {
  File K = new File("Temp.tem");
  File A = new File(archivo);
  A.delete();
  K.renameTo(A);
  }
  return;
 }

}//inicial



//**************** nuevas clase diálogos ************************
/**
 * @com.register ( clsid=6C90B118-7976-11D3-9623-444553540001, typelib=6C90B105-7976-11D3-9623-444553540001 )
 */
class errores_archivo1 extends Dialog
{
 public errores_archivo1(Frame padre,String titulo,String mensaje_error)
 {
  super(padre,titulo,true);
  Panel info     = new Panel();
  Panel aceptar = new Panel();

  info.setLayout(new GridLayout(3,1));
  info.add(new Label("Firma Digital . . .",Label.CENTER ));
  info.add(new Label(mensaje_error,Label.CENTER ));
  info.add(new Label(" LUCAS 1999 ",Label.CENTER ));
  add("North",info);

  aceptar.add(new Button("Aceptar"));
  add("South",aceptar);
  }             //err

  public boolean action(java.awt.Event evt,Object arg)
 {
   if (evt.target instanceof Button)
   {
                this.dispose();
                return true;
   }
   return false;
 } //boolean


}//clase     */

Contiene métodos utilizados en el proceso de ciframiento por medio de la secuencia de Lucas
*/
package paquete;
```

```
/**
 * @com.register ( clsid=6C90B117-7976-11D3-9623-444553540001, typelib=6C90B105-7976-11D3-9623-444553540001 )
 */
public class clave {

public java.math.BigInteger calculo(java.math.BigInteger P,java.math.BigInteger e,java.math.BigInteger N)
 {
 // Para calcular  Ve(P,1) modulo N , secuencia de Lucas
 java.math.BigInteger  D = new java.math.BigInteger("0");
 java.math.BigInteger ut = new java.math.BigInteger("0");
 java.math.BigInteger  u = new java.math.BigInteger("0");
 java.math.BigInteger  v = new java.math.BigInteger("0");
 java.math.BigInteger vt = new java.math.BigInteger("0");
 java.math.BigInteger  c = new java.math.BigInteger("0");

//Halla el determinante con el texto plano
 D = (P.multiply(P)).subtract(new java.math.BigInteger("4"));
 ut = new java.math.BigInteger ("1");
 vt = (java.math.BigInteger)P;
 u  = (java.math.BigInteger)ut;
 v  = (java.math.BigInteger)vt;

 if ((e.mod(new java.math.BigInteger("2"))).equals(new java.math.BigInteger("0")))
  {
  u = new java.math.BigInteger ("0");
  v = new java.math.BigInteger ("2");
  }
  e = e.divide(new java.math.BigInteger("2"));

    while( (e.compareTo( new java.math.BigInteger("0") ))== 1 )
     {
        ut = (ut.multiply(vt)).mod(N);
        vt = (vt.multiply(vt)).mod(N);

       if ( (vt.compareTo( new java.math.BigInteger("3") ) ) == -1)
            { vt = vt.add(N);}
             vt = vt.subtract(new java.math.BigInteger("2"));

        if ((e.mod(new java.math.BigInteger("2"))).equals(new java.math.BigInteger("1"))  )
         {
          c = (    (ut.multiply(v)) .add( u.multiply(vt))  ) .mod(N);
          v = (    (vt.multiply(v)).add( (D.multiply(u)).multiply(ut))   ) .mod(N);

                  if ((v.mod(new java.math.BigInteger("2"))).equals(new java.math.BigInteger("1")) )
                          { v = v.add(N);}
                    v = v.divide(new java.math.BigInteger("2"));
                 if ((c.mod(new java.math.BigInteger("2"))).equals(new java.math.BigInteger("1")) )
                          { c = c.add(N); }
                    u = c.divide(new java.math.BigInteger("2"));
        }
        e = e.divide(new java.math.BigInteger("2"));
     }//while
return v;
 } //

public Vector Euc_Ext(java.math.BigInteger a,java.math.BigInteger b)
 {
  Vector PP = new Vector(3,1);
  Vector QQ = new Vector(3,1);
  java.math.BigInteger g = new java.math.BigInteger("0");
  java.math.BigInteger x = new java.math.BigInteger("0");
  java.math.BigInteger y = new java.math.BigInteger("0");
  if(b.equals(java.math.BigInteger.valueOf(0)))
  {
  QQ.addElement(a);
  QQ.addElement(java.math.BigInteger.valueOf(1));
  QQ.addElement(java.math.BigInteger.valueOf(0));
  return QQ;
  }else{
                PP = Euc_Ext(b,(a.mod(b)) );
                        g = (java.math.BigInteger)PP.elementAt(0);
                        x = (java.math.BigInteger)PP.elementAt(1);
                        y = (java.math.BigInteger)PP.elementAt(2);
       QQ.addElement(g);
       QQ.addElement(y);
                        x = x.subtract((a.divide(b)).multiply(y));
       QQ.addElement(x);
       return QQ;
       }
 }//ee
//euclides extendido
```

```
public int FUNCION(java.math.BigInteger L , java.math.BigInteger M)
{
  int j=0;
  java.math.BigInteger auxM = new java.math.BigInteger("10");
  auxM =  M.multiply(auxM);
  for(j=9;j>=0;j--)
   {
    if(( ((auxM.add( java.math.BigInteger.valueOf(j) )).multiply
                (java.math.BigInteger.valueOf(j) )).compareTo(L)==-1 )||
                ( ((auxM.add( java.math.BigInteger.valueOf(j) )).multiply
                (java.math.BigInteger.valueOf(j) )).compareTo(L)== 0 ))
     {
     break;
     }
   }
 return j;
}
/**
 * Método utilizado para determinar si 1 valor con respecto a otro  tiene residuo cuadraticos
 */
public int jacobi(java.math.BigInteger a,java.math.BigInteger b) throws Exception
{
java.math.BigInteger sign,temp,test,bit;
sign = new java.math.BigInteger("1");
java.math.BigInteger ONE = new java.math.BigInteger("1");
java.math.BigInteger TWO = new java.math.BigInteger("2");
java.math.BigInteger CERO = new java.math.BigInteger("0");
java.math.BigInteger FOUR = new java.math.BigInteger("4");
while(a.compareTo(ONE)==1)
{
bit = a.mod(TWO);
if(CERO.equals(bit))
{
 test = (((b.multiply(b)).subtract(ONE)).divide(java.math.BigInteger.valueOf(8))).mod(TWO);
 a = a.divide(TWO);
 if(!( CERO.equals(test) ))  {sign = sign.multiply(java.math.BigInteger.valueOf( -1));}
}else{
                        test = (((a.subtract (ONE)).multiply(b.subtract(ONE))).divide(FOUR)).mod(TWO);
                        temp = (java.math.BigInteger)a;
                          a = b.mod(temp);
                          b = (java.math.BigInteger)temp;
                          if(!(CERO.equals(test)))
        {sign = sign.multiply(java.math.BigInteger.valueOf(-1));}
    }
}//while
if(CERO.equals(a))
{sign = new java.math.BigInteger("0");}
return sign.intValue();
}
            public static class ClassInfo extends com.ms.wfc.core.ClassInfo
            {
                        public void getEvents(IEvents events)
                        {
                                    super.getEvents(events);
                        }

                        public void getProperties(IProperties props)
                        {
                                    super.getProperties(props);
                        }
            }
 }

/**
Se encuentran los procesos de encripción y desencripción con el algoritmo idea recibe bloques de 64 bits y  una clave  de 128 bits
*/
package paquete;
/**
 * @com.register ( clsid=6C90B11B-7976-11D3-9623-444553540001, typelib=6C90B105-7976-11D3-9623-444553540001 )
 */
public class  Idea
{
            //utilizadas para crear valores UNSIGNED, obtener todos los bits
            public static java.math.BigInteger maxim = new java.math.BigInteger("65537");
   public static java.math.BigInteger fuyi  = new java.math.BigInteger("65536");
   public static java.math.BigInteger one   = new java.math.BigInteger("65535");
   public static int                  round = 8;

/**
Función de encripción recibe      :
```

```
Clave   - clave de 128 bits
textp[] - bloque 64 bits(4 palabras de 16 bits)
*/
public Vector iop(java.math.BigInteger texp[],java.math.BigInteger Clave)throws Exception
{
int i,j,k,x;
//guarda las subclaves de encripción
java.math.BigInteger Z[][]    = new java.math.BigInteger[7][10];
//valor a retornar encriptado
Vector                ENC   = new Vector(4,1);
for(i=0;i<7;i++)
{
 for(j=0;j<10;j++)
 {
  Z[i][j]  = new java.math.BigInteger("0");//Inicialización de las variables
  //que guardaran las subclaves de encripción
 }
}
java.math.BigInteger XX[]     = new java.math.BigInteger[5];//Guardara texto a encriptar
java.math.BigInteger YY[]     = new java.math.BigInteger[5]; //Guardara valor encriptado

for(i=0;i<5;i++)
{
 XX[i] = new java.math.BigInteger("0");
 YY[i] = new java.math.BigInteger("0");
}
java.math.BigInteger uskey[] = new java.math.BigInteger[9];//guarda la clave del usuario
for(i=1;i<9;i++)
{
 //convierte la clave recibida en valores numéricos, toma los 8 primeros valores, cada uno de 16 bits
 uskey[i] = Clave.and(java.math.BigInteger.valueOf(65535));//evita que pase el rango y UNSIGNED(65535), toma los 16 bits
primeros
 //actualiza el valor de Clave corriendo 16 bits a la derecha, pues los anteriores fueron sacados
 Clave = (java.math.BigInteger)Clave.shiftRight(16);
}

key(uskey,Z);//genera las sub claves de encripción Z[i][r]

XX[1] = (java.math.BigInteger)texp[0];
XX[2] = (java.math.BigInteger)texp[1];
XX[3] = (java.math.BigInteger)texp[2];
XX[4] = (java.math.BigInteger)texp[3];

cip(XX,YY,Z);//cifra XX lo guarda en YY con las subclaves Z

ENC.addElement(YY[1]);
ENC.addElement(YY[2]);
ENC.addElement(YY[3]);
ENC.addElement(YY[4]);


return ENC;
}
//*************************************************************
//algoritmo de Encripción
public void cip(java.math.BigInteger IN[],java.math.BigInteger OUT[],java.math.BigInteger Z[][])throws Exception
{
int r;
java.math.BigInteger x1 = new java.math.BigInteger("0");
java.math.BigInteger x2 = new java.math.BigInteger("0");
java.math.BigInteger x3 = new java.math.BigInteger("0");
java.math.BigInteger x4 = new java.math.BigInteger("0");
java.math.BigInteger kk = new java.math.BigInteger("0");
java.math.BigInteger t1 = new java.math.BigInteger("0");
java.math.BigInteger t2 = new java.math.BigInteger("0");
java.math.BigInteger a  = new java.math.BigInteger("0");
x1 =(java.math.BigInteger)IN[1];//el valor de los datos de entrada deben ser menor o igual que 65535 = 16 bits
x2 =(java.math.BigInteger)IN[2];
x3 =(java.math.BigInteger)IN[3];
x4 =(java.math.BigInteger)IN[4];
for(r=1;r<9;r++)
{
//*Grupo de operaciones dentro de bloques de 64 bits
 x1 = mul(x1,Z[1][r]);
 x4 = mul(x4,Z[4][r]);
 x2 = ((java.math.BigInteger)(x2.add(Z[2][r]))).and(one);
 x3 = ((java.math.BigInteger)(x3.add(Z[3][r]))).and(one);
//estructura de la función  MA ( Ver Descripción de IDEA)
 kk = mul( Z[5][r],( (x1.xor(x3))) );
 t1 = mul( Z[6][r],( ((kk.add(x2.xor(x4))).and(one)) ));
 t2 = (java.math.BigInteger)(kk.add(t1).and(one));
```

```
 //permutación involuntaria PI

 x1 = x1.xor(t1);
 x4 = x4.xor(t2);
  a  = x2.xor(t2);
 x2 = x3.xor(t1);
 x3 = a;
 //transformación de salida
}

 OUT[1] = mul(x1,Z[1][round+1]);
 OUT[4] = mul(x4,Z[4][round+1]);
 OUT[2] = (x3.add(Z[2][round+1])).and(one);
 OUT[3] = (x2.add(Z[3][round+1])).and(one);

return;
}
//*****************************************************
//genera subclaves de encripción Z[][], recibe la clave del usuario
public void key(java.math.BigInteger uskey[],java.math.BigInteger Z[][])throws Exception
{
// en S estarán los diferentes desplazamientos que se le realizan a
// los subbloques de la clave
 java.math.BigInteger S[] = new java.math.BigInteger[54];

 for(int w=0;w<54;w++)
 {
 S[w] = new java.math.BigInteger("0");
 }
int i,j,r;
// En las primeras 8 posiciones estarán los bits en forma original.
 for(i=1;i<9;i++)
  {
   S[i-1] = uskey[i];
  }
 // Se le realizan los diferentes desplazamientos a las subclaves.
 for(i=8;i<54;i++)
 {
  if(((i+2)%8) == 0)
  {
   S[i] = ((S[i-7].shiftLeft(9) ).xor( S[i-14].shiftRight(7) )).and(one);
  }else if(((i+1)%8) == 0)
        {
         S[i] = ((S[i-15].shiftLeft(9) ).xor( S[i-14].shiftRight(7) )).and(one);
        }
                else
    {
    S[i] = ((S[i-7].shiftLeft(9) ).xor( S[i-6].shiftRight(7) )).and(one);
    }
 }//for
//obtiene las sub claves  guardadas en S y según corresponda para cada iteración/
 for(r=1;r<=round+1;r++)
 {
   for(j=1;j<7;j++)
                {
                Z[j][r] = (java.math.BigInteger)S[6*(r-1)+j-1];
    }
 }
return;
}
//** Es llamado por la subclaves de desencripción invirtiendolas donde se necesitan   *******/
public java.math.BigInteger inv(java.math.BigInteger xin)throws Exception
{
java.math.BigInteger n1 = new java.math.BigInteger("0");
java.math.BigInteger n2 = new java.math.BigInteger("0");
java.math.BigInteger  q = new java.math.BigInteger("0");
java.math.BigInteger  r = new java.math.BigInteger("0");
java.math.BigInteger b1 = new java.math.BigInteger("0");
java.math.BigInteger b2 = new java.math.BigInteger("0");
java.math.BigInteger  t = new java.math.BigInteger("0");
// pregunta que si el valor de la subclave es igual a cero.
if(xin.equals(java.math.BigInteger.valueOf(0) ))
{
 b2 = java.math.BigInteger.valueOf(0);
 }else
   {
   n1 = (java.math.BigInteger)maxim;
   n2 = (java.math.BigInteger)xin;
   b2 = java.math.BigInteger.valueOf(1);
   b1 = java.math.BigInteger.valueOf(0);
```

```
   do{// se realiza hasta cuando el residuo entre n1 y xin sea 1
                r = (java.math.BigInteger)n1.mod(n2);
                q = (java.math.BigInteger)(n1.divide(n2));
                if(r.equals(java.math.BigInteger.valueOf(0)))
                  {
                          if(b2.compareTo(java.math.BigInteger.valueOf(0))== -1)
                                  {
                        b2 = maxim.add(b2);
                                  }
                  }else{
                          n1 = (java.math.BigInteger)n2;
                                  n2 = (java.math.BigInteger)r;
                                  t  = (java.math.BigInteger)b2;
                                  b2 = (java.math.BigInteger)( b1.subtract( q.multiply(b2)) );
                                  b1 = (java.math.BigInteger)t;
                                 }// if else
     }while(  r.compareTo(java.math.BigInteger.valueOf(0))==1) ;
   }//else
return b2; // retorna b2 el cual será el valor de la nueva sub clave
}
// genera subclaves de desecripción Z[][].
public void de_key(java.math.BigInteger Z[][],java.math.BigInteger DK[][])throws Exception
{
 int j;
 // genera las sub claves que se necesitan para cada iteración.
 for(j=1;j<=round+1;j++)
 {
   DK[1][round-j+2] = inv(Z[1][j]);
  DK[4][round-j+2] = inv(Z[4][j]);
    if((j==1)||(j==round+1))
   {
   DK[2][round-j+2] = (fuyi.subtract(Z[2][j])).and(one);
   DK[3][round-j+2] = (fuyi.subtract(Z[3][j])).and(one);
   }
   else
   {
   DK[2][round-j+2] = (fuyi.subtract(Z[3][j])).and(one);
   DK[3][round-j+2] = (fuyi.subtract(Z[2][j])).and(one);
   }
 }//for
  for(j=1;j<=round+1;j++)
             {
             DK[5][round+1-j] = (java.math.BigInteger)Z[5][j];
             DK[6][round+1-j] = (java.math.BigInteger)Z[6][j];
}
return;
}
/**
Algoritmo multiplicador sin tener en cuenta el bit de signo
Este algoritmo se conoce como Low-High
*/
public java.math.BigInteger  mul(java.math.BigInteger a,java.math.BigInteger b)throws Exception
{
 java.math.BigInteger p   = new java.math.BigInteger("0");
 java.math.BigInteger q   = new java.math.BigInteger("0");

 if(a.equals(java.math.BigInteger.valueOf(0)))
  {
   p = maxim.subtract(b);
  }
 else if( b.equals(java.math.BigInteger.valueOf(0)) )
       {
                       p = maxim.subtract(a);
       }
 else {          q = (a.multiply(b));
                 p = (q.and(one)).subtract(q.shiftRight(16));
                  if((p.compareTo(java.math.BigInteger.valueOf(0))==-1)||(p.compareTo(java.math.BigInteger.valueOf(0))==0))

                     {p = p.add(maxim);}
     }
 return p.and(one);
}
/**
 * utilizado para desencriptar un conjunto de bits encriptados con IDEA
 **/
public Vector iop1(java.math.BigInteger texp[],java.math.BigInteger Clave)throws Exception
{
int i,j,k,x;
java.math.BigInteger Z[][]    = new java.math.BigInteger[7][10];
java.math.BigInteger DK[][]   = new java.math.BigInteger[7][10];//guarda las subclaves de desencripción
Vector               ENC   = new Vector(4,1);//devuelve el texto desencriptado
```

```
for(i=0;i<7;i++)
{
 for(j=0;j<10;j++)
 {
  Z[i][j]  = new java.math.BigInteger("0");
 DK[i][j]  = new java.math.BigInteger("0");
 }
}
java.math.BigInteger TT[]     = new java.math.BigInteger[5];//texto provisional desencriptado
java.math.BigInteger YY[]     = new java.math.BigInteger[5];//texto encriptado
for(i=0;i<5;i++)
{
 TT[i] = new java.math.BigInteger("0");
 YY[i] = new java.math.BigInteger("0");
}
java.math.BigInteger uskey[] = new java.math.BigInteger[9];
for(i=1;i<9;i++)
{
//guarda la clave del usuario en uskey, el tamaño de la
 //clave recibida es de 8*16 =  128 bits
 uskey[i] = Clave.and(java.math.BigInteger.valueOf(65535));
 Clave = (java.math.BigInteger)Clave.shiftRight(16);
}
key(uskey,Z);//genera las sub claves de encripción Z[i][r]
de_key(Z,DK);//computa las claves de descencripción
YY[1] = (java.math.BigInteger)texp[0];
YY[2] = (java.math.BigInteger)texp[1];
YY[3] = (java.math.BigInteger)texp[2];
YY[4] = (java.math.BigInteger)texp[3];
cip(YY,TT,DK);//descifra YY lo guarda en TT con  las subclaves DK
ENC.addElement(TT[1]);
ENC.addElement(TT[2]);
ENC.addElement(TT[3]);
ENC.addElement(TT[4]);
return ENC; }
/**
Contiene métodos que hacen parte de RIPEMD-160

*/
package paquete;
import java.awt.*;

/**
 * @com.register ( clsid=6C90B11D-7976-11D3-9623-444553540001, typelib=6C90B105-7976-11D3-9623-444553540001 )
 */
public class  RMD160H  {



 //variable utilizada para trabajar con un nuevo especifico de bits
//con esta variable y la operación and se evita tener numero negativos
//y de un rango mayor
  long Rem = 0xFFFFFFFFL;

/**
  recibe un arreglo de bytes  , a es el numero donde inicia la creación de la palabra
  b
  */
 public long BYTES_TO_DWORD(byte data[],int a, int b)
 {
  //retorna una palabra de 32 bits
  return( (long)  (((long)data[a+b+3]) <<24)|
            (((long)data[a+b+2]) <<16)|
            (((long)data[a+b+1]) <<8 )|
            (((long)data[a+b]  )     )  );
 }
//*********************************************************
 public long ROL(long x , long n)
    {
                //realiza un rotación n bits de la palabra x a la izquierda
     return ((long)((((x)<<(n ))|((x)>>(32-(n ))))&Rem) );
    }
////////////////// Las cinco funciones   Básicas  ////////////////////////////
//*********************************************************          ///////////
  public long F(long x,long y,long z)                                ///////////
    {                                                 ///////////
    return( ((long)(x)^(y)^(z))&Rem );                                 ///////////
    }//                                               ///////////
//*********************************************************         ///////////
  public long G(long x,long y,long z)                                ///////////
    {                                                 ///////////
```

```
    return( ((long)((x)&(y)) | (((~x)&Rem )&(z) ))&Rem );                          //////////
    }//                                                 //////////
//*************************************************************          //////////
  public long H(long x,long y,long z)                                  //////////
    {                                                   //////////
    return( (long)(((x)| ((~y)&Rem)) ^(z))&Rem );                              //////////
    }//                                                 //////////
//*************************************************************          //////////
  public long I(long x,long y,long z)                                  //////////
    {                                                   //////////
    return( (long)(((x)&(z)) | ((y)& ((~z)&Rem) ))&Rem );                          //////////
    }//                                                 //////////
//*************************************************************          //////////
  public long J(long x,long y,long z)                                  //////////
    {                                                   //////////
    return( (long)((x)^ ((y) | ((~z)&Rem)))&Rem );                             //////////
    }//                                                 //////////
//*************************************************************          //////////
////////////////////////////////////////////////////////////////////////////////

/************************ 10 operaciones FF -> III          **********/
public void FF(Long MDaux[],int a,int b,int c,int d,int e,Long x,long s)
 {
 MDaux[a] = new Long((long) (MDaux[a].longValue() + F( MDaux[b].longValue(), MDaux[c].longValue(),
MDaux[d].longValue() ) + x.longValue())&Rem) ;
 MDaux[a] = new Long((long) (ROL( MDaux[a].longValue(), s ) + MDaux[e].longValue())&Rem );
 MDaux[c] = new Long((long) (ROL( MDaux[c].longValue(),          10))&Rem );
 return;
 }

public void GG(Long MDaux[],int a,int b,int c,int d,int e,Long x,long s)
 {
 MDaux[a] = new Long((long)( MDaux[a].longValue() + G( MDaux[b].longValue(), MDaux[c].longValue(),
MDaux[d].longValue() ) + x.longValue() + 0x5A827999L)&Rem);
 MDaux[a] = new Long((long) (ROL( MDaux[a].longValue(), s ) + MDaux[e].longValue())&Rem );
 MDaux[c] = new Long((long) (ROL( MDaux[c].longValue(),          10))&Rem );
 return;
 }

public void HH(Long MDaux[],int a,int b,int c,int d,int e,Long x,long s)
 {
 MDaux[a] = new Long((long)( MDaux[a].longValue() + H( MDaux[b].longValue(), MDaux[c].longValue(),
MDaux[d].longValue() ) + x.longValue()+ 0x6ED9EBA1L)&Rem);
 MDaux[a] = new Long((long) (ROL( MDaux[a].longValue(), s ) + MDaux[e].longValue())&Rem );
 MDaux[c] = new Long((long) (ROL( MDaux[c].longValue(),          10))&Rem );
 return;
 }


public void II(Long MDaux[],int a,int b,int c,int d,int e,Long x,long s)
 {
 MDaux[a] = new Long((long)( MDaux[a].longValue() + I( MDaux[b].longValue(), MDaux [c].longValue(),
MDaux[d].longValue() ) + x.longValue() + 0x8F1BBCDCL)&Rem);
 MDaux[a] = new Long((long) (ROL( MDaux[a].longValue(), s ) + MDaux[e].longValue())&Rem );
 MDaux[c] = new Long((long) (ROL( MDaux[c].longValue(),          10))&Rem );
 return;
 }

public void JJ(Long MDaux[],int a,int b,int c,int d,int e,Long x,long s)
 {

 MDaux[a] = new Long((long)( MDaux[a].longValue() + J( MDaux[b].longValue(), MDaux[c].longValue(),
MDaux[d].longValue() ) +  x.longValue()  + 0xA953FD4EL)&Rem);
 MDaux[a] = new Long((long) (ROL( MDaux[a].longValue(), s ) + MDaux[e].longValue())&Rem );
 MDaux[c] = new Long((long) (ROL( MDaux[c].longValue(),          10))&Rem );
 return;
 }

public void FFF(Long MDaux[],int a,int b,int c,int d,int e,Long x,long s)
 {
 MDaux[a] = new Long((long)( MDaux[a].longValue() + F( MDaux[b].longValue(), MDaux[c].longValue(),
MDaux[d].longValue() ) + x.longValue())&Rem );
 MDaux[a] = new Long((long) (ROL( MDaux[a].longValue(), s ) + MDaux[e].longValue())&Rem );
 MDaux[c] = new Long((long) (ROL( MDaux[c].longValue(),          10))&Rem );
 return;
 }
public void GGG(Long MDaux[],int a,int b,int c,int d,int e,Long x,long s)
 {
 MDaux[a] = new Long((long)( MDaux[a].longValue() + G( MDaux[b].longValue(), MDaux[c].longValue(),
MDaux[d].longValue() ) + x.longValue() + 0x7A6D76E9L)&Rem);
 MDaux[a] = new Long((long) (ROL( MDaux[a].longValue(), s ) + MDaux[e].longValue())&Rem );
```

```
 MDaux[c] = new Long((long) (ROL( MDaux[c].longValue(),         10))&Rem );
 return;
 }
public void HHH(Long MDaux[],int a,int b,int c,int d,int e,Long x,long s)
 {
 MDaux[a] = new Long((long)( MDaux[a].longValue() + H( MDaux[b].longValue(), MDaux[c].longValue(),
MDaux[d].longValue() ) + x.longValue() + 0x6D703EF3L)&Rem);
 MDaux[a] = new Long((long) (ROL( MDaux[a].longValue(), s ) + MDaux[e].longValue())&Rem );
 MDaux[c] = new Long((long) (ROL( MDaux[c].longValue(),         10))&Rem );
 return;
 }
public void III(Long MDaux[],int a,int b,int c,int d,int e,Long x,long s)
 {
 MDaux[a] = new Long((long)( MDaux[a].longValue() + I (MDaux[b].longValue(), MDaux[c].longValue(),
MDaux[d].longValue() ) + x.longValue() + 0x5C4DD124L)&Rem);
 MDaux[a] = new Long((long) (ROL( MDaux[a].longValue(), s ) + MDaux[e].longValue())&Rem );
 MDaux[c] = new Long((long) (ROL( MDaux[c].longValue(),         10))&Rem );
 return;
 }

public void JJJ(Long MDaux[],int a,int b,int c,int d,int e,Long x,long s)
 {

 MDaux[a] = new Long((long)( MDaux[a].longValue() + J( MDaux[b].longValue(), MDaux[c].longValue(),
MDaux[d].longValue() ) + x.longValue()+( 0x50A28BE6L))&Rem );
 MDaux[a] = new Long((long) (ROL( MDaux[a].longValue(), s ) + MDaux[e].longValue())&Rem );
 MDaux[c] = new Long((long) (ROL( MDaux[c].longValue(),         10))&Rem );
 return;
 }


}


import java.math.BigInteger.*;
import paquete.*;
import java.util.*;
import java.awt.*;
import java.io.*;

/**
 * @com.register ( clsid=6C90B11E-7976-11D3-9623-444553540001, typelib=6C90B105-7976-11D3-9623-444553540001 )
 */
class  RMD160 extends RMD160H{


//**********************************************************************
public void MDinit( Long MDbuf[])
 {
//Iniciación de las variables de transformación              para los registros
  MDbuf[0] = new Long( 0x67452301L);
  MDbuf[1] = new Long(0xEFCDAB89L);
  MDbuf[2] = new Long(0x98BADCFEL);
  MDbuf[3] = new Long(0x10325476L);
  MDbuf[4] = new Long(0xC3D2E1F0L);
  return;

 }
//**********************************************************************
public void compress(Long MDbuf[], Long X[] )
 {
   long Rem = 0xFFFFFFFFL;
   Long MDaux[] = new Long [10];
   for(int i=0;i<10;i++){ MDaux[i] = new Long(0L);}

   int aa = 0; int bb = 1; int cc = 2; int dd = 3; int ee = 4;
   int aaa = 5; int bbb = 6; int ccc = 7; int ddd = 8; int eee = 9;
   //Varibles toman los valores después del padding
   //quedan entonces 10 bloques de 32 bits
   MDaux[aa]  = new Long( MDbuf[0].longValue()&Rem );
   MDaux[bb]  = new Long( MDbuf[1].longValue()&Rem );
   MDaux[cc]  = new Long( MDbuf[2].longValue()&Rem );
   MDaux[dd]  = new Long( MDbuf[3].longValue()&Rem );
   MDaux[ee]  = new Long( MDbuf[4].longValue()&Rem );
   MDaux[aaa] = new Long( MDbuf[0].longValue()&Rem );
   MDaux[bbb] = new Long( MDbuf[1].longValue()&Rem );
   MDaux[ccc] = new Long( MDbuf[2].longValue()&Rem );
   MDaux[ddd] = new Long( MDbuf[3].longValue()&Rem );
   MDaux[eee] = new Long( MDbuf[4].longValue()&Rem );

   /* round 1 */
```

```
//Funciones que son usadas en la primera iteración
//realizando los desplazamientos recibidos en el último parámetro
FF(MDaux,aa, bb, cc, dd, ee, X[ 0], 11);
FF(MDaux,ee, aa, bb, cc, dd, X[ 1], 14);
FF(MDaux,dd, ee, aa, bb, cc, X[ 2], 15);
FF(MDaux,cc, dd, ee, aa, bb, X[ 3], 12);
FF(MDaux,bb, cc, dd, ee, aa, X[ 4],  5);
FF(MDaux,aa, bb, cc, dd, ee, X[ 5],  8);
FF(MDaux,ee, aa, bb, cc, dd, X[ 6],  7);
FF(MDaux,dd, ee, aa, bb, cc, X[ 7],  9);
FF(MDaux,cc, dd, ee, aa, bb, X[ 8], 11);
FF(MDaux,bb, cc, dd, ee, aa, X[ 9], 13);
FF(MDaux,aa, bb, cc, dd, ee, X[10], 14);
FF(MDaux,ee, aa, bb, cc, dd, X[11], 15);
FF(MDaux,dd, ee, aa, bb, cc, X[12],  6);
FF(MDaux,cc, dd, ee, aa, bb, X[13],  7);
FF(MDaux,bb, cc, dd, ee, aa, X[14],  9);
FF(MDaux,aa, bb, cc, dd, ee, X[15],  8);

/* round 2 */
//Funciones que son usadas en la segunda iteración
//realizando los desplazamientos  recibidos en el último parámetro
GG(MDaux,ee, aa, bb, cc, dd, X[ 7],  7);
GG(MDaux,dd, ee, aa, bb, cc, X[ 4],  6);
GG(MDaux,cc, dd, ee, aa, bb, X[13],  8);
GG(MDaux,bb, cc, dd, ee, aa, X[ 1], 13);
GG(MDaux,aa, bb, cc, dd, ee, X[10], 11);
GG(MDaux,ee, aa, bb, cc, dd, X[ 6],  9);
GG(MDaux,dd, ee, aa, bb, cc, X[15],  7);
GG(MDaux,cc, dd, ee, aa, bb, X[ 3], 15);
GG(MDaux,bb, cc, dd, ee, aa, X[12],  7);
GG(MDaux,aa, bb, cc, dd, ee, X[ 0], 12);
GG(MDaux,ee, aa, bb, cc, dd, X[ 9], 15);
GG(MDaux,dd, ee, aa, bb, cc, X[ 5],  9);
GG(MDaux,cc, dd, ee, aa, bb, X[ 2], 11);
GG(MDaux,bb, cc, dd, ee, aa, X[14],  7);
GG(MDaux,aa, bb, cc, dd, ee, X[11], 13);
GG(MDaux,ee, aa, bb, cc, dd, X[ 8], 12);

/* round 3 */
//Funciones que son usadas en la tercera iteración
//realizando los desplazamientos recibidos en el último parámetro
HH(MDaux,dd, ee, aa, bb, cc, X[ 3], 11);
HH(MDaux,cc, dd, ee, aa, bb, X[10], 13);
HH(MDaux,bb, cc, dd, ee, aa, X[14],  6);
HH(MDaux,aa, bb, cc, dd, ee, X[ 4],  7);
HH(MDaux,ee, aa, bb, cc, dd, X[ 9], 14);
HH(MDaux,dd, ee, aa, bb, cc, X[15],  9);
HH(MDaux,cc, dd, ee, aa, bb, X[ 8], 13);
HH(MDaux,bb, cc, dd, ee, aa, X[ 1], 15);
HH(MDaux,aa, bb, cc, dd, ee, X[ 2], 14);
HH(MDaux,ee, aa, bb, cc, dd, X[ 7],  8);
HH(MDaux,dd, ee, aa, bb, cc, X[ 0], 13);
HH(MDaux,cc, dd, ee, aa, bb, X[ 6],  6);
HH(MDaux,bb, cc, dd, ee, aa, X[13],  5);
HH(MDaux,aa, bb, cc, dd, ee, X[11], 12);
HH(MDaux,ee, aa, bb, cc, dd, X[ 5],  7);
HH(MDaux,dd, ee, aa, bb, cc, X[12],  5);

/* round 4 */
//Funciones que son usadas en la cuarta iteración
//realizando los desplazamientos recibidos en el último parámetro
II(MDaux,cc, dd, ee, aa, bb, X[ 1], 11);
II(MDaux,bb, cc, dd, ee, aa, X[ 9], 12);
II(MDaux,aa, bb, cc, dd, ee, X[11], 14);
II(MDaux,ee, aa, bb, cc, dd, X[10], 15);
II(MDaux,dd, ee, aa, bb, cc, X[ 0], 14);
II(MDaux,cc, dd, ee, aa, bb, X[ 8], 15);
II(MDaux,bb, cc, dd, ee, aa, X[12],  9);
II(MDaux,aa, bb, cc, dd, ee, X[ 4],  8);
II(MDaux,ee, aa, bb, cc, dd, X[13],  9);
II(MDaux,dd, ee, aa, bb, cc, X[ 3], 14);
II(MDaux,cc, dd, ee, aa, bb, X[ 7],  5);
II(MDaux,bb, cc, dd, ee, aa, X[15],  6);
II(MDaux,aa, bb, cc, dd, ee, X[14],  8);
II(MDaux,ee, aa, bb, cc, dd, X[ 5],  6);
II(MDaux,dd, ee, aa, bb, cc, X[ 6],  5);
II(MDaux,cc, dd, ee, aa, bb, X[ 2], 12);

/* round 5 */
//Funciones que son usadas en la quinta iteración
```

```
//realizando los desplazamientos recibidos en el último parámetro
JJ(MDaux,bb, cc, dd, ee, aa, X[ 4],  9);
JJ(MDaux,aa, bb, cc, dd, ee, X[ 0], 15);
JJ(MDaux,ee, aa, bb, cc, dd, X[ 5],  5);
JJ(MDaux,dd, ee, aa, bb, cc, X[ 9], 11);
JJ(MDaux,cc, dd, ee, aa, bb, X[ 7],  6);
JJ(MDaux,bb, cc, dd, ee, aa, X[12],  8);
JJ(MDaux,aa, bb, cc, dd, ee, X[ 2], 13);
JJ(MDaux,ee, aa, bb, cc, dd, X[10], 12);
JJ(MDaux,dd, ee, aa, bb, cc, X[14],  5);
JJ(MDaux,cc, dd, ee, aa, bb, X[ 1], 12);
JJ(MDaux,bb, cc, dd, ee, aa, X[ 3], 13);
JJ(MDaux,aa, bb, cc, dd, ee, X[ 8], 14);
JJ(MDaux,ee, aa, bb, cc, dd, X[11], 11);
JJ(MDaux,dd, ee, aa, bb, cc, X[ 6],  8);
JJ(MDaux,cc, dd, ee, aa, bb, X[15],  5);
JJ(MDaux,bb, cc, dd, ee, aa, X[13],  6);

/* iteración paralela a la primera */
//Funciones que son usadas en la iteración paralela a la primera
//realizando los desplazamientos recibidos en el último parámetro
JJJ(MDaux,aaa, bbb, ccc, ddd, eee, X[ 5],  8);
JJJ(MDaux,eee, aaa, bbb, ccc, ddd, X[14],  9);
JJJ(MDaux,ddd, eee, aaa, bbb, ccc, X[ 7],  9);
JJJ(MDaux,ccc, ddd, eee, aaa, bbb, X[ 0], 11);
JJJ(MDaux,bbb, ccc, ddd, eee, aaa, X[ 9], 13);
JJJ(MDaux,aaa, bbb, ccc, ddd, eee, X[ 2], 15);
JJJ(MDaux,eee, aaa, bbb, ccc, ddd, X[11], 15);
JJJ(MDaux,ddd, eee, aaa, bbb, ccc, X[ 4],  5);
JJJ(MDaux,ccc, ddd, eee, aaa, bbb, X[13],  7);
JJJ(MDaux,bbb, ccc, ddd, eee, aaa, X[ 6],  7);
JJJ(MDaux,aaa, bbb, ccc, ddd, eee, X[15],  8);
JJJ(MDaux,eee, aaa, bbb, ccc, ddd, X[ 8], 11);
JJJ(MDaux,ddd, eee, aaa, bbb, ccc, X[ 1], 14);
JJJ(MDaux,ccc, ddd, eee, aaa, bbb, X[10], 14);
JJJ(MDaux,bbb, ccc, ddd, eee, aaa, X[ 3], 12);
JJJ(MDaux,aaa, bbb, ccc, ddd, eee, X[12],  6);

/* iteración paralela a la segunda*/
//Funciones que son usadas en la iteración paralela a la segunda
//realizando los desplazamientos recibidos en el último parámetro
III(MDaux,eee, aaa, bbb, ccc, ddd, X[ 6],  9);
III(MDaux,ddd, eee, aaa, bbb, ccc, X[11], 13);
III(MDaux,ccc, ddd, eee, aaa, bbb, X[ 3], 15);
III(MDaux,bbb, ccc, ddd, eee, aaa, X[ 7],  7);
III(MDaux,aaa, bbb, ccc, ddd, eee, X[ 0], 12);
III(MDaux,eee, aaa, bbb, ccc, ddd, X[13],  8);
III(MDaux,ddd, eee, aaa, bbb, ccc, X[ 5],  9);
III(MDaux,ccc, ddd, eee, aaa, bbb, X[10], 11);
III(MDaux,bbb, ccc, ddd, eee, aaa, X[14],  7);
III(MDaux,aaa, bbb, ccc, ddd, eee, X[15],  7);
III(MDaux,eee, aaa, bbb, ccc, ddd, X[ 8], 12);
III(MDaux,ddd, eee, aaa, bbb, ccc, X[12],  7);
III(MDaux,ccc, ddd, eee, aaa, bbb, X[ 4],  6);
III(MDaux,bbb, ccc, ddd, eee, aaa, X[ 9], 15);
III(MDaux,aaa, bbb, ccc, ddd, eee, X[ 1], 13);
III(MDaux,eee, aaa, bbb, ccc, ddd, X[ 2], 11);

/* iteración paralela a la tercera*/
//Funciones que son usadas en la iteración paralela a la tercera
//realizando los desplazamientos recibidos en el último parámetro
HHH(MDaux,ddd, eee, aaa, bbb, ccc, X[15],  9);
HHH(MDaux,ccc, ddd, eee, aaa, bbb, X[ 5],  7);
HHH(MDaux,bbb, ccc, ddd, eee, aaa, X[ 1], 15);
HHH(MDaux,aaa, bbb, ccc, ddd, eee, X[ 3], 11);
HHH(MDaux,eee, aaa, bbb, ccc, ddd, X[ 7],  8);
HHH(MDaux,ddd, eee, aaa, bbb, ccc, X[14],  6);
HHH(MDaux,ccc, ddd, eee, aaa, bbb, X[ 6],  6);
HHH(MDaux,bbb, ccc, ddd, eee, aaa, X[ 9], 14);
HHH(MDaux,aaa, bbb, ccc, ddd, eee, X[11], 12);
HHH(MDaux ,eee, aaa, bbb, ccc, ddd, X[ 8], 13);
HHH(MDaux,ddd, eee, aaa, bbb, ccc, X[12],  5);
HHH(MDaux,ccc, ddd, eee, aaa, bbb, X[ 2], 14);
HHH(MDaux,bbb, ccc, ddd, eee, aaa, X[10], 13);
HHH(MDaux,aaa, bbb, ccc, ddd, eee, X[ 0], 13);
HHH(MDaux,eee, aaa, bbb, ccc, ddd, X[ 4],  7);
HHH(MDaux,ddd, eee, aaa, bbb, ccc, X[13],  5);

/* iteración paralela a la cuarta*/
//Funciones que son usadas en la iteración paralela a la cuarta
//realizando los desplazamientos recibidos en el último parámetro
```

```
   GGG(MDaux,ccc, ddd, eee, aaa, bbb, X[ 8], 15);
   GGG(MDaux,bbb, ccc, ddd, eee, aaa, X[ 6],  5);
   GGG(MDaux,aaa, bbb, ccc, ddd, eee, X[ 4],  8);
   GGG(MDaux,eee, aaa, bbb, ccc, ddd, X[ 1], 11);
   GGG(MDaux,ddd, eee, aaa, bbb, ccc, X[ 3], 14);
   GGG(MDaux,ccc, ddd, eee, aaa, bbb, X[11], 14);
   GGG(MDaux,bbb, ccc, ddd, eee, aaa, X[15],  6);
   GGG(MDaux,aaa, bbb, ccc, ddd, eee, X[ 0], 14);
   GGG(MDaux,eee, aaa, bbb, ccc, ddd, X[ 5],  6);
   GGG(MDaux,ddd, eee, aaa, bbb, ccc, X[12],  9);
   GGG(MDaux,ccc, ddd, eee, aaa, bbb, X[ 2], 12);
   GGG(MDaux,bbb, ccc, ddd, eee, aaa, X[13],  9);
   GGG(MDaux,aaa, bbb, ccc, ddd, eee, X[ 9], 12);
   GGG(MDaux,eee, aaa, bbb, ccc, ddd, X[ 7],  5);
   GGG(MDaux,ddd, eee, aaa, bbb, ccc, X[10], 15);
   GGG(MDaux,ccc, ddd, eee, aaa, bbb, X[14],  8);

    /* iteración paralela a la quinta*/
   //Funciones que son usadas en la iteración paralela a la quinta
   //realizando los desplazamientos recibidos en el último parámetro
   FFF(MDaux,bbb, ccc, ddd, eee, aaa, X[12] ,  8);
   FFF(MDaux,aaa, bbb, ccc, ddd, eee, X[15] ,  5);
   FFF(MDaux,eee, aaa, bbb, ccc, ddd, X[10] , 12);
   FFF(MDaux,ddd, eee, aaa, bbb, ccc, X[ 4] ,  9);
   FFF(MDaux,ccc, ddd, eee, aaa, bbb, X[ 1] , 12);
   FFF(MDaux,bbb, ccc, ddd, eee, aaa, X[ 5] ,  5);
   FFF(MDaux,aaa, bbb, ccc, ddd, eee, X[ 8] , 14);
   FFF(MDaux,eee, aaa, bbb, ccc, ddd, X[ 7] ,  6);
   FFF(MDaux,ddd, eee, aaa, bbb, ccc, X[ 6] ,  8);
   FFF(MDaux,ccc, ddd, eee, aaa, bbb, X[ 2] , 13);
   FFF(MDaux,bbb, ccc, ddd, eee, aaa, X[13] ,  6);
   FFF(MDaux,aaa, bbb, ccc, ddd, eee, X[14] ,  5);
   FFF(MDaux,eee, aaa, bbb, ccc, ddd, X[ 0] , 15);
   FFF(MDaux,ddd, eee, aaa, bbb, ccc, X[ 3] , 13);
   FFF(MDaux,ccc, ddd, eee, aaa, bbb, X[ 9] , 11);
   FFF(MDaux,bbb, ccc, ddd, eee, aaa, X[11] , 11);

   /* Combinación de los resultados obtenidos en cada iteración */
 MDaux[ddd] = new Long( (MDaux[ddd].longValue() + MDaux[cc].longValue() + MDbuf[1].longValue() ) &Rem );            /*
final result for MDbuf[0] */
   MDbuf[1] = new Long( (MDbuf[2].longValue() + MDaux[dd].longValue() + MDaux[eee].longValue())&Rem);
   MDbuf[2] = new Long( (MDbuf[3].longValue() + MDaux[ee].longValue() + MDaux[aaa].longValue())&Rem);
   MDbuf[3] = new Long( (MDbuf[4].longValue() + MDaux[aa].longValue() + MDaux[bbb].longValue())&Rem);
   MDbuf[4] = new Long( (MDbuf[0].longValue() + MDaux[bb].longValue() + MDaux[ccc].longValue())&Rem);
   MDbuf[0] = new Long( MDaux[ddd].longValue()&Rem);

   return;
 }
//**********************************************************************
/**
recibe un arreglo de bytes
offset indica la posición donde debe empezar a operar
MDbuf será la función has final
lswlen tamaño total de bytes leídos
mswlen

*/

public void MDfinish(Long MDbuf[], byte strptr[],int offset, int lswlen, long mswlen)
{

   long Rem = 0xFFFFFFFFL;
   int i;
   Long X[]= new Long[16];
   for(i=0;i<16;i++){
                    X[i] = new Long(0L);
                  }

  for (i=0; i<(lswlen&63); i++)//numero de bytes leídos que faltan por transformar < 64
   {
//cada 4 bytes guárdelos en la posición x[];o cree una palabra de 32 bits y guárdela
     X[i>>2] = new Long ( ((((long)(strptr[offset++]))<<(8*(i&3))) ^ X[i>>2].longValue())&Rem );
   }
   X[(lswlen>>2)&15] = new Long( ((X[(lswlen>>2)&15].longValue()) ^ (long)(1<<(8*(lswlen&3)+7)))&Rem );
  //en la posición que falta realice un relleno
   if ((lswlen & 63) > 55)
   {
     compress(MDbuf, X);//
     for(i=0;i<16;i++){
                      X[i] = new Long(0L);//rellene con ceros
                      }
```

```
   }
   X[14] = new Long (((long)lswlen << 3)&Rem);//guarde los últimos valores para completar el padding
   X[15] = new Long ((((long)lswlen >> 29) | ((long)mswlen << 3))&Rem);

   compress(MDbuf, X);//realiza las ultimas iteraciones

   return;
}

//*****************************************
public java.math.BigInteger iopRMD(String Archivo)
{
   Vector REV = new Vector(20,1);
   long Rem = 0xFFFFFFFFL;//variable para trabajar con números de 32 bits sin signo
   int nbytes;
   int i,j;
   int length[] = new int[2];
   int offset ;
   byte data[] = new byte[1024];
   Long MDbuf[] = new Long[5];//variable final que contendrá 32 bits * 5 = 160 bits resultado final
 java.math.BigInteger codigohash = new java.math.BigInteger("0");
 java.math.BigInteger aux        = new java.math.BigInteger("0");

   for(i=0;i<5;i++){
                MDbuf[i] = new Long(0L);
               }

    long hashcode[] = new long[20];//código has definitivo dividido en palabras de 8 bits *20 = 160 bitws
    for(i=0;i<20;i++){
                  hashcode[i] = 0;
                 }

  Long X[] = new Long[16];
   for( i=0;i<16;i++){
                  X[i] = new Long(0L) ;
                 }

  try{
    FileInputStream mf = new FileInputStream(Archivo);// se abre el archivo que se va a resumir
    MDinit(MDbuf);//inicializa la variable MDbuf con las constantes mágicas

    length[0]=0;
    length[1]=0;

    while((nbytes=mf.read(data,1,1024)) != -1)//se leen cada 1024 bytes y se guardan en data
    {
     for(i=0;i<(nbytes>>6);i++)////hasta que el ultimo bloque sea menor que 64 bytes
     {
      for(j=0;j<16;j++)
         {
                                    // 16 * 4 = 64;  16 = palabras de la primera vuelta, y así sucesivamente
                                    // 64*i es el desplazamiento en el vector
                                    // 4*j indica hasta donde llega en el vector, 4 por que cada 4 bytes es una palabra

          X[j] = new Long( ( BYTES_TO_DWORD(data,(64*i),(4*j)) )&Rem );//32*16 =
         }//for j
     compress(MDbuf,X);//realiza las iteraciones
    }//for i

    if( (length[0] + nbytes)<length[0]){ length[1]++;}

     length[0]+=nbytes;//contiene el numero total de bytes leídos
    }//while

    offset = length[0]&(0x30);
             //offset es un que indica la posición del arreglo en donde se debe empezar a hacer las iteraciones

    MDfinish(MDbuf,data,offset,length[0],length[1]);//realiza la ultima operación

             //guarda la función has de forma legible en caracteres
   for(i=0;i<20;i+=4)
     {

    hashcode[i  ] = (long)(((long)MDbuf[i>>>2].longValue())&(0xFFL));
    hashcode[i+1] = (long)(((long)MDbuf[i>>>2].longValue()>>8)&(0xFFL));
    hashcode[i+2] = (long)(((long)MDbuf[i>>>2].longValue()>>16)&(0xFFL));
    hashcode[i+3] = (long)(((long)MDbuf[i>>>2].longValue()>>24)&(0xFFL));

             REV.addElement( new Character((char)hashcode[i]));
             REV.addElement(new Character((char)hashcode[i+1]));
             REV.addElement(new Character((char)hashcode[i+2]));
```

```
                REV.addElement(new Character((char)hashcode[i+3]));
      }
                for(i=0;i<19;i++)
                {
                aux = (java.math.BigInteger.valueOf((((Character)REV.elementAt(i)).charValue())&255)).shiftLeft(i*8);
                codigohash = codigohash.or(aux);
                }
     }//try
    catch(IOException e)
     {
     System.out.println("Error al leer archivo Nø = "+ e);
     }
                return codigohash;
 }
}//final
```