INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO
Subdirección Académica

No. TT 0310 | Serie Amarilla | 13 de Mayo de 2002
Trabajo Terminal

"SUHCDES: Suite de Herramientas Criptográficas para Archivos y Documentos Electrónicos Seguros"

Carolina Cazares Becerra*
Citlali Paola Hernández Angeles**

Asesor
Ing. Arturo Pruneda Martínez
apruneda@hotmail.com
Instituto Politécnico Nacional
Escuela Superior de Cómputo

Resumen

El presente reporte contiene una descripción de la SUITE DE HERRAMIENTAS CRIPTOGRÁFICAS PARA ARCHIVOS Y DOCUMENTOS ELECTRÓNICOS SEGUROS, un sistema que brinda un conjunto de herramientas criptográficas que ayudan a resolver problemas de seguridad computacional. El uso de técnicas criptográficas en esta aplicación tiene como propósito prevenir algunas fallas de seguridad en un sistema computacional, usando los algoritmos de cifrado: RSA , DSA, DES y TDES . La aplicación esta creada en el lenguaje de programación JAVA bajo la arquitectura de JAVA Swing.

Palabras Llave: Algoritmo, cifrar, descifrar, llaves, firma digital, protocolo de seguridad, llave pública, llave privada, llave secreta, Java, SQL Server.

* Calle Mariquita No. 368 Col. Benito Juárez CP 57000 CD. Nezahualcoyotl Tel. 57343196 Correo: ccarolina90@hotmail.com
** Calle Temascalcingo Mz. 3 Lt. 32 Col. Sagitario 6 Ecatepec de Morelos CP 55297 Tel. 57123143 Correo: citllalipaola@hotmail.com

INDICE

INTRODUCCIÓN

En una comunicación segura los datos se envían a través del canal de comunicación tal como son, sin sufrir modificaciones de ningún tipo y para garantizarlo debemos preservar esta información frente a observadores no autorizados. Una de las disciplinas que ha demostrado tener más éxito para resolver algunos de estos problemas es la criptología. Desde la década de los treinta se descubrió que al aplicar algunas fórmulas y conceptos matemáticos, era posible solucionar la problemática de la seguridad de la información digital, a este conjunto de técnicas se le llamo Criptografía.

La seguridad en general debe de ser considerada como un aspecto de gran importancia en cualquier corporación que trabaje con sistemas computarizados. El hecho que gran parte de actividades humanas sea cada vez más dependiente de los sistemas computarizados hace que la seguridad juegue un papel importante .

SUHCDES es una herramienta que le permite al usuario garantizar la seguridad de documentos electrónicos, así como su autentificación, presentándose a través de una interfaz sencilla de utilizar, permitiendo al usuario la elección de niveles de seguridad según sus necesidades, sin utilizar tecnicismos que puedan resultar complejos y difíciles de entender.

Este sistema asegura la Confidencialidad, Integridad y Autentificación de información, cumpliendo así con los objetivos primordiales de la seguridad computacional.

1. ANTECEDENTES

1.1 Estado del Arte

La seguridad computacional ha dejado de ser un simple tema de moda, para convertirse en una necesidad de todo sistema de información. El concepto de seguridad en la información es mucho más amplio que la simple protección de los datos a nivel lógico. Para proporcionar una seguridad real hemos de tener en cuenta múltiples factores, tanto internos como externos. En primer lugar habría que caracterizar el sistema que va a albergar la información para poder identificar las amenazas, por tanto las clasificamos de la siguiente forma:

Sistemas aislados. Son los que no están conectados a ningún tipo de red. De unos años a esta parte se han convertido en minoría, debido al auge que ha experimentado Internet.

Sistemas interconectados: Hoy en día cualquier computadora pertenece a alguna red, enviando y recogiendo información del exterior casi constantemente. Esto hace que las redes de computadoras sean cada día más complejas y supongan un peligro potencial que no puede en ningún caso ser ignorado.

De tal manera que la seguridad en la información se refiere a la preservación de la información frente a observadores no autorizados.

Las tecnologías criptográficas están cada vez más integradas en los sistemas y las aplicaciones comerciales. Ya existen actualmente en el mercado no menos de 1,400 productos informáticos de encriptación. Sin embargo su importación es sumamente restringida por políticas de seguridad nacional de los países en los que se han desarrollado (por ejemplo, Estados Unidos de Norteamérica y Francia), esto obliga a que en nuestro país se promueva investigación en el área y se desarrollen herramientas que permitan satisfacer las necesidades que en seguridad en cómputo se tienen en México.

El comercio electrónico y muchas otras aplicaciones de la sociedad de la información sólo se expandirán y desplegarán sus beneficios económicos y sociales si puede garantizar la confidencialidad de la información que viaja a través de medios digitales.

1.2 Conceptos Básicos

Cifrado

El cifrado es la transformación de datos en signos ilegibles para quien no disponga de la llave secreta para descifrarlos. Su propósito consiste en asegurarle al usuario privacidad, ocultando la información de aquellos a quienes no está dirigida, incluso de aquellos que tienen acceso a la información cifrada.

Descifrado

El descifrado es la transformación de signos ilegibles a datos legibles (texto plano). Su propósito consiste en asegurarle al usuario privacidad.

Criptología

La criptología es una ciencia constituida por dos subciencias antagonistas: la criptografía y el criptoanálisis.

Criptografía

Rama de las matemáticas que hace uso de métodos y técnicas matemáticas con el objetivo principal de cifrar un mensaje o archivo por medio de un algoritmo, usando una o mas llaves. Esto da lugar a los Criptosistemas.

Entendemos por criptografía (Kriptos=ocultar, Graphos=escritura) como el arte de escribir mensajes con llave secreta o de un modo enigmático. Obviamente la Criptografía hace años que dejó de ser un arte para convertirse en una técnica, o más bien un conglomerado de técnicas, que tratan sobre la protección, ocultamiento frente a observadores no autorizados de la información.

El objetivo de la criptografía es el de proporcionar comunicaciones seguras (y secretas) a través de canales inseguros. Ahora bien, la criptografía no es sinónimo de seguridad.

No es más que una herramienta que es utilizada de forma integrada por mecanismos[1] de complejidad variable para proporcionar no solamente servicios de seguridad, sino también de confidencialidad.

La base de la Criptografía suele ser la aplicación de problemas matemáticos de difícil solución a aplicaciones específicas, denominándose **criptosistema** o **sistema de cifrado** a los fundamentos y procedimientos de operación involucrados en dicha aplicación.

Los sistemas analizados por la criptografía enmascaran la información con el objetivo de asegurar:

- **Confidencialidad o privacidad:** Que sólo pueda leer el contenido del mensaje la gente autorizada.

Ejemplos: en la comunicación por teléfono, que alguien intercepte la comunicación y escuche la conversación quiere decir que no existe privacidad. Si mandamos una carta y por alguna razón alguien rompe el sobre para leer la carta, ha violado la privacidad.

En la comunicación por Internet es muy difícil estar seguros de la privacidad de la comunicación, ya que no se tiene control de la línea de comunicación. Por lo tanto al cifrar (esconder) la información cualquier intercepción no autorizada no podrá revelar la información. Esto es posible si se usan técnicas criptográficas, en particular la privacidad se logra si se cifra el mensaje con un método simétrico.

- **Integridad:** proporciona la seguridad de que los datos recibidos fueron los originalmente emitidos.

Ejemplos: cuando compramos un boleto de avión y los datos del vuelo son cambiados, puede afectar los planes del viajero. Una vez hecho un deposito en el banco, si no es capturada la cantidad correcta causará problemas. La integridad es muy importante en las transmisiones militares ya que un cambio de información puede causar graves problemas.

En Internet las compras se pueden hacer desde dos ciudades muy distantes, la información tiene necesariamente que viajar por una línea de transmisión de la cual no se tiene control, si no existe integridad podrían cambiarse por ejemplo el número de una tarjeta de crédito, los datos del pedido, en fin, información que causaría problemas a cualquier comercio y cliente.

[1] Algoritmos de cifrado (simétricos y asimétricos), condensado de mensaje, firma digital, etc.

La integridad también se puede solucionar con técnicas criptográficas particularmente con procesos simétricos o asimétricos.

- **Autenticación:** proporciona la seguridad de que los datos recibidos fueron en realidad enviados por quien asegura haberlo hecho.

Ejemplo: cuando se quiere cobrar un cheque a nombre de alguien, quien lo cobra debe de someterse a un proceso de verificación de identidad para comprobar que en efecto es la persona quien dice ser, esto en general se lleva a cabo con una credencial que anteriormente fue certificada y acredita la identidad de la persona que la porta. La verificación se lleva a cabo comparando la persona con una foto o con la comparación de una firma convencional.

Por Internet es muy fácil engañar a una persona con quien se tiene comunicación respecto a la identidad, resolver este problema es por lo tanto muy importante para efectuar comunicación confiable.

Las técnicas necesarias para poder verificar la autenticidad tanto de personas como de mensajes usan quizá la más conocida aplicación de la criptografía asimétrica que es la firma digital, de algún modo ésta reemplaza a la firma autógrafa que se usa comúnmente. Para autenticar mensajes se usa criptografía simétrica.

- **No rechazo o no repudiación:** se refiere a que no se pueda negar la autoría de un mensaje enviado.

Cuando se diseña un sistema de seguridad, una gran cantidad de problemas pueden ser evitados si se puede comprobar la autenticidad, garantizar privacidad, asegurar integridad y el no-rechazo de un mensaje.

La criptografía simétrica y asimétrica conjuntamente con otras técnicas, como el buen manejo de las llaves y la legislación adecuada resuelven satisfactoriamente los anteriormente problemas planteados.

Algoritmo criptográfico

Es también llamado cifrador, que es la función matemática utilizada para cifrar y descifrar. Si la seguridad de un algoritmo se centra en mantenerlo en secreto, entonces es un algoritmo restringido.

Un grupo cambiante de usuarios no puede utilizarlo, porque cada vez que un usuario abandona el grupo cada uno debe cambiar un nuevo algoritmo, ya que si alguien accidentalmente revela el secreto, cualquiera podría tener acceso a la información.

La criptografía moderna resuelve este problema con una llave. Esta llave puede tener un gran número de valores. El conjunto de posibles valores de esta llave es llamado espacio llave. Tanto el cifrado como el descifrado utilizan las llaves.

Los sistemas criptográficos se pueden clasificar en dos grandes grupos según el modo en el que operan: en bloque o en flujo.

Cifradores en bloque

Un cifrador en bloque divide el mensaje, *M*, en bloques sucesivos, M_1, M_2,..., de una determinada longitud, y encripta cada uno con una llave única, *K*:

$$E_K(M) = E_K(M_1)E_K(M_2)...$$

La función opera sobre un bloque de texto de tamaño fijo y genera un bloque cifrado del mismo tamaño.

Los cifradores en bloque pueden operar en varios modos, siendo los más comunes el CBC (*Cipher Block Chaining*) y el CFB (*Cipher Feedback*). El modo básico de operación, el ECB (*Electronic Codebook*), no se suele emplear pues es considerado el más débil.

Cifradores en flujo

Un cifrador en flujo divide el mensaje, *M*, en sucesivos caracteres o bits, m_1, m_2,..., y cifra cada uno con el i-ésimo elemento de la llave, k_i:

$$E_K(M) = E_{K1}(M_1)E_{K2}(M_2)...$$
$$K = K_1\, K_2\, ...$$

El algoritmo opera sobre un texto de tamaño arbitrario y genera un texto cifrado del mismo tamaño.

El mecanismo de cifrado tiene una dependencia temporal dictada por la llave y por el estado interno del cifrador de tal forma que, después de cifrar cada elemento, el sistema cambia de estado siguiendo unas ciertas pautas. El algoritmo, que requiere un vector de inicialización, procesa los datos como un flujo de caracteres.

Criptoanálisis: El criptoanálisis, se ocupa de romper los procedimientos de cifrado para así recuperar la información original. Esto se puede hacer descifrando un mensaje sin conocer la llave, o bien obteniendo a partir de uno o más criptogramas la llave que ha sido empleada en su codificación.

El criptoanálisis consiste en comprometer la seguridad de un criptosistema. Esto se puede hacer descifrando un mensaje sin conocer la llave, o bien obteniendo a partir de uno o más criptogramas la llave que ha sido empleada en su codificación. No se considera criptoanálisis el descubrimiento de un algoritmo secreto de cifrado; hemos de suponer por el contrario que los algoritmos siempre son conocidos.

En general el criptoanálisis se suele llevar a cabo estudiando grandes cantidades de pares mensaje - criptograma generados con la misma llave. El mecanismo que se emplee para obtenerlos es indiferente, y puede ser resultado de escuchar un canal de comunicaciones, o de la posibilidad de que el objeto de nuestro ataque responda con un criptograma cuando le enviemos un mensaje. Obviamente, cuanto mayor sea la cantidad de pares, más probabilidades de éxito tendrá a el criptoanálisis.

Uno de los tipos de análisis más interesantes es el de texto plano escogido, que parte de que conocemos una serie de pares de textos planos elegidos por nosotros y sus criptogramas correspondientes. Esta situación se suele dar cuando tenemos acceso al dispositivo de cifrado y este nos permite efectuar operaciones, pero no nos permite leer su llave por ejemplo, las tarjetas de los teléfonos móviles GSM. El número de pares necesarios para obtener la llave desciende entonces significativamente. Cuando el sistema es débil, pueden ser suficientes unos cientos de mensajes para obtener información que permita deducir la llave empleada.

También podemos tratar de criptoanalizar un sistema aplicando el algoritmo de descifrado, con todas y cada una de las llaves, a un mensaje codificado que poseemos y comprobar cuáles de las salidas que se obtienen tienen sentido como posible texto plano.

Este método y todos los que buscan exhaustivamente por el espacio de llaves K, se denominan ataques por la fuerza bruta, y en muchos casos no suelen considerarse como auténticas técnicas de criptoanálisis, reservándose este término para aquellos mecanismos que explotan posibles debilidades intrínsecas en el algoritmo de cifrado.

Se da por supuesto que el espacio de llaves para cualquier criptosistema digno de interés ha de ser suficientemente grande como para que un ataque por la fuerza bruta sea inviable. Hemos de tener en cuenta no obstante que la capacidad de cálculo de las computadoras crece a gran velocidad, por lo que algoritmos que hace unos años eran resistentes frente a ataques por la fuerza bruta hoy pueden resultar inseguros, como es el caso de DES. Sin embargo, existen longitudes de llave para las que resultará imposible a todas luces un ataque de este tipo. Por ejemplo, si diseñáramos una máquina capaz de recorrer todas las combinaciones que pueden tomar 256 bits, cuyo consumo fuera mínimo en cada cambio de estado 1 , no habrá energía suficiente en el Universo para que pudiera completar su trabajo.

Un par de métodos de criptoanálisis que han dado interesantes resultados son el análisis diferencial y el análisis lineal.
El primero de ellos, partiendo de pares de mensajes con diferencias mínimas usualmente de un bit, estudia las variaciones que existen entre los mensajes cifrados correspondientes, tratando de identificar patrones comunes. El segundo emplea operaciones XOR entre algunos bits del texto plano y algunos bits del texto cifrado, obteniendo finalmente un único bit. Si realizamos esto con muchos pares de texto plano-texto cifrado podemos obtener una probabilidad p en ese bit que calculamos. Si p está suficientemente sesgada (no se aproxima a 1/2), tendremos la posibilidad de recuperar la llave.

Otro tipo de análisis, esta vez para los algoritmos asimétricos, consistirá en tratar de deducir la llave privada a partir de la pública. Suelen ser técnicas analíticas que básicamente intentan resolver los problemas de elevado coste computacional en los que se apoyan estos criptosistemas: factorización, logaritmos discretos, etc. Mientras estos problemas genéricos permanezcan sin solución eficiente, podremos seguir contando en estos algoritmos.

La Criptografía no sólo se emplea para proteger información, también se utiliza para permitir su autentificación, es decir, para identificar al autor de un mensaje e impedir que nadie suplante su personalidad. En estos casos surge un nuevo tipo de criptoanálisis que está encaminado únicamente a permitir que elementos falsos pasen por buenos. Puede que ni siquiera nos interese descifrar el mensaje original, sino simplemente poder sustituirlo por otro falso y que supere las pruebas de autentificación.

Como se puede apreciar, la gran variedad de sistemas criptográficos produce necesaria mente gran variedad de técnicas de criptoanálisis, cada una de ellas adaptada a un algoritmo o familia de ellos.

Con toda seguridad, cuando en el futuro aparezcan nuevos mecanismos de protección de la información, surgirán con ellos nuevos métodos de criptoanálisis. De hecho, la investigación en este campo es tan importante como el desarrollo de algoritmos criptográficos, y esto es debido a que, mientras que la presencia de fallos en un sistema es posible demostrarla, su ausencia es por definición indemostrable.

1.3 Métodos de Cifrado

1.3.1 Método simétrico

La criptografía simétrica se refiere al conjunto de métodos que permiten tener comunicación segura entre las partes siempre y cuando anteriormente se hayan intercambiado la llave correspondiente que llamaremos llave secreta.

Este método usa la misma llave para cifrar y para descifrar un mensaje y su seguridad se basa en el secreto de dicha llave (Ver figura 1.1). Generalmente se utilizan dos funciones: una para realizar el cifrado y otra para descifrar. Su principal desventaja es que hace falta que el emisor y el receptor compartan la llave.

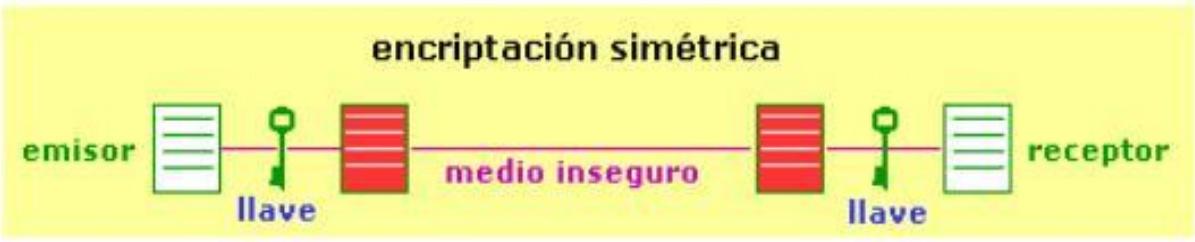


Figura 1.1

Como ejemplo de sistema simétrico está «Enigma»; Éste es un sistema que fue usado por Alemania, en el que las llaves se distribuían a diario en forma de libros de códigos. Cada día, un operador de radio, receptor o transmisor, consultaba su copia del libro de códigos para encontrar la llave del día. Todo el tráfico enviado por ondas de radio durante aquel día era cifrado y descifrado usando las llaves del día. Algunos ejemplos actuales de algoritmos simétricos son 3DES, Blowfish e IDEA.

Dado que toda la seguridad está en la llave, es importante que sea muy difícil adivinar el tipo de llave. Esto quiere decir que el abanico de llaves posibles, o sea, el *espacio de posibilidades de llaves*, debe ser amplio.

Inglaterra usó máquinas para adivinar las llaves durante la Segunda Guerra Mundial. El sistema alemán Enigma estaba provisto de un amplio abanico de llaves, pero los ingleses diseñaron máquinas de cómputo especializado, los Bombes, para probar las llaves de un modo mecánico hasta que la llave del día era encontrada. Esto significaba que algunas veces encontraban la llave del día unas pocas horas después de que ésta fuera puesta en uso, pero también que otros días no podían encontrar la llave correcta. Los Bombes no fueron máquinas de cómputo general, sino los precursores de las computadoras de hoy en día.

Hoy por hoy, las computadoras pueden adivinar llaves con extrema rapidez, y ésta es la razón por la cual el tamaño de la llave es importante en los «criptosistemas» modernos. El algoritmo de cifrado DES usa una llave de 56 bits, lo que significa que hay 2^{56} llaves posibles. 2^{56} son 72.057.594.037.927.936 llaves. Esto representa un número muy alto de llaves, pero una computadora de uso general puede comprobar todo el espacio posible de llaves en cuestión de días. Una máquina especializada lo puede hacer en horas. Por otra parte, algoritmos de cifrado de diseño más reciente como 3DES, Blowfish e IDEA usan todos llaves de 128 bits, lo que significa que existen 2^{128} llaves posibles. Esto representa muchísimas más llaves, y aun en el caso de que todas las máquinas del planeta estuvieran cooperando, todavía tardarían más tiempo que la misma edad del universo en encontrar la llave.

Distribución de llaves simétricas

Estos métodos suelen tener tres tipos de llaves:

- de sesión, empleadas para cifrar los datos de usuario y que son actualizadas frecuentemente,
- cifrado de llaves, para proteger las llaves de sesión, y
- secretas, que son distribuidas manualmente y se utilizan para proteger las de cifrado de llaves cuando son intercambiadas.

La distribución manual de las llaves secretas es el principal inconveniente de la administración mediante técnicas simétricas, por lo que la mayoría de los sistemas utilizan métodos asimétricos para llevar a cabo dicha distribución.

1.3.2 Método Asimétrico

La criptografía asimétrica usa dos llaves, una para cifrar y otra para descifrar, relacionadas matemáticamente de tal forma que los datos cifrados por una de las dos sólo pueden ser descifrados por la otra. Cada usuario tiene dos llaves, la **pública** y la **privada**, y distribuye la primera.

Estos algoritmos se pueden utilizar de dos formas, dependiendo de si la llave pública se emplea como llave de cifrado o de descifrar.

Los sistemas de cifrado de llave pública (cifrado asimétrico) se inventaron con el fin de evitar por completo el problema del intercambio de llaves. Un sistema de cifrado de llave pública usa un par de llaves para el envío de mensajes. Las dos llaves pertenecen a la misma persona a la que se ha enviado el mensaje. Una llave es *pública* y se puede entregar a cualquier persona. La otra llave es *privada* y el propietario debe guardarla para que nadie tenga acceso a ella. El remitente usa la llave pública del destinatario para cifrar el mensaje, y una vez cifrado, sólo la llave privada del destinatario podrá descifrar este mensaje.

Este protocolo resuelve el problema del intercambio de llaves, que es inherente a los sistemas de cifrado simétricos. No hay necesidad de que el remitente y el destinatario tengan que ponerse de acuerdo en una llave. Todo lo que se requiere es que, antes de iniciar la comunicación secreta, el remitente consiga una copia de la llave pública del destinatario. Es más, esa misma llave pública puede ser usada por cualquiera que desee comunicarse con su propietario. Por tanto, se necesitarán sólo *n* pares de llaves por cada *n* personas que deseen comunicarse entre ellas.

Los sistemas de cifrado de llave pública se basan en funciones-trampa de un sólo sentido. Una función de un sólo sentido es aquélla cuya computación es fácil, mientras que invertir la función es extremadamente difícil. Por ejemplo, es fácil multiplicar dos números primos juntos para sacar uno compuesto, pero es difícil factorizar uno compuesto en sus componentes primos. Una función-trampa de un sentido es algo parecido, pero tiene una trampa. Esto quiere decir que si se conociera alguna pieza de la información, sería fácil computar el inverso. Por ejemplo, si tenemos un número compuesto por dos factores primarios y conocemos uno de los factores, es fácil computar el segundo.

Dado un cifrado de llave pública basado en factorización de números primos, la llave pública contiene un número compuesto de dos factores primos grandes, y el algoritmo de cifrado usa ese compuesto para cifrar el mensaje. El algoritmo para descifrar el mensaje requiere el conocimiento de los factores primos, para que el descifrado sea fácil si poseemos la llave privada que contiene uno de los factores, pero extremadamente difícil en caso contrario.

Como con los sistemas de cifrado simétricos buenos, con un buen sistema de cifrado de llave pública toda la seguridad descansa en la llave. Por lo tanto el tamaño de la llave es una medida del seguridad del sistema, pero no se puede comparar el tamaño del cifrado simétrico con el de un cifrado de llave pública para medir la seguridad. En un ataque de fuerza bruta sobre un cifrado simétrico con una llave de un tamaño de 80 bits, el atacante debe enumerar hasta 2^{81}-1 llaves para encontrar la llave correcta. En un ataque de fuerza bruta sobre un cifrado de llave pública con una llave de un tamaño de 512 bits, el atacante debe factorizar un número compuesto codificado en 512 bits (hasta 155 dígitos decimales).

La cantidad de trabajo para el atacante será diferente dependiendo del cifrado que esté atacando. Mientras 128 bits es suficiente para cifrados simétricos, dada la tecnología de factorización de hoy en día, se recomienda el uso de llaves públicas de 1024 bits para la mayoría de los casos.

En el primer caso , cuando un usuario, A, quiere enviar información a otro usuario, B, utiliza la llave pública de B, Kpu_B, para cifrar los datos. El usuario B utilizará su llave privada (que sólo él conoce), Kpr_B, para obtener el texto en claro a partir de la información (cifrada) recibida. Si otro usuario, C, quiere enviar información al usuario B, también empleará la llave pública Kpu_B. Este modo se puede emplear para proporcionar el servicio de confidencialidad, pues sólo el usuario B es capaz de descifrar los mensajes que los usuarios A y C le han enviado (Ver figura 1.2).

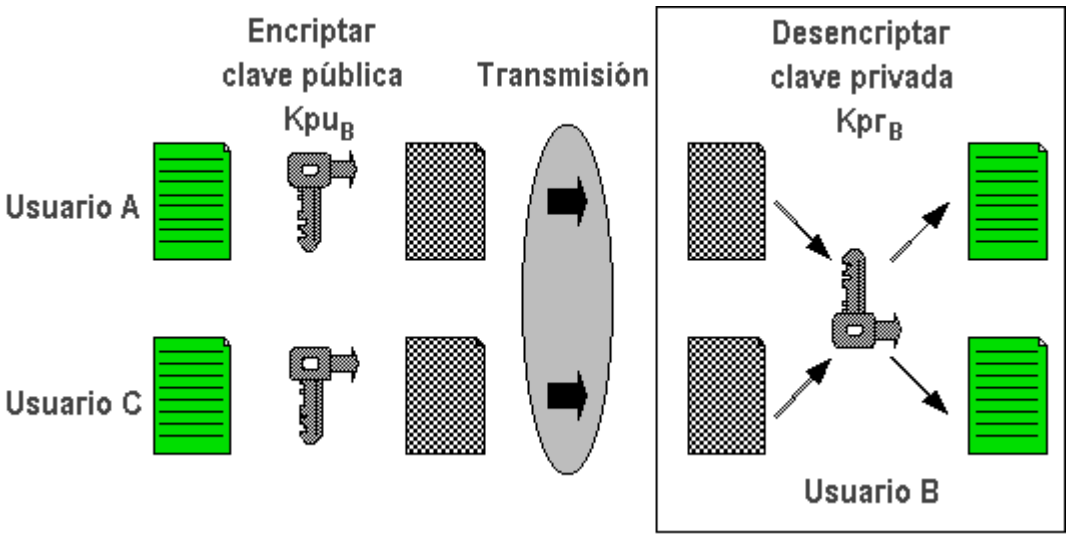


Figura 1.2 Criptografía asimétrica: confidencialidad.

En el otro modo de operación , es el usuario B quien cifra la información utilizando su llave privada, Kpr_B, de forma que cualquiera que conozca Kpu_B podrá descifrar la información transmitida. Este modo se puede emplear para proporcionar el servicio de autenticación, ya que la obtención del texto en claro a partir del texto cifrado es una garantía de que el emisor del mensaje es el propietario de Kpu_B (lógicamente, para saber que el mensaje obtenido del descifrado del texto cifrado es el texto en claro original, éste se ha de obtener por otros medios para realizar la comparación). Esto es la base de las firmas digitales (Ver figura 1.3).

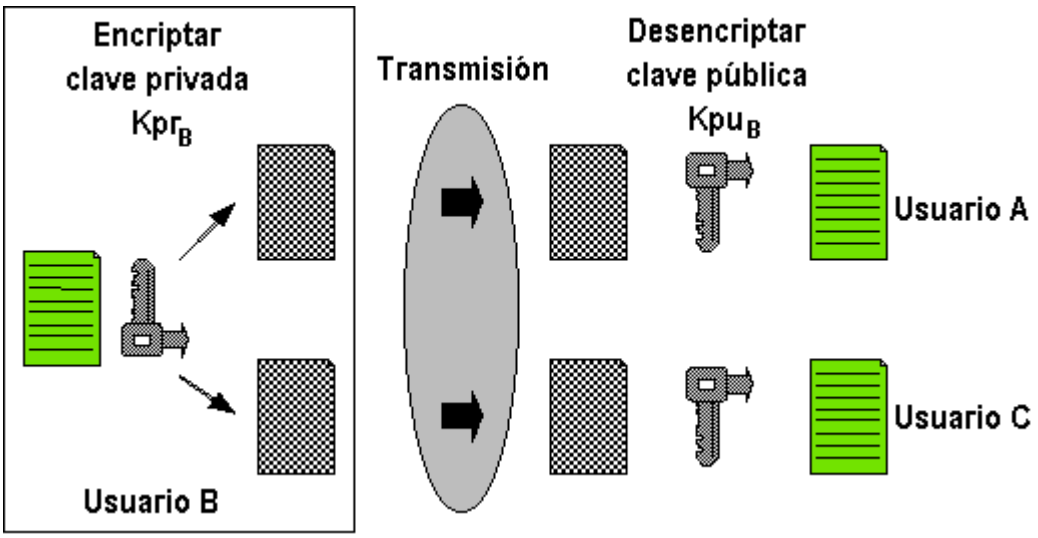


Figura 1.3 Criptografía asimétrica: autenticación.

Desde el punto de vista de la confidencialidad, los algoritmos asimétricos proporcionan una mayor seguridad que los simétricos a costa de una mayor carga computacional. Es por esta razón por lo que generalmente se emplea una combinación de ambos.

1.4 Funciones de Autentificación

1.4.1 Firma Digital

La Firma Digital es un conjunto de caracteres que ligan un documento digital con una llave de cifrado. Es un mecanismo utilizado para asegurar la integridad del mensaje y la autenticación del emisor. Este método consiste en la obtención de un valor hash del mensaje y su posterior encriptación con la llave privada del emisor. En recepción se desencripta el hash con la llave pública del emisor y se compara con otro valor hash obtenido en recepción de forma independiente a partir del mensaje recibido (Ver figura 1.4).

La firma digital permite soportar el no repudio, característica esencial en entornos de comercio electrónico, ya que la verificación de la firma garantiza que ésta sólo puede haber sido generada por el poseedor de la llave privada; o sea, su usuario legítimo.

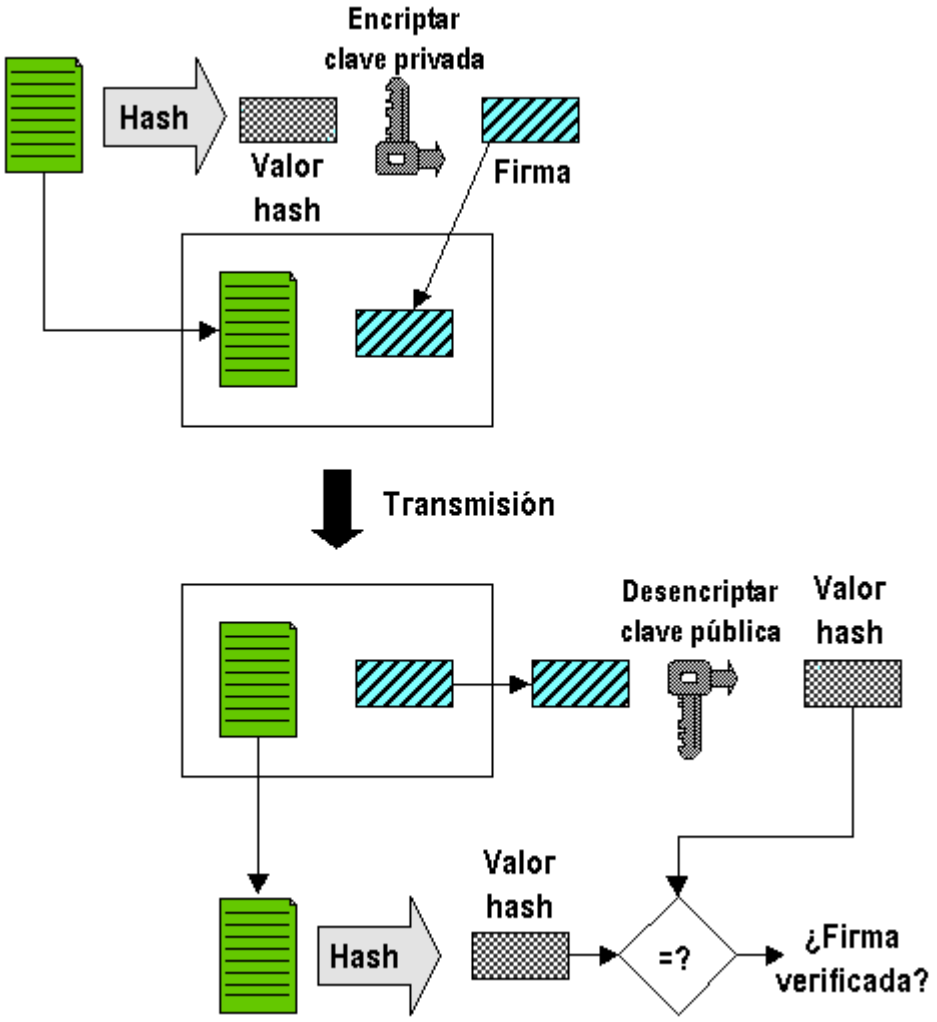


Figura 1.4

Realización de firma digital

Se aplica un algoritmo **hash** (revoltijo) sobre el texto a firmar, obteniendo un extracto de longitud fija, y absolutamente específico para ese mensaje (un mínimo cambio en el mensaje produce un extracto completamente diferente).

Los algoritmos hash más utilizados son **MD5** ó **SHA-1**. Este extracto, cuya longitud oscila entre 128 y 160 bits (en función del algoritmo utilizado), se somete a continuación a cifrado mediante la llave **secreta** del dueño de la llave pública. El algoritmo utilizado para cifrar el extracto puede ser el mismo **RSA** o DSA. El extracto cifrado constituye la firma y se añade al final del mensaje (o en un fichero adherido).

Verificación de Firma Digital

Se necesita disponer de la llave pública del firmante para poder verificar su firma. El receptor descifra el condensado cifrado que constituye la firma digital (de forma transparente al usuario), utilizando para ello la llave pública del remitente. Como resultado obtiene un bloque de caracteres. A continuación, calcula el condensado hash que corresponde al texto del mensaje. Si el resultado coincide **exactamente** con el bloque de caracteres obtenido en la operación anterior, la firma se considera **válida**. Si existe la menor diferencia, la firma se considera no válida.

1.4.2 Message Digest

Otro método que nos ayuda a mantener la autenticidad e integridad de los mensajes es el llamado Message Digest (condensado de mensaje).Utiliza una función hash que a partir de un mensaje de longitud variable genera una serie de bytes. Los Message Digest tienen varias propiedades muy importantes:

- Normalmente son de longitud fija, independientemente de la longitud del mensaje.
- A partir del Message Digest no se puede averiguar el mensaje.
- No se puede encontrar dos mensajes diferentes que tengan el mismo Message Digest, a pesar que la longitud del Message Digest es menor que la longitud del mensaje.

1.4.3 Funciones hash

Las funciones hash, son unas funciones matemáticas que realizan un resumen de un documento. Su forma de operar es comprimir el documento en un único bloque de longitud fija, bloque cuyo contenido es ilegible y no tiene ningún sentido real. Tanto es así que por definición las funciones hash son irreversibles, es decir, que a partir de un bloque comprimido no se puede obtener el bloque sin comprimir, y si no es así no es una función hash. Estas funciones son además de dominio público.

Aunque las funciones de "hash" en general cumplen diferentes funciones en los programas de computación, en criptografía se utilizan para generar un condensado de mensaje que

representa a un archivo o mensaje de mayor longitud. Como las funciones de "hash" son más rápidas que las de firma, es más eficiente procesar una firma digital utilizando un condensado de mensaje, que es pequeño, que en base al documento en toda su extensión. Además, un condensado se puede hacer público sin revelar los detalles del documento que lo origina.

Las funciones hash más conocidas y usadas son:

MD2, abreviatura de Message Digest 2, diseñado para computadoras con procesador de 8 bits. Todavía se usa, pero no es recomendable, debido a su lentitud de proceso.

MD4, abreviatura de Message Digest 4, desarrollado por Ron Rivest, uno de los fundadodres de RSA Data Security Inc. y padre del sistema asimétrico RSA. Aunque se considera un sistema inseguro, es importante porque ha servido de base para la creación de otras funciones hash.

MD5, abreviatura de Message Digest 5, también obra de Ron Rivest, que se creó para dar seguridad a MD4, y que ha sido ámpliamante usado en diversos campos, como autenticador de mensajes en el protocolo SSL y como firmador de mensajes en el programa de correo PGP. Sin embargo, fue reventado en 1996 por el mismo investigador que lo hizo con MD4, el señor Dobbertin, que consiguió crear colisiones en el sistema MD5, aunque por medio de ataques parciales. Pero lo peor es que también consiguió realizar ataques que comprometían la no-colisión, por lo que se podían obtener mensajes con igual hash que otro determinado. A pesar de todo esto, MD5 se sigue usando bastante en la actualidad.

SHA-1, Secure Hash Algorithm, desarrollado como parte integrante del Secure Hash Standar (SHS) y el Digital Signature Standar (DSS) por la Agencia de Seguridad Nacional Norteamericana, NSA. Sus creadores afirman que la base de este sistema es similar a la de MD4 de Rivest, y ha sido mejorado debido a ataques nunca desvelados. La versión actual se considera segura (por lo menos hasta que se demuestre lo contrario) y es muy utilizada algoritmo de firma, como en el programa PGP en sus nuevas llaves DH/DSS (Diffie-Hellman/Digital Signature Standar). Destacar también que en la actualidad se están estudiando versiones de SHA con longitudes de llave de 256, 384 y 512 bits.

1.5 Normativas sobre seguridad

Las normativas sobre seguridad empezaron a desarrollarse a finales de los años 7 cuando surgió la necesidad de proteger ciertas comunicaciones que no pertenecían a los restringidos ambientes militares y diplomáticos, hasta entonces únicos usuarios de comunicaciones seguras. Bancos y Multinacionales fueron los promotores y primeros beneficiarios de la normalización de los métodos de protección de información y de las comunicaciones. Dicha normalización, además de permitir una comunicación cifrada entre distintas entidades y distintos equipos, incrementa el nivel de seguridad de los algoritmos al hacerlos públicos, lo que facilita su estudio y posibilidad de analizarlos detalladamente.

Las organizaciones internacionales dedicadas a la elaboración de normativas han generado varios documentos relativos a las distintos aspectos de seguridad. Generalmente, en estas normas se tratan aspectos muy genéricos de la seguridad, como la situación de los elementos de seguridad en los distintos puntos de la red, sin entrar en detalles de algoritmos concretos. Así ocurre, por ejemplo, en la norma ISO 7498.2 y en distintas recomendaciones de las series X.2xx, X.4xx, X.5xx, X.7xx y X.8xx dadas por la Unión Internacional de Telecomunicaciones (ITU) y los trabajos realizados por el Sub Comité 27 (SC27) dependiente de la Organización Internacional de Estándares (ISO) y de la Comisión Electrónica Internacional (IEC).

Por otra parte, el Instituto Nacional Americano de Estándares (ANSI), en sus serie X3 y X9, describe con detalle el cifrado, la integridad, la autenticación y la administración de llaves. En la serie X12 esta contemplada la seguridad para el intercambio de datos electrónicos. Muchas de las normas ANSI han sido posteriormente modificadas y adoptadas por ISO.

Existen, además otro grupo y otras comisiones como son el Instituto Europeo de Estándares de Telecomunicaciones (ETSI) o la Asociación Europea de Fabricantes de Ordenadores (ECMA), que están trabajando en cuestiones de normativa de seguridad, aunque sus resultados aun no son definitivos.

Sub Comité 27 (SC27)

El SC27 realiza un trabajo de normativa sobre la seguridad genérica; es decir sin referencias a algoritmos criptográficos concretos. El SC27 existen 3 grupos de trabajo (WG1, WG2 y WG3), cada uno de los cuales esta encargado de generar la normativa correspondiente sobre distintos aspectos de seguridad.

WG1

Es el responsable de identificar las necesidades criptográficas, de definir de forma general los servicios de seguridad y de sugerir las líneas maestras para el uso y la administración de sistemas seguros relacionados con la tecnología de la información. Algunas de las normas generadas son:

- ISO/IEC 9979: Procedures for the registration of criyptographic algorithms
- ISO/IEC DIS 11770-1: Key management- part 1: Key management framework
- ISO/IEC WD 14516-1 y 14516-2: Guidelines on the use and manegement of Trusted Third-Party services.

WG2

Se encarga de la definición de técnicas y mecanismos necesarios para ofrecer los distintos servicios de seguridad en las redes comunicaciones. Algunas de las normas redactadas por este grupo son:

- ISO/IEC 9796: Digital Signature scheme giving message recovery
- ISO/IEC 9797: Data Integrity Mechanism Using a Cryptographic Check Function Employing a Block Cipher algorithm.
- ISO/IEC 9798-1, 9798-2, 9798-3, 9798-4 y 9798-5: Entity Authentication.
- ISO/IEC CD 10118-1 y 10118-2: Hash Functions.
- ISO/IEC DIS 11770-1: Key Management-Part 1: Key Management Framework
- ISO/IEC DIS 11770-2 y 11770-3: Key Management – Mechanisms using symmetric techniques.
- ISO/IEC DIS 11770-2 y 11770-3: Key Management – Mechanisms using asymmetric techniques.
- ISO/IEC CD 13338-1, 13338-2 y 13338-3: Non – repudiation
- ISO/IEC CD 14888-1, 14888-2 y 14888-3: Digital signatures

WG3

Se dedica a establecer criterios de la evaluación de la seguridad como los datos en:

- ISO/IEC WD: Evaluation criteria for information tecnology systems

La mayor parte de estas normas han aparecido durante 1994, 1995 y 1996, y esta prevista una ampliación para muchas de ellas en los próximos años. El texto redactado coincide en muchos casos con el de las recomendaciones dadas por ITU

2. Análisis y Diseño

2.1 Análisis de métodos de cifrado

Una de las diferencias fundamentales entre los métodos[2] de cifrado radica en el concepto de seguridad.

La ventaja principal de los cifrados asimétricos es su mayor seguridad. En el sistema simétrico siempre existe la posibilidad de que sea la llave descubierta durante la transmisión.

Otra ventaja importante del sistema asimétrico consiste en que proporciona un método de firma digital; por otro lado la autenticación por medio de sistemas simétricos requieren que algunos secretos se compartan y algunas veces hasta requiere de la confianza en un tercero. El remitente puede alegar que desconoce un mensaje que previamente ha firmado diciendo que la llave compartida no fue mantenida en el debido secreto por alguna de las partes.

La autenticación con los sistemas asimétricos evitan este tipo de inconvenientes ya que cada usuario es responsable de mantener en secreto su llave privada. Por ello, la autenticación con llave pública[3] no es repudiable. .

[2] Métodos de cifrado Simétrico y Asimétrico.
[3] Métodos de cifrado asimétricos.

La desventaja de utilizar el método de cifrado asimétrico es la velocidad: los algoritmos simétricos cifra bloques de texto del documento original, y son más sencillos que los sistemas de llave pública, por lo que sus procesos de cifrado y descifrado son más rápidos pero ambos métodos pueden combinarse para obtener lo mejor de cada uno: la seguridad de la llave pública y la velocidad de la llave privada.

Para el desarrollo de nuestro sistema utilizaremos los métodos simétricos y asimétricos para el cifrado de datos. Elegimos utilizar estos dos tipos de cifrados para que el usuario tenga las herramientas criptográficas necesarias para garantizar la seguridad de su información.

2.2 Análisis de algoritmos de cifrado

2.2.1 Algoritmo DES

DES[4] fue el primer algoritmo desarrollado comercialmente y surgió como resultado de la petición del Departamento de Defensa de EE.UU. a IBM.

Es un cifrador en bloque que utiliza una llave de 64 bits de longitud (de los cuales 8 son de paridad) para encriptar bloques de 64 bits de datos. Se basa en permutaciones, sustituciones y sumas módulo 2 (XOR) y consigue que la modificación de un bit del texto en claro produzca aproximadamente el cambio de los bits del criptograma con una probabilidad del 50%.

DES es un sistema criptográfico simétrico de llave secreta. Cuando se utiliza para comunicaciones, tanto el remitente como el receptor deben conocer la misma llave secreta que se usa tanto para cifrado como para el descifrado del mensaje. El DES puede ser utilizado también por un solo usuario para cifrado: se pueden guardar archivos cifrados en un disco duro. Si se utiliza en un medio de usuarios múltiples, distribuir la llave con seguridad puede llegar a ser un serio inconveniente; la criptografía con llave pública se ideó justamente para resolver este problema.

[4] Algoritmo de llave secreta (cifrado simétrico)

Así pues, las propiedades fundamentales del DES son:

- **Dependencia entre símbolos:** Cada bit del texto cifrado es una función compleja de todos los bits de la llave y todos los bits del texto original (por bloques).

- **Cambio de los bits de entrada:** Un cambio en un bit en el mensaje original produce el cambio del 50%, aproximadamente, de los bits del bloque cifrado.

- **Cambio de los bits de llave:** Un cambio en un bit de la llave produce, aproximadamente, el cambio de la mitad de los bits del bloque cifrado.

- Un error en la transmisión de un texto cifrado se "propaga" a todo el bloque del que forma parte, produciendo un conjunto de errores después del descifrado de 64 bits.

Los modos de operación de DES.

Dependiendo de la naturaleza de la aplicación **DES** tiene 4 modos de operación para poder implementarse: **ECB** (Electronic Codebook Mode) para mensajes cortos, de menos de 64 bits, **CBC** (Cipher Block Chaining Mode) para mensajes largos, **CFB** (Cipher Block Feedback) para cifrar bit por bit ó byte por byte y el **OFB** (Output Feedback Mode) el mismo uso pero evitando propagación de error.

ECB (Electronic CodeBook): cifra cada bloque de 64 bits del mensaje en claro uno tras otro con la misma llave de 56 bits. Un par de bloques idénticos de mensaje en claro producen bloques idénticos de mensaje cifrado.

CBC (Cipher Block Chaining): sobre cada bloque de 64 bits del mensaje en claro se ejecuta un OR exclusivo con el bloque previo del mensaje cifrado antes de proceder al cifrado con la llave DES. De este modo, el cifrado de cada bloque depende del anterior y bloques idénticos de mensaje en claro producen diferentes mensajes cifrados.

CFB (Cipher FeedBack): el cifrado de un bloque de mensaje en claro procede de ejecutar un OR exclusivo del bloque de mensaje en claro con el bloque previo cifrado. CFB puede modificarse para trabajar con bloques de longitud inferior a 64 bits.

OFB (Output FeedBack): similar al modo CFB excepto en que los datos sobre los que se ejecuta el OR exclusivo junto con los bloques de mensaje en claro es generada independientemente del mensaje en claro y del mensaje cifrado.

Las ventajas del algoritmo son las siguientes:

- Es el más utilizado en el mundo, lo que da lugar a que sea el más barato, más probado, utilizado por todo tipo de sistemas, etc.
- En 20 años nunca ha sido roto con un sistema práctico.
- Es muy rápido y fácil de implementar.

Cuando se utiliza el sistema DES, existen diversas consideraciones prácticas a tener en cuenta que pueden afectar la seguridad de la información cifrada. Se deben cambiar las llaves con frecuencia para prevenir ataques que requieran un sostenido análisis de la información. En un contexto de comunicaciones, se debe encontrar la forma de transmitir las llaves con seguridad entre el remitente y el receptor.

No existe ninguna prueba que garantice que un algoritmo de cifrado sea prácticamente indescifrable; lo único que existen son demostraciones de que ciertos algoritmos son vulnerables. No seria imposible diseñar una maquina que fuera capaz de probar 100 millones de llaves por segundo. Tal maquina tardaría 28 horas en probar exhaustivamente todas las llaves del DES.

2.2.2 Algoritmo Triple-DES (3DES).

Una mejora del algoritmo DES, que siempre había sido muy criticado debido a la pequeña longitud de la llave, es Triple-DES. Con este procedimiento, el mensaje es cifrado tres veces. Existen varias implementaciones:

1. DES-EEE3. Se cifra tres veces con una llave diferente cada vez.
2. DES-EDE3. Primero se cifra, luego se descifra y por último se vuelve a cifrar, cada vez con una llave diferente.
3. DES-EEE2 y DES-EDE2. Similares a los anteriores con la salvedad de que la llave usada en el primer y en el último paso coinciden.

Se estima que las dos primeras implementaciones, con llaves diferentes, son las más seguras.

El sistema Triple DES (**TDES**), esta basado en tres iteraciones sucesivas del algoritmo DES, con lo que se consigue una longitud de llave de 128 bits, y que es compatible con DES simple.

Este hecho se basa en que DES tiene la característica matemática de no ser un grupo, lo que implica que si se cifra el mismo bloque dos veces con dos llaves diferentes se aumenta el tamaño efectivo de la llave.

Para implementarlo, se toma una llave de 128 bits y se divide en 2 diferentes de 64 bits, aplicándose el siguiente proceso al documento en claro (Ver figura 2.1):

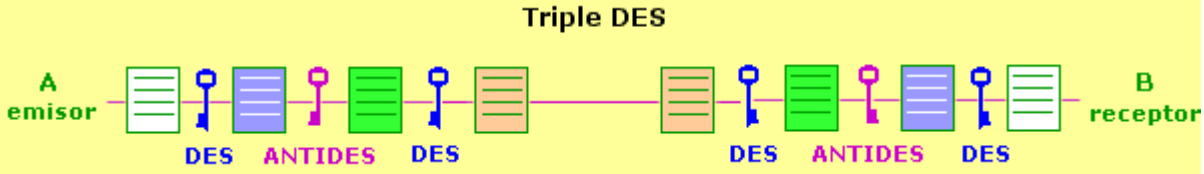


Figura 2.1

1. Se le aplica al documento a cifrar un primer cifrado mediante la primera llave, C1.
2. Al resultado (denominado ANTIDES) se le aplica un segundo cifrado con la segunda llave, C2.
3. Y al resultado se le vuelve a aplicar un tercer cifrado con la primera llave, C1.

Si la llave de 128 bits está formada por dos llaves iguales de 64 bits (C1=C2), entonces el sistema se comporta como un DES simple.

Tras un proceso inicial de búsqueda de compatibilidad con DES, que ha durado 3 años, actualmente TDES usa 3 llaves diferentes, lo que hace el sistema mucho más robusto, al conseguirse longitudes de llave de 192 bits (de los cuales son efectivos 168), mientras que el uso de DES simple no está aconsejado.

2.2.3 Algoritmo IDEA

Sistema criptográfico simétrico, creado en 1990 por Lai y Massey, que trabaja con bloques de texto de 64 bits, operando siempre con números de 16 bits usando operaciones como OR-Exclusiva y suma y multiplicación de enteros.

El algoritmo de descifrado es muy parecido al de cifrado, por lo que resulta muy fácil y rápido de programar, y hasta ahora no ha sido roto nunca, aportando su longitud de llave una seguridad fuerte ante los ataques por fuerza bruta (prueba y ensayo o diccionarios).

Este algoritmo es de libre difusión y no está sometido a ningún tipo de restricciones o permisos nacionales, por lo que se ha difundido ampliamente, utilizándose en sistemas como UNIX y en programas de cifrado de correo como PGP.

Este algoritmo IDEA[5] presenta las siguientes características:

- El espacio de llaves es mucho mas grande: $2^{128} = 3{,}4 * 10^{38}$.
- No hay operaciones a nivel de bit, facilitando su programación en alto nivel.
- El proceso de descifrado es esencialmente el mismo que el de cifrado, con la salvedad de que los sub-bloques de llave descifradas son distintos, calculándose como los inversos de los sub-bloques cifrados y también en orden inverso.

El ataque por fuerza bruta de IDEA resulta impracticable, ya que seria necesario probar 10^{38} llaves, cantidad imposible de manejar con los medios informáticos actuales y previsiblemente futuros.

2.2.4 Algoritmo RSA

Este algoritmo fue inventado por R. Rivest, A. Shamir y L. Adleman (de sus iniciales proviene el nombre del algoritmo) en el Massachusetts Institute of Technology (MIT).

[5] Algoritmo de llave secreta (cifrado simétrico).

RSA[6] emplea las ventajas proporcionadas por las propiedades de los números primos cuando se aplican sobre ellos operaciones matemáticas basadas en la función módulo. En concreto, emplea la función exponencial discreta para cifrar y descifrar, y cuya inversa, el logaritmo discreto, el muy difícil de calcular.

Ahora bien, si el número considerado es un número primo (el que sólo es divisible por 1 y por él mismo), tendremos que para factorizarlo habría que empezar por 1, 2, 3,..... hasta llegar a él mismo, ya que por ser primo ninguno de los números anteriores es divisor suyo. Y si el número primo es lo suficientemente grande, el proceso de factorización es complicado y lleva mucho tiempo.

Basado en la exponenciación modular de exponente y módulo fijos, el sistema RSA crea sus llaves de la siguiente forma:

1. Se buscan dos números primos lo suficientemente grandes: **p** y **q** (de entre 100 y 300 dígitos).
2. Se obtienen los números **n** = p * q y **X** = (p-1) * (q-1).
3. Se busca un número **e** tal que no tenga múltiplos comunes con X.
4. Se calcula **d** = e^{-1} mod X, con mod = resto de la división de números enteros.

Y ya con estos números obtenidos, n es la llave pública y d es la llave privada. Los números p, q y X se destruyen. También se hace público el número e, necesario para alimentar el algoritmo.

El RSA agrega dos funciones importantes: el intercambio seguro de llaves sin un previo intercambio de llaves secretas y firma digital. El sistema RSA es poco necesario en un entorno con un solo usuario, por ejemplo, si se desea mantener cifrados los archivos personales, solo se necesita de algún algoritmo simétrico. El RSA se aplica mejor en entornos con multiusuarios. Cualquier sistema que utilice firmas digitales necesita del RSA o de algún otro sistema con llave pública.

En cuanto a las longitudes de llaves, el sistema RSA permite longitudes variables, siendo aconsejable actualmente el uso de llaves de no menos de 1024 bits (se han roto llaves de hasta 512 bits, aunque se necesitaron más de 5 meses y casi 300 computadoras trabajando juntos para hacerlo).

[6] Algoritmo de llave publica (cifrado asimétrico).

RSA basa su seguridad es ser una función computacionalmente segura, ya que si bien realizar la exponenciación modular es fácil, su operación inversa, la extracción de raíces no es factible.

RSA es el más conocido y usado de los sistemas de llave pública, y también el más rápido de ellos. Presenta todas las ventajas de los sistemas asimétricos, incluyendo la firma digital, aunque resulta más útil a la hora de implementar la confidencialidad el uso de sistemas simétricos, por ser más rápidos. Se suele usar también en los sistemas mixtos para cifrar y enviar la llave simétrica que se usará posteriormente en la comunicación cifrada.

2.2.5 Algoritmo Digital Signature Algorithm (DSA)

Un algoritmo muy extendido es el Digital Signature Algorithm (DSA) definido en el Digital Signature Standard (DSS), el cual fue propuesto por el U.S. National Institute of Standards and Technology (NIST). Este algoritmo se basa en la función exponencial discreta en un campo de elementos finito, la cual tiene la característica de ser difícilmente reversible (logaritmo discreto).

2.3 Protocolos criptográficos

Un protocolo de seguridad es la parte visible de una aplicación, es el conjunto de programas y actividades programadas que cumplen con un objetivo especifico y que usan esquemas de seguridad criptográfica. Un protocolo resuelve problema de la distribución de llaves y de su almacenamiento.

El ejemplo más común es **SSL** (**S**ecure **S**ockets **L**ayer) que vemos integrado en el Browser de Netscape y hace su aparición cuando el candado de la barra de herramientas se cierra y también sí la dirección de Internet cambia de http a https, uno más es el conocido y muy publicitado **SET** que es un protocolo que permite dar seguridad en las transacciones por Internet usando tarjeta de crédito, **IPsec** que proporciona seguridad en la conexión de Internet a un nivel más bajo.

Estos y cualquier protocolo de seguridad procura resolver algunos de los problemas de la seguridad como la integridad, la confidencialidad, la autenticación y el no rechazo, mediante sus diferentes características

Las características de los protocolos se derivan de las múltiples posibilidades con que se puede romper un sistema, es decir, robar información, cambiar información, leer información no autorizada, y todo lo que se considere no autorizado por los usuarios de una comunicación por red.

Enseguida vemos un escenario donde puede ocurrir algo de esto:

Por ejemplo sobre la seguridad por Internet se deben de considerar las siguientes tres partes: seguridad en el browser (Netscape o Explorer), la seguridad en el Web server (el servidor al cual nos conectamos) y la seguridad de la conexión.

Un ejemplo de protocolo es **SET**, su objetivo es efectuar transacciones seguras con tarjeta de crédito, usa certificados digitales, criptografía de llave pública y criptografía llave privada.

2.4 Administración de llaves

En cualquier red de comunicaciones cifrada es preciso establecer una política de administración de llaves por cuatro motivos:

- Dado que en los actuales sistemas criptográficos no existe la seguridad perfecta, sino solamente una seguridad práctica, y que una llave queda de alguna forma expuesta cada vez que se usa, resultando más comprometida cada vez que se reutiliza, es conveniente renovar las llaves frecuentemente; en algunos casos se renuevan las llaves con cada mensaje.

- Por las mismas razones anteriores, hay que emplear llaves diferentes para cometidos diferentes: autenticación, transmisión, almacenamiento, distribución de llaves, etc.

- Es necesario dotar de llaves distintas a diferentes clientes o grupos de clientes de una red que no están autorizados a comunicarse entre sí.

- Es necesario anular las llaves utilizadas por clientes que han cesado en sus privilegios y que ya no deben tener acceso a la información.

Se puede establecer una política de administración de llaves utilizando -solamente sistemas de cifrado con llave simétrica, sólo con llave asimétrica o un método híbrido mezclando la llave asimétrica y la simétrica.

En general, es conveniente establecer una jerarquía de llaves piramidal en la que llaves de menor rango estén cifradas bajo llaves de mayor rango, de forma que las llaves de mayor rango se utilicen el mínimo de veces.

2.4.1 Tipos de llaves

Los diferentes tipos de llave que se utilizan en un sistema son :

Llave estructural

Llave común que pueden tener todos los equipos de una red y que está implementada en hardware o en memoria ROM o similar; no puede ser modificada por el usuario, sino que se cambia en fábrica o en los escalones de servicio. Se puede considerar como parte del algoritmo de cifrado; por ejemplo, las cajas 5 del DES serían una llave estructural. A veces se usa en redes en las que hay varios niveles de secreto, para aislarlos entre sí, dando a cada uno una llave estructural distinta, lo que impide que hablen entre sí clientes de distinto nivel, aunque conozcan las llaves operativas mutuas.

Llave maestra

Llave de menor jerarquía que la llave estructural, pero de máxima jerarquía entre las que puede cambiar el usuario. Solamente se usa para cifrar llaves secundarias, nunca mensajes. Se almacena sin *cifrar* en un módulo de seguridad.

Llave primaria o secundaria

Llave de menor jerarquía que la llave maestra. Se puede almacenar cifrada, bajo la llave maestra, en una memoria sin protección contra intromisiones.

Llave para generación de llaves

Llave primaria o secundaria utilizada para generar llaves de sesión y vectores de inicialización de forma aleatoria.

Llave para cifrado de llaves

Llave primaria o secundaria utilizada para cifrar otras llaves, a fin de protegerlas en su transmisión o almacenamiento.

Llave de sesión o de mensaje

Transmitida a través de la línea al principio del mensaje, cifrada bajo una llave secundaria de cifrado de llaves que modifica las llaves cargadas en el equipo. Es una llave de uso único que se utiliza solamente en el cifrado de un mensaje y luego se descarta. En su generación aleatoria se suele incluir un mecanismo que use como semilla, entre otros parámetros, la hora y fecha de generación, para evitar repeticiones involuntarias de la llave. Se puede hacer que la hora y fecha sirvan como mecanismo de rechazo de mensajes propios repetidos por un oponente.

Vector de inicialización

Puede considerarse parte de la llave de sesión; su función es determinar el punto de la serie cifrante en que empieza el cifrado. Es imprescindible en los sistemas que requieren sincronismo. En los sistemas autosincronizantes es optativa; si no se usa, se perderían los primeros bits del mensaje hasta que se produzca el autosincronismo. Se transmite a través de la línea al principio del mensaje, cifrada bajo una llave secundaria de cifrado de llaves. Es una llave que se emplea una sola vez para el cifrado de un mensaje y que posteriormente se descarta.

Llave de cifrado de archivos

Son llaves equivalentes a las dos anteriores, pero dedicadas a cifrar archivos. Se guardan en la cabecera del archivo correspondiente, cifradas bajo una llave secundaria. Se usan una sola vez para cifrar un solo archivo.

Llaves de conveniencia

Son llaves de baja seguridad para usos especiales.

Llave autollave

Procedimiento que utiliza una llave anterior para producir otra llave. Evidentemente, este procedimiento resulta poco seguro, pues si la llave anterior estaba comprometida, la nueva lo estará también.

Llave de red o de difusión

Llave que poseen todos los equipos de una red de cifra que permite que con ella puedan comunicarse todos ellos entre sí. Su seguridad es limitada, pues es fija y conocida por muchos clientes de la red.

Llave de emergencia

Llave de difusión disponible en una red de cifra para ser utilizada en sus equipos en circunstancias anormales. Su seguridad es limitada, pues es fija y conocida por muchos clientes de la red. Tiene sentido, sobre todo, en equipos militares tácticos; se usa cuando se teme que un enemigo pueda apropiarse del equipo de cifrado y ha sido preciso borrar las demás llaves. También se usa cuando las llaves ordinarias se han corrompido accidentalmente y han dejado de ser útiles. Esta llave puede ser una llave estructural de bajo nivel.

Variante de una llave

La variante de una llave consiste en otra llave que se mezcla con la primera por un procedimiento convenido, normalmente por suma módulo 2 bit a bit. Sirve para generar una tercera llave derivada de la primera. No se mantiene almacenada la llave final, sino solamente la variante. Es un buen sistema para hacer llaves secundarias, y tiene la ventaja de que al cambiar la llave maestra quedan cambiadas automáticamente todas las secundarias que se deriven de ella a través de una variante.

2.4.2 Llave maestra y almacenamiento de llaves

En todo equipo informático dotado de facilidades criptográficas, ha de existir un «módulo de seguridad» parecido al que se muestra en la figura 2.2. En este módulo está contenido el elemento que realiza el algoritmo de cifrado y descifrado, la llave maestra y, si se desea, algunas llaves de rango inferior. Está construido de modo que no es posible leer las llaves que contiene ni perturbar las operaciones de cifrado. Su apertura significa su destrucción y la de la información en él almacenada.

Las llaves de rango inferior a la maestra se podrán almacenar en el módulo de seguridad para reforzar su protección, pero ello resulta caro en caso de desear almacenar muchas llaves. En una estructura jerárquica de llaves óptimamente diseñada, el número de éstas crecerá.

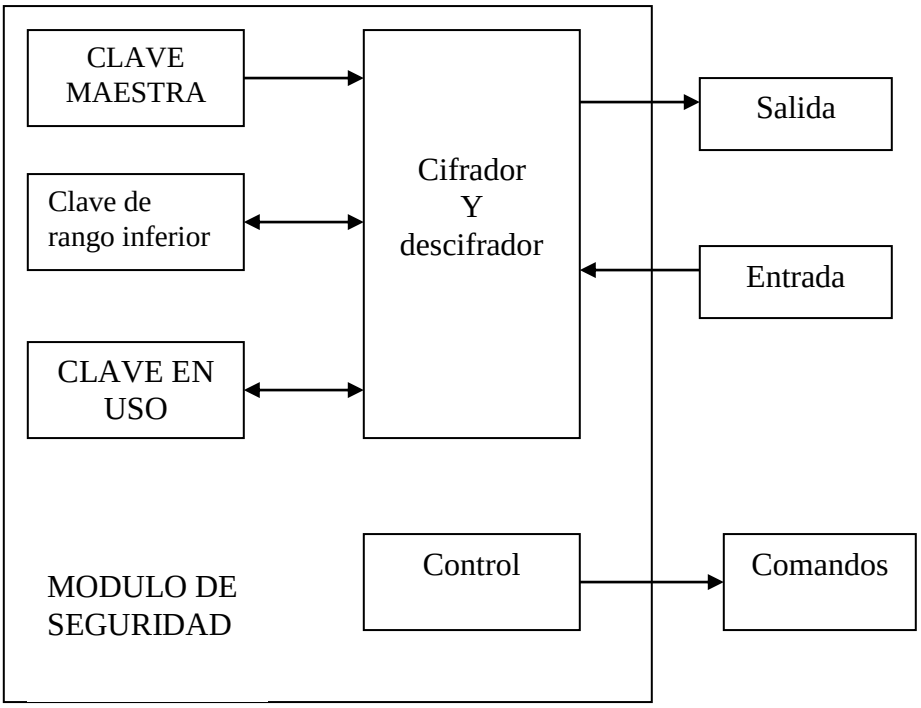


Figura 2.2 —Módulo de seguridad de un equipo criptográfico

De forma similar al de una pirámide; por tanto, sería aconsejable guardar los últimos rangos fuera del módulo seguro, en la memoria general del equipo, en donde quedarían igualmente protegidas, ya que están cifradas bajo una llave de rango superior.

2.5 JAVA

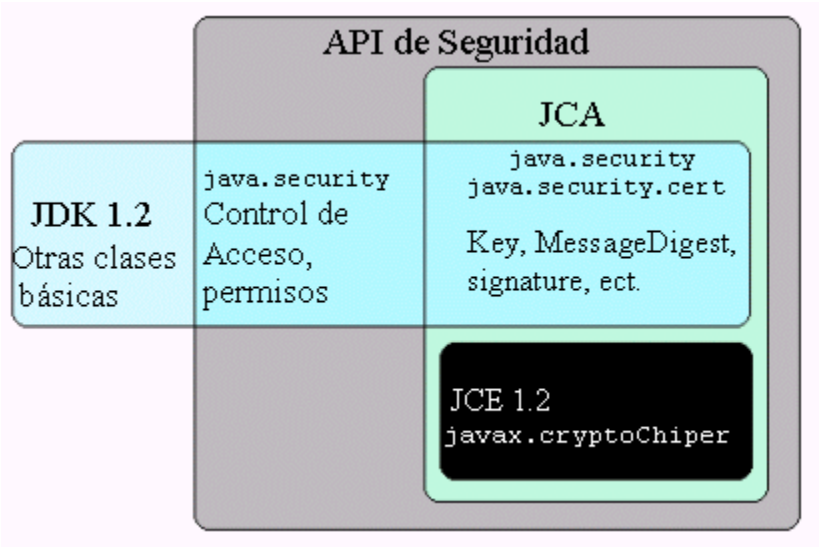
Existen bibliotecas de fines generales que ayudan a la implementación de algoritmos criptográficos; uno de los lenguajes que ha implementado estos algoritmos es Java.

Una ventaja de Java sobre otros lenguajes es que tiene integrados mecanismos de seguridad. Estos mecanismos están pensados para garantizar la seguridad en los siguientes puntos:

- Aspectos del diseño del lenguaje: verificación del tamaño de los arreglos; sólo se permite conversión legal de tipos; no hay aritmética de apuntadores, entre otros.
- Java es un lenguaje de programación multiplataforma.
- Un mecanismo llamado ``caja de arena" (*sandbox*) controla lo que puede hacer el código.
- Firma de código. Los autores del código pueden certificarlo utilizando algoritmos criptográficos; de esta manera es posible saber quién hizo el código y si sufrió modificaciones después de que fue firmado.
- El API de seguridad de Java es extendido por el paquete JCE (*Java Cryptography Extension*) que añade interfaces de encriptamiento. Este paquete está separado del API principal porque utiliza código que no es exportable fuera de Estados Unidos y Canadá.

El API de seguridad está incluido en Java API en la forma del paquete java.security. Este paquete provee dos API's, uno para los usuarios de los algoritmos de seguridad y otro para implementadores o proveedores de estos algoritmos.

Diagrama de la API de seguridad de Java



En los últimos 50 años los matemáticos y los científicos de la computación han desarrollado algoritmos que aseguran la integridad de los datos y que permiten hacer firmas digitales. El paquete java.security contiene implementaciones para muchos de estos algoritmos.

El API para usuarios está diseñado para que los distintos algoritmos criptográficos sean utilizados en una aplicación, sin tener que preocuparnos por la manera en la que éstos han sido implementados. Lo único que se necesita saber es el nombre del algoritmo. Una compañía o algún programador puede añadir sus propias implementaciones de los algoritmos usando la interfaz Provider.

En general el API de seguridad incluye interfaces para hacer manejo de identidades, para utilizar firmas digitales y para cifrar datos.

En el API de seguridad en Java se hace uso de la criptografía de llave pública y de llave privada. La interfaz Key y las clases KeyPair, KeyGenerator y KeyPairGenerator son la base para manejar y crear las llaves.

La interfaz Key representa una llave criptográfica. La llave puede ser pública o privada para algoritmos de llave pública, o puede ser una llave secreta. A cada Key se le asocia un algoritmo. La generación de llaves depende del algoritmo. PublicKey, PrivateKey y SecretKey son subclases de Key que sirven para distinguir entre los distintos tipos de llave.

La clase KeyPair contiene una PublicKey y la correspondiente PrivateKey. Los pares de llaves son generados usando la clase KeyPairGenerator. En los algoritmos de llave secreta se usa objetos del tipo SecretKey; estos son generados usando la clase KeyGenerator.

En Java están implementados los algoritmos *SHA1* y *MD5* para calcular los condensados de mensaje. La clase MessageDigest es una fábrica para crear objetos que encapsulan los algoritmos de huellas digitales.

El paquete de seguridad de Java tiene incluido al algoritmo *DSA* (*Digital Signature Algorithm*) que se basa en el uso de esta criptografía. Otro algoritmo conocido es *RSA* (inventado por Rivest, Shamir y Adleman), para utilizar este algoritmo en Java es necesario comprarlo.

Dadas esta características se instalo el Kit de Desarrollo de JAVA jdk versión 1.4[7], con el que sean realizado programas de prueba utilizando algoritmos implementados en JAVA como por ejemplo el algoritmo DSA[8], DES[9] y SHA1 .

2.5.1 JAVA Swing

Es un componente principal de la JFC lo cual es el resultado de un esfuerzo de colaboración muy grande entre Sun, Netscape, IBM y otras empresas. Swing proporciona una gran cantidad de controles GUI cuyo origen encontramos en las clases de fundamentos de Internet en Netscape (IFC).

Aunque Swing este separado del AWT (Abstract Windowing Toolkit), se implementa en términos de clases AWT básicas. El AWT proporciona la interfaz entre el sistema de ventanas nativos subyacente y los componentes GUI de JAVA. Swing utiliza esta interfaz, pero no se apoya en componentes del AWT para hacer uso de objetos nativos. En lugar de ello, los componentes Swing están escritos en JAVA puro.

[7] Versión beta
[8] Utilizado para realizar firma digital.
[9] Con el que realizamos encripcion y desencripcion de texto

Esto ofrece ventajas significativas. Permite a los componentes Swing ser independientes del sistema de ventanas nativo, lo cual implica que pueden ejecutarse en cualquier sistema de ventanas que admita el AWT.

Swing también ofrece una implementación de JAVA puro de muchos de los componentes tradicionales del AWT. Esos componentes tienen la misma funcionalidad que los componentes del AWT y todas las ventajas de Swing. Swing es compatible con el AWT, y los componentes Swing se pueden utilizar con los componentes de AWT.

2.6 SQL Server

El Lenguaje Normalizado de Consultas (SQL) es un lenguaje que sirve para interactuar con las Bases de Datos Relacionales, fue desarrollada por IBM durante los años 70 y 80 y estandarizada a finales de los 80.

SQL tiene muchos usos. Cuando SQL se usa para crear o diseñar una base de datos, se trata de un lenguaje de definición de datos. Cuando se utiliza para actualizar los datos que hay en una base de datos se trata de un lenguaje de mantenimiento de datos. Cuando se usa para recuperar información de una base de datos, se trata de un lenguaje de consulta de datos.

Uso de SQL como lenguaje de definición de datos: Se puede utilizar SQL para definir una base de datos. Por ejemplo, existen instrucciones SQL para crear una base de datos, para crear tablas y agregarlas a la base de datos, para actualizar el diseño de las tablas existentes y para eliminar tablas (CREATE DATABASE, CREATE TABLE, ALTER TABLE, DROP TABLE). Sin embargo, la mayoría de sistemas de bases de datos proporcionan herramientas GUI para definir la base de datos, y es mucho mas fácil trabajar con esas herramientas que hacerlo con SQL a la hora de diseñar la base.

Uso de SQL como lenguaje de mantenimiento de datos: Uno de los usos principales de SQL consiste en actualizar los datos de una base. Existen instrucciones SQL para insertar filas nuevas en una base de datos, para eliminar filas y para actualizar filas existentes (INSERT, DELETE, UPDATE).

Uso de SQL como lenguaje de consulta de datos: Para muchos usuarios el uso mas importante de SQL consiste en recuperar los datos que hay en una base de datos. La estructura básica de una instrucción SQL consiste en tres cláusulas: SELECT, FROM y WHERE.

2.6.1 Controladores JDBC

Los clientes de bases de datos utilizan controladores de bases de datos para enviar instrucciones SQL a servidores de bases de datos y recibir conjuntos de resultados y otras respuestas de los servidores. Los controladores JDBC los usan los Applet y aplicaciones de JAVA para comunicarse con los servidores de la base de datos. JDBC ofrece una API común de programación de base de datos para programas de JAVA. Sin embargo los controladores JDBC todavía no se comunica directamente con tantos productos de base de datos como los controladores ODBC (Conectividad Abierta de Base de Datos).

En lugar de ello muchos controladores JDBC se comunican con las bases de datos a través de la ODBC. De hecho uno de los primeros controladores JDBC fue el controlador puente JDBC-ODBC desarrollado por JavaSoft e InterSolv.

Para usar la JDBC, se necesita un servidor de base de datos (la JDBC ofrece un enfoque independiente del servidor al acceso de base de datos) y un controlador de base de datos, el cual proporciona el vínculo entre la JDBC y la base de datos. La JDBC viene con un puente JDBC-ODBC. Este puente permite acceder a bases de datos a través de la API Open DataBase Conectivity de Microsoft.

3. Análisis y Diseño del sistema

Se tiene un sistema que instalado en la máquina del usuario permite ejecutar las diferentes herramientas criptográficas, presentadas a través de una interfaz fácil de utilizar.

A continuación se presenta el diagrama 3.1 de contexto de nuestro sistema, en el cual podemos observar los flujos de entrada que acepta el sistema, así como los flujos de salida.

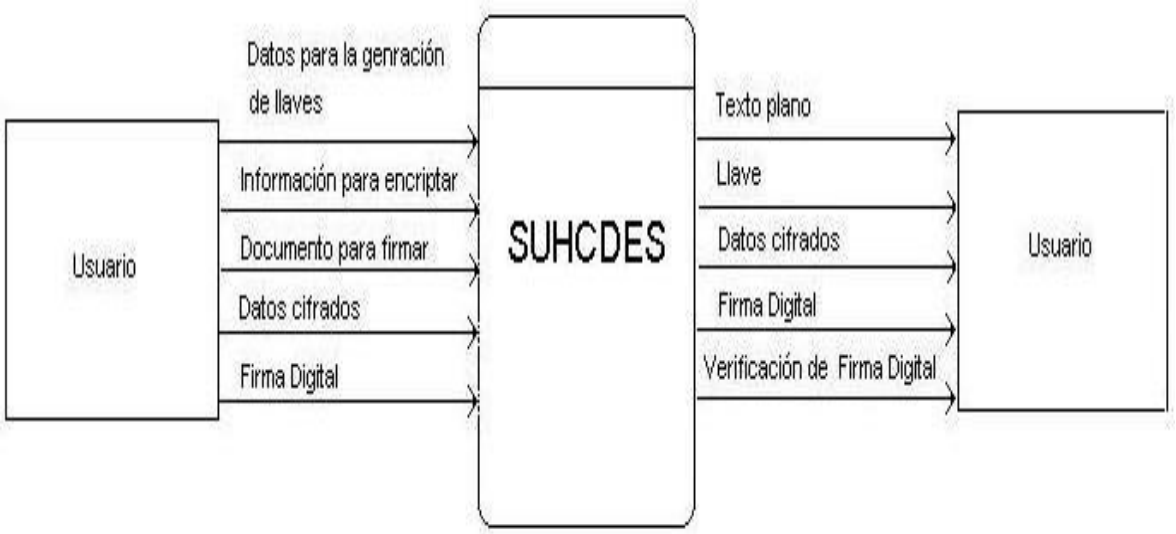


Figura 3.1

El sistema cuenta con ayuda, para que el usuario pueda consultarla si se le presenta alguna duda acerca del funcionamiento del programa. Vendrá acompañado de un manual de usuario que contiene información suficiente para que el propio usuario pueda operar el sistema.

El sistema cuenta con los siguientes módulos:

- Módulo de cifrado/descifrado de archivos y/o documentos electrónicos.
- Módulo de administración de llaves criptográficas.
- Módulo de creación y verificación de firmas digítales
- Módulo de administración del Sistema

DFD nivel 0

En este diagrama 3.2 se muestra los módulos que contendrá el sistema, así como los datos que se relacionan entre ellos.

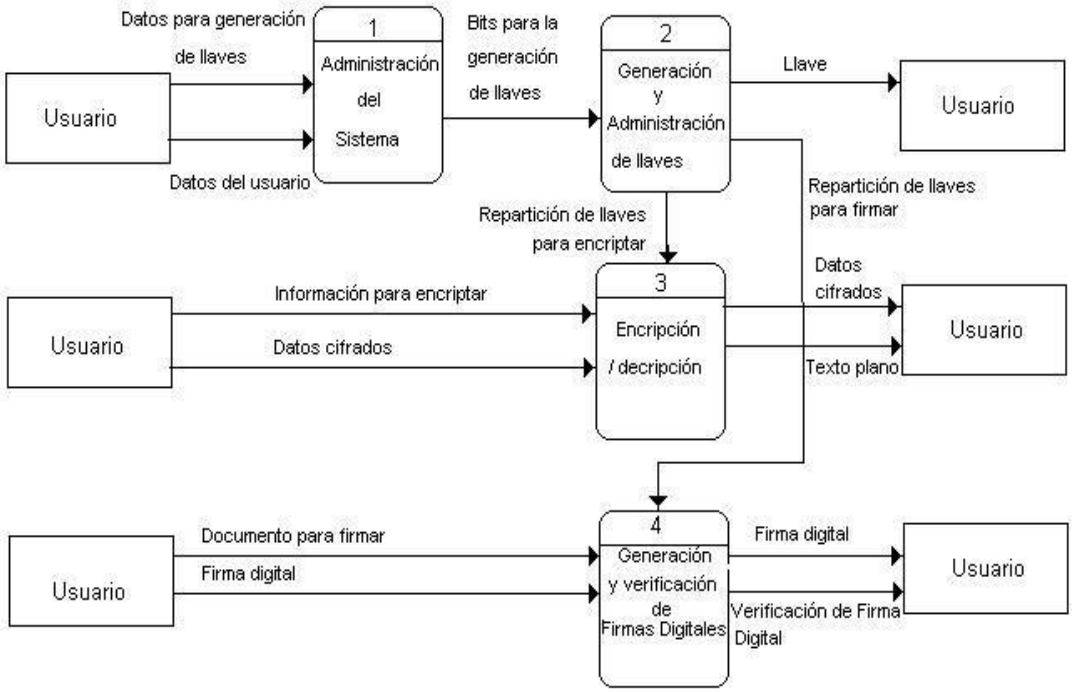


Figura 3.2

Usuario

Entidad externa al sistema que le proporcionara la información necesaria para que el sistema pueda garantizar la seguridad de la información que ingrese, dependiendo de la herramienta que desee utilizar; es decir, si requiere de cifrar y/o descifrar un archivo[10] y/o realizar un firmado digital[11].

[10] Garantizar la confidencialidad.
[11] Para garantizar la autentificación e integridad de la información.

3.1 Módulo de cifrado/descifrado de archivos y/o documentos electrónicos.

Este módulo tendrá tres entradas: el texto plano[12], el criptograma[13] y la llave de cifrado y/o descifrado (ver figura 3.3). El usuario introducirá al sistema el texto plano o el criptograma (según sea el papel que este desempeñando el usuario, emisor o receptor respectivamente), entendiendo que la información que se recibirá del usuario será cualquier documento o archivo electrónicos. El sistema[14] proporcionara las llaves para la cifrado y descifrado (Ver figura 3.4a y 3.4b).

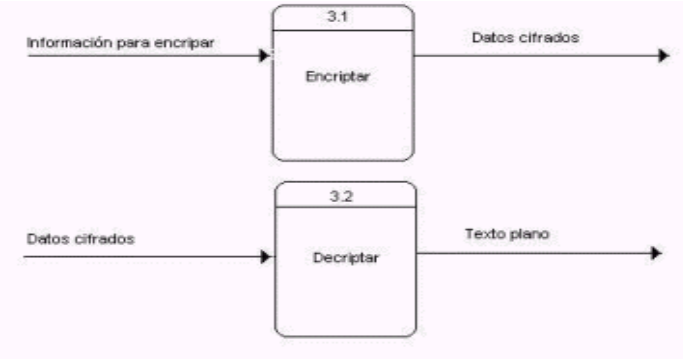


Figura 3.3

Se aplican algoritmos de cifrado (DES, TDES, DSA, RSA) que define dos transformaciones:

- **Cifrado:** se realiza la conversión del texto en claro en el texto cifrado o criptograma mediante el empleo de la denominada llave de cifrado.

- **Descifrado:** es el proceso inverso al cifrado para la cual se emplea la llamada llave de descifrado.

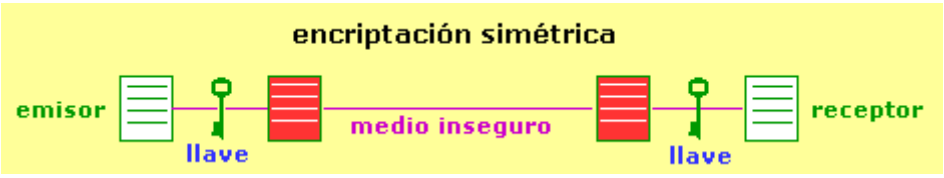


Figura 3.4a

[12] Se llama así al documento antes de ser encriptado.

[13] Documento o texto cifrado.

[14] Proporcionadas por el modulo de Generación y Administración de llaves.

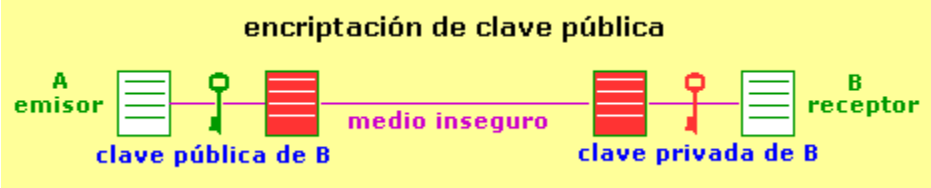


Figura 3.4b

La aplicación mas inmediata de este módulo (aunque no es la única) es la de asegurar el servicio de confidencialidad: la información transmitida no se podrá descifrar sin el conocimiento de la llave de descifrado.

3.2 Módulo de generación y administración de llaves criptográficas.

Este modulo tendrá únicamente como entrada, los bits para la generación de llaves. Una parte importante para la aplicación de los algoritmos criptográficos es la generación de llaves para cifrar y descifrar (Ver figura 3.5).

Esta generación se realizara a través de las clases que nos proporciona JAVA para este fin, especificando el tipo de algoritmo o función utilizar.

Pero también se necesita establecer una política de administración de llaves, ya que en los actuales sistemas criptográficos no existe la seguridad perfecta, sino solamente una seguridad practica y que una llave queda de alguna forma expuesta cada vez que se usa, resultando mas comprometida cada vez que se reutiliza, es conveniente renovar las llaves frecuentemente; que en algunos casos (según lo desee el usuario) se renovaran las llaves.
Para el almacenamiento de llaves e información de usuario se utiliza el manejador de Bases de Datos SQL Server 7.0

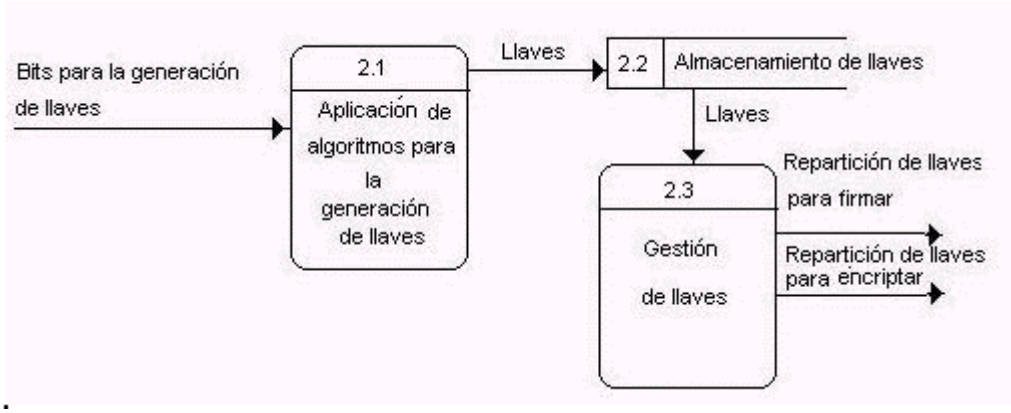
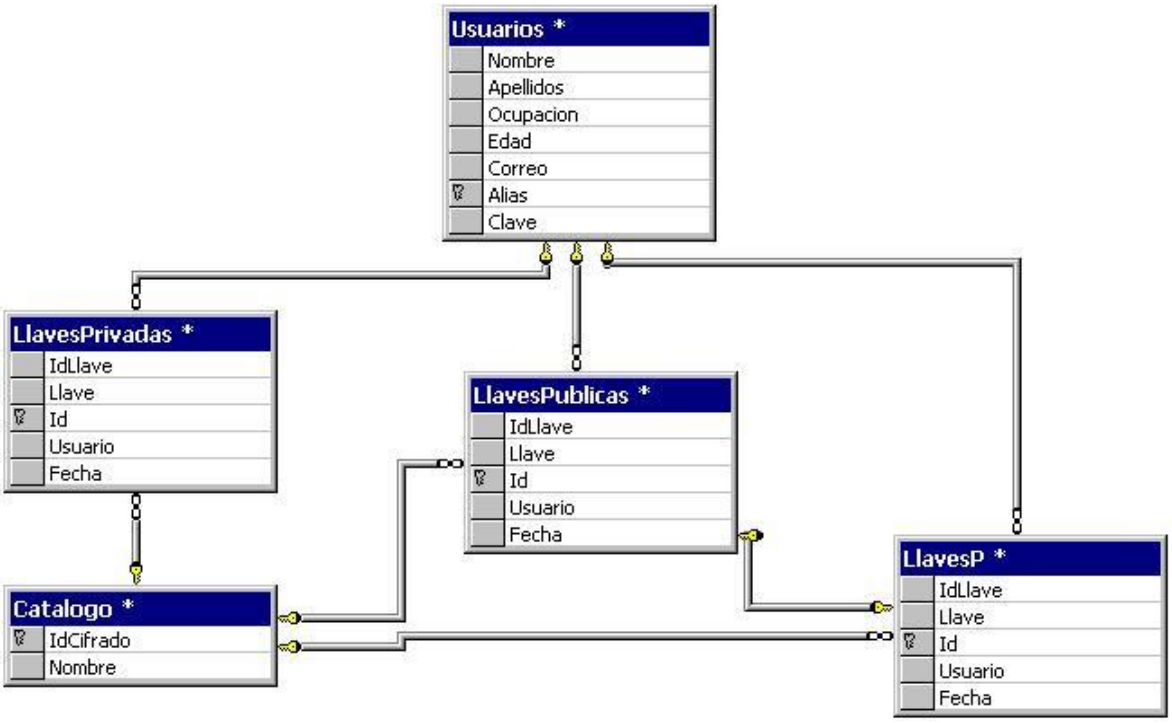


Figura 3.5

Diagrama de Relaciones



Los usuarios deben contar con un método de almacenamiento seguro de llaves: ningún intruso podrá encontrarlas pero deben estar al alcance de la mano para su uso. Las llaves son válidas hasta su fecha de vencimiento, que será elegida adecuadamente y publicada en un medio seguro.

3.3 Modulo de generación y verificación de firmas digitales.

Obtendrá como entrada el documento a firmar y/o el documento a verificar además de la llave correspondiente proporcionada por el sistema (módulo de generación y administración de llaves), realizando lo siguiente (ver Figura 3.6):

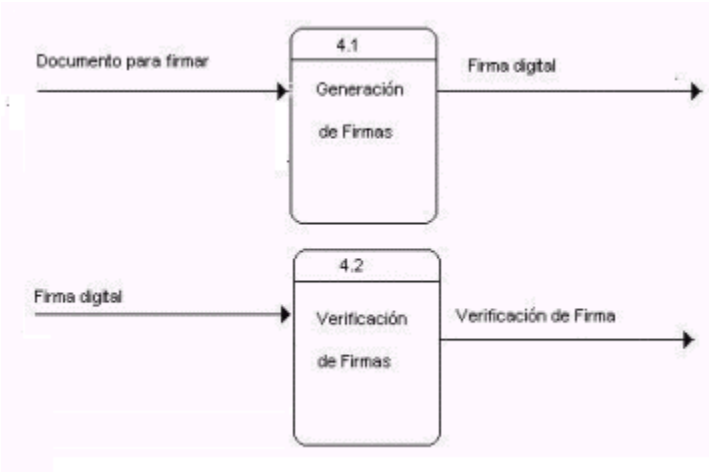


Figura 3.6

1. **Proceso de Firma**: 1. Se calcula el condensado de mensaje. 2. El condensado de mensaje se cifra a través de una llave privada de un par de llaves pública/privada, creándose la firma digital del mensaje (Ver figura 3.7).

2. **Proceso de Verificación de la Firma**: 1. La firma de descifra a través de la llave pública de un par de llaves pública/privada, creándose un valor de condensado de mensaje. 2. El valor del condensado de mensaje se compara con el condensado de mensaje calculado a partir del mensaje original. 3. Si ambos valores de condensado coinciden, la firma será autentica. Si no, la firma o el mensaje se abran entrometido.

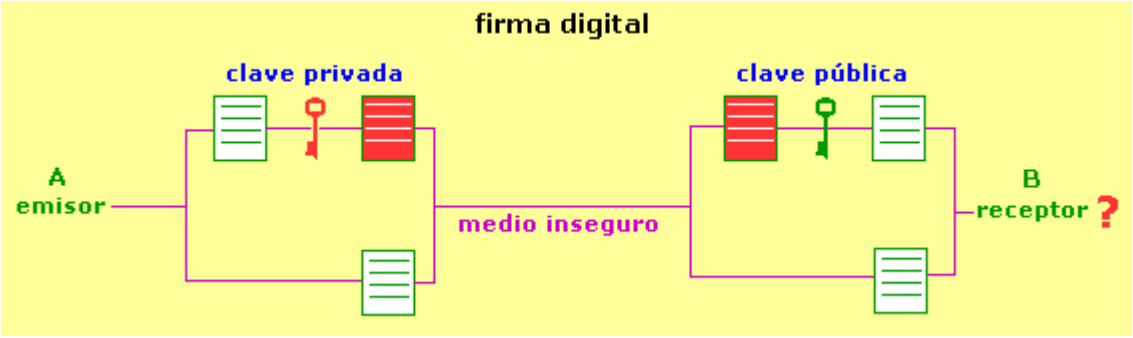


Figura 3.7

El receptor descifra el documento cifrado con la llave pública de A y comprueba que coincide con el documento original, lo que atestigua de forma total que el emisor del mismo ha sido efectivamente A.

3.4 Modulo de administración del sistema

Este modulo se encargara de manejar los datos del usuario y asegurar que estos tengan acceso únicamente a sus llaves, esto se va lograr por medio de un inicio de sesión el cual va a hacer ser accesado a través de un login y password que se obtendrán cuando se cree un nuevo usuario (Ver figura 3.8).

Los datos de cada usuario se almacenan en una base de datos local que además de contener el login y password de usuario contendrán datos generales de estos.

Figura 3.8

4. Alternativas

4.1 Solución a las administración de llaves

Keystores (Almacenes de Llaves)

Las llaves privadas y sus certificados de llaves públicas asociadas están almacenadas en unas bases de datos protegidas por passwords llamadas **keystores**. Un keystore puede contener dos tipos de entradas: las entradas de certificados verdaderos, y entradas llave/certificado, cada una de ellas contiene un llave privada y el correpondiente certificado de la llave pública. Cada entrada en el keystore está identificada por un **alias**.

Un propietario de un keystore puede tener múltiples llaves en él, accedidas mediante diferentes alias. Un alias se nombra típicamente según las reglas particulares en las que el propietario del keystore usa las llaves asociadas. Un alias también podría identificar el propósito de la llave. Por ejemplo, el alias **signPersonalEmail** podría usarse para identificar una entrada del keystore cuya llave privada es usada para firmar e-mail personales, y el alias **signJarFiles** podría usarse para identificar una entrada cuya llave privada se usara para firmar archivos JAR.

La herramienta **keytool** puede usarse para.

- Crear llaves privadas y sus certificados de llaves públicas asociadas.
- Emitir peticiones de certificados, que enviamos a la autoridad de certificación apropiada.
- Importar respuestas de certificados, obtenidos desde la autoridad de certificación contactada.
- Importar certificados de llaves privadas pertenecientes a otras partes como certificados verdaderos.
- Manejar nuestro keystore

De acuerdo a las necesidades de SUHCDES la herramienta que nos proporciona JAVA (KeyStore) no cumplía con las expectativas de nuestro sistema, por tal motivo se decidió implementar la administración y almacenamiento de llaves mediante el soporte de un manejador de base de datos como SQL Server.

4.2 Solución a la elección de nivel de seguridad en el sistema

Durante el desarrollo del sistema se estudiaron dos posibles alternativas para la elección en el nivel de seguridad para la administración de llaves:

- Que el administrador del sistema decidiera el nivel de seguridad que pudiera requerir, lo cual limitaba la garantía de seguridad que pudiera tener.
- Al utilizar un manejador de bases de datos el usuario tiene la posibilidad de elegir el nivel de seguridad que se adecuara a sus necesidades.

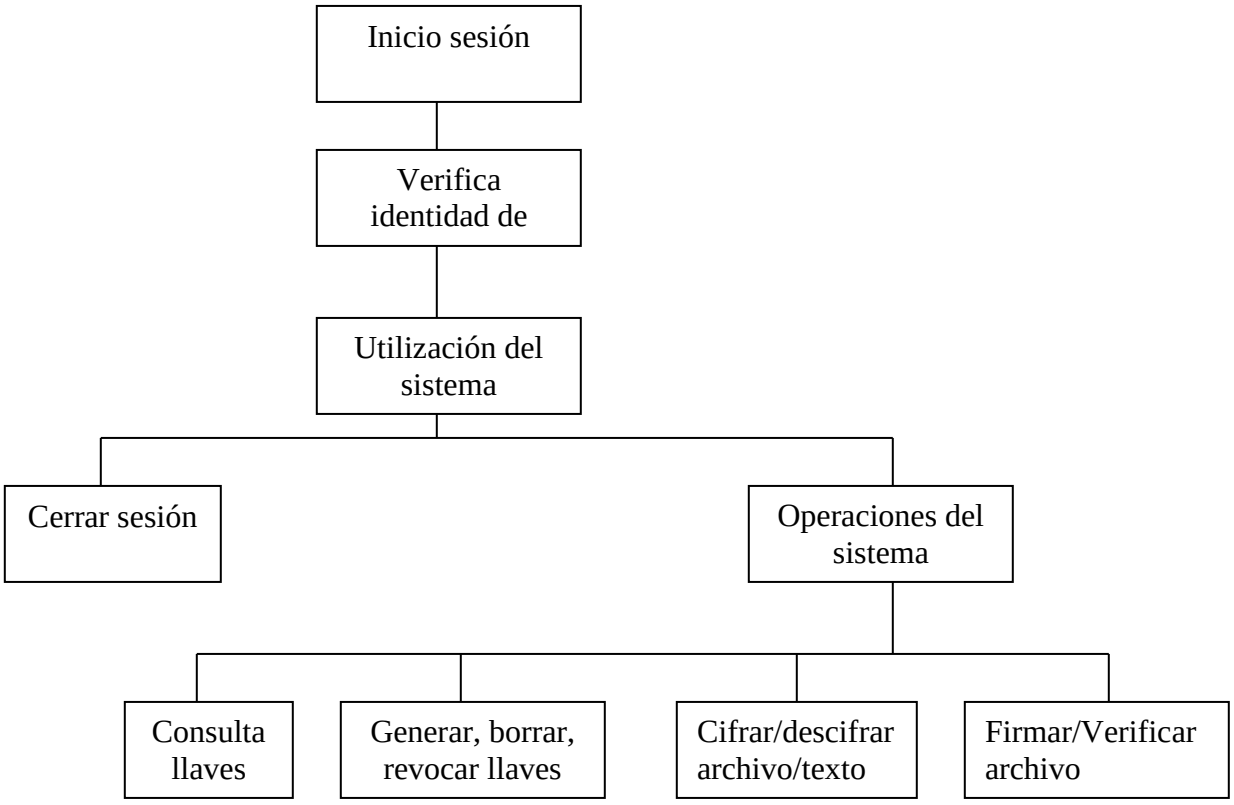
Gracias a las facilidades en el manejo de datos que nos proporciona SQL Server, el usuario al generar sus llaves podrá elegir el nivel se seguridad que requiera, evitando así que el administrador del sistema decida ese nivel.

5. Funcionamiento Del Sistema.

5.1 Aspectos generales.

En la búsqueda de la mejor solución para resolver el problema de seguridad, y así poder implantar una aplicación que pudiera mantener íntegra la información del usuario, se desarrollo un sistema que brinda cubrir las necesidades de confidencialidad, integridad y autentificación de la información mediante técnicas como son: el cifrado/descifrado de archivos o texto y generación/verificación de firmas digitales, dándole al usuario la opción de elegir el nivel de seguridad que se ajuste a sus necesidades.

Diagrama del funcionamiento de SUHCDES



5.2 Funcionalidades del sistema

Registro de Usuario

Esta función le permite al usuario darse de alta en el sistema (Ver figura 5.1), de tal manera que pueda utilizar las aplicaciones que este ofrece tal como cifrar/descifrar archivos o texto, así como generar/verificar firmas digitales obteniendo sus propias llaves (Llaves Secretas, Públicas-Privadas).

Figura 5.1

Los campos marcados con asterisco no se podrán omitir, ya que son datos llave para la identificación del usuario en el sistema y para la administración de sus llaves.

A través de una Llave Maestra se cifraran los campos Alias (login) y Llave antes de ser insertados en la base de datos, con la finalidad de resguardar la identidad del usuario.

Validaciones:

- El sistema no permitirá que existan Alias repetidos, para evitar conflictos con la administración de las llaves.
- No permite que el Alias y la Llave del usuario sean iguales, con la finalidad de garantizar la confidencialidad y autentificación del usuario en el sistema.

Generación de Llave Maestra

Para poder garantizar la seguridad de las llaves, alias y llaves de usuario dentro de la base de datos, es necesario generar una Llave Maestra (utilizando Cifrado Simétrico) que cifre la información antes de insertarla en la base de datos, de tal manera que esta permanezca confidencial ante observadores no autorizados.

Esta Llave Maestra será actualizada periódicamente a criterio del administrador del sistema.

Inicio de Sesión

Una vez el usuario dado de alta en el sistema podrá ingresar a este mediante el inicio de sesión, en donde se comprobará la identidad del usuario (Ver figura 5.2).

Figura 5.2

Al iniciar sesión, el usuario tendrá acceso únicamente a sus llaves secretas y públicas-privadas además de las llaves públicas de los demás usuarios registrados.

Validaciones:

- Si no son correctos los datos que ingresa el usuario a través del cuadro de dialogo, no se le permitirá el acceso al sistema.

Consultar Llavero de Usuario

El usuario visualizará sus propias llaves [Públicas-Privadas y Secretas], teniendo la oportunidad de consultar la información referente a ellas tal como su identificador, fecha de expedición, algoritmo de generación , etc (Ver figura 5.3).

Podrá generar nuevas llaves Públicas-Privadas (para el Cifrado Asimétrico o Firma Digital), así como llaves secretas (para el Cifrado Simétrico).

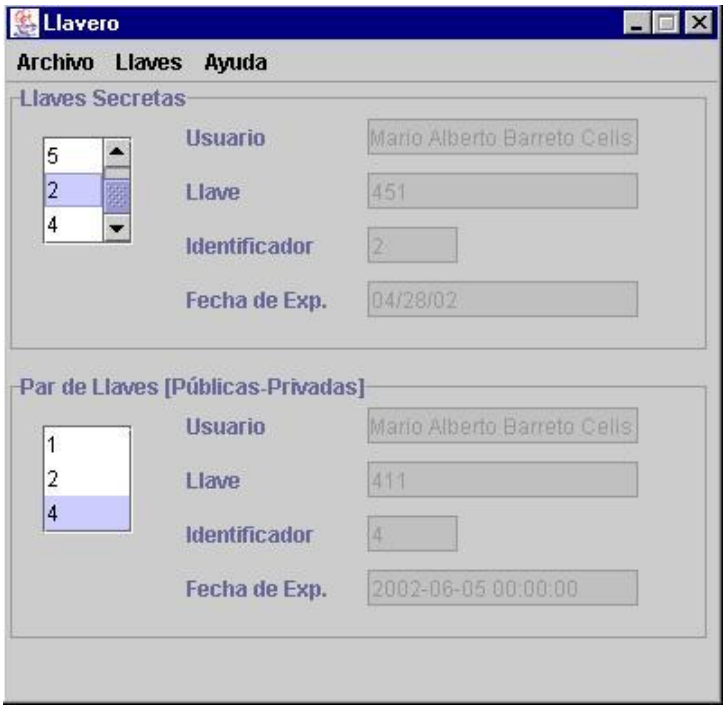


Figura 5.3

Desde el Llavero el usuario podrá eliminar o revocar la llave que seleccione en la lista, sea llave Secreta o Pública-Privada.

Validaciones:

- Para poder visualizar los datos de las llaves, tendrá que seleccionar primero un identificador de la lista.
- Cuando desee eliminar o revocar una llave el usuario tiene que seleccionar de la lista el identificador de esta.

Generación de Llaves [Públicas-Privadas, Secretas]

A través del llavero se podrán generar las llaves, pidiendo una cadena de caracteres para poder generarlas (Figura 5.4). Esta cadena de caracteres podrá identificar las llaves.
El usuario deberá indicar al sistema el nivel de seguridad que requiere antes de generar las llaves, teniendo como opción:

Nivel de seguridad alto: Para el caso de llaves secretas se usa el algoritmo de cifrado TDES y para las llaves Públicas-Privadas se usa el algoritmo de cifrado RSA con longitud de llave de 1024 bits.

Nivel de seguridad bajo: Para las llaves secretas se usa el algoritmo de cifrado DES y para las llaves Públicas-Privadas se usa el algoritmo de cifrado DSA con longitud de llave de 1024 bits.

Figura 5.4

Validaciones (ambos casos, llaves públicas-privadas y secretas):

- Validará que la cadena de caracteres ingresados no haya sido utilizada por otro usuario o por el mismo.
- No generara la llave si no se teclea la cadena de caracteres.

Cifrado/Descifrado de Texto

El usuario cifrará/descifrara el texto que teclee en el área señalada en la interfaz, eligiendo la llave con la cual desea cifrarlo, teniendo en cuenta que deberá utilizar la misma llave para descifrarlo; es decir, seguir el protocolo de cifrado del sistema. Al elegir la llave, se visualiza información referente a ella (Figura 5.5).

Figura 5.5

Una vez elegida la llave y tecleado el texto a cifrar, se procederá a cifrarlo; obteniendo el texto cifrado en el área señalada como “Texto Cifrado”.

Funciones utilizadas:

Para poder realizar el cifrado de texto, se implemento una clase llamada Base64. Base64 es un sistema para representar un arreglo de bytes de caracteres ASCII de forma eficiente.

Clase javax.cripto.Cipher: Cipher cifra o descifra datos. La clase Cipher contempla algoritmos simétricos (llaves secreta) y asimétricos (llave pública). Esta clase forma parte de JCE. Cipher es una clase abstracta y no puede ser inicializada de forma directa. Para usar Cipher seguimos tres pasos:

- Obtener el tipo de algoritmo a utilizar usando el método getInstance()
  Cipher cipher = Cipher.getInstance("DES/ECB/PKCS5Padding");
- Inicializar Cipher para cifrar o descifrar usando el método init(). Este método acepta dos modos, Cipher.ENCRYPT_MODE o Cipher.DECRYPT_MODE y una llave. El tipo de llave a usar depende de el tipo de algoritmo que se desee utilizar (llaves, secreta, llave pública o llave privada)

```
cipher.init(Cipher.ENCRYPT_MODE, key);
```

- Cifra o descifra los datos usando el método update() y doFinal().

```
Base64 decoder = new Base64();
byte[] raw = decoder.decode(String);
byte[] stringBytes = cipher.doFinal(raw);
```

Validaciones:

- Si no se elige la llave con la cual se cifrará, no llevará a cabo la aplicación de cifrado.

Cifrado/Descifrado de Archivo

En esta opción, el usuario tendrá la posibilidad de cifrar/descifrar un archivo, eligiendo la llave con la cual desea cifrarlo, teniendo en cuenta que deberá utilizar la misma llave para descifrarlo; es decir, seguir el protocolo de cifrado propio del sistema. Al elegir la llave, se visualiza información referente a ella (Figura 5.6).

Deberá especificar la ruta del archivo que desea cifrar, además del nombre del archivo cifrado (Figura 5.7).

Figura 5.6

Figura 5.7

Funciones utilizadas:

Clase javax.crypto.CipherOutputStream: CipherOutputStream es una subclase de java.io.FilterOutputStream.

```
FileOutputStream fileOut = new FileOutputStream(fileDestino);
CipherOutputStream out = new CipherOutputStream(fileOut, cipher);
```

Generación/Verificación de Firma

El usuario podrá asegurar la integridad y autentificación de archivos a través de llaves públicas, teniendo en cuenta que se utilizará la llave privada para firmarlo y la llave pública correspondiente para verificar.

Indicará el archivo a firmar/verificar, además del archivo donde se almacenara la firma (Figura 5.8).

Figura 5.8

Funciones utilizadas:

KeyPairGenerator: De la clase java.security.KeyPairGenerator creea una réplica de una llave pública y privada, retornando entonces un objeto KeyPair. Se puede crear el

KeyPairGenerator usando el método getInstance().

KeyPairGenerator keyGen = KeyPairGenerator.getInstance("DSA", "SUN");

SecureRandom: Esta clase genera números aleatorios. El estándar de la generación de números aleatorios es diferente a la generación de números a través de la clase de seguridad criptográfica.

SecureRandom random = SecureRandom.getInstance("SHA1PRNG", "SUN");

keyGen.initialize(1024, random);

java.security.PublicKey: Esta interfaz representa la llave pública del par de llaves, apropiado para utilizarlo en la generación de firmas digitales o para cifrado asimétrico

PublicKey pub = pair.getPublic();

java.security.PrivateKey: Esta interfaz representa la otra mitad del par de llaves, en este caso la llave secreta es usada para generar la firma.

PrivateKey priv = pair.getPrivate();

Signature: La clase Signature, es una clase motor diseñada para proporcionar la funcionalidad de un algoritmo de firma digital como el DSA o RSA con SHA1. Un algoritmo de firma toma una entrada de tamaño arbitrario y una llave privada lo cual generará una cadena de bytes relativamente cortas, llamada firma.

Signature dsa = Signature.getInstance("SHA1withDSA", "SUN");

dsa.initSign(key);

6 Conclusiones

En una comunicación segura los datos se envían a través del canal de comunicación tal como son, sin sufrir modificaciones de ningún tipo y para garantizarlo debemos preservar esta información frente a observadores no autorizados. Durante el desarrollo del sistema se pudo comprobar que una de las disciplinas que ha tenido mas éxito en la solución de problemas de seguridad computacional es la criptografía.

De tal forma que SUHCDES es un sistema capaz de poner al alcance del usuario un conjunto de herramientas criptográficas, que puedan garantizar la seguridad de su información a través de técnicas como el cifrado simétrico, cifrado asimétrico y firmas digitales.

7. Referencias bibliografías

Técnicas Criptográficas de protección de datos.
Amparo Fúster Sabater, Dolores de la Guía Martínez
Editorial ra-ma

JAVA Cryptography.
Jonathan Knudsen
O'Reilly

Applied Cryptography, Second Edition
Protocols, Algorithms, and Source Code in C.
Bruce Schneier

Java 1.2 Al descubierto
Jaimie Jaworski
Editorial:. Prentice hall

El ABC de los documentos electrónicos seguros.
Ignacio Mendìvil

Criptografía y Seguridad en Computadores.
Manuel José Lucena López
Departamento de Informática
Escuela Politécnica Superior

JAVA Magazine No. 3
Revista especializada para desarrolladores de JAVA.
Articulo: Criptografía y JAVA
Autor: J. Leonardo Antolì del Amo

http://www.java.sun.com

http://www.kriptopolis.com

http://www.cryptix.org

http://www.rsa.com