Sistema de mensajería instantánea punto a punto mediante cifrado por intercambio de clave de sesión.

Oriol Baratech Soler
ETIS
Consultor: **Antoni Martínez Ballesté**
10 de enero de 2006

INDICE

1 Especificación del problema

El objetivo del proyecto es crear una aplicación de mensajería instantánea[1] punto a punto del estilo de ICQ o MSN Messenger con la particularidad que la información viaje cifrada. De este modo los participantes en la comunicación generarán automáticamente una clave de sesión con la que se cifrarán los mensajes que intercambien.

Los requisitos mínimos son los siguientes:

- El programa ha de tener una pequeña interfaz gráfica.
- Se deben poder mantener varias conversaciones simultáneamente.
- Para generar las claves de sesión se utilizará el protocolo de intercambio de claves Diffie-Hellman. Los valores públicos residirán en una "agenda" que la aplicación consultará para poder realizar llamadas.

Este proyecto se realiza como parte de la asignatura **Treball de Final de Carrera** para completar los estudios de **Ingeniería Técnica en Informática de Sistemas** de la **Universitat Oberta de Catalunya**.

[1] La mensajería instantánea (conocida también en inglés como IM) requiere el uso de un cliente informático que realiza el servicio de mensajería instantánea y que se diferencia del correo electrónico en que las conversaciones se realizan en tiempo real. La mayoría de los servicios ofrecen el "aviso de presencia", indicando cuando el cliente de una persona en la lista de contactos se conecta o en que estado se encuentra, si está disponible para tener una conversación. En los primeros programas de mensajería instantánea, cada letra era enviada según se escribía y así, las correcciones de las erratas también se veían en tiempo real. Esto daba a las conversaciones mas la sensación de una conversación telefónica que un intercambio de texto. En los programas actuales, habitualmente, se envía cada frase de texto al terminarse de escribir. Además, en algunos, también se permite dejar mensajes aunque la otra parte no esté conectada al estilo de un contestador automático. Otra función que tienen muchos servicios es el envío de ficheros.

2 "Estado del arte" de la tecnología del cifrado de datos

El principio más importante que hay que recordar es que el hecho de que una comunicación esté cifrada, no es sinónimo de que sea segura.

El cifrado de la información consiste en disimular los datos con el fin de asegurar su confidencialidad cuando estos atraviesan canales no seguros. Durante mucho tiempo esta técnica de comunicación se ha utilizado con fines militares.

En algunas ocasiones, en épocas de censura, ha permitido a los intelectuales seguir en contacto. Pero es a raíz de la explosión de la Sociedad de la Información -a principios de la década de los noventa-cuando el cifrado realmente adquirió la relevancia que tiene hoy en nuestra vida cotidiana. Para que funcione más rápidamente, nuestro sistema necesita medios que garanticen de forma automática un cierto nivel de confianza. Esta es la razón por la cual la mayor parte de dominios de interacción de individuo con la sociedad recurre al cifrado de los intercambios. Para el pago electrónico, control de acceso, comunicaciones móviles o un número infinito de intercambios de todo tipo que utilizan redes públicas de comunicación. Hasta ahora nunca estas tecnologías han sido tan demandadas.

2.1 Introducción histórica

Aunque existen indicios circunstanciales de su uso anterior, sólo existen pruebas concretas del uso del cifrado desde hace 2400 años. Inicialmente se utilizaba para la transmisión de mensajes militares o comerciales secretos.

Resulta interesante comprobar que en 24 siglos, los principios de base del cifrado de datos no han cambiado en esencia. Así, el primer sistema documentado la "cifra de César" forma parte de la familia de los cifrados de clave privada. Pone en funcionamiento una técnica de substitución simple que consiste en desplazar cada letra tres posiciones en relación al orden del alfabeto. (la D por la A, la E por la B y así sucesivamente).

En el siglo XVI, el diplomático francés Vigenère hace una contribución decisiva: la utilización de una clave de cifrado, en el que las letras, repetidas en bucle, representan cada vez un valor de desplazamiento distinto. Así, tomando como base la cifra de César si el texto a cifrar es "CESAR" y la clave "ABC" el texto cifrado será DGVBT (esto es, C desplazado 1, E desplazado 2, etc.) Este tipo de cifrado, llamado polialfabético, tiene como ventaja que una misma letra se cifra de diferentes formas a lo largo de un mismo texto, de forma difícilmente previsible si no conocemos ni la clave ni su longitud. Además se les ha resistido a los criptonalistas durante casi tres siglos.

Otra técnica ineludible es la transposición que consiste en reordenar el conjunto del texto. Recientemente se ha descubierto que fue utilizada por primera vez 400 años A.C. en Asiria. Para cifrar un texto, se empezaba por enrollar una tira fina de papiro de forma muy apretada alrededor de un cilindro, después se escribía el texto horizontalmente sobreponiendo los círculos de papiro. Cuando la tira de papiro se desenrollaba el texto era completamente ininteligible. Era necesario utilizar un cilindro del mismo diámetro para enrollar nuevamente el papiro y el texto pudiera descifrarse.

2.2 Cifrado simétrico

El cifrado simétrico es el método criptográfico que usa una misma clave para cifrar y para descifrar mensajes. Las dos partes que se comunican han de ponerse de acuerdo de antemano sobre la clave a usar. Una vez ambas tienen acceso a esta clave, el remitente cifra un mensaje usándola, lo envía al destinatario, y éste lo descifra con la misma. Dado que toda la seguridad radica en la clave, es importante que sea muy difícil adivinar el tipo de clave. Esto quiere decir que el abanico de claves posibles, o sea, el espacio de posibilidades de claves, debe ser amplio.

Hoy por hoy, los ordenadores pueden averiguar claves con extrema rapidez, y ésta es la razón por la cual el tamaño de la clave es importante en los criptosistemas modernos.

Desde que existe el cifrado, se puede decir, que la criptografía de clave simétrica sólo ha evolucionado en los medios referidos a su implementación, hasta el punto de proporcionar a la operación un nivel de complejidad que asegura por sí misma la confidencialidad de las informaciones. En la práctica, los criptógrafos se han inclinado rápidamente por soluciones de tipo matemático recurriendo mayoritariamente a la teoría de los grandes números cuyo descifrado necesita de fuentes de cálculo que no existían en el momento que se inventó el cifrado.

Pero cada uno sabe que esta potencia de cálculo evolucione de manera exponencial, más incluso que los presupuestos destinados a su puesta en marcha. Es de esta forma que los algoritmos que se consideraban seguros no hace tanto tiempo están hoy en día a la merced de cualquier pirata informático medianamente hábil.

El principal problema con los sistemas de cifrado simétrico no está ligado a su seguridad, sino al intercambio de claves. Una vez que el remitente y el destinatario hayan intercambiado las claves pueden usarlas para comunicarse con seguridad, pero ¿qué canal de comunicación que sea seguro han usado para transmitirse la clave entre sí? Sería mucho más fácil para un atacante intentar interceptar una clave que probar las posibles combinaciones del espacio de claves. Otro problema es el número de claves que se necesitan. Si tenemos un número n de personas que necesitan comunicarse entre ellos, entonces se necesitan n(n-1)/2 claves para cada pareja de personas que tengan que comunicarse de modo privado. Esto puede funcionar con un grupo reducido de personas, pero sería imposible llevarlo a cabo con grupos más grandes.

2.2.1 Cifrado simétrico, algoritmos de flujo

Un algoritmo de cifrado de flujo es un tipo de algoritmo de cifrado simétrico en el que el texto en claro se cifra dígito a dígito y en el cual la transformación que se aplica a los sucesivos dígitos varía durante el proceso de cifrado. En la práctica los dígitos son bits o bytes.

Los algoritmos de cifrado de flujos representan una aproximación diferente al cifrado simétrico de la que dan los cifrados de bloque. Los cifrados de bloque operan en grandes bloques de números con una transformación fija, invariable. Esta distinción no queda siempre tan clara. Algunos modos de operación utilizan un bloque de cifrado de tal forma que en realidad actúan como si se tratase de un cifrado de flujo. Los cifrados de flujo se caracterizan por ejecutarse a una mayor velocidad que los cifrados de bloque y requieren un hardware menos complejo.

El cifrado de un texto plano en particular con un cifrado de bloque siempre da lugar al mismo texto cifrado siempre que se usa la misma clave. Con el cifrado de flujo la transformación de unidades de texto plano mas pequeñas variará dependiendo de dónde se encuentren durante el proceso de cifrado.

En un sistema de cifrado de flujo se produce el llamado flujo de calve (en inglés keystream), una secuencia de bits utilizada como clave. El cifrado se consigue combinando el flujo de clave con el texto en claro, generalmente aplicando la operación sobre bits XOR.

La generación del flujo de clave puede ser independiente del texto plano y del texto cifrado dando lugar a lo se que se llama cifrado de flujo síncrono, o puede depender de los datos y su cifrado, llamándose en este caso cifrado de flujo autosincronizando. La mayoría de algoritmos de cifrado de flujo se diseñan para ser autosincronizados.

2.2.1.1 Registros de desplazamiento realimentados linealmente (LFSR)

Los registros de desplazamiento realimentados linealmente, conocidos por sus siglas en inglés, LFSR son un mecanismo para generar secuencias binarias de bits. Consisten en una serie de celdas que se activan mediante un vector de inicialización, que habitualmente es la clave secreta.

Este es un registro de desplazamiento en el que el bit de entrada lo dirige una función linear del total del valor del registro de desplazamiento. El valor inicial del LFSR se denomina semilla, y ésta determina completamente la secuencia de valores. La más frecuente de las funciones lineares es el XOR de los valores de varias posiciones de los bits.

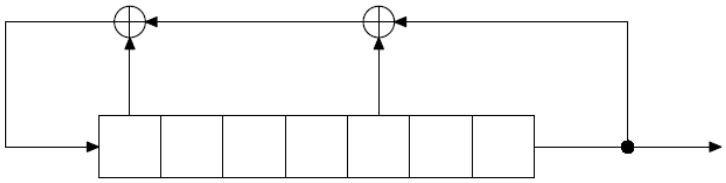

Figura 1 Esquema de un LFSR

El comportamiento del registro se regula por un contador (también llamado reloj). En cada instante el contenido de las celdas del registro se desplaza a la derecha una posición y el resultado de efectuar una operación XOR con un subconjunto de las celdas se pone en la celda de más a la izquierda. Generalmente se deriva un bit de la salida en este procedimiento de refresco.

Los LFSR son rápidos tanto en sus implementaciones en hardware como en software. Si se selecciona correctamente las patas de retroalimentación, las secuencias que se generan pueden ser tener una buena apariencia estadística. A pesar de ello, las secuencias generadas por un único LFSR no son seguras porque se conoce perfectamente su comportamiento ya que su salida es completamente lineal lo que permite un criptoanálisis relativamente sencillo. Aún así, los LFSR son las piezas con las que se construyen sistemas más seguros.

Los registros de desplazamiento en cascada son varios LFSR conectados de tal manera que el comportamiento de un LFSR en particular depende del comportamiento de un LFSR previo en la cascada. Este comportamiento dependiente se consigue utilizando un LFSR para controlar el contador (o reloj) del siguiente LFSR. Por ejemplo, un registro pude avanzar un paso si la salida del registro precedente es 1 o avanzar dos pasos en caso contrario. Existen muchas combinaciones posibles y varios parámetros que se pueden elegir para obtener buena seguridad. Buena parte de los algoritmos de cifrado de flujo se basan de combinaciones de LFSR.

2.2.1.2 Algoritmo RC4

El algoritmo RC4 es uno de los algoritmos de cifrado de flujo más utilizados. Se utiliza en protocolos tan comunes como el SSL o el cifrado WEB y WPA del las redes WiFi. Fue diseñado por Rivest para la actual RSA Security. Tiene una longitud de clave de flujo variable y sus operaciones están orientadas a byte.

El RC4 genera un flujo pseudoaleatorio de bits que para el cifrado, se combina mediante operaciones XOR con el texto plano. El descifrado se realiza de la misma manera. Para generar el flujo de clave, el algoritmo usa un estado interno secreto que consiste en una permutación de todos los posibles 256 bytes y dos punteros de 8 bits. La permutación se inicializa con una clave de longitud variables, habitualmente entre 40 y 256 bits.

Algunos análisis estiman que el periodo de repetición del algoritmo es superior a 10^{100}. A pesar de ello para los estándares actuales no se considera un algoritmo completamente seguro y no se recomienda para aplicaciones nuevas.

2.2.1.3 Algoritmo A5

Probablemente sea el algoritmo más utilizado. Millones de personas lo utilizan cada día sin saberlo cuando hablan por sus teléfonos móviles GSM.

El A5 es la familia de algoritmos de cifrado utilizado para asegurar la privacidad entre la estación base y el terminal móvil. Entre la estación base y el resto de la red telefónica, por lo general no hay mecanismos de cifrado. Existen dos versiones del algoritmo de

cifrado A5. Cuando se creó el estándar GSM, las fuerzas de seguridad y las agencias de seguridad nacionales pusieron objeciones, alegando motivos de seguridad nacional, a que el GSM dispusiera de un sistema de cifrado fuerte. Países como Francia querían un sistema de cifrado débil que fuera fácil de romper, otros países con leyes que protegen la privacidad como Alemania querían cifrado fuerte que fuera difícil de romper. La OTAN no quería que potencias extranjeras tuvieran acceso a cifrado fuerte. Al final el resultado fue que se crearon dos versiones del algoritmo: el A5/1 y el A5/2. Se supina que el A/5 era la versión completa y fue utilizada en Europa y EE. UU. El A5/2 era la de "calidad de exportación" o sea de cifrado débil.

Se trata de un ejemplo de algoritmo de cifrado de flujo basado en registros LFSR. Consiste en tres LFSRs cortos de una longitud total de 64 bits con una frecuencia de reloj irregular que se registran mutuamente de modo parada/arranque. Genera secuencias de clave de flujo muy cortas a partir de la combinación de una clave secreta de sesión de 64 bits y 22 bit de una clave conocida públicamente. En la implementación 10 de los bits de la clave se fijan a cero resultando una clave efectiva de 54 bits.

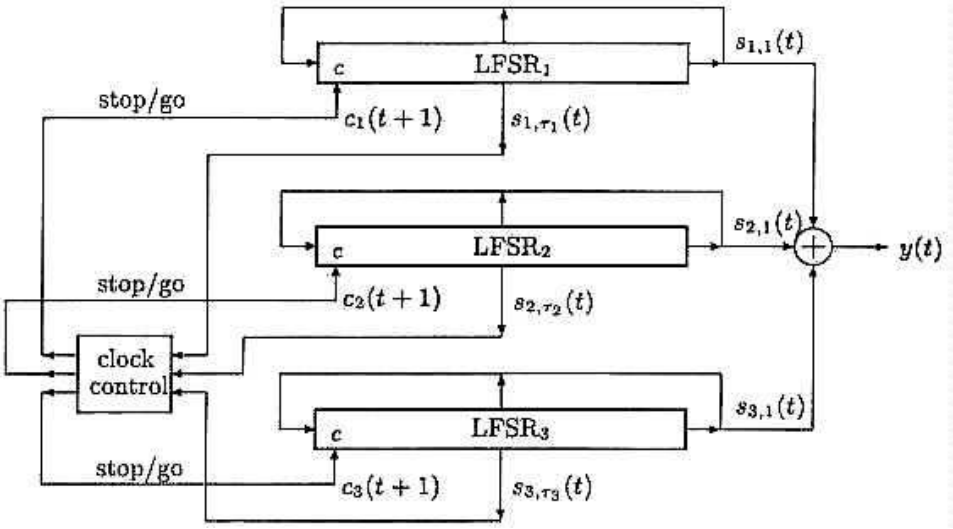

Figura 2 Esquema del algoritmo A5

Se conocen varios ataques contra este algoritmo. Algunos requieren un periodo de precomputación después del cual el algoritmo puede ser atacado en cuestión de minutos o segundos. Hasta hace poco se trataba de ataques pasivos que utilizaban texto en claro conocido pero en 2003 se descubrieron debilidades más serias que permiten ataques en los que sólo se conoce el texto cifrado.

2.2.2 Cifrado simétrico, algoritmos de bloque

Una algoritmo de cifrado simétrico de bloque es un tipo de cifrado que transforma bloques de texto plano (texto sin cifrar) de en un bloque de texto cifrado del mismo tamaño. Esta transformación se produce bajo la acción de una clave secreta que proporciona el usuario. El descifrado se realiza aplicando la transformación inversa al

bloque de texto cifrado utilizando la misma clave. Al tamaño fijo se le llama tamaño de bloque y para muchos cifrados de bloque suele ser de 64 bits. En los próximos años el tamaño de bloque se incrementará a 128 bits ya que hoy en día se considera que un tamaño de 64 bits está cercan a límite de lo que se considera seguro.

Cuando se usa un algoritmo de cifrado de bloque par cifrar un mensaje de longitud arbitraria se usan las técnicas llamadas modos de operación de cifrado de bloque. Un modo de operación, para ser útil, debe por lo menos ser tan seguro y eficiente como el algoritmo de cifrado al que se aplica.

Modo de operación ECB

ECB son las siglas en ingles de Electronic Code Book, libro de códigos electrónico. Es el modo más sencillo, cuando se aplica este modo, cada bloque del texto en claro se cifra aplicándole la transformación de modo independiente al resto de bloques. Este modo es tan seguro como el algoritmo de cifrado al que se aplica. Tiene dos desventajas:

- Bloques idénticos de texto en claro dan lugar bloques idénticos de texto cifrado por lo que no oculta los posibles patrones del texto en claro.
- Puede ser fáciliemnte atacado eliminando, añadiendo, repitiendo o intercambiando bloques del texto cifrado.

.

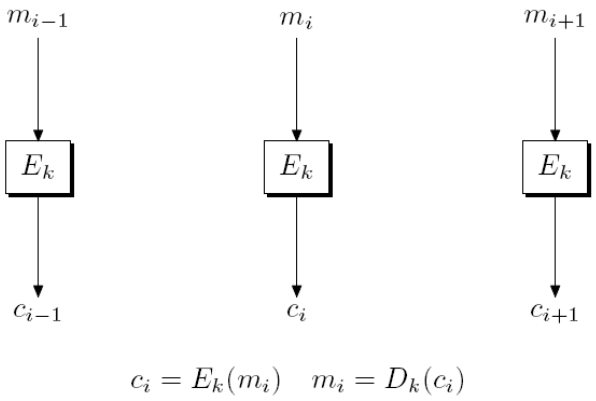

Figura 3 Modo de operación ECB

No se recomienda su uso en protocolos criptográficos salvo en algunos casos concretos, por ejemplo para cifrar datos aleatorios de tamaño pequeño como pueden ser claves de cifrado

Modo de operación CBC

En el modo CBC (del inglés cipher-block chaining), cada bloque de texto plano se le aplica una operación bit a bit XOR con el anterior bloque cifrado y después se cifra. Al primer bloque la operación XOR se aplica con un vector de inicialización, del mismo

tamaño que el bloque y que se escoge de forma aleatoria. Este vector, denotado el la figura con c_0 de inicialización sirve también para hacer cada mensaje único y se transmite en claro.

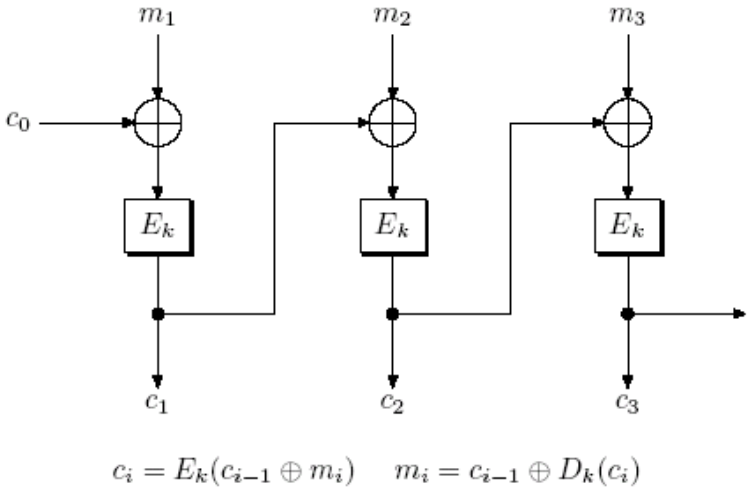

Figura 4 Modo de cifrado CBC

El modo CBC es tan seguro contra los ataques estándar como el algoritmo de cifrado al que se aplica. Además cualquier tipo de patrón repetitivo del texto en claro queda disimulado por las operaciones XOR con el bloque previo cifrado. Además el texto en plano no puede ser manipulado, excepto eliminando los bloques del principio o del fin del texto cifrado. El vector de inicialización debe ser distinto para cada mensaje y preferentemente se debe escoger de forma aleatoria.

Modo de operación CFB

En el modo CFB (Cipher Feedback Mode) el bloque de texto cifrado previo se cifra y la salida se combina con un bloque de texto plano mediante una operación XOR para producir el bloque de texto cifrado actual, tal como se muestra en la figura. El modo CFB se puede definir como si usase retroalimentación menor que la de un bloque entero de datos. También se usa un vector de inicialización para “cebar” el proceso.

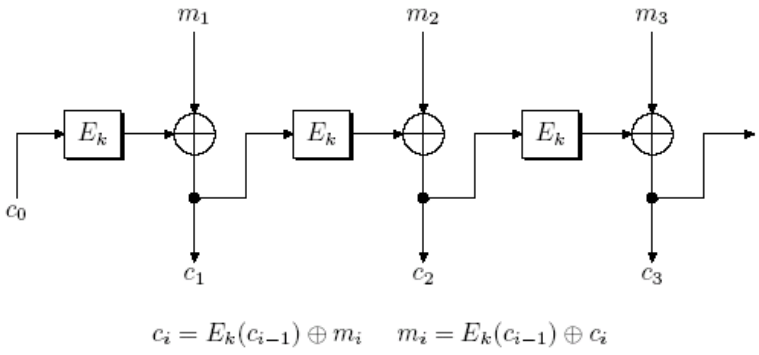

Figura 5 Modo de cifrado CFB

Modo de operación OFB

El modo OFB (Output Feedback Mode) es similar al CFB excepto en que la cantidad a la que se aplica la operación XOR con cada bloque de texto plano anterior se genera de forma independiente tanto del texto en plano como del texto cifrado. Se utiliza un vector de inicialización, denotado con s_0 en la figura, como "cebo o semilla" para un secuencia de bloques de datos, y cada bloque s_i se deriva mediante el cifrado del bloque previo s_i.

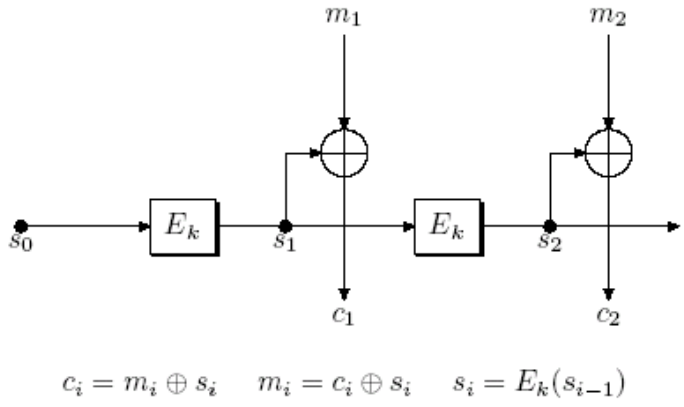

Figura 6 Modo de cifrado OFB

Dicho de otro modo, lo que se hace es cifrar sucesivamente el vector de inicialización y en cada paso se aplica la operación XOR contra un bloque de texto en plano.

2.2.2.1 Algoritmo DES

El ejemplo típico de cifrado simétrico de bloque es el algoritmo DES (Data Encryption Standard) que fue estandarizado en 1976 por el Federal Information Processing Standard (FIPS) de los EE. UU. Fue el primero de los grandes algoritmos del cifrado con clave secreta (o clave privada). Aún sirve de base a un determinado número de

sistemas actuales tales como el SSL que protege las compras electrónicas que se pagan con tarjeta de crédito.

DES es un algoritmo simétrico, es decir, que utiliza las mismas claves para cifrar que para descifrar.

El algoritmo fue controvertido desde un principio, con elementos de su diseño secretos y una clave de longitud relativamente corta. Durante su desarrollo por IBM en los años 70 del siglo pasado, la Agencia de Seguridad Nacional[2] de los EE. UU. (NSA en sus siglas en inglés) recomendó cambios en el algoritmo, entre otros, reducir el tamaño de la clave de 64 a 56 bits, se sospecha que porque la NSA estaba segura que ellos tendrían las suficiente potencia de cálculo para realizar un ataque de fuerza bruta contra el algoritmo varios años antes que el resto del mundo. También se sospecha que un componente crítico de este algoritmo, las llamadas S-boxes, fue alterado para insertar una "puerta trasera"

Debido a su publicación como estándar fue escrutado intensamente por el mundo académico, motivado por el interés en los modernos diseños de algoritmos de cifrado de bloque y su criptoanálisis.

Descripción del algoritmo

DES es el típico algoritmo de cifrado de bloque, un algoritmo que toma una cadena de bits del texto en claro de tamaño fijo y lo transforma a través de una serie de complicadas operaciones en otra cadena de bits de la misma longitud que el texto original. En el caso del DES el tamaño de bloque es de 64 bits. Este algoritmo también utiliza la clave para determinar la transformación, por lo que el descifrado sólo puede llevarse a cabo por quien conoce la clave utilizada para cifrar. La clave consiste en 64 bits pero el algoritmo usa 56 de estos. La clave utiliza ocho bits para comprobar la paridad y por lo tanto son descartados por el algoritmo, así que la longitud efectiva de este algoritmo es de 56 bits.

Como otros algoritmos de cifrado de bloque el DES debe usarse en modo de operación (CBC, CFB, OFB) si se aplica a mensajes más largos de 64 bits.

Estructura básica

La estructura básica del algoritmo se muestra en la figura 1. Hay 16 pasos idénticos de procesado denominados rondas. También hay una permutación inicial y una final denominadas IP y FP las cuales son una inversa de la otra (IP deshace lo que hace FP y viceversa). Estas permutaciones prácticamente no tienen sentido criptográfico pero aparentemente se incluyeron para facilitar la carga de bloques parra el hardware de mediados de los años 70 del siglo pasado. Antes de las rondas principales, cada bloque

[2] NSA Agencia Nacional de Seguridad de los EE. UU. Es la agencia de espionaje del gobierno de los EE. UU. encargada de la vigilancia, recolección y análisis de las comunicaciones del resto del mundo y de asegurar el secreto de las comunicaciones del gobierno de los EE.UU. frente a agencias similares de potencias extranjeras. Entre otras es responsable de la red ECHLEON dedicada a capturar emisiones de radio, satélite, comunicaciones telefónicas, faxes, correos electrónicos, etc. Se estima que la red ECHEON captura y analiza cerca de 3 millardos de comunicaciones al día.

se divide en dos subloques de 32 bits y se procesan alternativamente. Este entrecruzamiento se denomina el sistema de Feistel

La estructura de Feistel asegura que el descifrado y el cifrado son procesos muy similares, la única diferencia es que las subclaves se aplican en el orden inverso cuando se descifra. El resto del algoritmo es idéntico. Esto simplifica mucho la implementación, especialmente en hardware, ya que no se necesitan algoritmos distintos para el cifrado y descifrado.

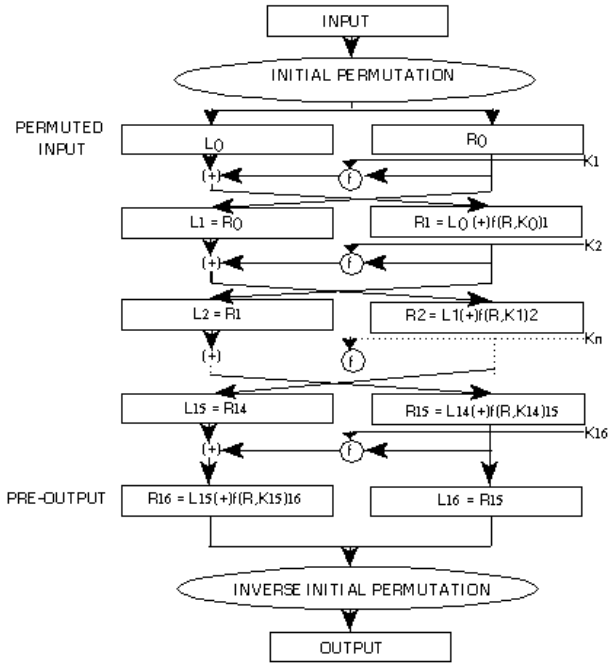

Figura 7 Esquema del algoritmo DES

El símbolo (+) representa la operación o exclusivo (XOR). La función f revuelve la mitad de un bloque junto con parte de la clave. La salida de la función f se combina con la otra mitad del bloque, y las mitades se intercambian antes de la siguiente ronda.

La función f o función de Feistel

La función Feistel (función f), tal como se explica en la figura 2, opera medio bloque (32 bits) cada vez y consiste en cuatro pasos:

1. Expansión. El medio bloque de 32 bits se expande a 48 usando la expansión-permutación denotada por E en la figura.
2. Mezclado de claves. El resultado se combina con una subclave utilizando la operación XOR.
3. Sustitución. Después del mezclado de la subclave, el bloque se divide en ocho partes de 6 bits cada una antes de ser procesado en las s-boxes, o cajas de

sustitución. Cada una de las ocho S-boxes reemplaza los seis bits de entrada con cuatro bits de salida de acuerdo con una transformación no lineal, proveída en forma de tabla de búsqueda. Las S-boxes proporcionan el núcleo de la seguridad del DES, sin ellas, el algoritmo de cifrado sería lineal y rompible de forma trivial.

4. Permutación. Finalmente, las 32 salidas de las S-boxes se reordenan de acuerdo con una permutación fijada, la denominada P-box.

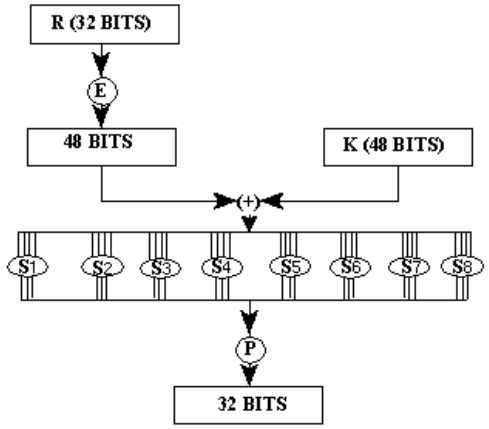

Figura 8 Esquema de la función f o función de Feistel

La alteración por sustitución de las S-boxes, y permutación de bits de la P-box y la E-expansion proporciona los llamados "confusión y difusión" respectivamente, unos conceptos identificados por Claude Shannon en la década de los 40 del siglo anterior condición necesaria para la seguridad de un algoritmo de cifrado.

Obtención de las subclaves

La figura 3 ilustra el proceso de obtención de las subclaves para el cifrado. Inicialmente se seleccionan 56 bits de los 64 iniciales mediante la permutación PC-1 (Permuted choice 1) los restantes ocho bits se descartan o se usan como bits de paridad. Los 56 bits se dividen en dos mitades de 28 bits, cada mitad se trata por separado. En sucesivas rondas, los bits de las dos mitades se desplazan a la derecha por uno o dos bits (especificados para cada ronda) y posteriormente se obtiene una subclave de 48 mediante la permutación PC-2 (denominada en inglés Permuted Choice 2), 24 bits de la mitad derecha y 24 de la mitad izquierda. Los desplazamientos de bits suponen que un conjunto distinto de bits se utiliza para cada subclave, cada bit se usa en aproximadamente 14 de las 16 subclaves.

La obtención de subclaves para el descifrado es similar, se deben generar las claves en orden inverso, por lo tanto los desplazamientos de bits son a la derecha en vez de a la izquierda.

Seguridad del algoritmo

El problema es que las claves que utiliza el algoritmo se han quedado demasiado cortas, los ordenadores actuales tienen suficiente potencia de cálculo para en un tiempo

razonable, mediante un ataque del tipo de fuerza bruta romper la seguridad de este algoritmo.

Actualmente el algoritmo DES no se considera seguro.

En el mundo académico se ha hecho varias propuestas para construir una máquina capaza de romper el DES. En 1977 Diffie y Hellman propusieron una máquina de un costo estimado de 20 millones de dólares capaz de romper el DES en menos de un día. En 1993, Wiener propuso una maquina buscadora de claves con un coste aproximado de 1 millón de dólares que sería capaza de romper el algoritmo en unas 7 horas. Pero en 1998 la Electronic Frontier Foundation[3], defensora de los ciber-derechos, demostró con hechos, construyendo un ordenador de un coste aproximado de 250.000 dólares que fue capaz de romper el DES mediante un ataque de fuerza bruta en apenas dos días.

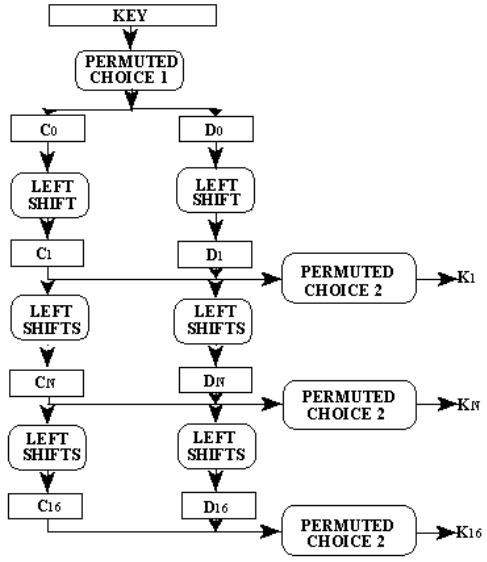

Figura 9 Esquema del procedimiento de obtención de subclaves

Por otro lado se han propuesto ataques teóricos contra este algoritmo aunque en la práctica no se puedan llevar a cabo:

- Criptoanálisis diferencial. Descubierto en los últimos años de la década de los 80, por Eli Biham y Adi Shamir, aunque se cree que era conocido por IBM y la NSA pero se mantuvo secreto. Para romper el DES sería necesarios 2^{47} textos en claro conocidos. DES se diseño para ser resistente al criptoanálisis diferencial.

[3] La "Electronic Frontier Foundation", conocida por sus siglas EFF es una orgranización sin ánimo de lucro establecida en los EE.UU. dedicada defender el derecho a la libertad de expresión, tal como protege la Primera Enmienda de la constitución de los EE. UU en el contexto de la era digital. Si principal objetivo es educar a la prensa, los políticos y al público en general sobre el debate de los derechos civiles relacionados con la tecnología, y para defender estos derechos.

Con formato: Español (España - alfab. internacional)

- Criptoanálisis lineal. Descubierto por Mitsuru Matsui, necesitaría 2^{43} textos en claro conocidos para llevarlo a cabo. El método fue implementado en 1994 y fue el primer criptoanálisis experimental sobre el DES conocido. No hay evidencias que el DES fuese proyectado para ser resistente a este tipo de ataque.
- Ataque de Davies mejorado. Mientras el criptoanálisis diferencial y lineal son técnicas generales y pueden aplicarse a un gran número de patrones, este ataque es una técnica especializada para el DES. En su forma más potente requiere 250 textos en claro conocidos.

2.2.2.2 Algoritmo 3DES

Identificada rápidamente, esta debilidad generó la necesidad de disponer de un sistema más sólido, de ahí la creación del 3DES (triple DES): Como su nombre indica, consiste en aplicar el algoritmo DES tres veces seguidas, generalmente en una secuencia del tipo cifrado/descifrado/cifrado movilizando tres claves diferentes sin ninguna relación entre ellas. Eficaz en términos de seguridad, sin embargo este procedimiento es demasiado lento para una implementación de manera más generalizada. Esta constatación llevó al NIST americano (National Institute of Standards and Technology) a concebir AES

2.2.2.3 Algoritmo AES

El algoritmo AES (Advanced Encryption Standard), cuya principal característica es la de utilizar claves más largas en bloques de datos igualmente más largos. Concretamente se trata en este caso de claves de 128, 192 y 256 bits que ya no se aplican a los caracteres sino a series de bits de tamaños predefinidos. Según los expertos, una clave de 128 bits aplicada de esta forma sería suficiente para cubrir las necesidades de la mayoría de los usuarios actuales y potenciales de aquí hasta el 2015-2020.

El algoritmo AES, también conocido como Rijndael, es un algoritmo de cifrado de bloque que ha sido adoptado como estándar por el gobierno de los EE. UU. Se espera que sea usado ampliamente y analizado de forma extensa como fue el caso de su predecesor el Data Encryption Standard (DES). Es de suponer que en el futuro sea el estándar de facto para las aplicaciones que requieran cifrado simétrico.

Origen

El algoritmo AES se adoptó por el Instituto Nacional de Estándares (NIST) de los EE. UU. como el US FIPS PUB 197 en noviembre de 2001 después de un proceso de estandarización que duró 5 años.

El algoritmo fue desarrollado por dos criptógrafos belgas, Joan Daemen y Vincent Rijmen y se envió al proceso de selección AES bajo el nombre “Rijndael”, un acrónimo formado por la contracción de los nombre de los inventores.

Al contrario que su predecesor DES, Rijndael está formado por una red de permutaciones-sustituciones, no por una red Feistel como el DES. El cifrado AES es rápido tanto en implementaciones por hardware como por software, es relativamente fácil de implementar y requiere poca memoria. Como nuevo estándar de cifrado actualmente se está desplegando a gran escala.

Descripción del algoritmo

Estrictamente, AES no es exactamente el algoritmo Rijndael aunque en la práctica se habla de los dos de forma intercambiable. Rijndael soporta un rango mayor de tamaños de clave y de bloque; AES tiene fijado el tamaño de bloque a 128 bytes y el tamaño de la clave a 128, 192 o 256 bits, mientras que el Rijndael se puede especificar con un tamaño de bloque de cualquier múltiplo de 32 bits con un mínimo de 128 bits y un máximo de 256.

La clave se expande usando el preprocesado de clave de Rijndael.

La mayoría de los cálculos del AES se realizan dentro de un grupo finito especial. El algoritmo AES opera en matrices de 4 por 4 bytes llamadas *estado*. Para el cifrado, cada ronda de la aplicación del algoritmo consiste en cuatro pasos:

1. **SubBytes**. En este paso se realiza una sustitución no lineal donde cada byte es reemplazado con otro de acuerdo a una tabla.
2. **ShiftRows**. En este paso se realiza un transposición donde cada fila del *estado* es rotado de manera cíclica un número determinado de veces.
3. **MixColumns**. Operación de mezclado que opera en las columnas del *estado*, combinando los cuatro bytes en cada columna usando una transformación lineal.

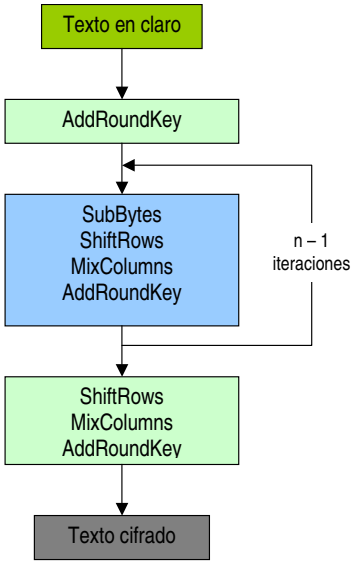

Figura 10 Esquema del algoritmo AES

4. **AddRoundKey**. Cada byte del *estado* se combina con la clave de la ronda; cada clave de ronda se deriva de la clave de cifrado usando un preprocesado de clave.

La ronda final omite el paso MixColumns.

Optimización del cifrado

En sistemas de 32 bits o de mayor tamaño de palabra, es posible acelerar la ejecución de este algoritmo mediante la conversión de las transformaciones SubBytes, ShiftRows y MixColumn en tablas. Se tienen cuatro tablas de 256 entradas de 32 bits que utilizan un total de 4 kilobytes (4096 bytes) de memoria, un Kb cada tabla. De esta manera, una ronda del algoritmo consiste en 16 búsquedas en una tabla seguida de 16 operaciones XOR de 32 bits en el paso AddRoundKey. Si el tamaño de 4 kilobytes de la tabla es demasiado grande para una plataforma determinada, la operación de búsqueda en la tabla se pude realizar mediante una sola tabla de 256 entradas de 32 bits mediante el uso de rotores circulares.

Seguridad

A día de hoy no se conoce ningún ataque eficaz contra el AES. Los únicos ataques conocidos son del tipo de ataques "por canal auxiliar", es decir ataques basados de la implementación física del criptosistema y no de la debilidad teórica del algoritmo, por ejemplo información de tiempo, de consumo de potencia del sistema, emanaciones electromagnéticas que pueden dar información que puede ser explotada para romper el sistema de cifrado. La NSA (Agencia de Seguridad Nacional de los EE. UU.) revisó todos los candidatos finalistas al AES, incluido el Rijndael y declaró que todos eran lo suficientemente seguros para su empleo en la protección de información no clasificada del gobierno de los EE. UU. En junio de 2003 el gobierno de los EE. UU. anunció que el AES también podía ser usado para información clasificada. Este hecho marca la primera vez que el público en general tiene acceso a un cifrador aprobado por la NSA para información secreta. Es interesante notar que muchos productos públicos usan llaves de 128 bits por defecto; es posible que la NSA sospeche de una debilidad fundamental en llaves de este tamaño, o simplemente prefieren tener un margen de seguridad para documentos confidenciales, (que pueden requerir una seguridad que dure décadas)

El método más común de ataque hacia un cifrador por bloques consiste en intentar varios ataques sobre versiones del cifrador con un número menor de rondas. El AES tiene 10 rondas para llaves de 128 bits, 12 rondas para llaves de 192 bits, y 14 rondas para llaves de 256 bits. Hasta 2005, los mejores ataques conocidos son sobre versiones reducidas a 7 rondas para llaves de 128 bits, 8 rondas para llaves de 192 bits, y 9 rondas para llaves de 256 bits.

Algunos criptógrafos muestran preocupación sobre la seguridad del AES. Opinan que el margen entre el número de rondas especificado en el cifrador y los mejores ataques conocidos es muy pequeño. El riesgo es que se puede encontrar alguna manera de mejorar los ataques y, de ser así, se podría romper la cifra. En el contexto criptográfico se considera "roto" un algoritmo si existe algún ataque más rápido que una búsqueda exhaustiva - ataque por fuerza bruta. De modo que un ataque contra el AES de clave de 128 bits que requiera "sólo" 2^{120} operaciones sería considerado como un ataque que rompe el AES aún tomando en cuenta que por ahora sería un ataque irrealizable. En la práctica, un ataque tan "bueno" como este es irrelevante. De momento estas debilidades se pueden ignorar. El mayor ataque de fuerza bruta conocido públicamente ha sido contra el algoritmo RC5 de clave 64 bits y fue llevado a cabo por **distributed.net**

Otra preocupación es la estructura matemática del AES. Al contrario que la mayoría de otros algoritmos de bloque, AES tiene una descripción matemática muy clara. Esto todavía no ha conducido a ningún ataque pero algunos investigadores están preocupados de que futuros ataques puedan explotar esta estructura.

En el 2002, Nicolas Courtois y Josef Pieprzyk publicaron un ataque teórico, denominado "ataque XSL", mostrando una potencial debilidad en el algoritmo AES.

Varios expertos en criptografía han encontrado problemas en los principios matemáticos en que se basa el algoritmo y han propuesto ataques que sugieren que los autores cometieron errores en sus estimaciones. Esta línea de ataque contra el AES de momento es una pregunta abierta. Por el momento, Para el momento, el ataque de XSL contra

AES parece más especulativo que real y es improbable que nadie sea capaz de llevarlo a la práctica.

En Abril del 2005, D. J. Bernstein anunció un ataque denominado “cache timing attack” utilizado para romper un servidor que usaba el cifrado AES proporcionado por OpenSSL. El servidor estaba diseñado para dar la mayor cantidad de información posible sobre el tiempo y el ataque requirió 200 millones mensajes escogidos de texto en claro. Dicen que este ataque no es posible en implementaciones reales.

En octubre de 2005 Adi Shamir y dos investigadores más presentaron un artículo que demostraba varios ataques más de “cache timing” contra el AES. Uno de estos ataques era capaz de obtener la clave AES entera tras sólo 800 escrituras, en 65 milisegundos. Todos estos ataques sólo son posibles si el atacantes es capaz de ejecutar programas en el mismo sistema en que se ejecuta el cifrado AES.

2.2.2.4 Otros algoritmos de cifrado simétrico

Existen gran variedad de algoritmos de clave secreta se han ido proponiendo a la comunidad. Entre estos otros algoritmos cito dos de los más conocidos, **Blowfish** e **Idea**. Una de las características principales de Blowfish (concebido por el experto en criptografía, Bruce Scheiner) es que acepta las claves de diferentes tamaños. La longitud de las claves puede llegar hasta los 448 bits. En Idea (International Data Encryption Algorithm), la clave se limita a 128 bits genera claves intermedias a lo largo de ocho etapas de modificación sucesivas y opera en bloques de 64 bits.

En todos los casos, la longitud de la clave juega un papel decisivo en la solidez del algoritmo. Pero he de añadir que su método de aplicación es también muy importante. Hace falta que su aplicación no sea lineal, que varíe en función del número de elementos más elevado posible, en relación con el contenido del mensaje para ser resistente a los ataques. Un poco más adelante explicaremos unas normas básicas para poner apunto una clave eficaz. Así, se establece que la extracción de una clave Idea mediante cálculo puro serían necesarios 30.000 millones de años. Como para esperarse….

2.3 Funciones de resumen HASH y códigos de autentificación de mensajes MAC

Funciones de resumen

Una función de resumen (conocidas por su nombre ingles **hash functions**) es una transformación H que toma la entrada **m** de tamaño variable y retorna una cadena de tamaño fijo al que se llama valor resumen h de m (h = H(m)). Las funciones de resumen con esta propiedad tienen diversas aplicaciones en el mundo de la informática pero cuando se quieren usar en criptografía se suelen elegir para que cumplan ciertas propiedades adicionales:

- La entrada debe poder ser de cualquier tamaño.
- La salida debe ser de longitud fija.
- Dado cualquier x, H(x) debe ser relativamente fácil de calcular.

- La función resumen debe ser unidireccional. Se dice que una función es unidireccional si es difícil de invertir, dado un resumen h, sea computacionalmente impracticable obtener una entrada x tal que h = H(x).
- La función resumen debe estar libre de colisiones. Es decir que sea altamente improbable que, al aplicar la función de resumen a dos mensajes distintos se obtenga el mismo resultado.

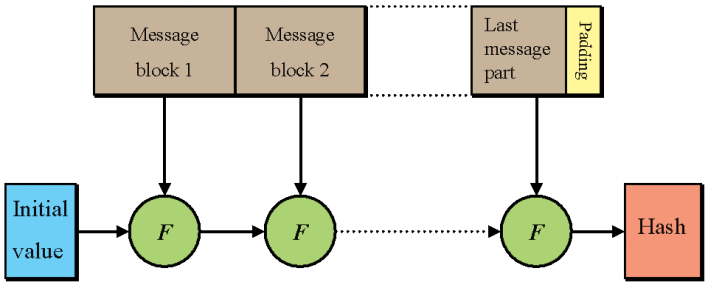

Figura 11 Estructura de un algoritmo de resumen.

Las funciones de resumen criptográficas, que cumplen estos requisitos, se utilizan como primitiva en varias aplicaciones tales como la autentificación, la comprobación de la integridad de mensajes y en la firma digital. Las funciones de resumen criptográficas también se conocen como huellas digitales electrónicas[4].

El principal uso de las funciones de resumen criptográficas se de en la comprobación de la integridad de mensajes y en firmas digitales. Como la funciones de resumen son generalmente más rápidas que el cifrado o el firmado digital es habitual calcular la firma digital o la comprobación de integridad de un documento aplicando el proceso criptográfico, no al documento en sí sino al resumen del mismo que es mucho menor comparado con el tamaño del documento. Además se puede publicar un resumen criptográfico de un documento revelar el documento en sí. Por ejemplo para un servicio de marcado de tiempo de fecha digital[5] donde se puede obtener un documento con una marca de tiempo emitida por una tercera persona (o entidad) sin necesidad de reveler a esta tercera persona el contenido que se quiere *timbrar*.

Códigos de autentificación de mensajes

Un código de autentificación de mensaje (conocido por sus siglas en inglés MAC) es un pequeño trozo de información usado para autentificar un mensaje. Un algoritmo MAC acepta como entrada un mensaje de longitud arbitraria y una clave secreta y retorna el código de autentificación de mensaje (MAC). El MAC sirve para proteger tanto la integridad del mensaje como para garantizar su autenticidad al permitir su verificación

[4] En la bibliografía en lengua inglesa se llama “digital fingerprint” lo he traducido por “huella digital electrónica” ya que el término “huela digital digital” suena muy forzado.

[5] Nota de traducción. No he encontrado como traducir el termino inglés *timestamp*. En castellano podría ser marca tiempo, timbrado de fecha, etc.

por el poseedor de la clave secreta y detectar cualquier cambio en el contenido del mensaje.

Existen cuatro tipos de códigos de autentificación de mensaje:

1. Incondicionalmente seguros, basados en el cifrado de un solo uso (one-time pad). El mensaje de autentifica a sí mismo ya que nadie más tiene acceso la clave de cifrado. También se puede conseguir un MAC incondicionalmente seguro si se utiliza sólo una vez una clave secreta.
2. Códigos de autentificación de mensajes basados en funciones de resumen criptográficas, también conocidos como HMAC. Utilizan una clave o claves en conjunción con una función de resumen para producir un *checksum* que se añade al mensaje. Ejemplos de este tipo son el HMAC-MD5 ó el HMAC-SHA-1
3. Códigos de autentificación de mensajes basados en algoritmos de cifrado de flujo. Se han propuesto algoritmos que, basándose los cifrados de flujo probablemente seguros, cifran el mensaje original mediante dos LFSR obteniendo el estado final de los dos LFSR como el código de autentificación del mensaje.
4. Códigos de autentificación de mensajes derivados de en algoritmos de cifrado de bloque. El ejemplo más habitual es cifrar el mensaje con un algoritmo de cifrado de bloque en modo CBC obtener el código MAC como el último bloque del texto cifrado. Ejemplos de este tipo son el DES-CBC MAC ó el AES-XCBC-MAC.

2.4 Cifrado asimétrico. Algoritmos de clave pública y clave privada

En lo que se refiere a los sistemas de clave compartida, todo es más simple. El sistema RSA, de los laboratorios del mismo nombre, domina la familia de los algoritmos denominados asimétricos (el método no es el mismo para cifrar que para descifrar). Un sistema derivado de la teoría de la complejidad (definida a finales de la década de los setenta). Según esta teoría, se puede generar dos claves sólidas, una pública y otra privada, de un solo problema cuya resolución exige un esfuerzo de varios millones de años. Con la clave pública, cada uno puede cifrar datos destinados al poseedor de la clave privada. A la inversa, es necesario poseer la clave privada para descifrar las informaciones cifradas con la clave pública correspondiente. De aquí derivan la mayoría de las implementaciones de tipo PKI (Public Key Infrastructure) tales como la PGP. Implementaciones cada vez más frecuentes, desde que en el año 2000 expiró la patente que protegía la RSA.

Las claves utilizadas en los sistemas de clave pública son generalmente más largas que las de los sistemas de clave secreta. Esto se debe al hecho de que un criptoanalista que quisiera descubrirlas, dispone de un mayor número de elementos e informaciones. El problema no radica tanto en adivinar estas claves privadas sino en el hecho de deducir una clave privada a partir de su correspondiente clave pública, que como su nombre indica, todo el mundo conoce. El principio de esta derivación es bastante sencillo. En realidad se trata de la descomposición en factores primos de un número entero de gran magnitud. Cuando este número corresponde a un valor de 256 bits, no importa que ordenador puede realizar esta factorización con éxito. La operación a 512 bits requiere

un ordenador especializada durante varias semanas. Hoy en día el descifrado se considera imposible en un límite de tiempo razonable a partir de 1024 bits.

Según RSA Security la sociedad creada por los inventores de RSA, las claves asimétricas de 1024 bits (de una magnitud superior a 10^{300}) tienen un grado de resistencia equivalente a las claves simétricas de 80 bits. De ahí que se recomiende no utilizarlas más allá del 2010. Por su parte, las claves asimétricas de 2048 bits, equivalen a las claves simétricas de 112 bits por lo que la preconización de utilización se estima alrededor del año 2030. Como anécdota, decir que se establece que la descomposición factorial de una clave de 2048 bits necesita alrededor de $4 \cdot 10^{14}$ mips/año (1 mip = 1 millón de instrucciones por segundo). Dicho de otra forma: para conseguirlo un ordenador con una potencia de 1 mips debería calcular durante 10^{14} años, es decir 100 billones de años.

2.4.1 Protocolo de intercambio de claves Diffie-Hellman

Diffie-Hellman fue el primer algoritmo de clave pública que se inventó, allá por el 1976. Su seguridad proviene de a dificultad de calcular logaritmos discretos en un grupo multiplicativo finito, frente a la dificultad de calcular exponenciación en ese mismo tipo de grupo. Diffie-Hellman puede utilizarse para el intercambio de claves. Dos usuarios A y B pueden utilizar este algoritmo para generar una clave secreta -pero no puede utilizarse para cifrar y descifrar mensajes.

Si dos usuarios, A y B, quieren establecer un canal seguro de comunicación el procedimiento es el siguiente:

1. Los dos usuarios, A y B, eligen un grupo multiplicativo finito G de orden n y un generador α perteneciente a G, que no tiene porque ser secreto.
2. A genera un número aleatorio **a**, y calcula α^a dentro del grupo multiplicativo G y transmite el resultado a B.
3. B genera un numero aleatorio **b**, calcula α^b y transmite el resultado a A.
4. A recibe α^b y calcula $(\alpha^b)^a$ dentro de G.
5. B recibe α^a y calcula $(\alpha^a)^b$ dentro de G.

Al final de este procedimiento A y B han convenido un elemento α^{ab} del grupo G que es su clave secreta común entre ambos, pero desconocida para el resto de usuarios que ignoran las claves privadas **a** y **b**

El algoritmo de intercambio de claves Diffie-Hellman estuvo patentado, al menos en EE. UU. y en Canadá hasta que la patente expiró en 1997.

2.4.2 Sistema de clave pública RSA

El algoritmo RSA recibe su nombre de las iniciales de ron Rivest, Adi Shamir y Len Adleman quienes lo inventaron en 1977. En realidad la técnica básica la descubrió Clifford Cocks del GCHQ[6] Británico pero fue secreta hasta 1997.

Este algoritmo sirve tanto para el cifrado de clave pública como para producir firmas digitales.

Cifrado con RSA

Descripción del algoritmo

1. Se generan dos números primos grandes, **p** y **q** de aproximadamente el mismo tamaño tales que su producto **n** = **p**·**q** tiene el tamaño en bits requerido, por ejemplo 1024 bits.
2. Se calcula **n** = **p**·**q** y **φ** = (**p**-1)(**q**-1)
3. Se escoge un entero e tal que 1 < **e** < **φ** tal que el máximo común divisor de **e** y **φ** sea 1
4. Se calcula el exponente secreto **d**, 1 < **d** < **φ** tal que **ed** = 1 mod **φ**
5. La clave pública está formada por el par (**n**, **e**) y la clave privada es el par (**n**, **d**). Los valores **p**, **q** y **φ** también se mantienen secretos.

El cifrado se produce de la siguiente manera:

El emisor **A** hace lo siguiente:

1. Obtiene la clave pública de **B** (**n**, **e**)
2. Representa el texto en claro del mensaje como un entero positivo
3. Calcula el texto cifrado **c** como $\mathbf{c} = \mathbf{m}^{\mathbf{e}} \bmod \mathbf{n}$
4. Envía el texto cifrado **c** a **B**

El receptor B, para descifrar el mensaje hace lo siguiente:

1. Utiliza su clave privada, (**n**, **d**) para calcular $m = \mathbf{c}^{\mathbf{d}} \bmod \mathbf{n}$.
2. Extrae el texto plano como el entero positivo m.

Actualmente es muy difícil obtener la clave privada a partir de la clave pública (**n**, **e**). Si fuera posible factorizar **n** para obtener **p** y **q** entonces se podría obtener la clave privada **d**. Por lo tanto la seguridad del sistema RSA se basa en la suposición de que factorizar números grandes es extremadamente costoso en términos de potencia de cálculo. Si se descubriese un método eficiente para factorizar números primos grandes significaría romper el RSA.

[6] GCHQ, siglas de Government Communications Headquarter, agencia británica de inteligencia similar a la NSA norteamericana, dedicada al espionaje electrónico.

Firma digital con RSA

Supongamos que A quiere enviar a B un mensaje de tal manera que B pueda verificar que el emisor auténtico del mensaje es A, esto es A quiere enviar un mensaje firmado a B. La manera de actuar sería la siguiente:

1. A crea un mensaje **m**.
2. A calcula la firma digital **s** mediante la exponenciación: $\mathbf{s} = \mathbf{m}^{\mathbf{d}} \bmod \mathbf{n}$ donde **d** y **n** son la clave privada de A.
3. A envía a B **m** y **s**.

Para verificar la firma, B exponencia y comprueba que el mensaje m coincide con el obtenido de la exponenciación: $\mathbf{m} = \mathbf{s}^{\mathbf{e}} \bmod \mathbf{n}$, donde **e** y **n** son la clave pública de A.

Por lo tanto cifrado y autentificación tienen lugar sin que se intercambien claves privadas. Cualquiera puede enviar un mensaje cifrado o verificar un mensaje firmado, pero sólo quien posee la clave privada correcta puede descifrar o firmar un mensaje.

2.4.3 Algoritmo de clave pública ElGamal

El algoritmo ElGamal es un algoritmo de clave asimétrica basado en intercambio de clave Diffie-Hellman. Fue descrito por el criptógrafo egipcio Taher Elgamal en 1984. Está basado en la exponenciación discreta sobre un grupo multiplicativo. Su seguridad recae en la dificultad de cierto problema relacionado con el cálculo de logaritmos discretos.

Descripción del algoritmo

El sistema ElGamal consiste en tres componentes: el generador de claves, el algoritmo de cifrado y el algoritmo de descifrado. El funcionamiento es el siguiente:

Generación de la clave pública:

1. A genera un número primo aleatorio grande **p** y un generador $\boldsymbol{\alpha}$ del grupo multiplicativo Z_p^* de los enteros módulo **p**.
2. Se escoge un entero aleatorio **a** tal que $1 \leq \mathbf{a} \leq p - 2$, y se calcula $\boldsymbol{\alpha}^{\mathbf{a}} \bmod \mathbf{p}$.
3. La clave pública de A esta formada por $(\mathbf{p}, \boldsymbol{\alpha}, \boldsymbol{\alpha}^{\mathbf{a}})$; la clave privada de A es **a.**

Cifrado. Una vez generada la clave pública de A, si B quiere enviar el mensaje **m** a A, B debe proceder de la siguiente manera:

1. B obtiene la clave pública de A $(\mathbf{p}, \boldsymbol{\alpha}, \boldsymbol{\alpha}^{\mathbf{a}})$.
2. B representa el mensaje **m** como un entero dentro del rango {1, 2, ..., **p** - 1}.
3. Elige un entero aleatorio **k** tal que $1 \leq \mathbf{k} \leq \mathbf{p} - 2$.
4. Calcula $\boldsymbol{\gamma} = \boldsymbol{\alpha}^{\mathbf{k}} \bmod \mathbf{p}$ y $\boldsymbol{\delta} = m = (\boldsymbol{\alpha}^{\mathbf{a}})^{\mathbf{k}} \bmod \mathbf{p}$.
5. B envía del mensaje cifrado $c = (\gamma, \delta)$ a A.

Descifrado. Para recuperar el texto en claro A debe hacer lo siguiente:

1. Usando su clave privada **a** calcula $\gamma^{\mathbf{p}-1-\mathbf{a}} \bmod \mathbf{p}$. (Observación: $\gamma^{\mathbf{p}-1-\mathbf{a}} = \gamma^{-\mathbf{a}}$

2. $= \boldsymbol{\alpha}^{-\mathbf{ak}}$).
3. Recupera el mensaje en claro m calculando m = $(\gamma^{-\mathbf{a}})\cdot\boldsymbol{\delta}$ mod **p**.

A y B pueden elegir utilizar el mismo primo p y generador α, en este caso, p y α no tienen porque ser publicados como parte de la clave pública. En este caso la clave pública resulta de menor tamaño. Otra ventaja adicional de tener fijados los parámetros **p** y **α** es que la exponenciación puede precalcularse, reduciéndose el tiempo para generar un mensaje cifrado.

ElGamal es un ejemplo de cifrado asimétrico semánticamente seguro bajo supuestos razonables. Es probabilístico, en el sentido que un único texto en claro cuando se cifra puede dar lugar a varios posibles textos cifrados, con la consecuencia de que en general ElGamal produce una expansión 2:1 del tamaño del texto en claro al tamaño del texto cifrado.

La seguridad de ElGamal se basa, en parte, en la dificultad de resolver logaritmos discretos dentro de un grupo multiplicativo. Específicamente si el problema del logaritmo discreto pudiese resolverse de forma eficiente, entonces se rompería el algoritmo ElGamal. A pesar de ello, la seguridad de ElGamal también descansa en el suposición denominada Diffie-Hellman Decisional (sus siglas en inglés DDH). Esta suposición es mas fuerte que la suposición del logaritmo discreto, pero todavía se cree que puede ser cierta para varias clases de grupos.

2.4.4 Sistemas de cifrado de curva elíptica

Los sistemas de cifrado mediante curva elíptica fueron propuestos a mediados de los años 80 del siglo pasado por Victor Miller y Neal Koblitz. Son sistemas análogos a los sistemas de clave pública en los que la aritmética modular se reemplaza por operaciones definidas sobre curvas elípticas. Los sistemas de cifrado de cura elíptica que han aparecido en la literatura pueden clasificarse en dos categorías de acuerdo con si son análogos al sistema RSA o a los basados en logaritmos discretos.

Como en todos los sistemas de clave pública, la seguridad del los sistemas de curva elíptica descansa en problemas matemáticos difíciles de resolver. Los análogos de curva elíptica al RSA tienen sólo interés académico ya que no ofrecen ninguna ventaja práctica con respecto del sistema RSA ya que su seguridad está basada en el mismo problema, la factorización de enteros grandes.

La situación es diferente para el sistema de curva elíptica que son variantes del problema del logaritmo discreto. La seguridad de estos sistemas depende de la dificultad del siguiente problema:

> Datos dos puntos G e Y de una curva elíptica tal que Y = k G, donde k es un entero, encontrar el entro k. Este problema se conoce como el problema del logaritmo discreto elíptico.

Actualmente los métodos para calcular logaritmos discretos elípticos son mucho menos eficientes que los que existen para factorizar o calcular logaritmos discretos convencionales. Como resultado, se pueden usa tamaños de clave menores para conseguir el mismo grado de seguridad que con los sistemas de clave pública convencionales y al mismo tiempo necesitan menor potencia de cálculo. En los

próximos años se espera que con la popularización de dispositivos móviles conectados tales como las PDA's o teléfonos móviles de tercera generación que disponen de recursos de cálculo limitados.

Se pueden construir sistema de curva elíptica de cifrado, firmado e intercambio de claves análogos a ElGamal, DSA y Diffie-Hellman. Estas variantes parecen ofrecer ventajas de implementación sobre los sistemas originales y recientemente han despertado el interés de la industria y de la comunidad académica.

Seguridad

En general los mejore ataque conocidos al problema del logaritmo discreto elíptico son todos métodos de fuerza bruta. La presente falta de ataques específicos significa que una menor longitud de claves para los sistemas de curva elíptica proporciona una seguridad similar que las claves de mayor longitud que deben usarse para los sistemas criptográficos basados en logaritmos discretos y en la factorización de enteros. Para algunos casos de curvas elípticas existen ataque s eficientes. Para ciertas curvas elípticas se ha podido reducir el problema del logaritmo discreto sobre curvas elípticas al tradicional problema del logaritmo discreto con lo que se necesitaría el mismo tamaño de clave que en los sistemas tradicionales. A pesar de ello, estos caso particulares son de sobras conocidos y clasificados se pueden evitar fácilmente.

En 1997 la criptografía de curva elíptica empezó a recibir mas atención, al final de 1999 no había desarrollos significativos en la seguridad de estos sistemas. Mientras la situación continua, crece la confianza de que los sistemas de curva elíptica ofrecen mayor seguridad de la que actualmente se les supone. En particular hay algunas pruebas de que el uso de ciertas curvas elípticas especiales, conocidas como curvas de Koblitz, ofrecen implementaciones muy rápidas y podrían permitir nuevos tipos de ataques especializados. Como punto de partida, los ataques básicos por fuerza bruta pueden mejorar cuando se atacan estar curvas.

2.5 Firma digital

La firma digital es un método de autentificación digital de la información similar a la firma ordinaria (física) en papel, pero implementado utilizando técnicas del campo del **cifrado de clave pública**. Un método de firma digital generalmente define dos algoritmos complementarios, uno para firmar y otro para verificar y el resultado del proceso de firmado también se llama firma digital.

La firma digital difiere en algunos aspectos de su contraparte física.

La firma digital, no confundir con certificado digital, es una firma electrónica que puede ser utilizada para autentificar la identidad del emisor de un mensaje o del firmante de un documento y asegurar que el contenido original del mensaje o documento que se ha enviado no ha sido alterado.

Las firmas digitales son fácilmente transportables, no pueden ser imitadas por una tercera persona y pueden ser "timbradas" con la fecha y hora de emisión. Otro requisito de la firma digital es el de no repudio, es decir que una vez firmado un documento o mensaje, el emisor firmante de ese mensaje no pueda negar su autoría.

La firma digital puede utilizarse con cualquier tipo de mensaje, tanto si se trata de un mensaje cifrado como de un mensaje de texto en claro para asegurar al receptor la identidad del emisor y que el mensaje a llegado intacto.

Habitualmente el firmado de un mensaje consiste en cifrar con la clave privada un resumen del mensaje. Así, si A envía a B un mensaje firmado, para verificar la autenticidad del mensaje B calcula el resumen del mensaje, descifra con la clave pública A firma digital y comprueba que coincide con el resumen del mensaje. Como sólo A posee la clave privada se asegura que la firma ha sido producida por A.

2.6 Certificados digitales e Infraestructura de clave pública

Un certificado de clave pública (o certificado de identidad) es un documento digital usa una firma digital para asociar una clave pública con una identidad, (información tal como el nombre de una persona física o jurídica, su dirección, etc). El certificado se utiliza para verificar que una clave pública pertenece a un individuo.

En algunos casos puede ser necesario crear una cadena de certificados en las que cada certificado de la cadena certifica la autenticidad del anterior hasta llegar al de la identidad que se necesita verificar.

En su forma más simple, los certificados contienen una clave pública y un nombre. Habitualmente también contiene una fecha de caducidad, un número de serie y posiblemente otra información adicional y, lo más importante, contiene la firma digital de la entidad emisora.

El formato de certificado digital más ampliamente utilizado es el definido en la norma ITU-T X.509 que constituye el estándar de facto.

En un sistema típico de Infraestructura de Clave Pública (conocida por sus siglas en inglés PKI), un certificado digital se firma con la clave privada de una Autoridad de Certificación (conocida por sus siglas en inglés CA) en la que se supone que los participantes confían y que es la encargada de verificar la identidad de las personas físicas o jurídicas para las que emite los certificados.

En un sistema típico de Infraestructura de Clave Pública (conocida por sus siglas en inglés PKI), un certificado digital se firma con la clave privada de una Autoridad de Certificación (conocida por sus siglas en inglés CA) en la que se supone que los participantes confían y que es la encargada de verificar la identidad de las personas físicas o jurídicas a las que firma sus certificados.

Los certificados se usan para generar confianza en la legitimación de la clave pública. Los certificados son esencialmente firmas digitales que protegen la clave pública de posibles suplantaciones, robos o alteraciones.

La verificación de una firma puede incluir la verificación de la validez del certificado de la clave pública implicada. Esta verificación se puede llevar a cabo con mayor o menor rigor según el contexto.

El uso más seguro de la autentificación implica asociar un o más certificados con cada mensaje firmado. El receptor del mensaje debe verificar el certificado usando la clave pública de la autoridad de certificación y, una vez confía en la clave pública del emisor

del mensaje, verificar la firma del mensaje. Puede haber dos o más certificados incluidos en el mensaje, formando una cadena jerárquica de certificación donde un certificado atestigua la autenticidad del certificado previo. Al final de la jerarquía de certificados está el certificado de la autoridad de certificación en el cual se confía sin el certificado de otra autoridad. La clave pública de la autoridad de certificación raíz debe ser conocido de forma independiente, por ejemplo, siendo ampliamente publicado. Es necesario denotar que hay modelos de confianza alternativos respaldados por varios investigadores que evitan el modelo jerárquico.

2.7 El futuro de la criptografía

El futuro. La criptografía cuántica: de la mesa de pruebas al uso industrial

En palabras de los expertos, el próximo paso decisivo en campo de la seguridad parece ser la criptografía cuántica, al menos en lo que se refiere a su aplicación para el transporte de datos por fibra óptica. Según la teoría cuántica, las propiedades de los fotones (quanta de luz) cambian desde el momento en que se intentan observar. Una premisa ideal para salvaguardar los secretos de las informaciones de cifrado. En la práctica para garantizar que el mensaje a cifrar es inviolable, es suficiente con cifrar la clave de cifrado utilizando fotones, cuyas propiedades tienen tal o tal otro valor, y enviarlos por fibra óptica al destinatario. El destinatario mide estos valores y se los comunica al remitente. Cuando la clave se intercepta, los valores en cuestión han cambiado, y deja de ser válida. Está claro que la utilización de la tecnología de cifrado basada en las propiedades cuánticas implica la autentificación segura de las partes implicadas, con el fin que ninguna de ellas pueda ser usurpada. Además, hoy en día desconocemos cómo transmitir datos cuánticos vía fibra óptica, a más de 70 km. En contrapartida, las ventajas están claras, ya que romper códigos cuánticos sobrepasa las capacidades de cálculo de los ordenadores actuales. Además, el cifrado cuántico es mucho menos costoso en términos económicos, que los sistemas actuales ya que no precisa más que lectores generadores de fotones. Este tipo de aparatos está previsto se comercialicen dentro de 3 o 4 años. En la actualidad empresas europeas y japonesas han puesto a punto prototipos. Se prevé que el coste del equipo por unidad esté incluso por debajo de los 50$. Nada que ver con los costes actuales…

Para concluir, si tiene que elegir un sistema criptográfico para sus datos personales o profesionales, evite cuidadosamente los algoritmos propietarios, es decir, aquellos que no han sido publicados. Esta precaución puede parecer paradójica: después de todo, son los datos publicados y analizados los que los piratas explotan con más frecuencia. Pero en el caso de la criptografía, todo funciona de modo distinto. Es porque, efectivamente una ecuación es compleja es segura, y porque su resolución sea conocida por exigir enormes medios técnicos que nadie intentará atacarla. Dicho de otro modo, la publicación de un algoritmo, con la condición de que sea sólido, representa una garantía de seguridad. A la inversa, los algoritmos secretos no ofrecen más que una mediocre protección, ya que si se descubre el secreto, las informaciones ya no están protegidas. Se trata de una auténtica amenaza habida cuenta de las competencias en desensamblado y en ingeniería inversa de la que hacen gala los especialistas. Sin contar con que la mayoría de veces, los algoritmos propietarios una vez revelados, se han mostrado de

una debilidad ridícula; puede que sea por esto que sus propietarios eran tan reticentes a hacerlos públicos….

3 Diseño del programa y método empleado

3.1 Análisis de requerimientos y funcionalidades

3.1 Casos de uso

Se han identificado los siguientes casos de uso:

1. Arrancar el servidor
2. Llamar
3. Recibir una llamada

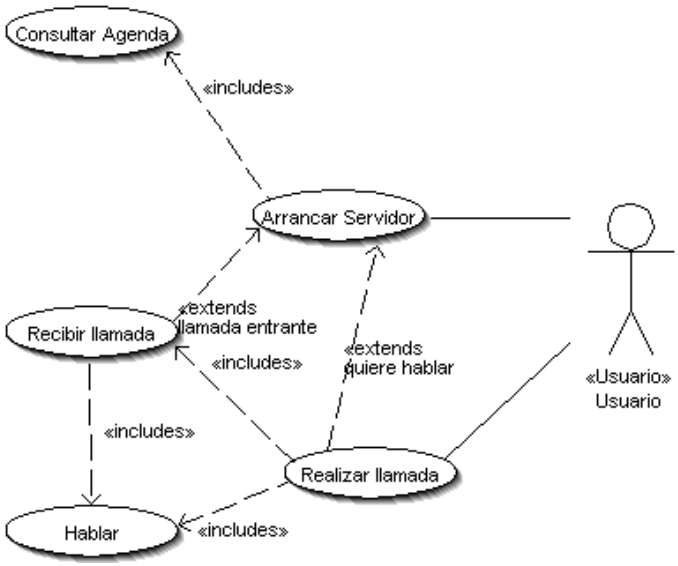

Figura 12 Diagrama de casos de uso.

3.2 Diseño de la interfaz de usuario

La interfaz de usuario consta de una ventana principal en la que se tiene la agenda con los contactos y n ventanas de conversación, una para cada usuario remoto con el que se establece una conversación. Las ventanas de conversación se pueden abrir de forma activa, cuando se llama a un usuario remoto o de forma pasiva cuando un usuario remoto es el que inicia la llamada.

Para iniciar una conversación en el modo activo, el usuario selecciona un contacto de la agenda y pulsa el botón “conectar”. A continuación se abrirá una ventana de conversación y quedará establecida una conexión con el usuario remoto, lo que permitirá iniciar la conversación.

En el modo pasivo es decir, cuando se recibe la llamada de un usuario remoto, se abre una ventana de conversación. Esto significa que la llamada ha sido aceptada de forma automática.

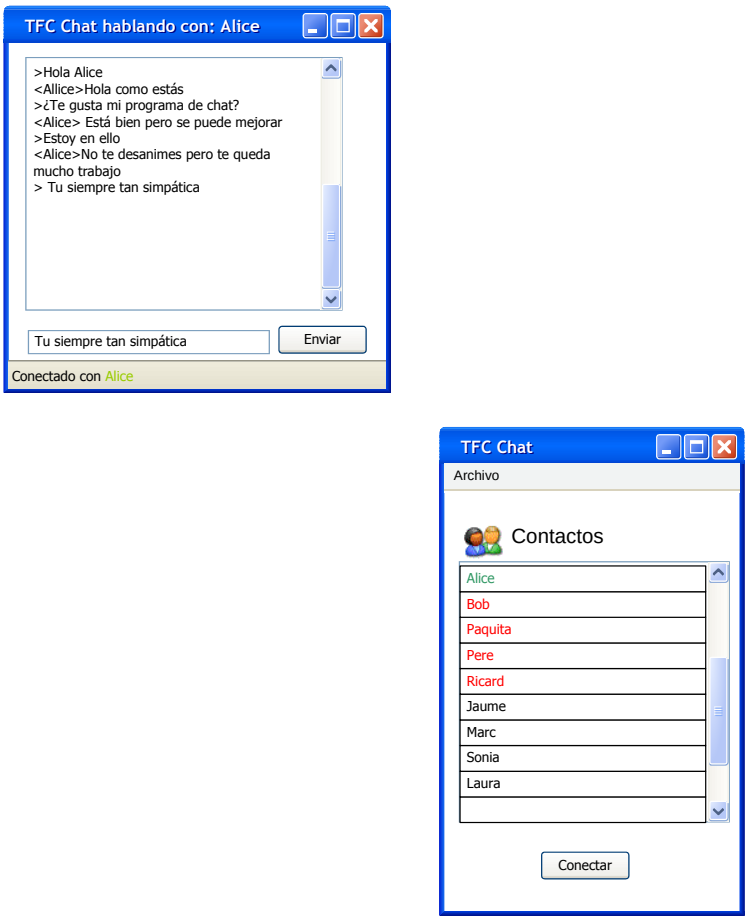

Figura 13 Diseño de la interfaz gráfica.

3.3 Diseño del sistema de comunicación, protocolo y sistema cliente-servidor.

El diseño se distingue tres elementos: el servidor, los clientes locales y los clientes remotos. El servidor y los clientes locales forman parte de la aplicación, los clientes remotos podrían ser cualquier otra aplicación que utilizase el mismo protocolo de comunicación. Para hacer una analogía con la interfaz de usuario, el servidor sería la ventana principal que contiene la libreta de direcciones y los clientes locales las ventanas que se abren para cada conversación. El resto de usuarios (sus respectivos programas) son vistos como clientes.

Como protocolo de comunicaciones utilizo TCP. El Protocolo de Control de Transmisión (TCP en sus siglas en inglés, Transmission Control Protocol) es uno de los protocolos fundamentales en Internet. Muchos programas dentro de una red de

ordenadores pueden usar TCP para crear conexiones entre ellos a través de las cuales enviarse datos. El protocolo garantiza que los datos serán entregados en su destino sin errores y en el mismo orden en que se transmitieron. También proporciona un mecanismo para distinguir distintas aplicaciones dentro de una misma máquina, a través del concepto de puerto. Desde el punto de vista del programador, usar TCP frente a UDP simplifica el diseño de la aplicación.

La única ventaja que podría tener UDP es el menor coste en tiempo para establecer una conexión y la menor latencia. Tratándose de un programa de mensajería instantánea, en el que lo que se transmite es texto, un retardo de unas pocas décimas de segundo no es crítico para el funcionamiento del sistema. De haber usado UDP se requeriría programar al menos un sistema de control de pérdidas de paquetes como el que lleva incorporado el TCP y siguiendo la filosofía general de este proyecto, intentar hacer de nuevo algo que ya existe y ampliamente probado por personas que saben mucho mas que el autor de este proyecto sería como reinventar la rueda.

De modo muy esquemático, el servidor actúa de la siguiente manera:

Si actúa como servidor

- Arranca el servidor y espera a recibir conexiones

- Recibe una conexión de un cliente remoto

- Para esa conexión cada conexión que recibe, lanza un cliente local en un hilo que se encarga de atender a esa conexión

-Vuelve a esperar más conexiones.

Si actúa como cliente:

- El servidor establece una conexión con el cliente remoto

- Lanza un cliente local que se encarga del envío y recepción con el cliente remoto.

El diagrama de clases simplificado de la aplicación es el siguiente,

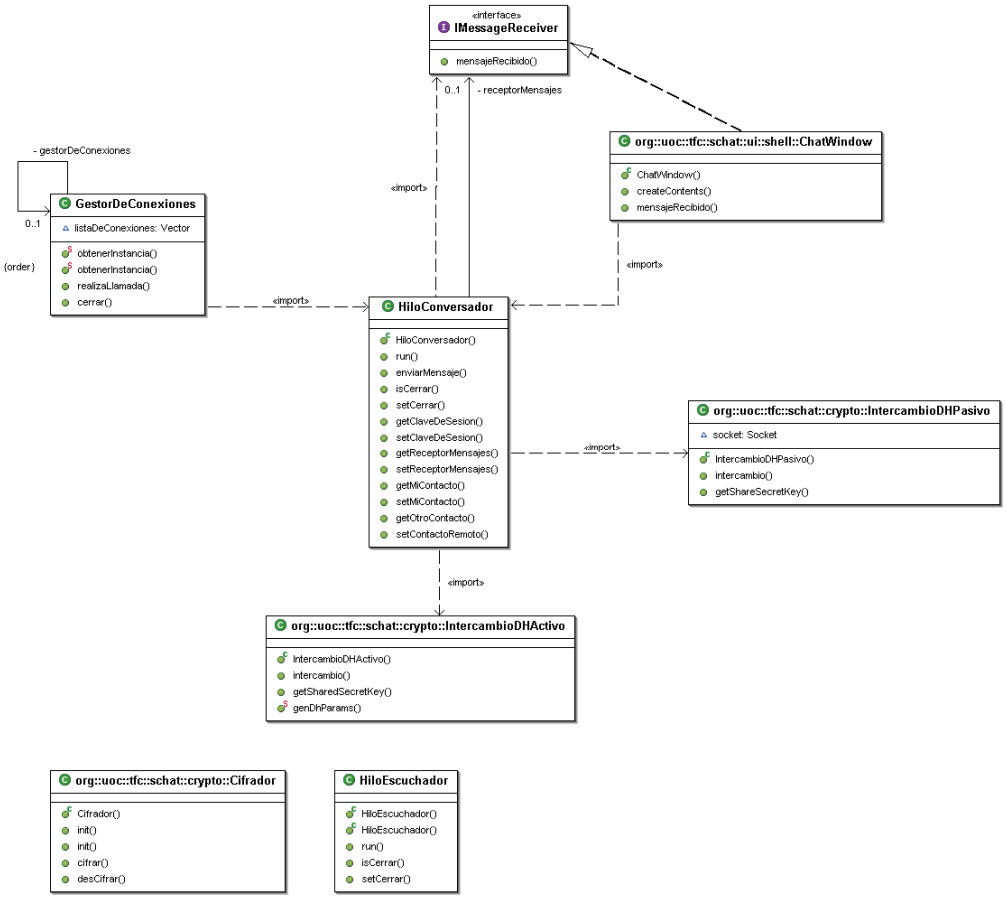

Figura 14 Diagrama de clases.

Para ayudarme a estructurar las clases y los procesos hice los siguientes diagramas de secuencia:

Diagrama de secuencia 1: Realizar una llamada.

El usuario arranca la aplicación para ello se invoca el método obtener instancia del gestor de conexiones. El gestor de conexiones, representado por la clase del mismo nombre es una clase en la que sólo se puede instanciar un único objeto, es decir se corresponde con el patrón de diseño Singleton.

Seguidamente se realiza una llamada a un usuario remoto. Se obtiene una instancia de la clase HiloConversador que es la que se encarga del proceso de comunicación entre los dos usuarios.

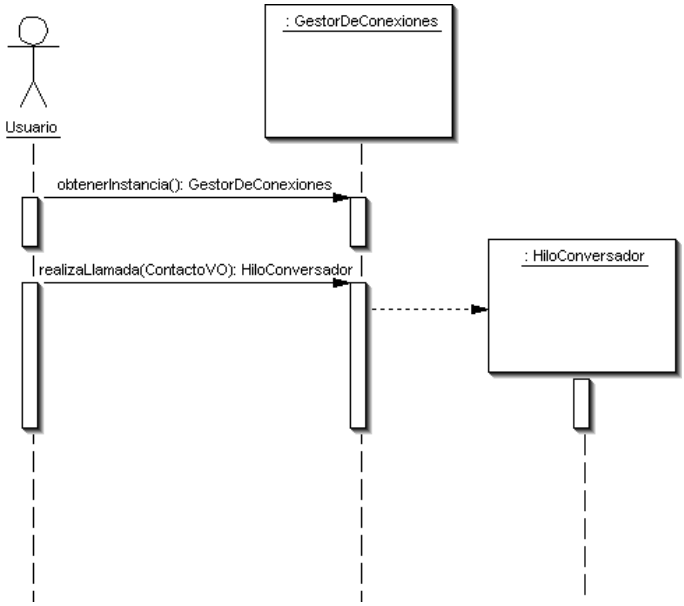

Figura 15 Diagrama de secuencia 1 Realizar llamada.

Diagrama de secuencia 2 Recibir una llamada.

El usuario arranca la el gestor de conexiones. El gestor de conexiones instancia en un hilo (thread) la clase HiloEscuchador, este se encarga de atender las conexiones entrantes. Cuando se recibe una conexión lanza un threat HiloConversador que se encarga de comunicar al interfaz de usuario los mensajes recibidos y de enviar el texto introducido desde la interfaz de usuario al programa remoto.

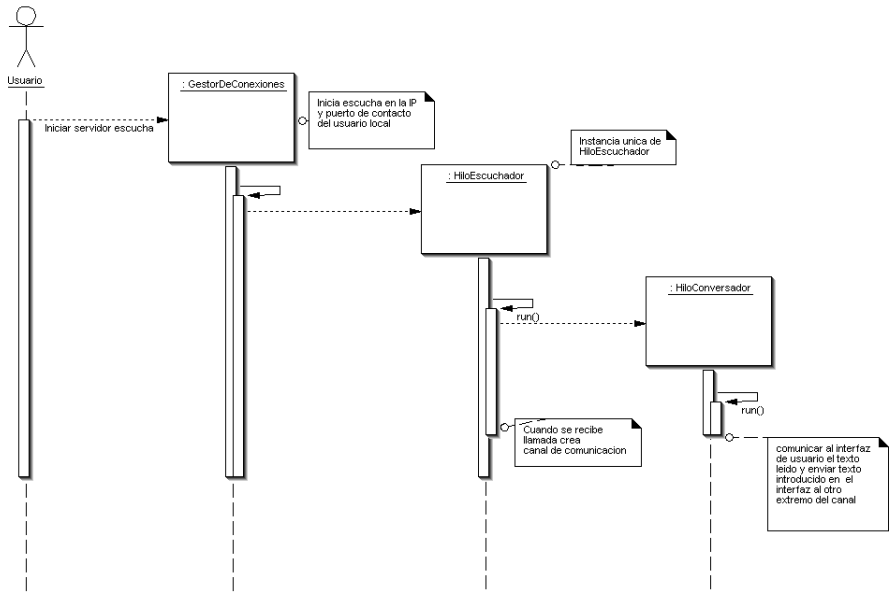

Figura 16 Diagrama de secuencia 2 Recibir llamada

3.4 Diseño del sistema de cifrado

3.4.1 Diseño del sistema de cifrado asimétrico: intercambio de claves

Tal como indican los requisitos del proyecto el intercambio de claves se realiza mediante el protocolo Diffie-Hellman. Los fundamentos matemáticos de este algoritmo/protocolo están explicados en el apartado 2.4.3. Otra manera de describir este protocolo es la siguiente:

1 A genera una clave privada aleatoria **a** Diffie-Hellman y se le envía a B los parámetros Diffie-Hellman de A G (la base generadora del grupo multiplicativo), P (el módulo primo) y L, la longitud en bits del exponente) y el resultado de calcular G^a mod P.

2 B genera una clave privada aleatoria **b** y, usando los parámetros recibidos de A calcula G^b mod P.

3 Con las claves públicas intercambiadas, A y B calculan su clave secreta común.

4 A partir de este momento se pueden intercambiar mensajes cifrados con un algoritmo simétrico.

Desde el punto de vista de la aplicación, si llamamos A a la parte que inicia la conexión, la manera de implementar el intercambio de claves es la siguiente:

1. A envía un mensaje con su nombre de usuario

2. A genera su clave pública Diffie-Hellman y se la envía a B
3. B, genera su clave pública a partir de la clave pública de A
4. Con las claves intercambiadas, ambas partes disponen de la clave secreta común y a partir de este momento, toda la información la envían cifrando los mensajes con el algoritmo de cifrado simétrico.

El *api* estándar de java proporciona la clase KeyAgreement que permite realizar el intercambio de claves Diffie-Hellman definido en el estándar in PKCS #3 de los laboratorios RSA.

3.4 Diseño del sistema de cifrado simétrico, cifrado de las comunicaciones

Para el cifrado simétrico he escogido como algoritmo de cifrado el AES en modo CBC.

El algoritmo AES ya se ha descrito en el apartado 2.2.3.

El cifrado simétrico funciona de la siguiente manera: una vez intercambiada la clave de sesión, cada mensaje, se cifra mediante el algoritmo AES. Se concatena el vector de inicialización con el resultado de cifrar el texto del mensaje; seguidamente se transforma en texto imprimible mediante la codificación Base64.

En el diseño inicial se utilizaba el algoritmo de cifrado simétrico en modo ECB pero, tal como se explica en el apartado de pruebas, al producir dos textos en claro iguales textos cifrados iguales. Esta propiedad puede no ser deseable pues de este modo se podría revelar ciertos patrones de los mensajes o de la forma de comunicación. Se dice, por ejemplo, que parte del éxito de los británicos en descifrar los códigos producidos por las máquinas Enigma durante Segunda Guerra Mundial se debió a que los alemanes siempre enviaban los mensajes con cabeceras estándar lo que facilitó en gran manera la rotura del código de las Enigma.

Durante la fase inicial de programación debido a ciertos problemas con la manipulación de los sockets en Java lo que me llevó a implementar el intercambio de mensajes en modo texto de tal manera que los datos cifrados que se transmiten son codificados en Base64 para que se puedan enviar en forma de cadenas de texto.

4 Aspectos concretos de la implementación

4.1 Metodología

En lo referente al método de programación, he intentando utilizar un método basado en las premisas utilizadas por la **programación extrema**[7]. Como es sabido, el ejemplo más popular y claro de las metodologías ágiles de programación, tiene ciertas características estándar de las cuales he intentado utilizar las que más se adaptaban a mi trabajo.

He realizado pruebas unitarias continuas para cada una de las clases principales que he programado, con el objeto de asegurar que su funcionamiento era el correcto. De este modo, dentro del árbol del código fuente existen los directorios *testSrc* en los que se encuentran las pruebas unitarias y el directorio *scripts* en el que hay unos guiones que en para arrancar estas pruebas unitarias.

4.2 Herramientas

Entorno de desarrollo Eclipse

Eclipse es un Entorno Integrado de desarrollo (IDE) libre para crear aplicaciones clientes de cualquier tipo. La primera y más importante aplicación que ha sido realizada con este entorno es el propio IDE Java llamado Java Development Toolkit (JDT) y el compilador incluido en Eclipse, que se usaron para desarrollar el propio Eclipse.

---

[7] La programación extrema o eXtreme Programming (XP) es una aproximación a la ingeniería de software formulada por Kent Beck, autor del primer libro sobre la materia, *Extreme Programming Explained: Embrace Change*. Se trata de un proceso ágil de desarrollo de software.

Las fundamentales del método son:

- Desarrollo iterativo e incremental: pequeñas mejoras, unas tras otras.
- Pruebas unitarias continuas, frecuentemente repetidas y automatizadas.
- Programación por parejas: se recomienda que las tareas de desarrollo se lleven a cabo por dos personas en un mismo puesto. Se supone que la mayor calidad del código escrito de esta manera es más importante que la posible pérdida de productividad inmediata.
- Frecuente interacción del equipo de programación con el cliente o usuario. Se recomienda que un representante del cliente trabaje junto al equipo de desarrollo.
- Corrección de todos los errores antes de añadir nueva funcionalidad. Hacer entregas frecuentes.
- Refactorización del código, es decir, rescribir ciertas partes del código para aumentar su legibilidad y mantenibilidad pero sin modificar su comportamiento.
- Propiedad del código compartida: en vez de dividir la responsabilidad en el desarrollo de cada módulo en grupos de trabajo distintos, este método promueve el que todo el personal pueda corregir y extender cualquier parte del proyecto.
- Simplicidad en el código: es la mejor manera de que las cosas funcionen. Cuando todo funcione se podrá añadir funcionalidad si es necesario.

El proyecto Eclipse se divide en tres subproyectos:

El Core (o núcleo) de la aplicación, que incluye el subsistema de ayuda, la plataforma para trabajo colaborativo, el Workbench (construido sobre SWT y JFace) y el Workspace para gestionar proyectos.

Java Development Toolkit (JDT), donde la contribución de Erich Gamma ha sido fundamental.

Plug-in Development Enviroment (PDE), que proporciona las herramientas para el desarrollo de nuevos módulos.

Eclipse fue creado originalmente por IBM. Ahora lo desarrolla la Fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios.

El entorno integrado de desarrollo (IDE) de Eclipse emplea módulos (en inglés plug-in) para proporcionar toda su funcionalidad, a diferencia de otros entornos monolíticos donde las funcionalidades están generalmente prefijadas, las necesite el usuario o no. El mecanismo de módulos permite que el entorno de desarrollo soporte otros lenguajes además de Java. Por ejemplo, existe un módulo para dar soporte a C/C++. Existen módulos para añadir un poco de todo, desde Telnet hasta soporte a bases de datos.

Los componentes gráficos (widget) de Eclipse están basados en un juego de herramientas de tercera generación para Java de IBM llamado SWT que mejora los de primera y segunda generación de Sun (AWT y Swing, respectivamente). La interfaz de usuario de Eclipse cuenta con una capa intermedia de interfaz gráfica (GUI) llamada JFace, lo que simplifica la creación de aplicaciones basadas en SWT.

La definición que da el proyecto Eclipse acerca de su software es: “una especie de herramienta universal - un IDE abierto y extensible para todo y nada en particular”.

Una de sus grandes ventajas es que basa su funcionamiento en plugins con lo que es ampliable para que haga prácticamente cualquier cosa, desde edición de XML a control del Tomcat, pasando por plugins para otros lenguajes como Perl o Shell Script.

Omondo UML

Se trata de un “plug-in” de Eclipse para realizar diagramas UML integrados con el con el código. Es un producto comercial pero existe una versión gratuita para proyectos “unipersonales”.

4.3 Bibliotecas externas

Utilizar el API estándar de Java evitando, dentro de lo posible, recurrir a bibliotecas externas, para evitar las bibliotecas criptográficas externas que suelen dar muchos problemas, especialmente a la hora de la instalación del programa en otros sistemas, ya que puede requerir modificaciones de la configuración de los permisos de seguridad la Máquina virtual de Java (JVM), modificaciones que no suelen ser triviales, es decir pueden representar una dificultad sobretodo si se trata de un usuario con conocimientos medios de informática.

Esto se ha cumplido especialmente con las bibliotecas criptográficas ya que solo se han usado las JCE (Java Cryptography Extension) que incorpora la JDK 1.4.2 "de serie".

Por otro lado, han sido necesarias algunas bibliotecas externas, en particular para la manipulación de archivos XML como el paquete *Digester* del *Proyecto Jakarta* de Apache, que es de código abierto.

En caso de necesitar bibliotecas externas, siempre he elegido las que tienen licencia abierta de tipo GPL o similar.

Log4j

Es una utilidad basada en Java para producir trazas (en inglés logs) del funcionamiento de un programa. Se utiliza principalmente como herramienta de depurado para los casos en los que no es posible utilizar un *debugger*. Es uno de los muchos proyectos de la Fundación de Software Apache. Como todos los proyectos de esta fundación log4J se distribuye bajo licencia de código abierto.

Jakarta Digester

Digester es muy útil para leer archivos de configuración. De hecho originalmente se creo para leer el archivo de configuración de Struts, posteriormente se extrajo del Struts para colocarlo dentro del paquete Commons del proyecto Jakarta[8].

El paquete Commons Digester proporciona una de las maneras más sencillas de convertir un documento XML en objetos. La clase Digester permite al programador especificar una serie de acciones que se realizan cuando el parser encuentra ciertos patrones simples en el documento XML. Lo he utilizado para leer los archivos XML en los que se definen los contactos de la aplicación.

Base64

Clase Java, con licencia dominio público, procedente de http://iharder.net/base64 para realizar la conversión de binario a codifiación Base64

Eclipse RCP y SWT

Tanto RCP como SWT son tecnologías procedentes del proyecto Eclipse. RPC, acrónimo de Plataforma de Cliente Rico (en inglés Rich Client Plataform) es una plataforma de desarrollo de software que ayuda al desarrollo rápido de aplicaciones Java mediante la aportación de los servicios más usados frecuentemente. Permite al programador construir aplicaciones completas basándose en una plataforma preexistente sin tener que empezar desde cero. Permite beneficiarse de funcionalidades probadas.

SWT es el acrónimo de Standard Widget Toolkit. Se trata de un una infraestructura de software (en inglés *software framework*) parra desarrollar interfaces gráficos de usuario en Java. Se trata de la alternativa a AWT y Swing para la creación de aplicaciones con

[8] El Proyecto Jakarta (en ingles The Jakarta Project) ofrece un diverso repertorio de soluciones Java de código abierto y es parte de la Fundación Apache de Software (The Apache Software Foundation ASF) que promueve un proceso de colaboración en el desarrollo, basado en el consenso bajo una licencia de software abierto.

interfaz de usuario. Está escrito en Java estándar y accede a bibliotecas específicas de la plataforma en que se ejecuta mediante JNI (Java Native Interface) para conseguir un rendimiento similar al de aplicaciones nativas de la plataforma.

5 Manual de instalación

Instalación para entornos Windows

Requisitos

Máquina virtual de Java, J2SE Java Runtime Environment (JRE) versión 1.4 o posterior correctamente instalada.

Instalación

1. Descomprimir archivo **`SChat.zip`** en la carpeta **`c:\SChat`** .
2. Editar el archivo **`chat.bat`** que se encuentra en la carpeta **`c:\SChat`** y modificar, si es necesario, los parámetros correspondientes al entorno en el que se va a realizar la instalación. Esto significa que se ha de sustituir la siguiente línea

   ```
   set JAVA_HOME=C:\j2sdk1.4.2_08\jre
   ```

   por el nombre de la carpeta donde está instalada la máquina virtual de java. Por ejemplo, en el caso de que sólo se tenga instalado la JRE sin el JSDK, y según la versión que se tenga, esta línea quedaría de la siguiente forma:

   ```
   set JAVA_HOME=C:\Archivos de programa\Java\j2re1.4.2_08
   ```

3. Editar el archivo `user.xml` e introducir el nombre de usuario, la dirección IP y el puerto a través del cual se quieren recibir las llamadas de clientes remotos. Por ejemplo:

   ```
   <nombre>xavi</nombre>
   <direccionIP>192.168.10.19</direccionIP>
   <puerto>8000</puerto>
   ```

4. Si se desea se puede editar el arvhivo **`agenda.xml`** para añadir o borrar contactos. Para introducir un nuevo contacto es necesario conocer:
   - El nombre de usuario.
   - La dirección IP y el puerto por el que recibe llamadas.
   - Opcionalmente se pueden introducir los parámetros **p**, **g** y **l** para el intercambio Diffie-Hellman, aunque esto último no es indispensable para el funcionamiento del chat.

Una vez completado el proceso de instalación, para iniciar el programa bastaría con ejecutar el archivo **`chat.bat`**, que se encuentra en la carpeta **`c:\SChat`** bien desde la línea de comandos MSDOS o bien haciendo doble clic con el ratón sobre el.

6 Manual de usuario

Una vez iniciado el programa aparece la ventana como la de la figura 11.

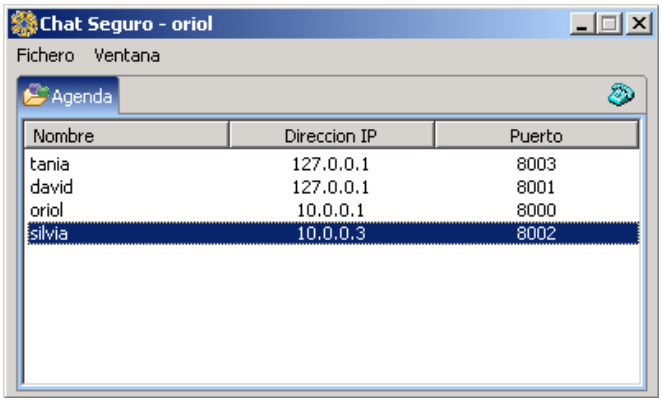

Figura 17 Ventana principal del programa o ventana de la agenda.

Para iniciar una conversación se selecciona con el ratón un contacto y se pulsa sobre el icono con forma de teléfono  situado arriba a la derecha.

Al pulsar sobre icono con el teléfono  se establece la conexión con el contacto de la agenda y se abre otra ventana como la de la figura 12.

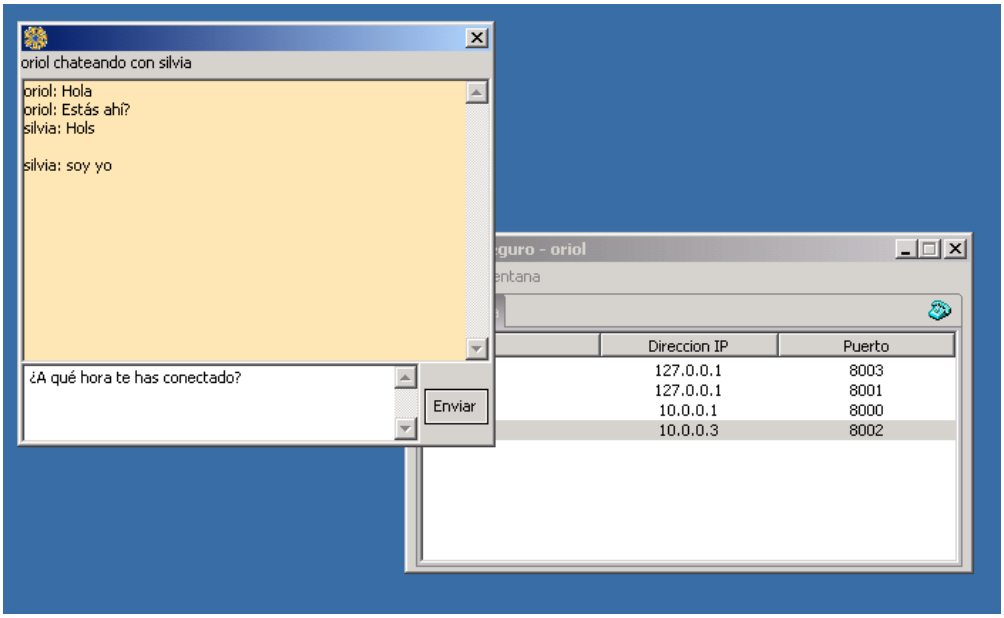

Figura 18 Ventana de la agenda y ventana de conversación.

Para enviar un mensaje se escribe en el cuadro de color blanco y se pulsa el botón enviar 

. También se puede enviar un mensaje escribiendo una frase y al final de la frase se pulsa la tecla retorno de carro.

Para acabar una conversación se cierra la ventana de conversación pulsando sobre el botón de cerrar ventana  , situado arriba a la derecha.

Finalmente si se quiere cerrar el programa basta con pulsar sobre el botón de cerrar ventana de la ventana de la agenda.

7 Descripción de las pruebas de funcionamiento

Aparte de las pruebas unitarias que se realizaron durante la programación de la aplicación con para comprobar el funcionamiento de las clases principales, se ha hecho una prueba consistente en simular un *sniffer* mediante un programa que realiza la función de proxy.

Recibe una conexión TCP entrante por un puerto determinado, copia por pantalla y en un archivo lo que recibe y lo retransmite a la dirección y el puerto del que debería ser el programa cliente en condiciones normales.

La situación que se pretende simular es la que se describe en la figura TO-DO. El usuario A se comunica con el usuario B a través de una red no segura. Un pirata informático captura la comunicación.

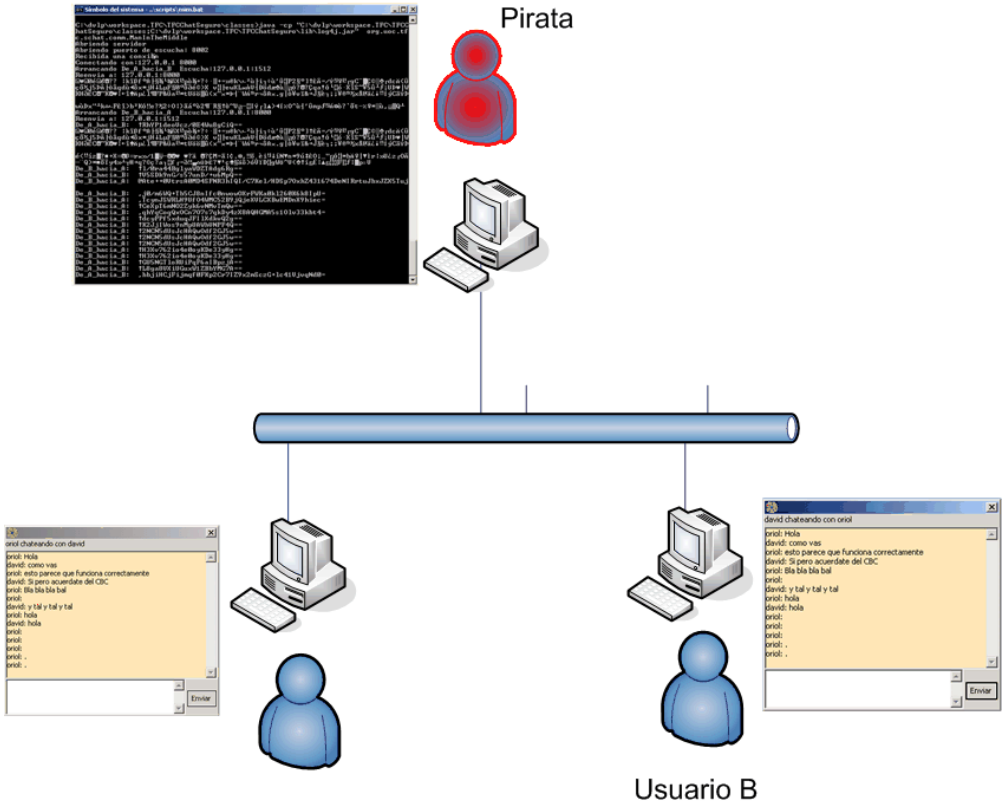

Figura 19 Esquema de la red en la simulación.

Esta prueba se hizo toda sobre un mismo ordenador. Para ello se utilizo el programa Java llamad `ManInTheMiddle.java` que se incluye en el directorio testSrc del código fuente. El programa lanza dos hilos. Cada uno de los dos hilos se encarga de retransmitir el tráfico de red que recibe en un sentido.

La captura de pantalla de la prueba , ya que por motivos de disponibilidad de hardware se hizo en un único ordenador

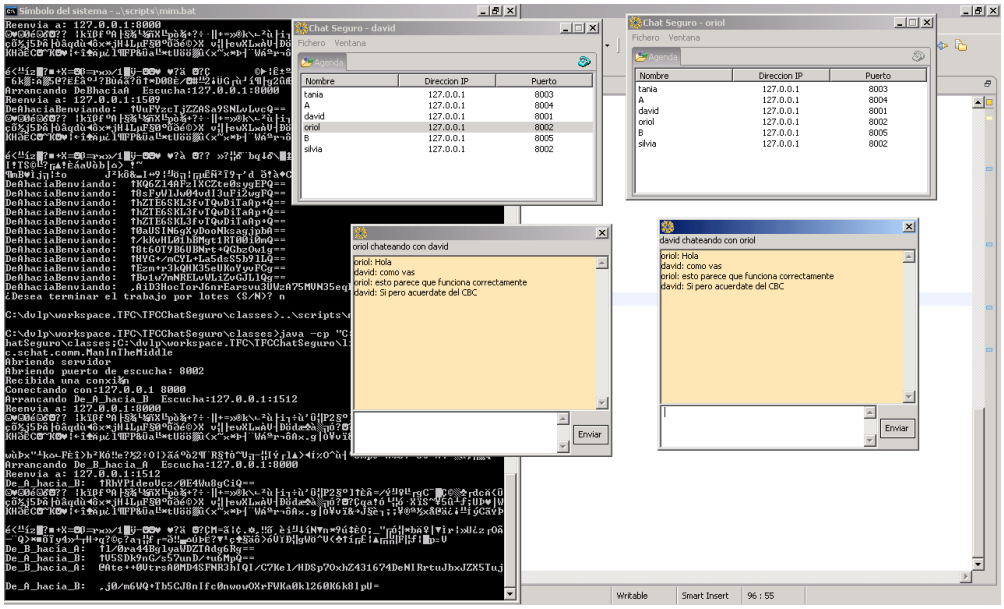

Figura 20 Captura de la simulación

El contenido del fichero de captura se puede verse en el cuadro de la página siguiente. Tal como puede verse en este cuadro, en la primera comunicación del cliente A al cliente B es se realiza en binario, igual que la primera de B hacia A. Estos dos intercambios de datos binarios corresponden al proceso de intercambio de clave de Diffie-Hellman. Una vez acordada la clave secreta común o clave de sesión el tráfico pasa a ser cifrado y se envía en formato Base64. Lo primero que se envía cifrado es el nombre de usuario del cliente A al cliente B, a partir de este punto los usuarios pueden empezar a hablar.

Esta prueba sirvió para constatar que el diseño del sistema de cifrado no era el adecuado. Al principio se implementó el cifrado simétrico con el algoritmo AES en modo ECB (Electronic Code Book). Con este modo de aplicar el algoritmo dos textos planos iguales dan lugar a dos textos cifrados idénticos. Por lo tanto no se esconden posibles patrones en las comunicaciones.

El sistema de cifrado simétrico, que se encuentra en la clase java `Cifrador.java` se modificó para que cifre en modo CBC. Con este modo de funcionamiento del cifrado de bloque cifrar se añade un vector de inicialización con datos aleatorios cada vez que se cifra un mensaje, este vector se transmite en claro ya que la seguridad del cifrado sólo depende de la clave, que se mantiene secreta.

```
De_A_hacia_B: 0,┌¦0,┌→⁻       *†H†÷
┌└┌0,┌
┐?? Ýk‹áføAÃõóÀ¬¤XÈp•·ó+?öúº+=¯©k\ ý—Ãi¿ö—'ê¹P2⊥ø]↑Ô¶=/í¼
ÈÚgCîÛ€☼0◘¸°│ÚdcŽ{ê‡å«j5è¶Ãã ƒqd—◄"x•*jH├LæFõ0øåÐ,¸>X
v¹ÃewXLÏ·V´Ñ"d'│…°»à?┐?€qa!àÀÎ¢úX‹S~¾5–
ÁŸ;Uè└³WKHÐÒC┐~K┐└Ý←Œ⁻Žæ¨l¶FP&êaÈ*tU""²–
(x~ž*ç´ùWµ¦rª"Až.g³ã¾v‹&→JõŠ¿;;¾©¦«x†@„¨-ÊÖìCÆíè
,…◘<Ê¡zÛ?þ+XÍ┐áÍrÏ¯/1Û˜-┐┐└ÿ└?… ┐??
Ë8┼w«K☼áûà%e™_^G\Ù¥‰ŒÑ┬Xy«•TÓã$(0"└L²0ü#ñü•û¢Âs,7OCCE£ð[DzŸíÍ…ø!ýFô×
»^│®—n Â$ig_Œ¶d
Ö÷xê,-Ü ªØ¶$«◄™Ã_ëŽ¯µ-ÎÍ ó°I8%g•2S.-t(Ýeâ│ê

De_B_hacia_A: 0,┌¦0,┌→⁻       *†H†÷
┌└┌0,┌
┐?? Ýk‹áføAÃõóÀ¬¤XÈp•·ó+?öúº+=¯©k\ ý—Ãi¿ö—
'ê¹P2⊥ø]↑Ô¶=/í¼ÈÚgCîÛ€☼0◘¸°│ÚdcŽ{ê‡å«j5è¶Ãã ƒqd—◄"x•*jH├LæFõ0øåÐ,¸>X
v¹ÃewXLÏ·V´Ñ"d'│…°»à?┐?€qa!àÀÎ¢úX‹S~¾5–
ÁŸ;Uè└³WKHÐÒC┐~K┐└Ý←Œ⁻Žæ¨l¶FP&êaÈ*tU""²–
(x~ž*ç´ùWµ¦rª"Až.g³ã¾v‹&→JõŠ¿;;¾©¦«x†@„¨-ÊÖìCÆíè
,…◘<Ê¡zÛ?þ+XÍ┐áÍrÏ¯/1Û˜-┐┐└ÿ└?… ┐??
²c†®¯]þ÷8M┬‹o¸¦Õ?Dï ªaÒƒŠ4Y[ŸQg¦A¤àE»D'UwOkú.£Å┘—c—øYÃQ,┴E
       ◘Š¥¹â(Æª/iŒëJÕÎî®÷X¢8O┬†│êNY¹[½-lÃ´IwaùÍÇ◄?#¶ü
y‡{´ît0Hö¸ë Z$¡ÙÃzÃ
De_A_hacia_B: ↑Re/XBBkMpodqnSqaNP9zJQ==
De_A_hacia_B: ↑rSwkuxe3JEgRkuOWg+BJBA==
De_B_hacia_A: ↑CWfvzSSnchecosBpjhS/rQ==
De_B_hacia_A: ↑3zIF1v/ocYepHWToIJvQ0g==
De_A_hacia_B: ,nO5Jiiby64Hq6NesAd7n9HIXRbP1ZEeCMoxgCYlytkc=
```

8 Comentarios y conclusiones

Los objetivos mínimos del proyecto se han cumplido. Se ha creado un sistema de mensajería instantánea en el que la comunicación es segura mediante cifrado y con un sistema de intercambio de clave de sesión mediante el sistema Diffie-Hellman.

Este sistema se ha implementado utilizando dentro de lo posible herramientas de código abierto y multiplataforma, incrementando el posible número de usuarios potenciales del sistema. Sin embargo su desarrollo y prueba se ha realizado, dentro de la plataforma más utilizada actualmente: Microsoft Windows XP.

La realización de este proyecto me ha permitido profundizar en los conocimientos adquiridos en las asignaturas de "Xarxes, Aplicacions y Protocols d'Internet", todas las asignaturas relacionadas con la programación así como las relacionadas con el análisis y diseño de software y muy especialmente en los conocimientos adquiridos en "Criptografia" y "Seguretat en Xarxes de computadors". Además he aprendido a utilizar en profundidad Internet como una herramienta de búsqueda de información, como demuestra el hecho que gran parte de mi bibliografía se de recursos en línea.

Personalmente este proyecto me ha servido para tomar plena conciencia de mi capacidad de trabajo y de los conocimientos que he adquirido a lo largo de la carrera, así como de lo profundo y complejo que llega a ser el campo de la seguridad informática, en el cual siempre hay innovaciones constantes.

Por último mencionar que el trabajo está lejos de estar acabado: existen multitud de posibilidades de ampliación y mejora.

- El sistema criptográfico, a pesar de funcionar correctamente no protege de posibles ataques del tipo "hombre de en medio". Hubiese sido aconsejable utilizar algún mecanismo para autentificar las identidades como son las firmas digitales.
- Las funcionalidades relacionadas con la gestión de la agenda de contactos son muy limitadas. Falta la posibilidad de añadir y borrar contactos desde la interfaz gráfica sin tener que editar a mano el archivo XML.
- Por diversos motivos se ha utilizado la versión de java 1.4.2 cuando la actual es la 1.5, además, no he utilizado las posibilidades que brinda el JDK 1.4 tales como las nueva API nio (new Input/Output) que mejora en algunos casos la gestión de Sockets y entre otras cosas permite programar un servidor sin necesidad de lanzar hilos (threads en inglés).
- Por otra parte el sistema no ha sido sometido a pruebas exhaustivas. Sería necesario probar como funciona ante cargas de varios cientos o incluso miles de conexiones a pesar de que es poco probable, dada la naturaleza de la aplicación, que un usuario intentase conectarse simultáneamente con tantos contactos.
- Otra ampliación interesante y casi necesaria si se pretende un uso por el público en general, sería la de un pequeño servidor que controlase la presencia de usuarios conectados y sirviese de punto central para distribuir las direcciones IP de los clientes conectados. Debido a la escasez de direcciones IP en el protocolo

IPv4 la tendencia actual es que los usuarios residenciales dispongan de dirección IP dinámica, haciéndose casi imposible la comunicación punto a punto por dirección IP. El día que se complete el despliegue de la nueva versión del protocolo IPv6 habrá disponibles el suficiente número de direcciones IP como para olvidarnos de que alguna vez existieron direcciones IP dinámicas.

9 Bibliografía y recursos utilizados

9.1 Libros

[B1] DOMINGO FERRER, Josep; HERRERA JUANCOMARTÍ, Jordi; RIFÀ POUS, Helena, *Criptografia*, 2ª edición. Barcelona. UOC, 2004. 270 páginas. ISBN: 84-9788-055-2.

[B2] WEISS, JASON; *Java Cryptography extensions: practical guide for programmers*, San Francisco, (California), Morgan Kaufmann, 2004, ISBN 0-12-742751-1.

[B3] KNUDSEN, Jonathan B, *Java Cryptography*, 1ª edición. Sebastopol (California) O'Reilly & Associates,1998, 362 páginas. ISBN: 1-56592-402-9.

[B4] SCHNEIER, B. *Applied Cryptography. Protocols, Algorithms and Source code in C*. 2ª edición. New York [etc.] John Wiley & Sons cop.1996. ISBN: 0471128457.

[B5] MENEZES, Alfred J. et alia, *Handbook of applied cryptography*, Boca Raton (Florida). CRC Press, 1997. ISBN: 084-9385-237.

9.1 Recursos en línea

[R1] Miller, Roy W.; Williams, Adam; *Java sockets 101* [en línea]; IBM developer works. Disponible en Internet: http://www-128.ibm.com/developerworks/edu/j-dw-javasocks-i.html

[R2] MCGRAW, Gary; FELTEN, E. *Securing Java: Getting Down to Business with Mobile Code* [en línea]. 2ª edición enero de 1999.Wiley. Disponible en Internet: http://www.securingjava.com

[R3] *Java Cryptography Extension (JCE) for the Java 2 SDK, v 1.4.*Documentación oficial de SUN para las extensiones criptográficas de Java. Disponible en Internet: http://java.sun.com/products/jce/index-14.html

[R4] *Jabber Sofware Foundation*. Sitio del estándar abierto para programas de mensajería instantánea. Disponible en Internet: http://www.jabber.org

[R5] *The Whisper Protocol* (Protocolo de cifrado para Jabber). Disponible en Internet: https://whisperim.dev.java.net/protocol/

[R6] Philipp K. Janert, Ph.D., *Learning and Using Jakarta Digester* [en línea]; O'Reilly onJava.com; Disponible en Internet: http://www.onjava.com/lpt/a/2746

[R7] Burnette, Ed; *Rich Client Tutorial* [en línea]. Julio de 2004. The Eclipse Foundation. Disponible en Internet: http://www.eclipse.org/articles/Article-RCP-1/tutorial1.html

[R8] *Announcing the Standard for DATA ENCRYPTION STANDARD (DES) FIPS PUB 46-2*. Federal Information Processing Standards Publications, diciembre de 1993 Disponible en Internet: http://www.itl.nist.gov/fipspubs/fip46-2.htm

[R9] *Announcing the ADVANCED ENCRYPTION STANDARD (AES) FIPS PUB 197*. Federal Information Processing Standards Publications, noviembre de 2001 Disponible en Internet: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[R10] RSA Laboratories. *RSA Laboratories' Frequently Asked Questions About Today's Cryptography, Version 4.1*. RSA Security Inc. 2000. Disponible en Internet: http://www.rsasecurity.com/rsalabs/node.asp?id=2152