UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA
SECCIÓN DE POST-GRADO Y SEGUNDA ESPECIALIZACIÓN

DESARROLLO DE UN SISTEMA SEGURO DE TRANSMISIÓN DE VOZ MEDIANTE INTERNET

TESIS

Para Optar el Grado de Maestro en Ciencias
Mención: Telecomunicaciones

PRESENTADA POR:

Flor Elizabeth Díaz Vilca

LIMA - PERU
2005

UNIVERSIDAD NACIONAL DE INGENIERÍA
Facultad de Ingeniería Eléctrica y Electrónica

Desarrollo de un Sistema Seguro de Transmisión de voz mediante Internet

TESIS

Para optar el grado de Maestro en Ciencias
Mención: Ingeniería Electrónica, Especialidad: Telecomunicaciones

Presentado por:

Flor Elizabeth Díaz Vilca

Lima – Perú

Resumen

La presente tesis "Desarrollo de un Sistema Seguro de Transmisión de Voz mediante Internet" proporciona, en base a diferentes elementos criptográficos, alternativas para la seguridad de las comunicaciones de voz en Internet. Asimismo, con el fin de que la voz digitalizada viaje de manera segura en Internet, se han seleccionado algoritmos simétricos TripleDES, IDEA, Blowfish, AES, CAST5 que sean resistentes a ataques criptográficos, y dos protocolos de comunicación para el intercambio de la clave: uno diseñado en base a la criptografía de clave pública RSA y el otro, en el SSL (Secure Socket Layer). En ambos protocolos, los usuarios se autentican el uno al otro, a la vez que construyen un canal por donde intercambiarán las claves de sesión cada minuto, con el fin de aumentar la dificultad ante cualquier posible ataque.

El desarrollo de este sistema ha tomado en cuenta las consideraciones en cuanto al ancho de banda y a la calidad de servicio en las comunicaciones de voz. Para esto, se ha seleccionado el uso de codificadores de voz como G723 y GSM, los cuales disminuyen el uso del ancho de banda y utilizan detectores de actividad de la voz para incorporar tramas de silencio. Con esto, se consiguió que al tener menor cantidad de información que viaje por Internet, exista menos cantidad de paquetes perdidos. Asimismo, para la transmisión de voz, se emplea el protocolo de tiempo real (RTP, Real Time Protocol), el cual incorpora sellos de tiempos y números de secuencia en los paquetes, asegurando el reordenamiento de estos y su reproducción en el tiempo

esperado. Del mismo modo, dentro de este sistema se ha considerado el uso del protocolo de control de tiempo real (RTCP, Real Time Control Protocol), pues éste permite monitorear la calidad de servicio percibido por el receptor, y almacenar la información de los usuarios de las distintas sesiones establecidas.

La implementación del sistema propuesto ha sido desarrollada en Java, utilizando: el Java Media Framework (JMF) como Interfase de Programación de Aplicaciones multimedia API (Application Programming Interface) desarrollada por Sun. El JMF permite la reproducción y la transmisión de audio en tiempo real a través de Internet (utilizando el protocolo RTP/RTCP), la captura de audio del hardware disponible, etc. Asimismo, se ha utilizado el Java Secure Socket Extensión (JSSE) para la incorporación de SSL en el intercambio de la clave de sesión y la librería de seguridad BouncyCastle para el uso de algoritmos criptográficos dentro del sistema.

Contenido

Página

Contenido (continuación)

Página

Contenido (continuación)

Página

Contenido (continuación)

Página

Lista de figuras (continuación)

Lista de figuras (continuación)

Lista de figuras (continuación)

Lista de figuras (continuación)

Lista de tablas

Lista de tablas (continuación)

Glosario

A

ADSL (Asymmetric Digital Subcriber Line): Línea digital de abonado asimétrica; tecnología que permite la transmisión de señales analógica y digitales en sentido descendente – hacia el abonado- a velocidades de 1,5 a 8 Mbit/s y ascendente – hacia la central- de 16 a 640 kbit/s, utilizando par de cobre.

ATM (Asynchronous Transfer Mode): El modo de transferencia definido para la RDSI de Banda Ancha, en el que la información se organiza en celdas de tamaño fijo (53 octetos). Asimismo, es específica pues está orientada a paquetes, en los ciuales que utiliza un multiplexado por división en el tiempo síncrono.

API (Application Programming Interface): Un conjunto de reglas que definen cómo se accede a un servicio de un ordenador por medio de un programa de aplicación.

E

ETSI (European Telecommunications Standards Institute): Desde 1988, es el organismo europeo que reemplaza a la CEPT en la emisión de estándares técnicos de telecomunicaciones europeos. En él están representados, además de los operadores de redes públicas, los fabricantes, investigadores y usuarios.

F

FIPS: El gobierno de los Estados Unidos ha redactado las normas denominadas Federal Information Processing Standard (FIPS).

H

H.323: La recomendación H.323 del ITU-T define una arquitectura para el soporte de conferencias multimedia (audio, video, datos) sobre redes de conmutación de paquetes con dos o más participantes. El caso IP sólo es un caso particular de los incluidos en H.323, aunque en la práctica es la aplicación más difundida. La arquitectura propuesta por H.323 es muy completa y cubre prácticamente todos los aspectos necesarios para la definición de un sistema de comunicaciones multimedia: tipos y características de los equipos, servicios que soporta y protocolos necesarios en el plano de usuario y en el de control (señalización). Son sistemas ampliamente utilizados porque ofrecen garantías de estándar consolidado ya que existen numerosos equipos y aplicaciones en el mercado.

HTTP (HyperText Transmission Protocol : Protocolo de transmisión de hipertexto usado en Internet para la transferencia de documentos WWW. Cada documento –o recurso- tiene en la Web una dirección única, denominada URL. La mayoría de las URLs de la Web comienzan con http://, indicando que el documento está contenido en un servidor de hipertexto.

I

IMAP (Internet Messaging Access Protocol): Ampliamente aceptado por toda la industria, es un servicio de correo Internet avanzado, que aporta funciones de almacenamiento y envío, permite acceder a los usuarios a sus buzones desde cualquier estación de trabajo, realiza búsquedas, etc. Su última versión, IMAP4 se recoge en la RFC 1730 y permite a los usuarios una gestión muy eficiente de su correo, pudiendo moverse entre varios terminales, y mejorar el ancho de banda en situaciones críticas.

IANA (Internet Assigned Numbers Authority) : Es el organismo de la ISOC (Internet Society http://info.isoc.org) de la administración de las direcciones Internet (direcciones IP) así como de la creación de nuevos dominios (DNS) (Actualmente, se encuentra en estudio la creación de nuevos dominios como inc, co etc). La IANA delega la asignación de dominios ya creados a la InterNIC. Para conocer más de la IANA consulte: http://www.iana.org/iana/ o la definición por la ISOC en http://info.isoc.org:80/adopsec/index.html.

ISP (Internet Service Provider) : El proveedor de servicios de Internet es una empresa que suministra a otras empresas o individuos acceso a y presencia en Internet, además de otros servicios opcionales.

L

LDAP : Se diseñó para acceder al directorio X.500.

M

Mixer : Se usan mixers RTP para combinar múltiples flujos de datos en un solo flujo RTP. Estos dispositivos son usados para soportar aplicaciones de transmisiones de audio donde existe uno o dos usuarios en forma simultánea. Los mixers RTP no soportan aplicaciones de video.

R

Reseaux IP Europeenes -- RIPE (Redes IP Europeas): Conjunto de redes europeas que utilizan el conjunto de protocolos TCP/IP.

RTSP (Real Time Streaming Protocol): Es un protocolo del nivel-aplicación que lleva el control de la entrega de datos multimedia. RTSP provee una infraestructura extensible para habilitar la entrega y control de datos multimedia en-demanda, tales como audio y video. El origen de datos puede incluir la entrega de datos en vivo y datos almacenados. La finalidad de este protocolo es controlar múltiples sesiones de entrega de datos, proveer un medio para la selección de los canales de entrega tales como UDP, UDP multicast y TCP, y para seleccionar los mecanismos de entrega basados en RTP.

S

SIP (Session Initiation Protocol): El protocolo SIP es la alternativa del IETF (Internet Engineering Task Force) al estándar H.323 del ITU-T para el control de sesiones multimedia sobre redes IP. La principal ventaja de SIP sobre H.323 es la menor complejidad. Esta cualidad se deriva de dos características: utiliza una arquitectura genérica apoyada en un modelo cliente/servidor y realiza el intercambio de información a través de mensajes textuales.

U

UIT (ITU/International Telecommunications Union) : Unión Internacional de Telecomunicaciones, es uno de los organismos más antiguos de normalización. Recientemente, se ha reestructurado en tres sectores: el de normalización de telecomunicaciones (ITU-T), establecido para gestionar todas las actividades de normalización del antiguo CCITT; el de comunicaciones vía radio (ITU-R); y el sector desarrollo, que gestiona la asistencia a países en vía de desarrollo en materia de telecomunicaciones.

UTC (Coordinated Universal Time) : La hora UTC se basa en la rotación de la Tierra alrededor de su eje y en el calendario Gregoriano, que a su vez está basado en la rotación de la Tierra alrededor del Sol. Ha sido adoptado como la escala de tiempo estándar por la mayoría de las naciones del mundo.

X

X.500: Desarrollado por la Organización International para la Normalización (International Standards Organizations, ISO), el X.500 es un conjunto de reglas que definen un servicio de directorios distribuido que pueden contener casi toda la información útil relativa al usuario y a la infraestructura del sistema al que pertenece.

Introducción

Las aplicaciones tradicionales con requerimientos de tiempo real han venido utilizando las redes de conmutación de circuitos, las cuales transmiten a velocidad constante con retardos cortos. Las redes de paquetes (redes IP) se desarrollaron fundamentalmente para el transporte de datos que no requerían de temporización. Los avances en los algoritmos de compresión, las mejoras de la potencia de cálculo de los sistemas informáticos, el aumento en la capacidad del procesamiento de los paquetes en los nodos, la introducción de nuevas tecnologías de transmisión, así como el incremento y el abaratamiento de ancho de banda disponible están haciendo posible las aplicaciones en tiempo real (audio y video) a través de las redes IP [45].

Sin embargo, las transmisiones de paquetes en tiempo real deben resolver problemas inherentes a las redes de paquetes: el retardo y pérdida de paquetes, la entrega desordenada y la aparición de paquetes duplicados, etc [45]. Real Time Transport Protocol (RTP) es un protocolo basado en redes IP que permite enviar y recibir flujos de audio o de video en tiempo real por una red de propósito general. RTP fue diseñado principalmente para su uso con transmisiones multidestino; aunque también puede ser utilizado eficientemente con transmisiones punto a punto.

La problemática de la seguridad en las comunicaciones por Internet se ha extendido a las comunicaciones en tiempo real. La necesidad de proteger los contenidos multimedia distribuidos por Internet ha obligado cifrar los flujos de audio y video transportados por RTP. Además, es necesario considerar cómo se realizará la distribución de esta clave de sesión que se utilizará para el cifrado, la cual solo debe ser compartida o conocida por ambos usuarios de la comunicación.

Como parte inicial de esta tesis, en el Capítulo II, presentaremos los factores que se han tenido en cuenta para desarrollar un Sistema Seguro de Transmisión de Voz mediante Internet y los elementos que conforman la solución de este sistema.

En el Capitulo III, detallaremos las clases y librerías utilizadas para la implementación del sistema propuesto. Aquí, empleamos el lenguaje Java.

En el Capitulo IV, describimos las pruebas y mediciones del ancho de banda; conjuntamente con la evaluación de los parámetros que afectan la buena calidad de las transmisiones de voz dentro de dos escenarios distintos. Uno es una red LAN de 10 Mbps y el otro es una WAN (Internet), la cual utiliza una conexión ADSL.

Finalmente, en el Capitulo V, presentaremos las conclusiones obtenidas en nuestra tesis, así como recomendaciones para trabajos futuros que deseen garantizar la seguridad en las comunicaciones.

Los apéndices (1 al 6) presentan los diferentes conceptos teóricos fundamentales: la calidad del servicio de voz sobre IP, las técnicas de codificación de voz, los protocolos de tiempo real, la criptografía y el SSL (Secure Socket Layer). Todo este marco teórico nos servirá para la comprensión de la estructura y funcionamiento del sistema propuesto.

Capítulo 1

Planteamiento del problema

1.1 Antecedentes

Al ser Internet una red totalmente abierta y pública sin ningún tipo de jerarquía establecida ni de autoridad central, el número de usuarios aumenta diariamente, al igual que el tipo de operaciones realizadas. Internet se usa para una amplia gama de servicios como telefonía, videoconferencia, búsquedas por Web, comercio y correo electrónico, entre otros. Al contrario de la red telefónica, que es de conmutación de circuitos, Internet es una red de conmutación de paquetes, lo que significa que la información se maneja en forma de paquetes de datos. El formato exacto de estos paquetes lo define el protocolo IP (Internet Protocol), pero los datos pueden ser de cualquier tipo, incluida voz digitalizada. Así, la voz sobre IP, es áquella que se ha paquetizado (conforme lo define el protocolo IP) y que se envía sobre cualquier red IP [60]. Considerando que la voz viaja en paquetes IP; estos pueden ser detectados, capturados y reensamblados con herramientas que se encuentran disponibles en Internet, lo cual presenta una amenaza a la privacidad de las comunicaciones de voz.

1.2 Delimitación del problema

La comunicación de voz a través de Internet (VoIP) puede ser: de PC-PC, de PC-teléfono, y de teléfono a teléfono [60]. La presente tesis se basa en el modelo de PC-PC, en el cual ambos usuarios deben tener computadoras multimedia que se encuentren conectados a Internet. Estas computadoras pueden estar en una red LAN dentro de su centro laboral, con un servicio de banda ancha (ADSL, Cable Modem, etc.), o simplemente a través de la línea telefónica por medio de un proveedor de Internet. Asimismo, las computadoras deben estar equipadas con el mismo software (utilizando los mismos algoritmos de compresión y encriptamiento de datos). Todo el procesamiento que involucra la captura de voz, la reproducción, la compresión, la

descompresión, el encriptamiento, el desencriptamiento, el empaquetamiento y el desempaquetamiento de las señales de voz ocurren en el hardware y software de la PC de los usuarios.

Por otro lado, Internet es una red que ofrece un servicio de "Best Effort" sin considerar funciones de calidad en el servicio. Sin embargo, la incorporación de calidad de servicio en las redes se consigue a través de la reserva de los recursos de la red o por medio de la prioridad de los paquetes. Como sabemos, estas funciones están operativas en las redes ATM y ya se han planteado para las redes IP; pero su implementación en redes públicas como Internet demorará unos años, pues los cambios en el reemplazamiento de muchos de los actuales routers son costosos. De este modo, mientras estas funciones de calidad no entren en pleno servicio, resulta fundamental el diseño de estrategias, integradas en los propios codificadores de voz, que reduzcan el efecto de las pérdidas de paquetes de voz sobre las redes de datos [35]. Por esto, utilizaremos los codificadores de bajas tasas binarias con buenos detectores de actividad.

1.3 Formulación del problema

1.3.1 Problema general

El problema de la seguridad en Internet es de suma importancia. Por eso, nuestra tesis parte de la siguiente interrogante: ¿Qué hacer para asegurar que las comunicaciones de voz dentro de Internet se realicen de forma segura?

La incorporación de mecanismos de seguridad criptográficos dentro del sistema, nos ayuda a cumplir con los requerimientos de seguridad establecidos para las comunicaciones de datos, es decir: la confidencialidad, la autentificación y la integridad de datos. La primera se logra cuando la información intercambiada entre ambos participantes es mantenida en secreto; la segunda se obtiene cuando la parte receptora del mensaje es capaz de verificar su procedencia; y la tercera verifica que los datos no hayan sido corrompidos por terceros.

1.3.2 Problemas específicos

Para poder solucionar los problemas de seguridad en la transmisión de voz mediante Internet, se requiere resolver lo siguiente:

a.) Conocer la dirección IP del usuario destino

b.) Lograr la confidencialidad de las comunicaciones de voz realizadas en Internet

c.) Garantizar que cada usuario se autentifique antes de iniciar la comunicación de voz

d.) Realizar el intercambio de las claves de sesión entre ambos usuarios para el cifrado de los paquetes de voz

e.) Reproducir los paquetes de voz en el tiempo pre-establecido

f.) Asegurar un nivel de voz aceptable en las comunicaciones

1.4 Objetivos de la tesis

1.4.1 Objetivo general

El objetivo general de nuestra tesis consiste en desarrollar un sistema de comunicación de voz entre dos usuarios con calidad aceptable y con servicios de seguridad. De tal manera, que estos usuarios no tengan que preocuparse de los ataques que puedan sufrir en Internet o en su propia red.

1.4.2 Objetivos específicos

Para poder cumplir con el objetivo general, se requiere cumplir lo siguiente:

a) Incluir un servidor de directorio que registre las direcciones IP de los usuarios que hacen uso del sistema.

b) Cifrar los paquetes de voz utilizando algoritmos criptográficos simétricos que sean robustos frente a ataques de criptoanálisis.

c) Autentificar a ambos usuarios antes del inicio de cualquier comunicación, para evitar la suplantación por parte de un tercero.

d) Distribuir las claves de sesión entre ambos usuarios por medio de un canal seguro.

e) Utilizar el protocolo de tiempo real (RTP) para permitir la reproducción de los paquetes de voz en el orden correcto.

f) Utilizar codificadores de voz que utilicen menor cantidad de ancho de banda, de modo que reduzcan las pérdidas de paquetes.

1.5 Justificación

En la actualidad, todo tipo de comunicación que se realiza a través de Internet está expuesta a posibles ataques. De este modo, las empresas que ofrecen servicios a sus clientes por medio de esta red, se protegen con Firewalls o barreras de seguridad que impiden el acceso a intrusos. Asimismo, la información que estas envían debe viajar en forma encriptada para evitar, por ejemplo, la captura de las claves de las tarjetas de crédito. Del mismo modo, la transmisión de servicios de voz y video sobre redes IP debe ser protegida para evitar que pueda ser capturada y reproducida. Por tanto, se justifica la incorporación de un sistema seguro de transmisión de las comunicaciones de voz privadas.

Por otro lado, este sistema emplea la misma infraestructura de red, lo que supone una reducción de los costos y un ahorro en las cuentas telefónicas.

Capítulo 2

Análisis y diseño del sistema

2.1 Factores considerados en el sistema propuesto

Para el desarrollo del "Sistema Seguro de Transmisión de Voz mediante Internet", se han considerado los siguientes aspectos:

1) La inclusión de un servidor de directorio para que los usuarios que ingresen al sistema registren sus actuales direcciones IP y sus claves públicas
2) El cifrado de los paquetes de voz para brindar confidencialidad en las comunicaciones.
3) La autentificación de los usuarios a través de los certificados digitales. Para su realización, cada usuario deberá contar con un certificado digital firmado por una autoridad de certificación confiable y un almacenador de claves para guardar los certificados y las firmas digitales de los que la certifican.
4) La distribución de las claves de sesión para el cifrado de la voz entre ambos usuarios que participan en el sistema seguro de transmisión
5) El uso del protocolo RTP para las transmisiones de los flujos de audio encriptados a través de la red IP, y el uso del RTCP para el monitoreo de la calidad de servicio.
6) El uso de codificadores de voz , G723.1 y GSM, que consumen poco ancho de banda y, al mismo tiempo, entregan una calidad de voz aceptable

2.2 Estructura del sistema propuesto

La finalidad del sistema propuesto es garantizar la privacidad y confidencialidad de las conversaciones realizadas por Internet. Para conseguirlo, se la ha estructurado en tres partes:

1. El registro de los usuarios en un servidor de directorio
2. El intercambio de las claves de sesión entre ambos usuarios
3. La transmisión de paquetes de voz cifrados

2.2.1 El Registro de los usuarios en un servidor de directorio

En primer lugar, mostramos (figura 2.1) los componentes básicos de una comunicación de voz realizada por medio de Internet: dos computadoras, que tengan gran capacidad de procesamiento, que estén equipadas con dispositivos multimedia, que incluyan tarjetas de sonido full-duplex para las funciones de transmisión y recepción de voz, y, además, tengan una dirección IP para su localización en Internet.

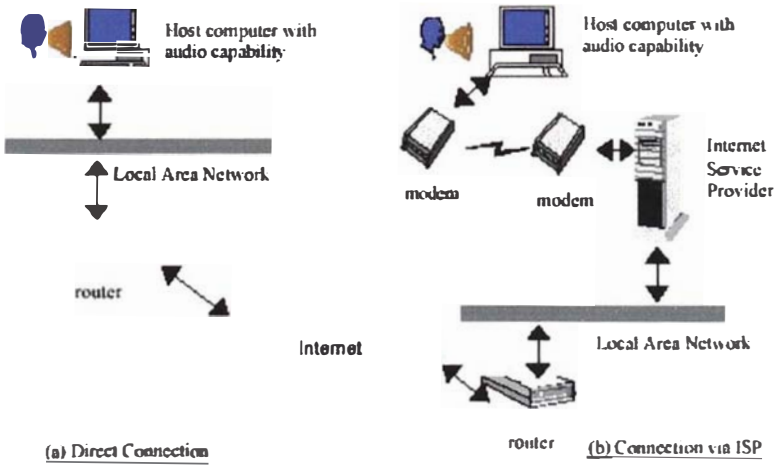

Figura 2.1: ambiente de comunicación de voz mediante Internet. [39]

En segundo lugar, este gráfico muestra dos modos posibles de conexión a Internet: conexión directa y conexión vía ISP [39]. En la primera, los usuarios son conectados directamente a Internet y tienen direcciones IP fijas. En la segunda, los usuarios acceden a Internet por medio de un Proveedor de Servicios de Internet (ISP, Internet Service Provider); asimismo, tienen una dirección IP asignada dinámicamente

durante el tiempo de conexión; de este modo, esta dirección permanecerá fija durante la sesión actual; luego será reutilizada por otros. Así, cada vez que un usuario se conecta a Internet, el ISP le asigna una dirección IP diferente.

De lo anterior, podemos deducir que el primer modo de conexión no brinda problemas; pues las direcciones IP son fijas, conocidas y de fácil reconocimiento. Sin embargo, el modo de direccionamiento dinámico tiene algunos problemas, ya que la dirección IP es conocida solo durante el tiempo de conexión; de modo que esta forma de conexión termina afectando a la mayoría de usuarios en Internet.

Así, para solucionar el problema del direccionamiento dinámico y, aún, el del direccionamiento directo (donde los usuarios que comparten una misma LAN, muchas veces desconocen sus direcciones IP asignadas), se ha optado por la utilización de un servidor de directorio. Éste se encargará de mantener la lista de usuarios activos y registrados en Internet. Lo que significa que este servidor almacenará las direcciones IP actuales junto con los datos personales proporcionados por los usuarios en el momento de su registro. De este modo, los usuarios que hagan uso del sistema, deberán primero registrarse en este servidor para obtener el listado de todos los usuarios activos conjuntamente con sus direcciones IP.

El procedimiento para que dos usuarios entablen una comunicación se muestra en la figura 2.2. Aquí, el usuario 1 desea entablar una comunicación con el usuario 3; Así, para lograrlo se debe realizar lo siguiente:

1. El usuario 1 se registra en el servidor de directorio.
2. El servidor de directorio acepta el registro del usuario 1, le retorna la lista de usuarios activos y comunica a los de la lista que se ha incorporado un nuevo usuario.
3. El usuario 1 obtiene el listado de usuarios activos con sus direcciones IP, busca en la lista al usuario 3, recoge la dirección IP, y se comunica con él.

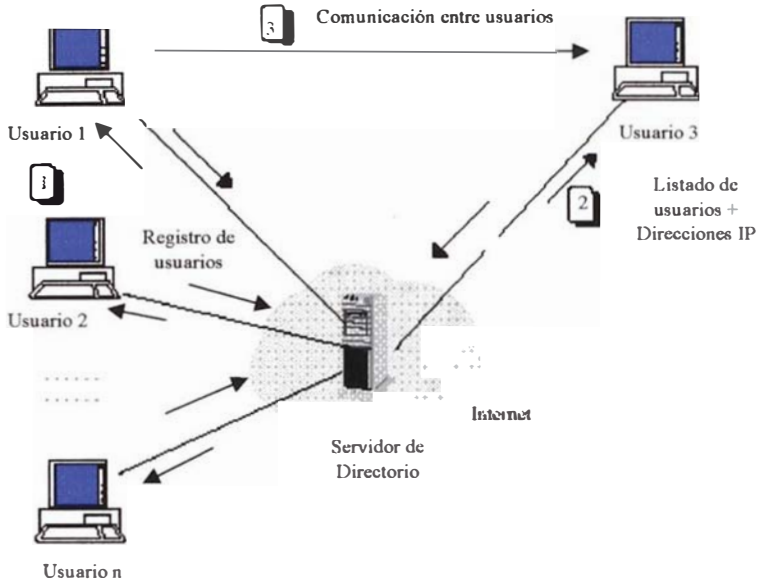

Figura 2.2: registro de usuarios en el servidor de directorio.[39] modificado

2.2.2 El intercambio de las claves de sesión entre ambos usuarios.

El intercambio de la clave de sesión entre ambos usuarios se realiza a través de protocolos de comunicación. Estos [24] son los procedimientos para realizar operaciones de comunicación, distribución de claves, establecimiento de claves de sesión, autentificación, firma digital, etc., con la utilización de algoritmos criptográficos como herramientas de protección. Para nuestra tesis, hemos planteado dos alternativas de solución: la primera crea un protocolo de comunicación utilizando el algoritmo asimétrico RSA, y la segunda usa el protocolo Secure Socket Layer (SSL).

2.2.2.1. Alternativa N° 1: protocolo de comunicación utilizando el algoritmo asimétrico RSA.

En este protocolo, dos usuarios intercambian información del acuerdo de claves para el encriptamiento de la voz. La comunicación entre ellos se realiza sobre TCP/IP.

7.2.2.1.1 Notación

C	Usuario C.
S	Usuario S.
rc	Número aleatorio , generado por el usuario C.
rs	Número aleatorio, generado por el usuario S.
Pc	Clave pública del usuario C para el encriptamiento de datos intercambiados.
Ps	Clave pública del usuario S para el encriptamiento de datos intercambiados.
Certc	Certificado correspondiente al usuario C para verificación de firmas.
Certs	Certificado correspondiente al usuario S para verificación de firmas.
Pc(y)	Resultado de aplicar la clave pública de encriptamiento del usuario C a la data y.
Ps(y)	Resultado de aplicar la clave pública de encriptamiento del usuario S a la data y.
Sc(y)	Firma del dato y con la clave privada del usuario C.
Ss(y)	Firma del dato y con la clave privada del usuario S.
Ec(y)	Encriptamiento con la clave secreta y vector de inicialización seleccionada por el usuario C utilizando el algoritmo AES.
LOCKc	Variable generada por el usuario C para agregarle como dato a su firma.
AES KEYc	Clave de encriptamiento para el algoritmo AES generado por el usuario C.
IVc	Vector de inicialización para el algoritmo AES generado por el usuario C.
HMAC KEYc	Clave para el algoritmo HMAC-SHA1 generado por el usuario C.

2.2.2.1.2 Configuración del sistema

1. Cada usuario debe tener dos pares de claves públicas, una para la firma y otra para el encriptamiento de mensajes.
2. Cada usuario debe distribuir su clave pública a los demás usuarios para el encriptamiento de los mensajes intercambiados. Así, nosotros proponemos que cada vez que el usuario ingresa al sistema genera un par de clave pública. De este modo, mientras la clave privada es guardada por el usuario, la clave pública junto con los datos de registro son enviados al servidor de directorio. De esta forma, este se encarga de distribuir las claves públicas de los usuarios.
3. Los certificados deben estar firmados por una entidad confiable.

2.2.2.1.3 Mensajes del protocolo de comunicación

El protocolo de comunicación hace intercambio de tres mensajes:

Paso N° 1	C → S	Certc, Ps(rc), Sc(rc, Certc)
Paso N° 2	C ← S	Certs, Pc(rs), Ss(rc, rs, Certs)
Paso N° 3	C → S	Ps(AES_KEYc,Ivc,HMAC_KEYc, Ec(LOCKc)), Sc(rc, rs, Ps(AES_KEYc,Ivc,HMAC_KEYc), LOCKc)

2.2.2.1.4 Acciones del protocolo:

En el Paso N° 1, el usuario C genera una variable aleatoria rc, lo encripta con la clave pública del usuario S, y envía tres datos a S: el certificado que utiliza para las firmas de mensajes (Certc), la variable aleatoria rc encriptada y su firma que incluye la variable aleatoria rc. y su certificado Certc.

En el Paso N°2, el usuario S verifica la autenticidad del certificado (verificando si se encuentra firmado por una entidad confiable), y si es conforme, se extrae la clave pública del certificado del usuario C; luego, se desencripta la variable aleatoria rc con su clave privada y procede a verificar si la firma enviada ha sido generada por C. De ser así, el usuario S genera una variable aleatoria rs y lo encripta con la clave pública del

usuario C. Finalmente, envía tres datos a C: el certificado que utiliza para la firma de mensajes (Certs), la variable aleatoria rs encriptada y su firma que incluye ambos números aleatorios (rc y rs), y su certificado Certs.

En el paso N° 3, el usuario C recibe el certificado del usuario S (Certs) y verifica su autenticidad, desencripta la variable aleatoria rs con su clave privada, luego verifica si la firma enviada ha sido generada por este usuario. De ser así, procede a generar las siguientes variables aleatorias: una clave secreta y un vector de inicialización para utilizarlo en el algoritmo AES; luego, otra clave para utilizarlo en el algoritmo HMAC y una variable LOCK para utilizarlo en esta última firma. Seguidamente, le envía al usuario S los dos datos siguientes:

1. Las claves de sesión encriptadas con la clave pública del usuario S. Entre estas, se incluye: la clave secreta, el vector de inicialización a utilizar con el algoritmo AES, el código de autentificación para utilizarlo con el algoritmo HMAC, y la variable LOCK (cifrada con el algoritmo AES que emplea las mismas claves de sesión y vector de inicialización enviados).
2. Firma del usuario C que incluye las variables aleatorias de ambos usuarios (rc y rs), las claves de sesión encriptadas con la clave pública del usuario S, y la variable LOCK (variable en texto plano).

De esta forma, el usuario S, al recepcionar los mensajes enviados, procede a desencriptar las variables para obtener el valor LOCK y comprobar la veracidad de la firma del usuario C. Luego de la autentificación de la firma, se acepta las claves de sesión enviadas.

2.2.2.1.5 Resultado del protocolo:

Las claves de sesión intercambiadas (entre el usuario C y el usuario S) son las que se utilizarán para la construcción de un canal de comunicación seguro. En la figura 2.3 y en la tabla 2.1 se muestra la configuración de este canal.

Figura 2.3: canal de comunicación entre el usuario C y el usuario S

Usuario C	Usuario S
C → S (1)	**C → S (2)**
Encriptamiento ALGORITMO : AES Clave : AES_KEY_C IV : IV_C Firma ALGORITMO : HMAC-SHA1 Clave : $hmac\_key_C$	Desencriptamiento ALGORITMO : AES Clave : AES_KEY_C IV : IV_C Verificación Firma ALGORITMO HMAC-SHA1 Clave : $hmac\_key_C$

Tabla 2.1: configuración del canal de comunicación utilizando el algoritmo asimétrico RSA.

2.2.2.2 Alternativa N° 2: Protocolo SSL

La comunicación usando SSL comienza con el intercambio de información entre el cliente y el servidor. Este intercambio forma parte del protocolo de handshake SSL.

Los propósitos principales del protocolo de handshake SSL [3] son:

1. La negociación de la suite de cifrado
2. La autentificación del servidor y del cliente
3. El establecimiento de la seguridad de información a través de los mecanismos de encriptamiento

1. ***La negociación de la suite de cifrado***. La sesión SSL empieza con una negociación entre el cliente y el servidor para determinar la suite de cifrado que usarán. Una suite de cifrado es una serie de algoritmos criptográficos y tamaños de claves que un computador puede usar para encriptar los datos. Esta incluye información acerca del algoritmo de intercambio de clave pública, del algoritmo de encriptamiento de clave secreta, y de la función hash. El cliente le informa al servidor que suite de cifrado tiene disponible, y el servidor elige la mejor.
2. ***La autentificación del servidor y del cliente.*** En el sistema seguro de transmisión de voz mediante Internet, se hace imprescindible la autentificación tanto del cliente como del servidor. De esta forma, ambos realizan un intercambio de certificados para la autentificación mutua. Para esto, cuentan con un almacén de claves donde se encuentran registrados las firmas de entidades de certificación confiables; de este modo, se puede verificar si el certificado enviado es legítimo.

3. ***El establecimiento de la seguridad de información a través de los mecanismos de encriptamiento.*** El cliente y el servidor intercambian información que les permite tener la misma clave secreta. Por ejemplo, con RSA, el cliente usa la clave pública del servidor (obtenida del certificado) para encriptar información tal como la clave secreta. El cliente envía la clave secreta encriptada al servidor. De este modo, solo el servidor puede desencriptar el mensaje ya que éste es el único que conoce la clave privada.

Como resultado de lo anterior, obtenemos e***l envío de datos encriptados***. Ambos, el cliente y el servidor, tienen acceso a la misma clave secreta. Así, con cada mensaje, hacen uso de una función hash criptográfica (seleccionada en el paso uno), y de un código de autentificación compartido para calcular el HMAC que adicionan al mensaje. Luego, se usa un algoritmo simétrico y una clave secreta (negociados en el paso uno) para encriptar el mensaje junto con el HMAC. De esta forma, el cliente y el servidor pueden comunicarse de manera segura.

2.2.2.3. Resultado del uso de los protocolos de comunicación

Como resultado de los protocolos de comunicación (Alternativa Nro 1 o Alternativa Nro 2 tratados anteriormente) se logra construir un canal seguro de comunicación, la cual enviará información de forma encriptada y autentificada. Luego, las informaciones tales como la clave secreta y el vector de inicialización (utilizados en el encriptamiento de voz) serán intercambiados cada minuto por medio de este canal.

2.2.3 La transmisión de paquetes de voz cifrados

La figura 2.4 muestra el flujo de voz encriptada que viaja sobre paquetes RTP. Una vez que se ha localizado al usuario destino con su correspondiente dirección IP, se entabla una comunicación transmitiendo la voz en forma digitalizada y segmentada en flujos de paquetes. El primer paso en este proceso es la conversión de señales de voz analógica a digitales, usando un convertidor analógico-digital. Debido a que la voz digitalizada requiere un gran número de bits, se debe usar un algoritmo de compresión para reducir el volumen de datos transmitidos. A continuación, la voz comprimida se

encripta con un algoritmo simétrico utilizando la clave de sesión y el vector de inicialización transmitida mediante un canal seguro de comunicación. Estos paquetes de voz encriptados se insertan dentro de los paquetes de datos para ser transmitidos por Internet. El protocolo utilizado para el envío de paquetes de voz es el RTP. Este es transportado como dato por UDP, el cual puede ser procesado por los nodos de redes. Por otro lado, el proceso se revierte en el lado receptor: los paquetes son desensamblados y colocados en orden correcto, los datos de voz encriptados son extraídos del paquete para ser desencriptados con la correspondiente clave secreta y vector de inicialización y luego descomprimidos; por último, la voz digitalizada es procesada por un convertidor digital-a-analógico para reproducir las señales analógicas por el parlante en el lado receptor.

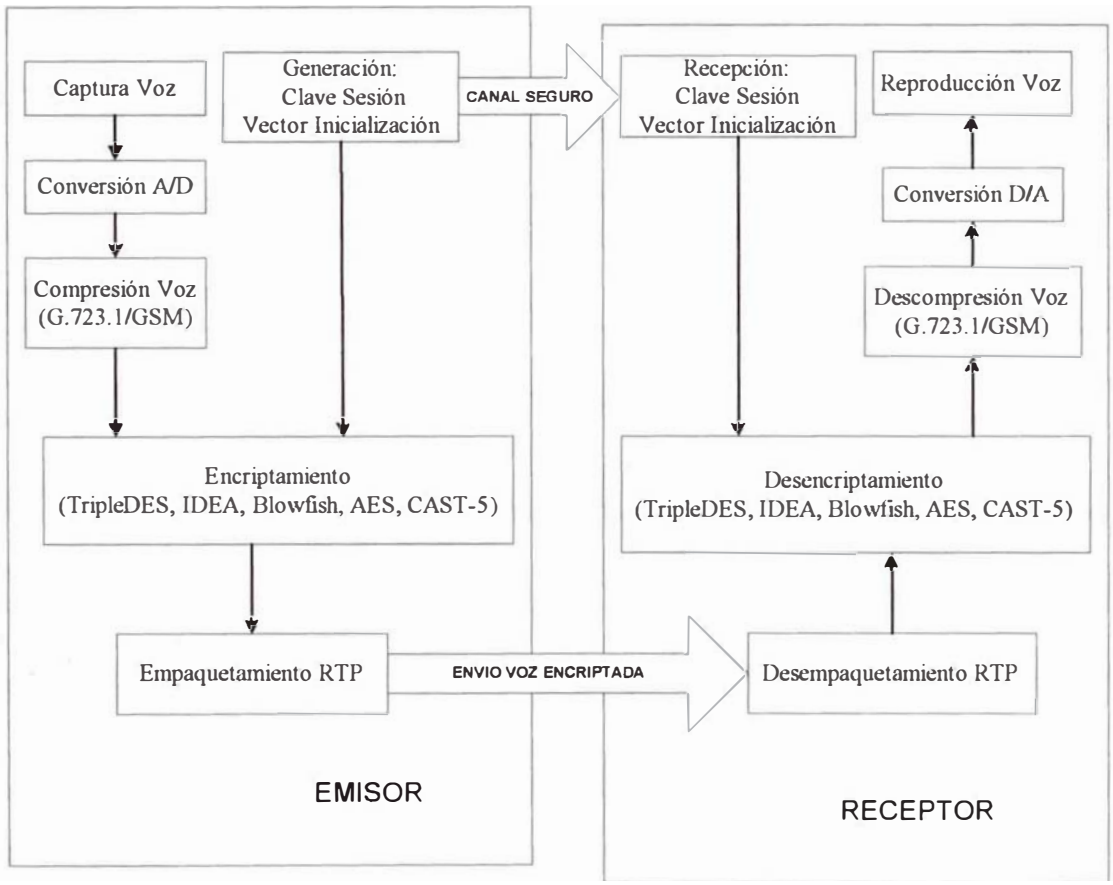

Figura 2.4: transmisión de paquetes de voz cifrados

A continuación, mostramos (tabla 2.2) los procesos para lograr el cifrado de los flujos de audio en la estación emisora y su reproducción en la estación receptora.

Procesos en la estación emisora	Procesos en la estación receptora
1 Captura de voz y conversión A/D	1 Desempaquetamiento RTP
2. Compresión de voz	2. Desencriptamiento de Voz
3. Encriptamiento de voz	3. Descompresión de voz
4. Empaquetamiento RTP	4. Conversión D/A y reproducción de voz

Tabla 2.2: procesos involucrados con el cifrado de flujos de audio en la estación emisora y receptora

2.2.3.1 Captura de voz y conversión A/D – conversión D/A y reproducción de la voz

La obtención y la reproducción de las muestras de audio se realizan a través de la tarjeta de sonido del computador. La captura de voz se realiza por el micrófono y la reproducción de voz, por los parlantes. Tanto para la obtención como para la reproducción, se debe tener un formato de tipo lineal, es decir, con una tasa de muestreo de 8000 Hz, donde cada muestra sea representada por 16 bits. Por otro lado, para la captura, se utiliza un formato estéreo (dos canales), mientras que, para la reproducción, se emplea un formato mono (un canal).

2.2.3.2 Compresión y descompresión de voz

Una vez capturadas las muestras de audio, observamos que, para su transmisión por la red, se requiere de un ancho de banda de 256 Kbps.; por lo que es preferible, aplicar un tipo de codec que comprima la voz, aunque se pierda un poco de calidad.

La compresión de voz se realiza antes del proceso de encriptamiento, por las siguientes razones:

Considerando que el objetivo del cifrado es crear datos completamente aleatorios difíciles de descifrar; debemos decir que si se comprime después de cifrar los datos de voz, no se podrá realizar una adecuada compresión de estos, puesto que no se encontrarían patrones similares.

Con la compresión, se logra velocidad de transmisión, pues reducimos el tamaño de los paquetes a encriptar.

Con la compresión, se fortalece la seguridad criptográfica del algoritmo [40]. La mayoría de las técnicas de desciframiento de códigos buscan patrones que se

encuentran en el texto plano para descifrar la clave. Con la compresión, el número de patrones en el texto sin cifrar se reduce de manera significativa, dando lugar a un grado de dificultad mucho más alto a la hora de descifrar el código.

Standard	GSM 06.10	G723.1	
Tasa Bit (Kbps)	13	5.3	6.4
Bits de Información por trama	260	160	192
Duración trama (ms)	20	30	30

Tabla 2.3: codificadores utilizados en voz sobre IP [61] Modificado.

Se utilizan los codificadores de la tabla 2.3 por las siguientes razones:

1. Estos aprovechan que el 50% de las conversaciones de voz son períodos de silencios para utilizar la detección de la actividad de la voz (Voice Activity Detection, VAD), el cual permite el envío de paquetes RTP sólo cuando se detecta la voz. Las especificaciones de estos codificadores se encuentran en:
    a. El anexo A del G723.1 [9] especifica un algoritmo de compresión de silencio.
    b. Las normas de referencia GSM 06.31, GSM 06.32 y GSM 06.12 [41] especifican funciones de transmisiones discontinuas (DTX), detección de actividad de voz (VAD) y de generación de ruido ambiental (CNG) para el algoritmo GSM 06.10.

    El uso de estos codificadores con tasas binarias intermedias y bajas, con calidad de voz aceptables y con buenos detectores de actividad, disminuyen la demora en las transmisiones y las pérdidas de paquetes en redes congestionadas, como Internet.

2. Para el tamaño del buffer (solución al problema del jitter), se tuvo en cuenta la recomendación que aparece en el estándar RFC3551. Este sugiere que el dispositivo receptor debe aceptar paquetes que representen entre 0 y 200 ms. de datos de voz. De esta forma, el jitter máximo al que se puede hacer frente utilizando estos codificadores son:
    a. 200 ms utilizando GSM 06.10, que reciben diez paquetes por cada trama.
    b. 180 ms utilizando G723.1, que reciben seis paquetes por cada trama.

2.2.3.3 Encriptamiento y desencriptamiento de voz:

Un buen esquema de encriptamiento debe usar un algoritmo lo suficientemente robusto ante los ataques de criptoanálisis. Además, una selección de claves lo suficientemente larga hace que la búsqueda de ésta sea extremadamente imposible o que tarde lo suficiente; de manera que si dan con ella, la información transmitida ya no tiene validez.

La tabla 2.4 muestran los algoritmos de cifrado en bloque con sus correspondientes longitudes de claves, vectores de inicialización y modos de funcionamiento utilizados para el desarrollo de nuestra tesis.

Algoritmo Simétrico	Longitud de la clave (bits)	Tamaño del Bloque datos	Longitud del Vector de Inic.	Modo de funcionamiento
TripleDES	128	64 bit	64 bits	OFB64
IDEA	128	64 bit	64 bits	OFB64
Blowfish	192	64 bit	64 bits	OFB64
AES	192	128 bit	128 bits	OFB128
CAST-5	128	64 bit	64 bits	OFB64

Tabla 2.4: algoritmos de cifrados de bloque para el cifrado de la voz

Características de seguridad de los algoritmos criptográficos seleccionados:

El objetivo fundamental del criptoanálisis es sin duda la recuperación de la información cifrada. En el estudio de los métodos de defensa frente al criptoanálisis [42], se supone que el criptoanalista tiene acceso a todo el criptograma. También se suele admitir que el algoritmo de cifrado es conocido. En la criptografía comercial, los usuarios potenciales de los productos criptográficos no pueden fiarse de algoritmos cuya seguridad no ha sido comprobada por la comunidad científica. Por esta razón, se han seleccionado los siguientes algoritmos criptográficos:

1. **Triple-DES**
    a. El Triple-DES utiliza tres iteraciones de la función DES [24], utilizando una longitud efectiva de clave de 112 bits, con lo que aumenta la seguridad frente a ataques por fuerza bruta.
    b. Es inmune al ataque de encuentro a la mitad [24], que se presenta cuando se aplica dos veces el algoritmo DES, con dos claves de 56 bits distintas.

2. **IDEA (International Data Encryption Algorithm)**
    a. El algoritmo IDEA fue desarrollado por el Instituto de Tecnología Federal de Suiza. El cifrado se basa en mezclar operaciones aritméticas de grupos algebraicos diferentes, con un espacio de claves de 2^{128}.
    b. Es más eficiente que los algoritmos de tipo Feistel, porque a cada vuelta se modifican todos los bits del bloque y no solo la mitad.
    c. IDEA ha resistido mucho mejor a ataques que otros métodos de cifrado como FEAL, REDOC-II, LOKI, Snefru y Khafre [43].
    d. IDEA es mucho más resistente que DES frente al altamente exitoso ataque mediante criptoanálisis diferencial de Biham y Shamir, así como ante los ataques de criptoanálisis lineal [43].
    e. Conforme este sistema de cifrado continúa atrayendo esfuerzos de ataque por parte de los integrantes del mundo criptográfico, la confianza en IDEA va creciendo con el paso del tiempo [43].

3. **Blowfish**
    a. El algoritmo Blowfish fue desarrollado por Bruce Schneier en los años 1993-94; y tiene una clave de tamaño variable desde 32 a 448 bits.
    b. Usa solo dos operaciones: adición modulo 2^{32} y XOR; Tiene 16 rondas, usa cuatro S-cajas dependiente de la clave y tiene un algoritmo de generación de subclaves bastante largo. Este último aspecto determina que la clave secreta no deban cambiarse frecuentemente. Luego, una vez calculadas estas deben ser almacenadas; lo que dificulta el ataque por fuerza bruta.
    c. El algoritmo debe tener un espacio de claves, que no sean débiles. Existe una falla ligera cuando una clave de 32 bit "A" es la misma que una de 64-bit "AA".Asimismo, se debe cuidar de no tener ninguna cadena repetida de 32 bit cuando se selecciona una clave [44].

d. El algoritmo debe ser fácilmente modificable para diferentes niveles de seguridad. Así, Blowfish lo logra fácilmente con su tamaño de clave escalable.

4. AES (Rijndael)

a. El Algoritmo AES fue desarrollado por Vincent Rijmen y Joan Daemen (Department of Electrical Engineering, K.U. Leuven) y fue seleccionado por el NIST (2/10/2000) como el nuevo estándar de cifrado para el siglo XXI y para el próximo estándar federal del gobierno de los Estados Unidos. Tanto como su tamaño de clave y de bloque de cifrado pueden ser de 128, 192 o 256 bits.

b. Las operaciones de Rijndael están entre las más sencillas de defender contra ataques por análisis temporal, criptoanálisis diferencial y lineal (siempre y cuando no se implemente una versión del algoritmo con menos de cuatro vueltas en la red interna).

c. Ha sido diseñado para proporcionar resistencia frente a las claves débiles.

5. CAST5

a. El algoritmo CAST5 es un cifrado de bloque con una clave de 128 bits. Su nombre se deriva de las iniciales de sus diseñadores, Carlisle Adams y Stafford Tavares, de la empresa Northern Telecom (Nortel) [43].

b. El algoritmo CAST no tiene claves débiles o semidébiles [43].

c. Se afirma que CAST es completamente inmune al criptoanálisis lineal y diferencial [43].

d. CAST es demasiado nuevo como para haber desarrollado un buen historial; sin embargo, su diseño formal y la buena reputación de sus diseñadores atraerán sin duda la atención y los ataques criptográficos del resto de la comunidad académica criptográfica [43].

Importancia del tamaño de las claves seleccionadas para los algoritmos simétricos

Si suponemos que el algoritmo de cifrado es conocido por el criptoanalista, existe siempre la posibilidad teórica de realizar un ataque por fuerza bruta: por eso, es necesario recorrer todos los posibles valores de la clave de descifrado hasta dar con la auténtica. La aplicación práctica de este mecanismo depende del tamaño del espacio de claves. Si bien este tipo de ataque ha servido en el pasado para descartar algunos criptosistemas (caso de DES), hoy en día no puede considerarse que un algoritmo criptográfico sea serio si admite la posibilidad de un ataque por fuerza bruta. El tamaño de las claves se calcula de forma que el conjunto de valores posibles sea imposible de explorar exhaustivamente, ni con la tecnología actual ni con la futura [42].

De entre todos nuestros algoritmos seleccionados, los algoritmos TripleDES, IDEA y CAST5 utilizan una clave de 128 bits; mientras que Blowfish y AES utilizan una clave de 192 bits. De esta forma, se dificulta el ataque por fuerza bruta.

Para el cifrado de la voz se ha seleccionado el modo de funcionamiento OFB (Output Feedback)

1. El OFB constituye un ejemplo de cifrado en flujo [14]; y se utiliza para encriptar un flujo de bits. Cualquier número de estos se puede encriptar al mismo tiempo haciendo un XOR de los bits del texto plano con la salida del registro de desplazamiento.
2. En el modo de OFB, un solo error en el texto cifrado implica un error de bits en el texto plano recuperado [14]. Esto supone una ventaja en las emisiones de audio, donde los errores tienen efectos mínimos sobre el flujo de medio descrito.
3. El OFB es vulnerable a la periodicidad en la salida generada del registro de desplazamiento [14]. Un registro de 64 bits puede variar de 2^{32} bits a 2^{64}-1, dependiendo del tamaño de la información (salida del registro de desplazamiento que vuelve a sí misma). La solución a esta vulnerabilidad consiste en usar el mismo tamaño de información y del registro de desplazamiento (64 bits).

En el sistema propuesto, se usa el modo OFB64 para los algoritmos TripleDES, Blowfish, IDEA y CAST5; y el modo OFB128 para AES. Se ha seleccionado el modo OFB64 (trabaja con 64 bits) y el modo OFB128 (trabaja con 128 bits) porque estos dos modos representan el mismo número de tamaño de bloque de entrada de datos. Con esto, se evita la vulnerabilidad a la periodicidad en la salida generada del registro de desplazamiento.

2.2.3.4 Empaquetamiento y desempaquetamiento RTP

Nuestro sistema utiliza el protocolo RTP porque proporciona solución al problema de la transmisión y recepción de flujos de datos multimedia sobre una red de propósito general asíncrona. El RTP presenta las siguientes características:

1. Incluye sellos de tiempo en los paquetes enviados para poder reproducirlos en el orden correcto, lo que permite al nivel de aplicación realizar la sincronización de varios flujos de datos como, por ejemplo, audio y video [45].
2. Añade un número de secuencia a los paquetes para poder reensamblarlos en el mismo orden en el que fueron enviados. En una sesión multimedia, cada flujo de datos será enviado en su propia sesión RTP, realizando la sincronización de ambos en el nivel de aplicación [45].

El RTP trabaja conjuntamente con el protocolo RTCP (RTP Control Protocol) por las siguientes razones:

1. Para monitorear la calidad de servicio percibido por el receptor y almacenar la información de los usuarios de las distintas sesiones que son establecidas [45].
2. Para ofrecer la identificación del origen de la comunicación y para resincronizarlo en una sesión RTP ya existente [45].

Características de un paquete RTP

Una vez que la sesión RTP se ha logrado, cada usuario podrá transmitir bloques de información de audio con una duración pre-establecida en el campo de carga útil del paquete RTP, el cual será encapsulado en datagramas UDP y posteriormente en paquetes IP (figura 2.5).

Cabecera I P 20 Oct.	Cabecera UDP 8 Oct.	Cabecera RTP 12 Oct.	Carga útil RTP

Figura 2.5: transporte de paquetes rtp en redes IP. [46]

Un paquete RTP esta formado de un encabezado y de una carga útil. El primero contiene informaciones tales como el tipo de carga útil, el número de secuencia, los sellos de tiempo, etc. La segunda contiene la voz digitalizada que ha sido comprimida y encriptada. El propósito principal en las comunicaciones seguras de voz es lograr la confidencialidad y la privacidad. Esto se consigue con el encriptamiento de la carga útil (payload), que hace uso de la clave secreta y el vector de inicialización que son compartidos a través del canal seguro de comunicación. (figura 2.6).

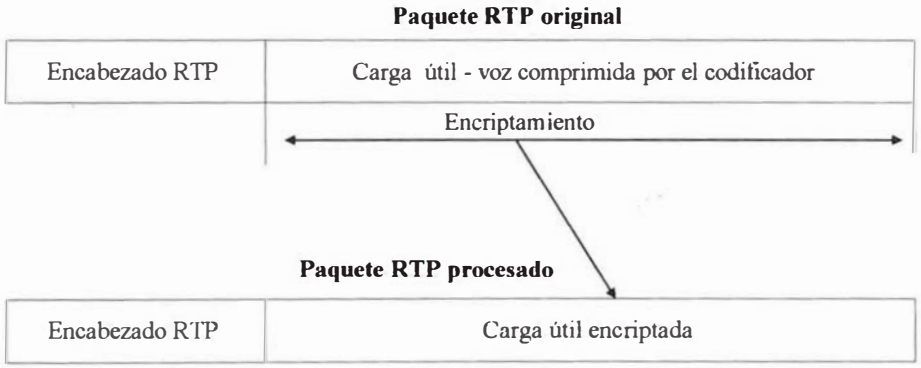

Figura 2.6: paquete RTP antes y después del procesamiento del emisor. [47] Modificado

2.2.3.5 Uso del UDP en la transmisión de voz

Si una aplicación quiere transmitir datos, usa uno de los protocolos ubicados en la capa de transporte, TCP o UDP ubicados dentro de la arquitectura TCP/IP.

Desventajas en el uso de TCP para las transmisiones de voz

1. TCP ofrece un servicio de control de flujo, basándose en la retransmisión de paquetes perdidos o corruptos, es decir, un servicio confiable que mantiene el orden de lo enviado. Si algún paquete se pierde o se corrompe, se produce un tiempo de espera en el receptor para la retransmisión de estos paquetes. Esto adiciona retardo extra a la comunicación [25].

2. El control de flujo y congestión son muy útiles, pero una aplicación tiene poco control sobre estas características. TCP puede decidir por si mismo si decrementa la tasa de envío de datos, y esto también incrementará el retardo total [25].

Como podemos observar, TCP tiene muchas características buenas, pero bastante complejidad que no es muy útil en las aplicaciones de voz. Así, se puede obtener un rendimiento mejor o igual usando protocolos menos elaborados como el UDP para no presentar complejidad alguna. Las ventajas que se han encontrado en el UDP para el sistema propuesto son las siguientes:

1. UDP es simplemente una extensión del protocolo IP, de manera que sólo ofrece un servicio de best-effort [25].
2. En UDP, no existe la retransmisión de paquetes [25]
3. UDP no provee mecanismos para sincronización ni para control de la congestión y de flujos de datos multimedia [25].

Como vemos el UDP es apropiado para las transmisiones de aplicaciones multimedia debido a su sencillez y rapidez en el envío de información. Sin embargo, las características faltantes de éste son suplidas con los protocolos RTP y RTCP.

2.3 Características de seguridad del sistema

Los mecanismos de implementación de seguridad usados en el sistema propuesto, se han distribuido en dos partes: la primera involucra el intercambio de las claves de sesión entre ambos usuarios y la segunda consiste en la transmisión de paquetes de voz cifrados.

1. **Los algoritmos criptográficos utilizados en el intercambio de las claves de sesión entre ambos usuarios.**

En esta parte, se presentan dos alternativas de solución para el protocolo de comunicación:

a. Alternativa N° 1: protocolo de comunicación utilizando el algoritmo asimétrico RSA.

La negociación entre los usuarios para la selección de las claves de sesión (clave secreta y vector de inicialización para el algoritmo AES, y el código de autentificación para el algoritmo HMAC-SHA1) se realiza utilizando:

- El cifrado asimétrico RSA para el intercambio de mensajes.
- El algoritmo MD5withRSA para las firmas digitales, en él se utiliza el algoritmo MD5 como función resumen. La firma digital se realiza con la clave privada correspondiente al certificado digital y su autentificación es hecha por la parte contraria mediante el uso de la clave pública del certificado.

La construcción del canal seguro se realiza utilizando:

- El cifrado simétrico AES
- La función MAC llamada HMACSHA1

b. Alternativa N° 2: utilizando el protocolo SSL.

La negociación entre los usuarios para la selección de las claves de sesión se realiza utilizando:

- El cifrado asimétrico acordado entre ambas partes

- La firma digital y su autentificación. La firma se realiza con la clave privada correspondiente al certificado digital, y su autentificación es hecha por la parte contraria mediante la clave pública del certificado. En el sistema propuesto, ambas partes realizan intercambio de certificados, y autentificación mutua.

La construcción del canal seguro, se realiza con:

- El cifrado simétrico acordado entre ambas partes.
- La función HMAC acordada entre ambas

2. Los algoritmos utilizados en la transmisión de paquetes de voz cifrada.

Los algoritmos simétricos utilizados para el cifrado de la voz depende de la selección del usuario. Entre ellos, tenemos a los algoritmos TripleDES, IDEA, Blowfish, AES, y el CAST5.

La tabla 2.5 muestra la relación entre los mecanismos de implementación criptográfica, con los servicios de seguridad.

	Confidencialidad	**Autentificación**	**Integridad**	**Control de Acceso**	**No repudio**
Acuerdo entre los participantes para la selección de la clave de sesión (RSA o SSL)					
Cifrado asimétrico	X	X	X		
Firma		X	X		X
Función Hash	X		X		
Construcción del canal seguro					
Cifrado simétrico	X				
Función HMAC	X	X	X		
Transmisión de paquetes RTP cifrados					
Algoritmo simétrico	X				

Tabla 2.5: relación entre los mecanismos de implementación criptográfica con los servicios de seguridad.

Capítulo 3

Implementación del sistema

3.1 Descripción del sistema

El objetivo de la presente tesis consiste en desarrollar un sistema de comunicación de voz entre dos usuarios con calidad aceptable y dotada de servicios de seguridad, de forma que no tengan que preocuparse de cualquier ataque que puedan sufrir en Internet o dentro de su propia red. La implementación de este sistema permite la distribución de audio comprimido y encriptado en una sesión RTP compartida entre dos usuarios del sistema a través de una red IP. De este modo, para nuestra tesis, hemos empleado el código Java con el fin de garantizar la portabilidad entre distintas plataformas de desarrollo, la simplicidad del mantenimiento y la facilidad para las aplicaciones multimedia.

Los requisitos para la transmisión en tiempo real hacen que la señal codificada y encriptada se vaya fragmentando en paquetes RTP, las cuales se envían a la red encapsulada en paquetes UDP y, posteriormente, en datagramas IP. El API de java Media Framework (JMF) nos proporciona todas las clases y métodos necesarios para el establecimiento de sesiones RTP, así como la posibilidad de realizar modificaciones de acuerdo a los requerimientos de las aplicaciones.

La incorporación de servicios de seguridad implica cifrar los fragmentos de la señal codificada (G723.1 o GSM) antes de su incorporación al paquete RTP. Para esto, brindamos al usuario la posibilidad de elegir entre cinco alternativas de algoritmos simétricos empeados para el cifrado de la voz: TripleDES, IDEA, Blowfish, AES, y CAST5 (proporcionados por la librería de seguridad criptográfica BouncyCastle). Cualquiera de estos es incorporado como un nuevo codec dentro de la infraestructura JMF y, luego, se utiliza un protocolo de comunicación para la distribución de la clave secreta y el vector de inicialización. Por otro lado, es conveniente indicar que uno de estos protocolos ha sido diseñado en base al algoritmo asimétrico RSA mientras que el

otro es el protocolo SSL (proporcionado por el API Java Secure Socket Extensión JSSE).

3.2 Diseño del aplicativo del sistema

El sistema propuesto ha sido desarrollado en Java y consta de dos partes (figura 3.1):

1. La parte usuaria : usuario 1, usuario 2 , ….., usuario n
2. La parte del servidor de directorio.

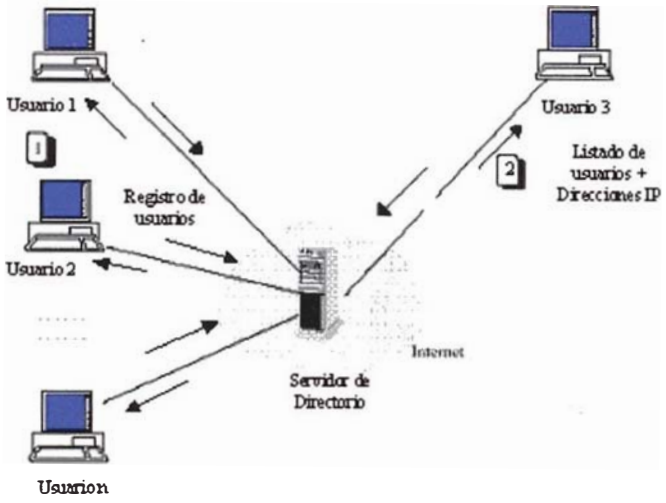

Figura 3.1: las partes del sistema propuesto desarrollado en Java [39] Modificado.

La parte usuaria se instala en las computadoras de los que las emplean. Para acceder al sistema, los usuarios deben primero registrarse en el servidor de directorio proporcionando sus datos personales, dirección IP y clave pública generada. De la misma forma, el servidor de directorio le retornará el listado de los usuarios activos con sus direcciones IP y claves públicas; luego, se selecciona al usuario con el que se desea comunicar; posteriormente, se inicia la negociación de las claves de sesión; y, finalmente, se entabla una comunicación segura.

De este modo, el servidor de directorio se encontrará activo todo el tiempo, de manera que los usuarios puedan registrarse y obtener la lista de usuarios en línea con sus correspondientes direcciones IP y claves públicas.

3.3 Clases utilizadas en el aplicativo del sistema

Las clases han sido agrupadas respetando la estructuración de nuestro sistema en tres partes (sección 2.2).

3.3.1 El registro de los usuarios en un servidor de directorio

1. ***Clase RegisterList***. Esta clase corresponde a la parte usuaria del sistema. Con esta se puede realizar:
   - El registro de los usuarios en el servidor de directorio (Clase Server).
   - La selección del algoritmo de encriptamiento que se empleará para la comunicación, pudiendo ser los algoritmos simétricos TripleDES, IDEA, Blowfish, AES, y el CAST5. Si no se desea el cifrado de la comunicación, se selecciona la opción NINGUNA. Esto con el fin de realizar la comunicación sin criptografía.
   - La selección del protocolo de comunicación para el intercambio de claves: RSA, SSL. Si no se desea el cifrado de la comunicación, se selecciona la opción NINGUNA.
   - El inicio de la transmisión y la recepción de los paquetes de voz cifrados

2. ***Clase Server.*** Cumple la función de servidor de directorio. Aquí, los usuarios se registran al ingresar y al dejar el sistema. Es decir, el usuario se registra con la clase RegisterList y automáticamente se conecta con la clase Server, quien lleva el registro de todos los usuarios; al aceptar la conexión, la clase Server crea un hilo ServerThread que mantiene la comunicación con el usuario hasta el final, mientras espera por más conexiones de usuarios.

3. ***Clase ServerThread.*** Es un hilo creado por la clase Server para que mantenga comunicación con el usuario y, al mismo tiempo, lo mantenga informado de los usuarios que se han conectado o abandonado el sistema.

4. ***Clase usuarios.*** Contiene la estructura de los datos que el usuario debe de proporcionar a la clase Server. Los atributos de esta clase son: el nombre, la dirección IP, el e-mail, y la clave pública del usuario.

3.3.2 El intercambio de las claves de sesión entre ambos usuarios.

3.3.2.1 Alternativa N° 1: utilizando el algoritmo asimétrico RSA

1. ***Clase SessionConstants***: Incluye las constantes a utilizar entre la clases RSAClient y RSAServerThread, las cuales indican en qué paso se encuentran el intercambio de las claves de sesión.

2. ***Clase Keytable:*** Almacena el par de claves generadas (para el intercambio de mensajes utilizando el algoritmo RSA), el almacén de claves y el certificado (para las firmas digitales), los cuales son utilizados en la negociación de las claves de sesión entre ambos usuarios. Hemos considerado lo siguiente:

   - Para la generación del par de claves se utiliza:
     - Algoritmo asimétrico : RSA
     - Longitud de la clave : 1024
     - El proveedor de la librería criptográfica : Bouncy Castle

   - Para el almacén de claves se hace uso de:
     - Tipo de almacén de claves : JKS
     - Proveedor : SUN

3. ***Clase ExDataInputtStream***: Recoge los datos por medio del canal seguro de comunicaciones. Por este medio, se recepciona la clave secreta y el vector de inicialización empleado en el cifrado de la voz. Se configura de la siguiente forma:

   - Para el desencriptamiento de los datos, se utiliza :
     - El algoritmo simétrico : AES
     - El modo de funcionamiento del algoritmo simétrico : CBC
     - El Padding (Relleno) para el bloque de datos : PKCS7Padding
     - El proveedor de la librería criptográfica : Bouncy Castle

- Para validar los datos enviados por el usuario, se emplea:
    - El algoritmo MAC : HMACSHA1
    - El proveedor de la librería criptográfica : Bouncy Castle

4. ***Clase ExDataOutputStream.*** Se utiliza para la salida de datos por medio del canal seguro de comunicaciones. Por este medio, se transmite la clave secreta y el vector de inicialización empleado en el cifrado de la voz. Se configura de la siguiente forma:
    - Para el encriptamiento de los datos, se emplea :
        - El algoritmo simétrico : AES
        - El modo de funcionamiento del algoritmo simétrico : CBC
        - El Padding(Relleno) para el bloque de datos : PKCS7Padding
        - El proveedor de la librería criptográfica : Bouncy Castle

    - Para la generación de un código de autentificación de mensaje, se utiliza :
        - El algoritmo MAC : HMACSHA1
        - El proveedor de la librería criptográfica : Bouncy Castle

5. ***Clase RSAClient***: Crea un socket de conexión con el RSAServer utilizando la dirección IP del otro usuario y el puerto 1887. La RSAServer acepta la conexión y crea un hilo llamado RSAServerThread, el cual se encarga de manejar la conexión con la clase RSAClient. Esta entablará una comunicación con el hilo RSAServerThread para negociar la configuración del canal de comunicación; luego, se verifica la autentificación de los usuarios antes de proceder al envío de las claves de sesión. El canal de comunicación construido consta de un algoritmo simétrico AES y un MAC llamado HMACSHA1. Dentro de esta negociación, se utiliza la misma clave secreta y vector de inicialización en el cifrado de la data que viaja por este canal. Así también, se acuerda el código de autentificación a utilizarse con el HMACSHA1. Luego de la configuración del canal de comunicación, cada minuto RSAClient genera la clave secreta y el vector de inicialización de acuerdo al algoritmo de encriptamiento seleccionado por el usuario y se lo envía al hilo RSAServerThread.

6. ***Clase RSAServer:*** Crea un socket llamado ServerSocket y espera una conexión de un cliente representado por RSAClient, a través del puerto 1887. Después de haber aceptado la conexión, esta clase crea un hilo llamado RSAServerThread, el cual se encarga de mantener la comunicación con RSAClient.

7. ***Clase RSAServerThread:*** Es un hilo creado por la clase RSAServer para que mantenga la comunicación con RSAClient aún después de la configuración del canal de comunicación seguro. Esta clase recepciona cada minuto la clave secreta y el vector de inicialización (enviado por RSAClient), ya que lo utilizan en el cifrado de la voz.

A continuación, mostramos (figura 3.2) la comunicación existente entre: RSAClient y: RSAServerthread para la construcción del canal seguro de comunicación.

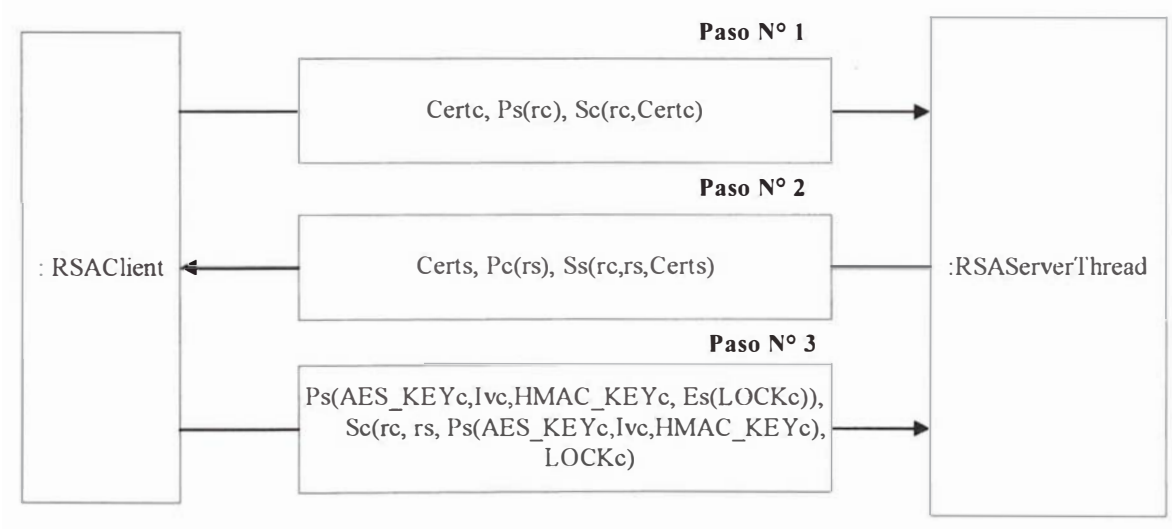

Figura 3.2: construcción del canal de comunicación utilizando el algoritmo RSA.

Seguidamente, detallamos los algoritmos utilizados dentro de esta negociación del canal de comunicación:

- La clase RSAServerThread representa al usuario S.
- La clase RSAClient representa al usuario C.

En la negociación entre los usuarios para ***la selección de las claves de sesión***, se considera lo siguiente:

a) Para la generación de números aleatorios de 32 bytes (rc y rs), se emplea :
   i) El algoritmo de generación de números pseudo-aleatorios : SHA1PRNG
   ii) El proveedor de la librería criptográfica : SUN

b) Para la generación de los certificados (Certc y Certs), se ha utilizado la herramienta ***keytool***, con una clave de longitud de 1024 bits y del tipo de algoritmo asimétrico RSA.

c) Para la firma y su verificación, se utiliza:
   i) El algoritmo MAC : MD5withRSA
   ii) El proveedor de la librería criptográfica : Bouncy Castle

d) Para el encriptamiento con clave pública, se emplea:
   i) El algoritmo asimétrico : RSA
   ii) El modo de funcionamiento del algoritmo asimétrico : None
   iii) El padding para el bloque de datos : OAEPPadding
   iv) El proveedor de la librería criptográfica : Bouncy Castle

e) Para :
   i) la clave privada del algoritmo AES del canal seguro de comunicación, se usa:
      (1) El proveedor de la librería criptográfica : Bouncy Castle
      (2) La longitud de la clave : 192 bits.

   ii) el vector de inicialización del algoritmo AES en el canal seguro de comunicación, se hace uso de:
      (1) El algoritmo de generación de números pseudoaleatorios : SHA1PRNG
      (2) El proveedor de la librería criptográfica : SUN
      (3) La longitud del vector : 16 bytes

   iii) la clave del algoritmo HMAC-SHA1 del canal seguro de comunicación, se emplea:
      (1) El proveedor de la librería criptográfica : BouncyCaslte
      (2) La longitud de la clave :160 bits

iv) la variable LOCK, se utiliza:

(1) El algoritmo de generación de números pseudo-aleatorios : SHA1PRNG

(2) El proveedor de la librería criptográfica : SUN

(3) La longitud de la variable : 24 bytes

f) el cifrado de la variable LOCK, se usa:

i) El algoritmo simétrico : AES

ii) El modo de funcionamiento del algoritmo simétrico : CBC

iii) El padding para el bloque de datos : PKCS7Padding

iv) El proveedor de la librería criptográfica : SunJCE

En la ***construcción del canal seguro***, se considera lo siguiente:

a) El cifrado de los datos encriptados utiliza:

i) El algoritmo simétrico : AES

ii) El modo de funcionamiento del algoritmo simétrico : CBC

iii) El padding para el bloque de datos : PKCS7Padding

iv) El proveedor de la librería criptográfica : Bouncy Castle

b) Para la verificación de la integridad de los datos, se utiliza :

i) El algoritmo MAC : HMACSHA1

ii) El proveedor de la librería criptográfica : Bouncy Castle

La tabla 3.1 resume las características tomadas en cuenta en la generación de las claves privadas y los vectores de inicialización de los diferentes algoritmos simétricos.

Algoritmo Simétrico	**Clave privada**		**Vector de inicialización**		
	Longitud de clave	**Proveedor de la librería criptográfica**	**Longitud del vector**	**Proveedor de la librería criptográfica**	**Algoritmo de generación de números pseudo aleatorios**
TripleDES	128	BouncyCastle	8 bytes	SUN	SHA1PRNG
IDEA	128	BouncyCastle	8 bytes	SUN	SHA1PRNG
Blowfish	192	BouncyCastle	8 bytes	SUN	SHA1PRNG
AES	192	BouncyCastle	16 bytes	SUN	SHA1PRNG
CAST-5	128	BouncyCastle	8 bytes	SUN	SHA1PRNG

Tabla 3.1: características en la generación de las claves privadas y vectores de inicialización de los diferentes algoritmos criptográficos.

3.3.2.2 Alternativa N° 2: utilizando SSL

1. ***Clase SSLClient:*** Crea un socket de conexión llamado SSLSocket utilizando la dirección IP del otro usuario y el puerto 2001. Cuando el SSLServer aceptar la conexión de SSLClient, inicia la negociación de las claves de sesión para la configuración del canal de comunicación; luego, crea un hilo llamado SSLServerThread, que se encarga de manejar la comunicación con SSLClient. Cada minuto, RSAClient se encarga de generar la clave secreta y el vector de inicialización de acuerdo al algoritmo de encriptamiento seleccionado por el usuario, y, luego, se lo envía al hilo SSLServerThread.

2. ***Clase SSLServer:*** Crea un socket de conexión llamado SSLServerSocket y espera por el puerto 2001; mediante una conexión de un cliente representado por SSLClient. Una vez aceptada la conexión y la negociación de las claves de sesión para la configuración de canal de comunicación, esta clase crea un hilo SSLServerThread para que mantenga la comunicación con SSLClient.

3. ***Clase SSLServerThread:*** Es un hilo creado por SSLServer para que mantenga la comunicación con SSLClient después de la configuración del canal seguro de comunicación. Cada minuto, esta clase espera por la clave secreta y el vector de inicialización enviado por SSLClient con la finalidad de utilizarlo en el cifrado de la voz.

Las características consideradas para la generación de las claves privadas y los vectores de inicialización de los diferentes algoritmos simétricos son los mismos que para RSAClient (tabla 3.1).

3.3.3 La transmisión y recepción de los paquetes de voz cifrados

1. ***Clase target:*** Almacena las direcciones IP, puertos orígenes y destinos tanto para la transmisión como para la recepción. Luego, esta información pasa a las clases AVTransmiter y AvReceiver. Los puertos han sido configurados de la siguiente manera

   Se transmite por el puerto 42050 y

   Se recibe por el 42052

2. ***Clase DeviceInfo:*** Es utilizada por AVTransmiter para comparar entre diferentes formatos de audio y convertir de un formato de audio a una clase String.

3. ***Clase AVTransmiter:*** Se encarga de la captura de voz, la compresión, el encriptamiento y el envío de paquetes RTP por la red. Los procesos involucrados en esta clase, se detallan a continuación:

   - ***La Inicialización de variables.*** Coge las direcciones IP, puertos origen y destino de la clase Target para utilizarlos en AVTransmiter.

   - ***El registro del codec SecureGsmPacketizer.*** Se registra en el sistema mediante la clase PlugInManager (JMF puede adicionar nuevos codecs). Este codec se encarga de realizar el encriptamiento de los datos de voz.

   - ***La captura de las muestras de audio.*** Se realiza con el dispositivo llamado DirectSoundCapture utilizando un formato lineal, de 8000.0 Hz, de 16-bits y estéreo.

   - **La creación del origen de datos (DataSource) para el procesador (processor).** Se considera al dispositivo de captura de audio como originador de las señales de entrada para el procesador.

   - ***La creación del procesador.*** Una vez obtenidas las muestras de voz, deben tratárselas para poder transmitir por la red existente. Así, se aplica un codec a las muestras de voz, el cual las comprime. Los codificadores a emplear son el G723.1 (con un tasa de 5.3 y 6.4 Kbps) y el GSM (con una tasa de 13 Kbps), los cuales están implementados en JMF. Debemos notar que es el procesador, el encargado de aplicar cualquiera de estos dos codificadores (el que sea seleccionado por el usuario); luego, la voz comprimida pasa a SecureGsmPacketizer para ser encriptada; y, por último, se crea una sesión RTP para el envío de esta.

- ***La creación del origen de datos para la administración de la sesión RTP (RTPSessionManager).*** Los datos de salida del procesador se consideran como entrada para la sesión RTP.

- ***La creación de un administrador de la sesión RTP (RTPSessionManager).*** Este administrador mantiene el registro de los usuarios y el flujo de datos que entre ellos se transmite. El administrador de RTP realiza el control de las sesiones junto con RTCP. El primero entrega estadísticas de todos los paquetes RTP y RTCP enviados y transmitidos en una sesión. El administrador de RTP realiza:

    - El registro dinámico de un nuevo payload RTP, colocando como ID el número 101 que indica el cifrado de la voz. Se coloca este número ya que, según el RFC3551, el rango comprendido entre 96 hasta 127 está designado para el payload dinámico.

    - La asignación de puertos a utilizar para el envío de la voz a través del protocolo RTP. Se configura como el puerto origen de transmisión al 42050 y como puerto destino al 42052. Por consiguiente, los puertos RTCP en el origen será el 42051 y en el destino será el 42053.

    - El envío de los paquetes por la red a través del objeto SendStream. Luego de haber creado e inicializado el administrador de sesión RTP, se crea el objeto SendStream que pasará como DataSource al RTPSessionManager.

4. ***Clase SecureGsmPacketizer:*** Es una implementación de la interface "Codec". Esta recibe el audio comprimido (formato G723.1 ó GSM) enviado por el procesador de AVTransmiter; y, luego, lo encripta y devuelve a la misma clase para su transmisión. Para el encriptamiento, se necesita de la clave secreta y el vector de inicialización, los cuales son tomados de RegisterList. Debemos decir, que estos elementos fueron colocados allí por las clases que participan en el

intercambio de las claves de sesión entre los usuarios (RSAClient, RSAServerThread, SSLClient, SSLServerThread).

Seguidamente, mostraremos (tabla 3.2) los algoritmos simétricos, involucrados en el cifrado de la voz:

Algoritmo simétrico	Modo funcionamiento	Padding	Provider
TripleDES	OFB64	NoPadding	BouncyCastle
IDEA	OFB64	NoPadding	BouncyCastle
Blowfish	OFB64	NoPadding	BouncyCastle
AES	OFB128	NoPadding	BouncyCastle
CAST-5	OFB64	NoPadding	BouncyCastle

Tabla 3.2: configuración de los algoritmos criptográficos.

5. ***Clase AVReceiver.*** Se encarga de la recepción de los paquetes RTP por la red, el desencriptamiento, la descompresión, y la reproducción de la voz. Los procesos involucrados en esta clase se detallan a continuación:

   - ***La Inicialización de variables***. Coge las direcciones IP, puertos origen y destino de la clase Target para utilizarlos en AVReceiver.

   - ***El registro del codec SecureGsmDepacketizer***. Se registra en el sistema con la clase PlugInManager (facilidad que JMF para poder adicionar nuevos codecs). Este codec se encarga de realizar el desencriptamiento de los datos de voz.

   - ***La creación de un administrador de la sesión RTP.*** Este administrador mantiene el registro de los usuarios y el flujo de datos que entre ellos se transmite. El administrador de RTP realiza el control de las sesiones junto con RTCP. El primero entrega estadísticas de todos los paquetes RTP y RTCP enviados y transmitidos en una sesión. El administrador de RTP realiza:

     - El registro dinámico de un nuevo payload RTP, colocando como ID el número 101 que indica el cifrado de la voz. Se coloca este número ya

que, según el RFC3551, el rango comprendido entre 96 hasta 127 está designado para el payload dinámico.

- La asignación de puertos a utilizar para el envío de la voz a través del protocolo RTP. Se configura como el puerto origen de transmisión al 42050 y como puerto destino al 42052. Por consiguiente, los puertos RTCP en el origen será el 42051 y en el destino será el 42053.

- El registro del ReceiveStreamListener para recibir la notificación de las llegadas de nuevos paquetes RTP.

- ***La creación del origen de datos para el procesador.*** El procesador toma como entrada de datos los flujos de audio recuperados por el administrador de la sesión RTP.

- ***Creación del processor***: Se realiza lo mismo que en AVTransmiter pero en sentido contrario: los datos recogidos del SessionManager se envían al codec SecureGsmDepacketizer para que los desencripte; luego, se pasa por el codificador de voz (G723.1 o GSM); por último, se reproduce la voz en los parlantes. La reproducción de la voz se representa mediante una pequeña ventana en el lado receptor.

6. ***Clase SecureGsmDepacketizer***: Es una implementación de la interfase "Codec". Esta clase recibe el audio encriptado de AVReceiver, lo desencripta y lo vuelve a esta misma clase para que continúe con el proceso de descompresión y posterior reproducción de la voz. Para el desencriptamiento, se necesita de la clave secreta y el vector de inicialización, los cuales son tomados de RegisterList, que fueron colocados allí por las clases que participan en el intercambio de las claves de sesión entre los usuarios (RSAClient, RSAServerThread, SSLClient, SSLServerThread).

3.4 Herramientas utilizadas

3.4.1 JAVA

Por sus características y su adecuación a nuestros objetivos, *Java* ha sido el lenguaje seleccionado para el "Desarrollo del Sistema Seguro de Transmisión de Voz mediante Internet".

Java es un lenguaje de propósito general orientado a objetos. Fue concebido por *James Gosling, Patrick Naughton, Chris Warth, Ed Frank* y *Mike Sheridan en Sun Microsystems Inc.* en 1991 [26]. Este lenguaje se llamó inicialmente "*Oak*", y se cambio al nombre de "*Java*" en 1995 y surgió como un lenguaje que fuese independiente de la plataforma, es decir, con arquitectura neutral, que se pudiese utilizar para crear software para diversos dispositivos electrónicos. Esto solucionaría el problema de tener que disponer de un compilador del lenguaje para cada tipo de CPU, puesto que el desarrollo de los compiladores es caro y requiere un tiempo de desarrollo elevado. Luego, mientras se ultimaban los detalles de Java, surgió la World Wide Web. Así, debido a que ésta precisaba de un lenguaje portable, Java se vio potenciado a convertirse en uno de los lenguajes de programación más destacados.

Sus principales características, resumidas por el equipo del diseño original, son:

- Simple
- Seguro
- Portable
- Orientado a objetos
- Robusto
- Multihilo
- Arquitectura neutral
- Interpretado
- Alto rendimiento
- Distribuido
- Dinámico

La portabilidad de Java se debe al hecho de ser un lenguaje interpretado y no compilado. El compilador de Java genera código binario (*bytecode*) y no ejecutable como sería lo habitual. Este código binario es un conjunto de instrucciones altamente optimizadas, diseñadas para ser ejecutadas por una máquina virtual que emula el intérprete de Java. De este modo, no es preciso tener un compilador para cada plataforma, sino solo se necesita implementar el intérprete de Java. Una vez disponible el programa de ejecución para un sistema dado, cualquier programa Java podrá ejecutarse en esa plataforma.

3.4.2 El API Java Media Framework (JMF)

El Java Media Framework (JMF) es un API que permite el desarrollo de aplicaciones multimedia (audio y video) en Java. Asimismo, el API JMF 1.0 permite la reproducción de ficheros multimedia. Por otro lado, el API JMF 2.0 extiende la arquitectura para soportar la captura y almacenamiento de datos multimedia, control del tipo de procesamiento durante la reproducción, y procesamiento personalizado en sus datos.

El JMF 2.0 [2] permite:

- La reproducción de ficheros multimedia (formatos, AVI, MPEG, QuickTime, AVI, WAV, MIDI…)
- La reproducción y transmisión de audio y video en tiempo real a través de Internet (utilizando el protocolo RTP/RTCP)
- La captura de audio y video del hardware disponible y el almacenamiento en el formato deseado
- La difusión de señales de audio/video a un grupo de usuarios
- La implementación de soluciones propias basadas en los APIs existentes y la integración de nuevas características con la estructura pre-establecida.
- El acceso a datos multimedia para su procesamiento
- El desarrollo de APIs personalizados: multiplexores, demultiplexores, codecs, procesadores de efecto y reproductores de datos multimedia (JMF plug-Ins)
- La compatibilidad con JMF 1.0

Arquitectura de alto nivel

Los dispositivos como las cintas de video y los VHS proveen un modelo para grabar, procesar y reproducir los datos multimedia. JMF utiliza el mismo modelo básico [2]:

1. Un ***DataSource*** representa a la cinta de video.
2. Un ***Player*** o ***Processor*** representa al VHS.
3. Los ***dispositivos de entrada*** y ***salida***, son representados por los monitores y los parlantes.

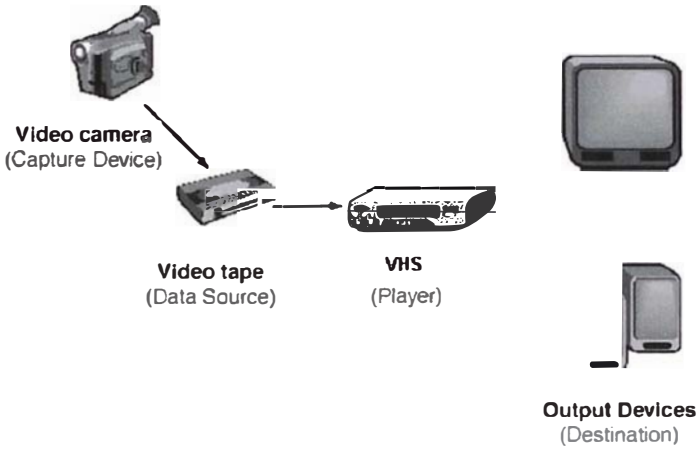

Figura 3.3: captura, almacenamiento, procesamiento y reproducción de datos multimedia [2].

Los DataSources y los Players son partes integrales de los API de alto nivel del JMF. Así, estos manejan la captura, la presentación y el procesamiento de datos en tiempo real. JMF también provee de APIs de bajo nivel que permiten la integración de componentes de procesamiento personalizados.

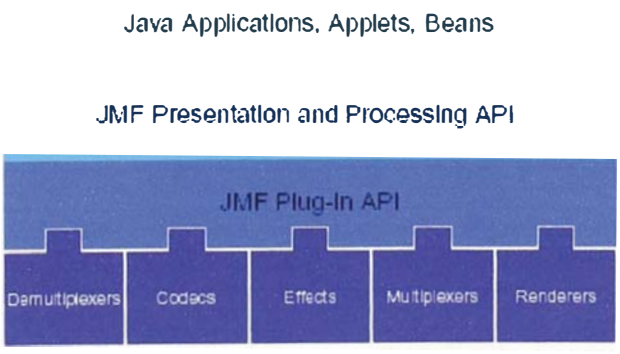

Figura 3.4: arquitectura de alto nivel de JMF [2].

3.4.3 Java Cryptography Extensión (JCE)

El JCE provee una infraestructura e implementación para el encriptamiento, la generación y el acuerdo de claves, y para los algoritmos de autentificación de mensajes (MAC). El soporte para el encriptamiento incluye cifrados simétricos, asimétricos, de bloque y de flujo [5].

Anteriormente, el JCE era un paquete opcional (de extensión) al Java ™ 2 SDK, Standard Edition (Java 2 SDK), versiones 1.2.x y 1.3.x; ahora, este ha sido integrado en Java 2 SDK, v 1.4.

El JCE está basado en los mismos principios de diseño encontrados en el JCA (Java Cryptography Architecture): la independencia de implementación y de algoritmo. Asimismo, utiliza la arquitectura "provider". Por otro lado, los proveedores que sean firmados por una entidad confiable pueden ser insertados en la infraestructura JCE, y algoritmos nuevos pueden ser fácilmente adicionados.

El JCE abarca:

- El encriptamiento simétrico como DES, RC2, e IDEA.
- El encriptamiento de flujo simétrico como RC4.
- El encriptamiento asimétrico como RSA.
- El encriptamiento-en-base-a-password (PBE).
- El acuerdo de claves.
- El código de autentificación de mensajes (MAC)

El Java 2 SDK, v 1.4 se entrega con un proveedor JCE llamado "SunJCE", que se encuentra pre-instalado y registrado; asimimo, provee los siguientes servicios criptográficos [5]:

- Una implementación de los algoritmos DES (FIPS PUB 46-1), TripleDES, y Blowfish en Electronic Code Book (ECB), Cipher Block Chaining (CBC), Cipher Feedback (CFB), Output Feedback (OFB), y Propagating Cipher Block Chaining (PCBC).
- Los generadores de clave para los algoritmos DES, TripleDES, Blowfish, HMAC-MD5, y HMAC-SHA1.

- Una implementación de MD5 con el algoritmo DES-CBC password-based encryption (PBE) definido en PKCS#5.
- Una implementación del algoritmo de acuerdo entre claves Diffie-Hellman de dos o más partes.
- Un generador del par de claves Diffie-Hellman
- Una implementación de los algoritmos hashing con claves como HMAC-MD5 y HMAC-SHA1 definidos en RFC 2104
- Una implementación del esquema de padding (relleno) descrito en PKCS#5.
- Una implementación del almacén de claves (keystore) para el propietario llamado "JCEKS".

3.4.4 Bouncy Castle Provider

El Bouncy Castle es un proveedor del JCE [13]. La lista completa de algoritmos que soporta incluye:

- Los Algoritmos simétricos (bloques)
    - Los modos de funcionamiento. ECB, CBC, OFB(n), CFB(n). n múltiplo de 8.
    - El relleno (Padding). No padding, PKCS5/7, ISO10126/ISO10126-2, X9.23/X923, CTS.
    - Los algoritmos. AES, AESWrap, Blowfish, CAST5, CAST6, DES, DESede, IDEA, RC2, RC5, RC5-64, RC6, Rijndael, Skipjack, Twofish, Serpent.
- Los Algoritmos simétricos (de flujos). RC4.
- Los Algoritmos asimétricos (de bloques).
    - Formatos de codificación.OAEP, PCKS1 – PKCS v1.5, ISO9796-1 – ISO9796-1 edición 1 padding.
    - El algoritmo. RSA
- El acuerdo de claves. Diffie-Hellman
- Las funciones resumenes (Hashes). MD2, MD4, MD5, RipeMD128, RipeMD160, SHA1, SHA-256, SHA-384, SHA-512, Tigre.
- Los códigos de autentificación de mensajes (MACs). MD2-HMac, MD4-HMac, MD5-HMac, RipeMD128-HMac, RipeMD160-HMAC, SHA1-HMac, Tigre-HMac.
- Las firmas digitales.

- Los esquemas de las firmas: MD2withRSA, MD5withRSA, SHA1withRSA, RIPEMD160withRSA, SHA1withDSA, SHA1withECDSA.

3.4.5 Algoritmo generador de números aleatorios (RNG)

Sun proporciona un algoritmo predeterminado SecureRandom llamado **SHA1PRNG**. Este sigue al apéndice G.7 del estándar 1363 del IEEE, y genera una secuencia de número aleatorio que está a 64 bits del valor de dispersión SHA-1 del valor de siembra y un contador de 64 bits. El contador se incrementa en uno con el fin de generar valores subsiguientes en un flujo aleatorio [4].

3.4.6 El API Java Secure Socket Extensión (JSSE)

El JSSE permite realizar comunicaciones seguras en Internet. Provee una infraestructura e implementación para una versión en Java de SSL y TLS e incluye funcionalidades para el encriptamiento de datos, la autentificación del servidor, la integridad de mensajes, y la autentificación opcional del cliente [3].

Anteriormente, el JSSE fue un paquete opcional al Java ™ 2 SDK, Standard Edition (J2SDK) versiones 1.2 y 1.3. Ahora, el JSSE ha sido integrado en J2SDK, v1.4.

El API JSSE es capaz de soportar las versiones 2.0 y 3.0 de SSL y la versión 1.0 de Transport Layer Security (TLS). Estos protocolos de seguridad encapsulan un socket de flujo bidireccional y el API JSSE adiciona soporte transparente para la autentificación, el encriptamiento, y la protección de integridad. El JSSE en J2SDK v1.4 implementa el SSL 3.0 y TLS 1.0, pero no implementa SSL 2.0.

JSSE es un componente de seguridad de la plataforma Java 2 que está basado en los mismos principios de diseño encontrados en la infraestructura de Java Cryptography Architecture (JCA). Esta infraestructura permite a la criptografía tener la independencia de implementación, y cuando es posible, la independencia de algoritmo. El JSSE usa la arquitectura "provider" definida en el JCA.

El JSSE incluye las siguientes características [3]:

- Está implementado 100% en Java.
- Puede ser exportado a la mayoría de países.
- Provee soporte para los APIs de las versiones 2.0 y 3.0 de SSL, e implementación de la versión 3.0 de SSL.
- Provee soporte para el API y una implementación para la versión 1.0 de TLS.
- Incluye clases que pueden ser instanciadas para crear canales seguros (SSLSocket y SSLServerSocket).
- Provee soporte para la negociación de la suite de cifrado, que forma parte de la negociación de SSL usada para inicializar o verificar las comunicaciones seguras.
- Provee soporte para la autentificación del cliente y del servidor, que forma parte de la negociación de SSL.
- Provee soporte para el Hytertext Transfer Protocol (http) encapsulado en el protocolo SSL (HTTPS).
- Provee los APIs de administración de una sesión para la memoria.
- Incluye los códigos licenciados del RSA Data Security Inc. tales como el RSA y el RC4.
- Provee soporte para varios algoritmos criptográficos usados en la suite de cifrado.

Algoritmo criptográfico	Proceso criptográfico	Longitud de clave (Bits)
RSA	Autentificación y intercambio de clave	2048 (autentificación) 2048 (intercambio de clave) 512 (intercambio de clave)
RC4	Encriptamiento de bloque	128 128 (40 efectivo)
DES	Encriptamiento de bloque	64 (56 efectivo) 64 (40 efectivo)
TripleDES	Encriptamiento de bloque	192 (112 efectivo)
Diffie-Hellman	Acuerdo de claves	1024 512
DSA	Autentificación	1024

Tabla 3.3: funcionalidad criptográfica disponible con JSSE [3].

Capítulo 4

Pruebas y evaluación del sistema

4.1 Introducción

La transmisión de aplicaciones de audio en Internet enfrenta una serie de dificultades tales como la disponibilidad de ancho de banda, la velocidad de procesamiento de los computadores, el retardo en el ruteamiento de los paquetes, y la congestión de la red. En el presente trabajo, se ha evaluado el ancho de banda que consume la transmisión de la voz cifrada y los parámetros (es decir, el retardo, el jitter y la pérdida de paquetes) considerados para una buena calidad de transmisión.

Estas evaluaciones se han realizado en dos escenarios, uno de ellos es una red LAN de 10 Mbps, y el otro es Internet. Las mediciones del ancho de banda, el retardo, el jitter y la pérdida de paquetes se han hecho a través de un software encargado del monitoreo de paquetes transmitidos por la red.

4.2 Evaluación del sistema

En la presente tesis, se ha evaluado:

- ***El ancho de banda consumido por la codificación y el cifrado de la voz***
  Empleamos el ancho de banda ya que es una condición esencial para el transporte de audio; pues desde esta dependen el porcentaje de los retardos y las pérdidas de paquetes. En otras palabras, un usuario que acceda a Internet con un servicio de banda ancha (ADSL, cable MODEM, etc.) tendrá un mejor rendimiento en las aplicaciones en tiempo-real a comparación de otro que acceda con un servicio dial-up.

- ***La calidad de transmisión de la voz***

  La calidad de la voz se ve afectada por el retardo, el jitter y las pérdidas de paquetes. Los procesos de codificación y cifrado adicionan retardo fijo a la red; sin embargo, el transporte de los paquetes RTP adicionan retardo variable dependiente de la condición de la red. Si las condiciones son pobres, el retardo promedio y la varianza del retardo del paquete (jitter) serán altos. El jitter se soluciona a través de un buffer en el receptor, al costo de un retardo adicional; sin embargo, los paquetes retrasados, que superen el máximo de tiempo para ser reproducidos, serán perdidos.

Características del ambiente de evaluación:

Software utilizado:

Java 2, Standard Edition (J2SE), versión 1.4.2.

Java Media Framework 2.1.1e

Jbuilder Enterprise 8.0

Ethereal – Network Protocol Analyser versión 0.10.4

Escenario N° 1: LAN de 10 Mbps

Máquina A: Pentium II de 233 Mhz con 384 MB.

Sistema Operativo: Microsoft Windows 2000 Server.

Máquina B: Pentium IV de 2.4 Ghz con 256 MB de memoria.

Sistema Operativo: Microsoft Windows 2000 Server.

Escenario N° 2: Internet

Máquina A: Pentium III de 864 Mhz con 128 MB.

Sistema Operativo: Microsoft Windows XP

Cabina Internet

Máquina B Pentium IV de 2.4 Ghz con 256 MB de memoria.

Sistema operativo: Microsoft Windows 2000 Server.

Acceso a Internet con el servicio Speedy 200 Kbps.

En esta sección, presentamos los resultados obtenidos en las pruebas realizadas de ambos escenarios de una comunicación half-duplex. La máquina A realiza la transmisión y la B, la recepción. Para el análisis del ancho de banda, se consideraron tres casos: 1) Los paquetes con inclusión de cabeceras, 2) los paquetes con compresión de cabeceras y 3) los que solo tienen datos de audio (payload).

4.2.1 Escenario N° 1: LAN de 10 Mbps

Caso A1: medición del ancho de banda de voz utilizando el codificador G.723.1 con sobrecarga de cabeceras

	Codificador	**codificador G723.1 + algoritmo de cifrado**				
	G723.1	**TripleDES**	**IDEA**	**Blowfish**	**AES**	**CAST**
Voice Payload (bytes)	48	48	48	48	48	48
Cabecera RTP (bytes)	12	12	12	12	12	12
Cabecera UDP (bytes)	8	8	8	8	8	8
Cabecera IP (bytes)	20	20	20	20	20	20
Cabecera Ethernet II (bytes)	14	14	14	14	14	14
Total Paquete	102	102	102	102	102	102
Packet count	174	185	176	181	177	176
Avg. Packets/sec	16.733	16.818	16.763	16.762	16.709	16.761
Avg. Packet size (bytes)	102.00	102.00	102.00	102.00	102.00	102.00
Avg. Bytes/sec	1706.765	1715.421	1709.820	1709.691	1704.349	1709.647
BW Half-Duplex (Kbps)	13.654	13.723	13.678	13.677	13.634	13.677

Tabla 4.1: ancho de banda del sistema utilizando el codificador G.723.1 en una LAN de 10 Mbps.

Análisis de la tabla 4.1

El codificador G723.1 está especificado en la recomendación G.723.1 de la ITU, "Codificador de voz de tasa-dual para las comunicaciones multimedia que se transmiten a 5.3 y 6.3 Kbps". El audio es codificado en tramas de 30 ms. con un retardo adicional de 7.5 ms de look-ahead. Una trama G.723.1 puede ser de tres tamaños:

- 24 octetos (6.3 Kbps frame),
- 20 octetos (5.3 Kbps frame),
- 4 octetos (SID frame, Silence Insertion Descriptor) usados para especificar los parámetros de ruido confortable.

Las tres medidas de 24, 20 y 4 octetos pueden ser combinadas sin restricciones.

Si el tamaño del payload lo limitamos a 48 bytes y consideramos una tasa de 6.3 Kbps, tendríamos dos tramas G723.1 dentro de un paquete de voz; por lo que necesitaríamos 60 ms para su empaquetamiento.

El ancho de banda se calcula de la siguiente forma:

BW = Avg. Packets/sec * Avg. Packet size .....(ecuación 4.1)

De esta forma, los resultados de la tabla 4.1 muestran el ancho de banda que consume el proceso de codificación G723.1 más un algoritmo de cifrado.

El tiempo de procesamiento consumido por el empaquetamiento de la voz cifrada utilizando el codificador G723.1 se muestra a continuación:

	Avg. Packets/sec	**s/paquete**
TripleDES	16.818	0.0594601022 s
IDEA	16.763	0.0596551929 s
Blowfish	16.762	0.0596587519 s
AES	16.709	0.0598479861 s
CAST	16.761	0.0596623113 s

Tabla 4.2: tiempo de empaquetamiento de la voz cifrada utilizando el codificador G723.1 en una LAN de 10 Mbps.

Los resultados correspondientes al tiempo de empaquetamiento utilizando los diferentes algoritmos criptográficos mostrados en la tabla 4.2, concuerda con el tiempo consumido por el empaquetamiento de dos tramas G723.1 equivalentes a 60 ms.

Caso A2: medición del ancho de banda de voz utilizando el codificador G.723.1 considerando solamente los datos de voz cifrados (payload)

	codificador	codificador g723.1 + algoritmo de cifrado				
	G723.1	TripleDES	IDEA	Blowfish	AES	CAST
Voice Payload (bytes)	48	48	48	48	48	48
Packet count	174	185	176	181	177	176
Avg. Packets/sec	16.733	16.818	16.763	16.762	16.709	16.761
Avg. Packet size (bytes)	48.00	48.00	48.00	48.00	48.00	48.00
Avg. Bytes/sec	803.184	807.264	804.624	804.576	802.032	804.528
BW Half-Duplex (Kbps)	6.425	6.458	6.436	6.436	6.416	6.436

Tabla 4.3: análisis del ancho de banda que consume el payload utilizando el codificador G723.1 en una LAN de 10 Mbps.

A continuación, (tabla 4.4) mostramos un resumen del ancho de banda consumido por el empaquetamiento de la voz cifrada con y sin sobrecarga de cabeceras utilizando el codificador G723.1.

	Con cabeceras	Sólo Payload
TripleDES	13.703 Kbps	6.448 Kbps
IDEA	13.693 Kbps	6.443 Kbps
Blowfish	13.725 Kbps	6.447 Kbps
AES	13.684 Kbps	6.439 Kbps
CAST	13.688 Kbps	6.441 Kbps

Tabla 4.4: utilización del ancho de banda de los diferentes algoritmos criptográficos utilizando el codificador G723.1 en una LAN de 10 Mbps.

En la tabla 4.4 se observa cómo las cabeceras de las distintas capas aumentan el consumo del ancho de banda.

Evaluación del Delay, del Jitter y del retardo de paquetes (voz cifrada) utilizando el codificador G723.1

	Codificador	codificador g723.1 + algoritmo de cifrado				
	G.723.1	TripleDES	IDEA	Blowfish	AES	CAST
Packet received	174	185	176	181	177	176
Packet expected	174	185	176	181	177	176
Lost RTP Packet	0	0	0	0	0	0
Max delay (ms)	110.361	109.954	110.243	110.601	109.581	110.245
Max Jitter (ms)	61.903	61.847	61.835	61.849	61.838	61.880

Tabla 4.5: máximo delay, jitter y número de pérdidas de los paquetes cifrados utilizando el codificador G723.1 en una LAN de 10 Mbps.

Gráficas del delay y del jitter observados en la transmisión de paquetes

Considerando el uso del **codificador G723.1 sin ningún algoritmo de cifrado** se obtuvo 174 paquetes en el lapso de 10 segundos. Asimismo, las distribuciones del retardo y del jitter con respecto al número de paquetes fueron las siguientes:

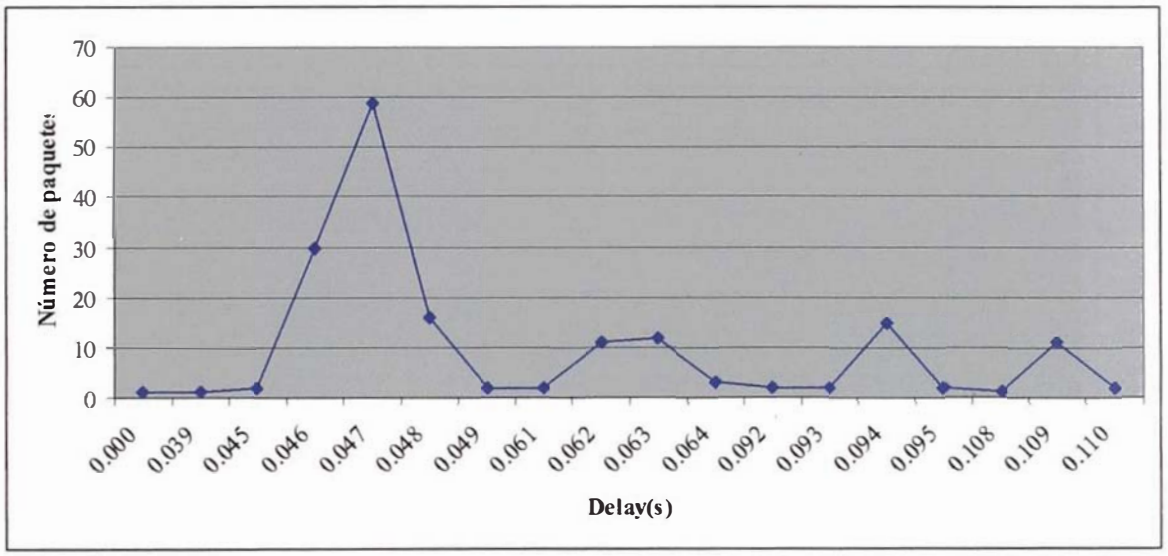

Figura 4.1: distribución de retardo entre el número de paquetes utilizando solamente el codificador G723.1 sin ningún tipo de encriptamiento en una LAN de 10 Mbps.

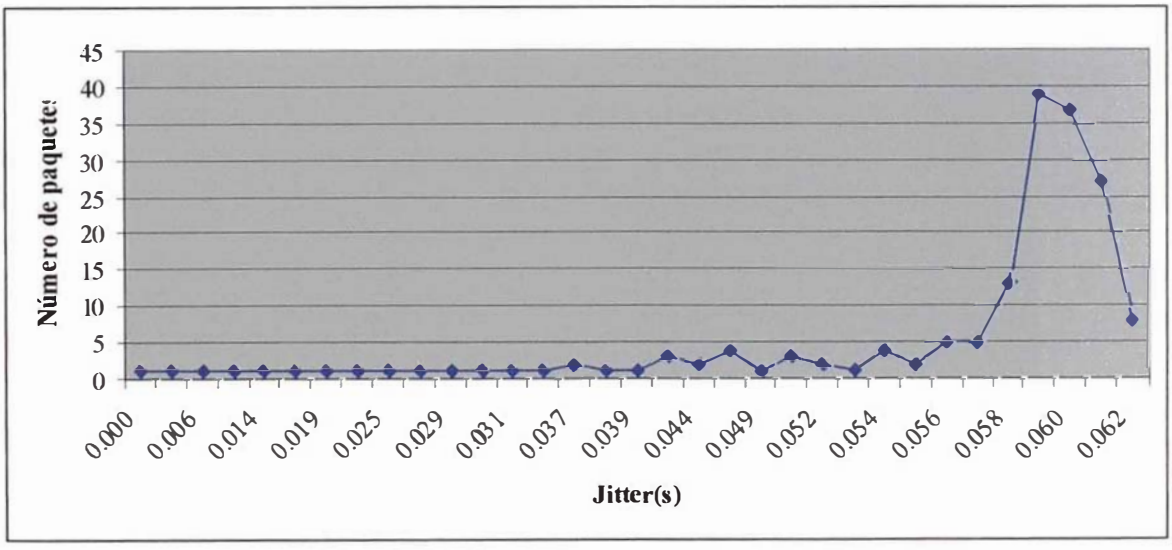

Figura 4.2: distribución del jitter entre el número de paquetes utilizando solamente el codificador G723.1 sin ningún tipo de encriptamiento en una LAN de 10 Mbps.

Considerando el uso del **codificador G723.1 más el algoritmo de cifrado IDEA** se obtuvo 176 paquetes en el lapso de 10 segundos. Del mismo modo, las distribuciones del retardo y del jitter con respecto al número de paquetes fueron las siguientes:

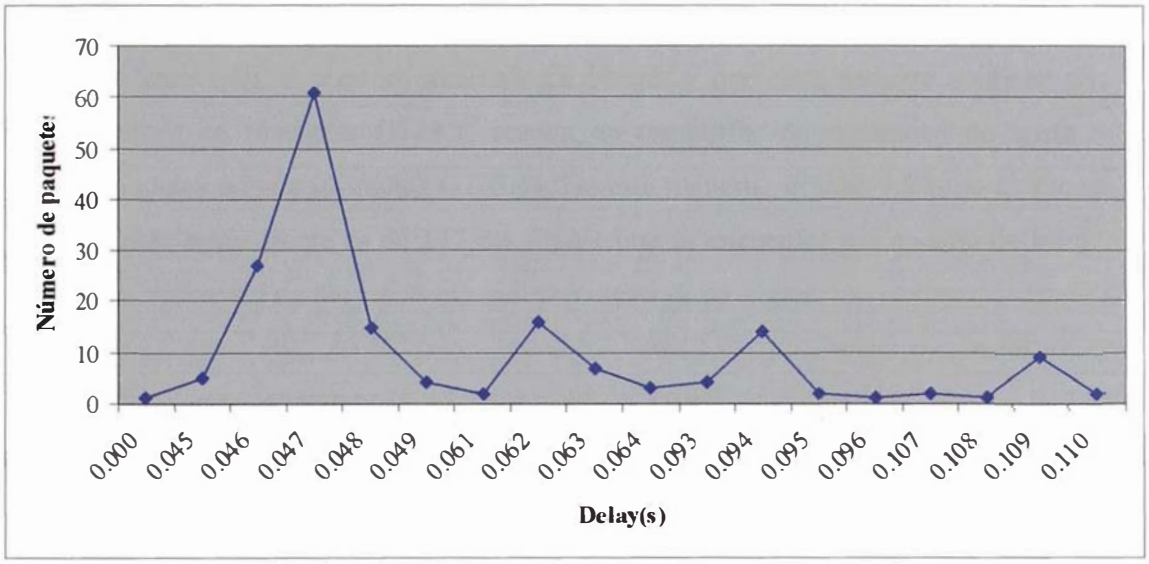

Figura 4.3: distribución de retardo entre el número de paquetes utilizando el codificador G723.1 más el algoritmo de cifrado IDEA en una LAN de 10 Mbps.

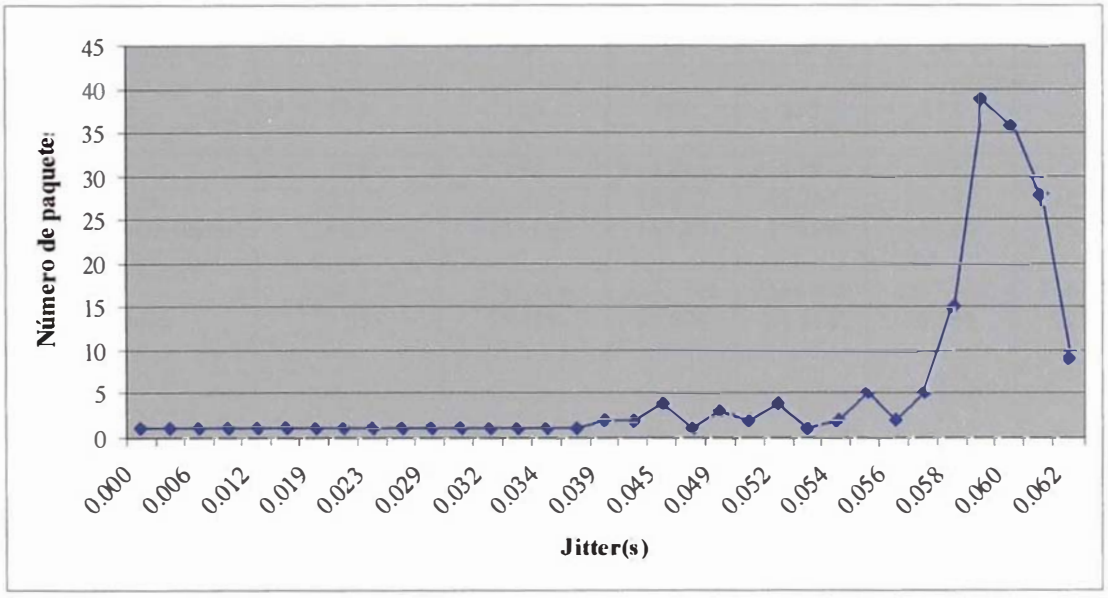

Figura 4.4: distribución del jitter entre el número de paquetes utilizando el codificador G723.1 más el algoritmo de cifrado IDEA en una LAN de 10 Mbps.

Análisis de los resultados de la tabla 4.5 y de las figuras del 4.1-4.4

De acuerdo a los resultados de la tabla 4.5, observamos que el máximo delay es aproximadamente de 110 ms.

1. Se tiene 0% de paquetes perdidos.
2. Teniendo en cuenta que el máximo jitter es de aproximadamente 60 ms., se utiliza un buffer de 200 ms. en el lado receptor. Conociendo que cada paquete de voz cifrada tiene un intervalo de 59 ms, y que cada paquete contiene dos frames de muestras G723.1, resulta en un buffer de capacidad de hasta 3 paquetes de voz (6 tramas G723.1): De esta manera, el jitter máximo al que se puede hacer frente es de 177 ms., valor que se encuentra por encima de los 61 ms. (tabla 4.5).

Caso B1: ancho de banda de voz utilizando el codificador GSM con sobrecarga de cabeceras

	codificador	codificador gsm + algoritmo de cifrado				
	GSM	TripleDES	IDEA	Blowfish	AES	CAST
Voice Payload (bytes)	99	99	99	99	99	99
Cabecera RTP (bytes)	12	12	12	12	12	12
Cabecera UDP (bytes)	8	8	8	8	8	8
Cabecera IP (bytes)	20	20	20	20	20	20
Cabecera Ethernet II (bytes)	14	14	14	14	14	14
Total Paquete	153	153	153	153	153	153
Packet count	178	174	179	176	177	172
Avg. Packets/sec	16.777	16.746	16.822	16.762	16.781	16.780
Avg. Packet size (bytes)	153.00	153.00	153.00	153.00	153.00	153.00
Avg. Bytes/sec	2566.920	2562.066	2573.765	2564.528	2567.433	2567.370
BW Half-Duplex (Kbps)	20.535	20.496	20.590	20.516	20.539	20.538

Tabla 4.6: ancho de banda del sistema utilizando el codificador GSM en una LAN de 10 Mbps.

Análisis de la tabla 4.6

La codificación GSM muestra el estándar europeo GSM 06.10 para la codificación de voz full-rate, la cual está basada en la codificación RPE/LTP (Residual Pulse Excitation/Long Term Prediction) a una tasa de 13 Kbps. Una trama GSM tiene un tamaño de 33 octetos y el audio es codificado en tramas de 20ms.

Si el tamaño del payload lo limitamos a 99 bytes, entonces tenemos tres tramas GSM dentro de un paquete de voz. Por lo tanto, se necesita 60 ms. para su empaquetamiento.

El ancho de banda se calcula según la ecuación 4.1.

El tiempo de procesamiento consumido por el empaquetamiento de la voz crifada, utilizando el codificador GSM, se muestra a continuación:

	Avg. Packets/sec	**s/paquete**
TripleDES	16.746	0.059715753 s
IDEA	16.822	0.059445963 s
Blowfish	16.762	0.059658751 s
AES	16.781	0.059591204 s
CAST	16.780	0.059594755 s

Tabla 4.7: tiempo de empaquetamiento de la voz cifrada utilizando el codificador GSM en una LAN de 10 Mbps.

Los resultados del tiempo de empaquetamiento de los diferentes algoritmos criptográficos (tabla 4.7) concuerdan con el empaquetamiento de tres tramas GSM equivalentes a 60 ms.

Caso B2: ancho de banda de voz utilizando el codificador GSM considerando el payload

	codificador	**codificador + algoritmo de cifrado**				
	GSM	**TripleDES**	**IDEA**	**Blowfish**	**AES**	**CAST**
Voice Payload (bytes)	99	99	99	99	99	99
Packet count	178	174	179	176	177	172
Avg. Packets/sec	16.777	16.746	16.822	16.762	16.781	16.780
Avg. Packet size (bytes)	99.00	99.00	99.00	99.00	99.00	99.00
Avg. Bytes/sec	1660.923	1657.854	1665.378	1659.438	1661.319	1661.22
BW Half-Duplex (Kbps)	13.287	13.262	13.323	13.275	13.290	13.289

Tabla 4.8: análisis del ancho de banda que consume sólo el payload utilizando el codificador GSM en una LAN de 10 Mbps.

A continuación, (tabla 4.9) mostramos un resumen del ancho de banda consumido por el empaquetamiento de la voz cifrada con y sin sobrecarga de cabeceras utilizando el codificador GSM.

	Con cabeceras	Sólo Payload
TripleDES	13.703 Kbps	6.448 Kbps
IDEA	13.693 Kbps	6.443 Kbps
Blowfish	13.725 Kbps	6.447 Kbps
AES	13.684 Kbps	6.439 Kbps
CAST	13.688 Kbps	6.441 Kbps

Tabla 4.9: utilización del ancho de banda de los diferentes algoritmos criptográficos utilizando el codificador GSM en una LAN de 10 Mbps.

En la tabla 4.9, se observa como las cabeceras de los paquetes de voz aumentan el consumo del ancho de banda.

Evaluación del Delay, del Jitter y del retardo de paquetes (voz cifrada) utilizando el codificador GSM.

	codificador	codificador gsm + algoritmo de cifrado				
	GSM	TripleDES	IDEA	Blowfish	AES	CAST
Packet received	178	174	179	176	177	172
Packet expected	178	174	179	176	177	172
Lost RTP Packet	0	0	0	0	0	0
Max delay (ms)	110.711	110.325	110.300	110.577	110.580	110.374
Max Jitter (ms)	61.917	61.837	61.835	61.880	62.012	61.898

Tabla 4.10: máximo delay, jitter y número de pérdidas de los paquetes cifrados utilizando el codificador GSM en una LAN de 10 Mpbs.

Gráficas del delay y del jitter observados en la transmisión de paquetes

Considerando el uso del **codificador GSM sin ningún algoritmo de cifrado** se obtuvo 178 paquetes en el lapso de 10 segundos, y las distribuciones del retardo y del jitter con respecto al número de paquetes fueron las siguientes:

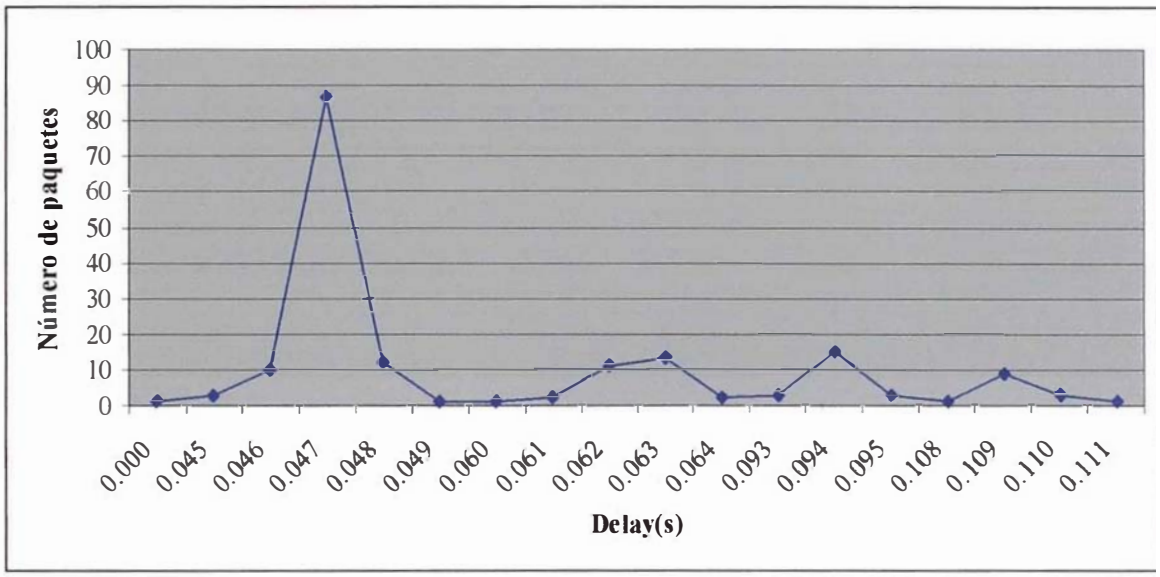

Figura 4.5: distribución de retardo entre el número de paquetes utilizando solamente el codificador GSM sin ningún tipo de encriptamiento en una LAN de 10 Mbps.

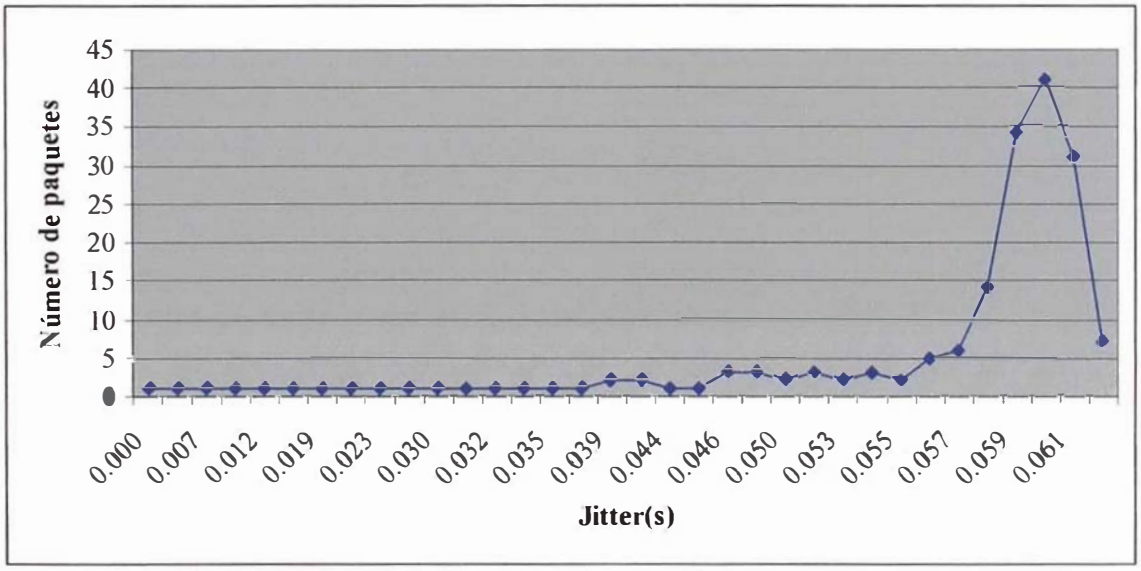

Figura 4.6: distribución del jitter entre el número de paquetes utilizando solamente el codificador GSM sin ningún tipo de encriptamiento en una LAN de 10 Mbps.

Considerando el uso del **codificador GSM más el algoritmo de cifrado IDEA,** se obtuvo 179 paquetes en el lapso de 10 segundos. Asimismo, las distribuciones del retardo y del jitter con respecto al número de paquetes fueron las siguientes:

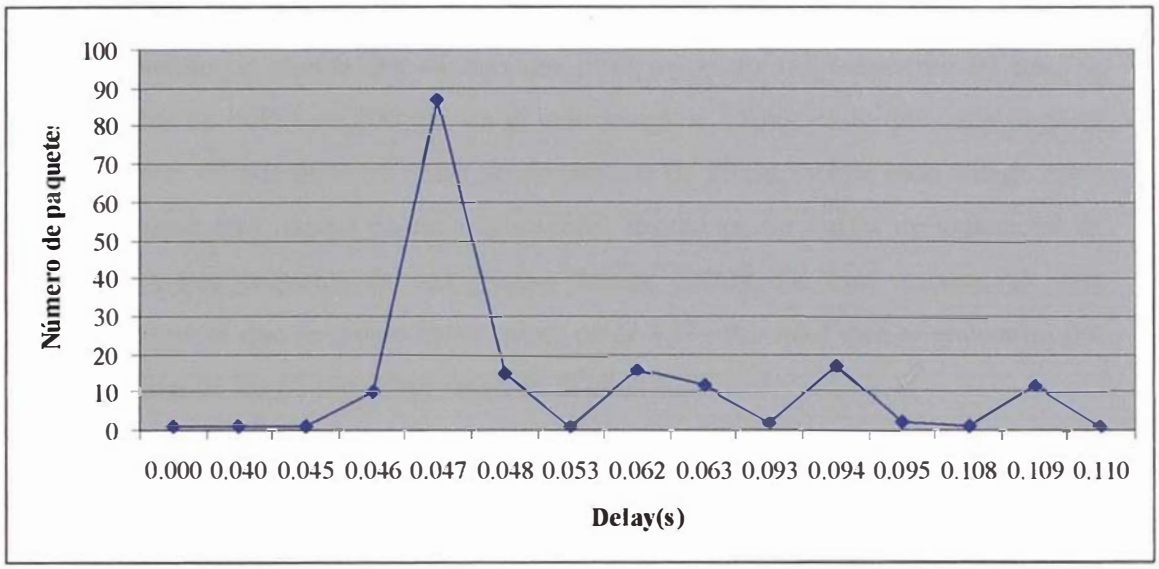

Figura 4.7: distribución de retardo entre el número de paquetes utilizando el codificador GSM más el algoritmo de cifrado IDEA en una LAN de 10 Mbps.

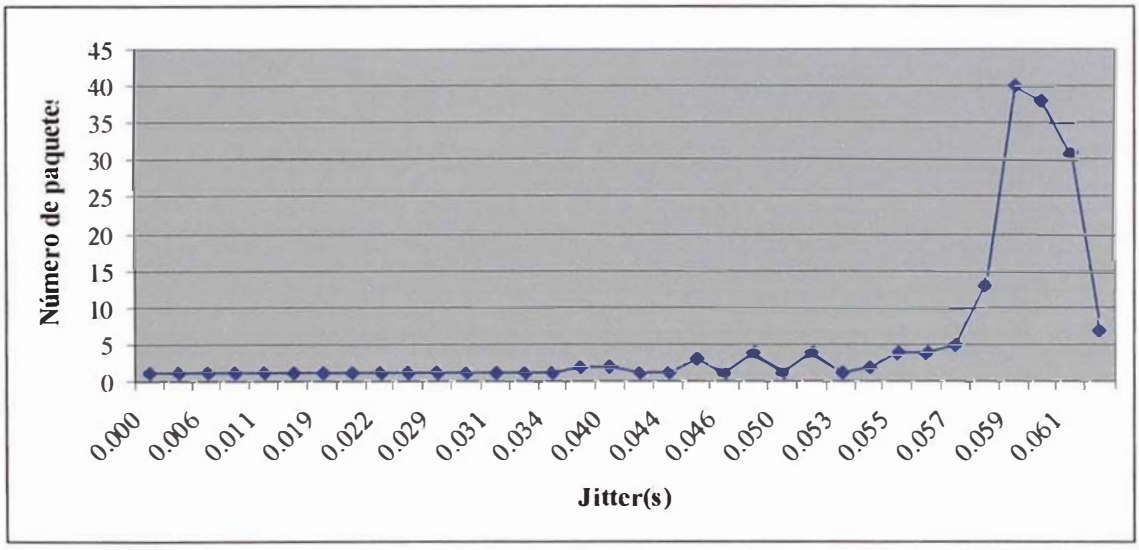

Figura 4.8: distribución del jitter entre el número de paquetes utilizando el codificador GSM más el algoritmo de cifrado IDEA en una LAN de 10 Mbps.

Análisis de los resultados de la tabla 4.10 y de las figuras del 4.5-4.8

1. De acuerdo a los resultados de la tabla 4.10, observamos que el máximo delay es de 110 ms.
2. Se tiene 0% de paquetes perdidos
3. Teniendo en cuenta que el máximo jitter es de aproximadamente 60 ms., se utiliza un buffer de 200 ms. en el lado receptor. Conociendo que cada paquete de voz cifrada tiene un intervalo de paquete de 59 ms, y que cada uno de estos contiene tres tramas de muestras GSM, resulta en un buffer de capacidad de hasta tres paquetes de voz (nueve tramas GSM). De esta manera, el jitter máximo al que se puede hacer frente es de 177 ms., valor que se encuentra por encima de los 61 ms. observados en la tabla 4.10.

4.2.2 Escenario N° 2: Internet

Caso C1: ancho de banda de voz utilizando el codificador G723.1 con sobrecarga de cabeceras

	codificador	Codificador g723.1 + algoritmo cifrado				
	G723.1	TripleDES	IDEA	Blowfish	AES	CAST
Voice Payload (bytes)	48	48	48	48	48	48
Cabecera RTP (bytes)	12	12	12	12	12	12
Cabecera UDP (bytes)	8	8	8	8	8	8
Cabecera IP (bytes)	20	20	20	20	20	20
Point-to-Point	2	2	2	2	2	2
PPP-over-Ethernet Session (bytes)	6	6	6	6	6	6
Ethernet II	14	14	14	14	14	14
Total Paquete	110	110	110	110	110	110
Packet count	175	169	168	170	172	171
Avg. Packets/sec	16.760	16.811	16.699	17.068	16.052	17.531
Avg. Packet size (bytes)	110.00	110.00	110.00	110.00	110.00	110.00
Avg. Bytes/sec	1843.653	1849.212	1836.874	1877.501	1765.673	1928.426
BW Half-Duplex (Kbps)	14.749	14.793	14.694	15.020	14.125	15.427

Tabla 4.11: ancho de banda del sistema utilizando el codificador G723.1 en Internet

Análisis de la tabla 4.11

Siguiendo las mismas consideraciones que en la tabla 4.1, y considerando que el payload ocupa 48 bytes, entonces tenemos dos tramas G723.1 dentro de un paquete de voz, y por lo tanto demandaría 60ms. para su empaquetamiento.

El tiempo de procesamiento consumido por el empaquetamiento de la voz cifrada utilizando el codificador G723.1, se muestra a continuación:

	Avg. Packets/sec	s/paquete
TripleDES	16.811	0.059484861 s
IDEA	16.699	0.059883825 s
Blowfish	17.068	0.058589172 s
AES	16.052	0.062297533 s
CAST	17.531	0.057041811 s

Tabla 4.12: tiempo de empaquetamiento de la voz cifrada utilizando el codificador G.723.1 en Internet.

Los resultados correspondientes al tiempo de empaquetamiento utilizando los diferentes algoritmos criptográficos (tabla 4.12), concuerdan con el tiempo consumido por el empaquetamiento de dos tramas G723.1 equivalentes a 60 ms.

Caso C2: ancho de banda de voz utilizando el codificador G723.1 con compresión de cabeceras (IP Header + UDP Header + RTP Header)

	codificador	codificador g723.1 + algoritmo de cifrado				
	G723.1	**TripleDES**	**IDEA**	**Blowfish**	**AES**	**CAST**
Voice Payload (bytes)	48	48	48	48	48	48
Compresión cabeceras	4	4	4	4	4	4
Point-to-Point	2	2	2	2	2	2
PPP-over-Ethernet Session (bytes)	6	6	6	6	6	6
Ethernet II	14	14	14	14	14	14
Total Paquete	74	74	74	74	74	74
Packet count	175	169	168	170	172	171
Avg. Packets/sec	16.760	16.811	16.699	17.068	16.052	17.531
Avg. Packet size (bytes)	74.00	74.00	74.00	74.00	74.00	74.00
Avg. Bytes/sec	1240.24	1244.014	1235.726	1263.032	1187.848	1297.294
BW Half-Duplex (Kbps)	9.921	9.952	9.885	10.104	9.502	10.378

Tabla 4.13: análisis del ancho de banda del sistema considerando la compresión de cabeceras utilizando el codificador G.723.1 en Internet.

Caso C3: ancho de banda de voz utilizando la codificación G723.1 considerando el payload

	codificador	codificador g723.1 + algoritmo de cifrado				
	G723.1	**TripleDES**	**IDEA**	**Blowfish**	**AES**	**CAST**
Voice Payload (bytes)	48	48	48	48	48	48
Packet count	175	169	168	170	172	171
Avg. Packets/sec	16.760	16.811	16.699	17.068	16.052	17.531
Avg. Packet size (bytes)	48.00	48.00	48.00	48.00	48.00	48.00
Avg. Bytes/sec	804.48	806.928	801.552	819.264	770.496	841.488
BW Half-Duplex (Kbps)	6.435	6.455	6.412	6.554	6.163	6.731

Tabla 4.14: análisis del ancho de banda que consume el payload utilizando el codificador G.723.1 en Internet.

A continuación, (tabla 4.15) mostramos un resumen del ancho de banda consumido por el empaquetamiento de la voz cifrada con cabeceras, con compresión de cabeceras y sin cabeceras.

	Con cabeceras	Compresión Cabeceras	Sólo Payload
DES	14.776	9.940	6.448
TripleDES	14.793	9.952	6.455
IDEA	14.694	9.885	6.412
Blowfish	15.020	10.104	6.554
AES	14.125	9.502	6.163
CAST	15.427	10.378	6.731

Tabla 4.15: utilización del ancho de banda de los diferentes algoritmos criptográficos utilizando el codificador G.723.1 en Internet.

La Tabla 4.15 indica que sólo el procesamiento del payload ocupa el 43% del ancho de banda total, y que 57% se utiliza para el empaquetamiento de cabeceras. Si utilizáramos algunos de los estándares de compresión de cabeceras (RFCs 3545 - CRTP Enhanced Compressed RTP - y RFC 3095 – ROHC Robust Header Compression - permiten la compresión de cabeceras IP, UPD y RTP), se lograría disminuir en un 33% el consumo del ancho de banda. Estos mecanismos suprimen la información redundante que se transmite en una sesión RTP y, a la vez, consiguen reducir la sobrecarga de cabeceras a 2 ó 4 octetos.

Evaluación del Delay, del Jitter y del retardo de paquetes (voz cifrada) utilizando el codificador G723.1

	codificador	codificador g723.1 + algoritmo de cifrado				
	G723.1	TripleDES	IDEA	Blowfish	AES	CAST
Packet received	175	169	168	170	172	171
Packet expected	175	169	168	170	172	171
Lost RTP Packet	0	0	0	0	0	0
Max delay (ms)	151.521	109.650	342.87	306.491	486.604	563.969
Max Jitter (ms)	62.937	62.123	76.901	71.429	87.260	90.389

Tabla 4.16: máximo delay, jitter y número de pérdidas de los paquetes cifrados utilizando el codificador G.723.1 en Internet.

Gráficas del delay y del jitter observados en la transmisión de paquetes

Considerando el uso del **codificador G723.1 sin ningún tipo de algoritmo de cifrado,** se obtuvo 175 muestras de paquetes en el lapso de 10 segundos, y las distribuciones del retardo y del jitter con respecto al número de paquetes fueron las siguientes:

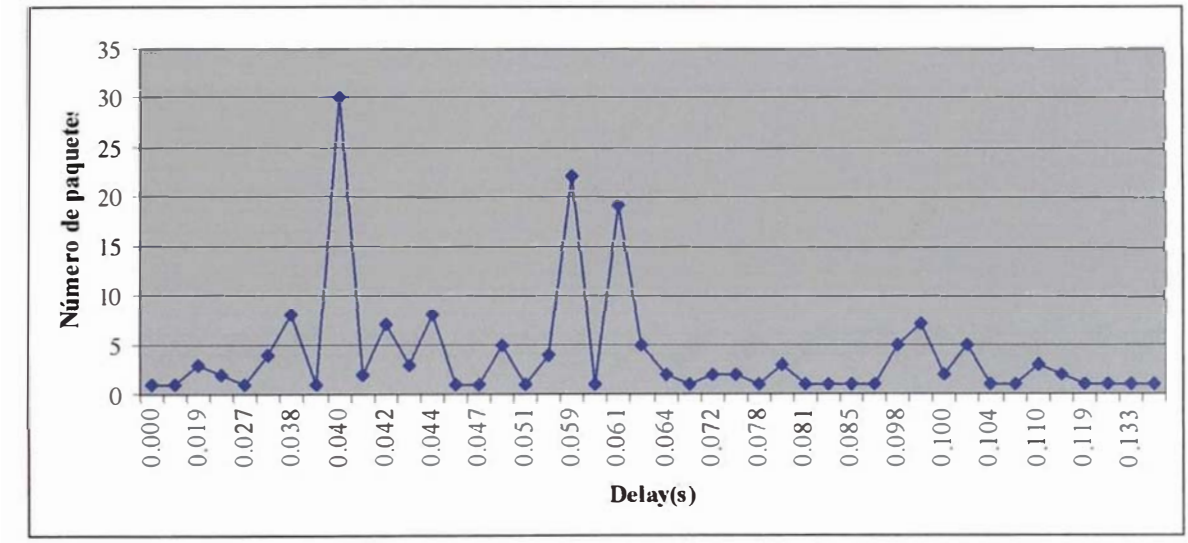

Figura 4.9: distribución de retardo entre el número de paquetes utilizando solamente el codificador G723.1 sin ningún tipo de encriptamiento en Internet.

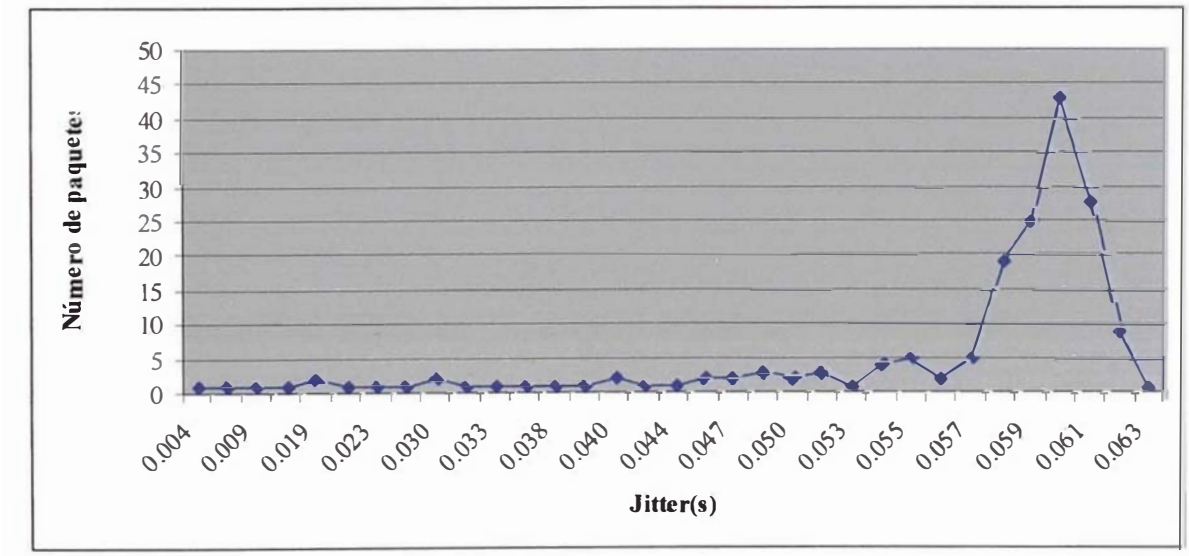

Figura 4.10: distribución del jitter entre el número de paquetes utilizando solamente el codificador G723.1 sin ningún tipo de encriptamiento en Internet.

Considerando el uso del **codificador G723.1 más el algoritmo de cifrado TripleDES,** se obtuvo 169 paquetes en el lapso de 10 segundos. Asimismo, las distribuciones del retardo y del jitter con respecto al número de paquetes fueron las siguientes:

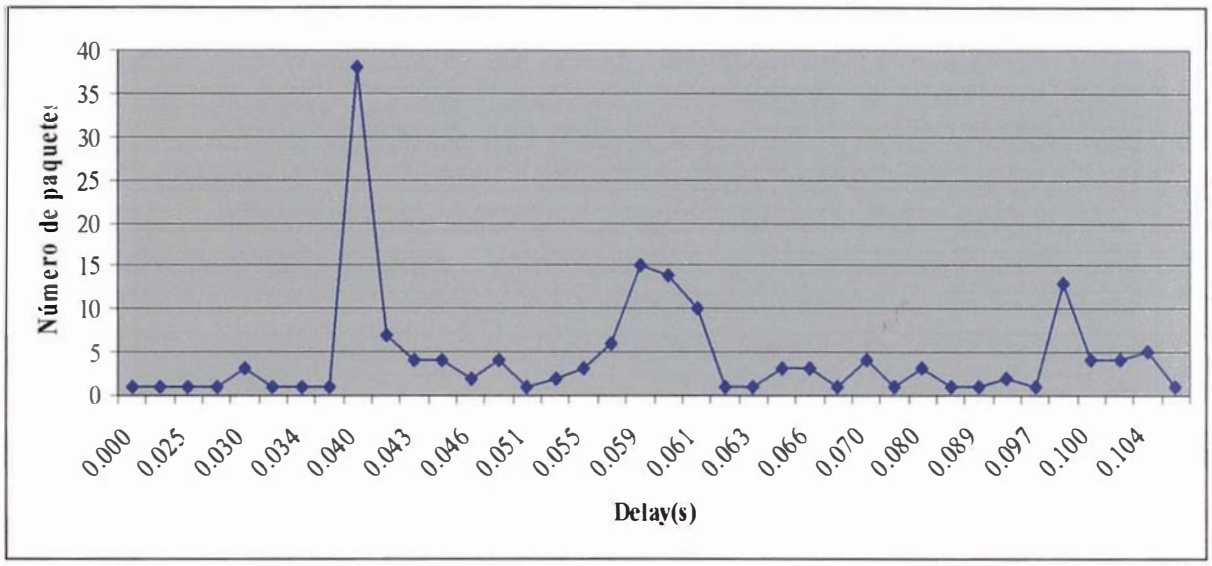

Figura 4.11: distribución de retardo entre el número de paquetes utilizando el codificador G723.1 más el algoritmo de cifrado TripleDES en Internet.

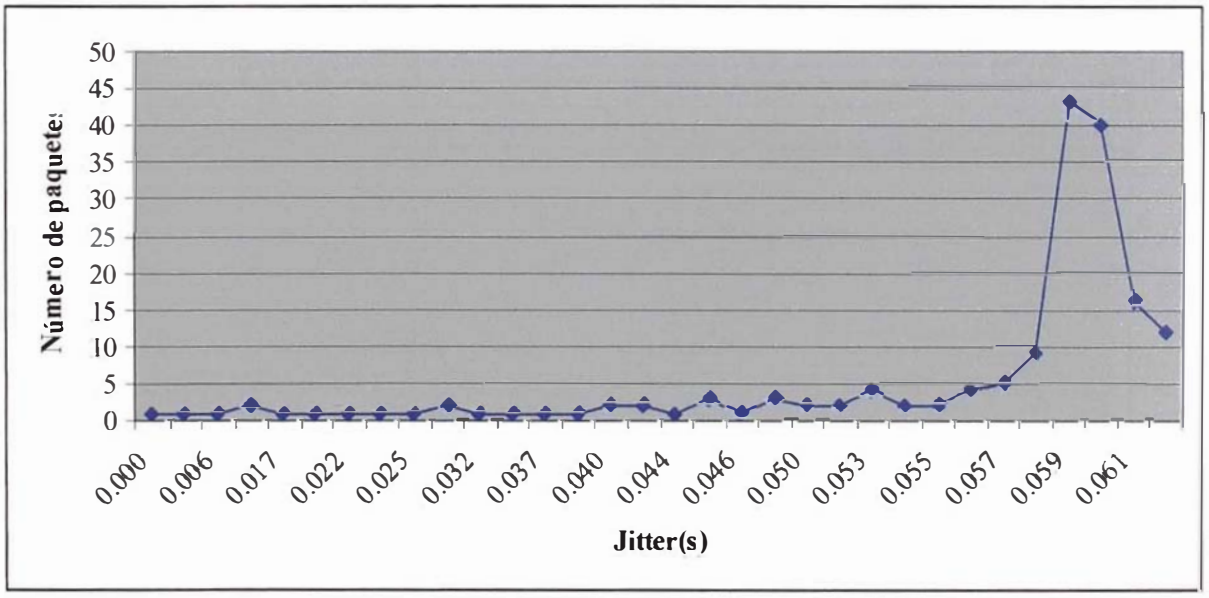

Figura 4.12: distribución del jitter entre el número de paquetes para el codificador G723.1 más el algoritmo de cifrado TripleDES en Internet.

Considerando el uso del **codificador G723.1 más el algoritmo de cifrado IDEA,** se obtuvo 168 paquetes en el lapso de 10 segundos. Por otra parte, las distribuciones del retardo y del jitter con respecto al número de paquetes fueron las siguientes:

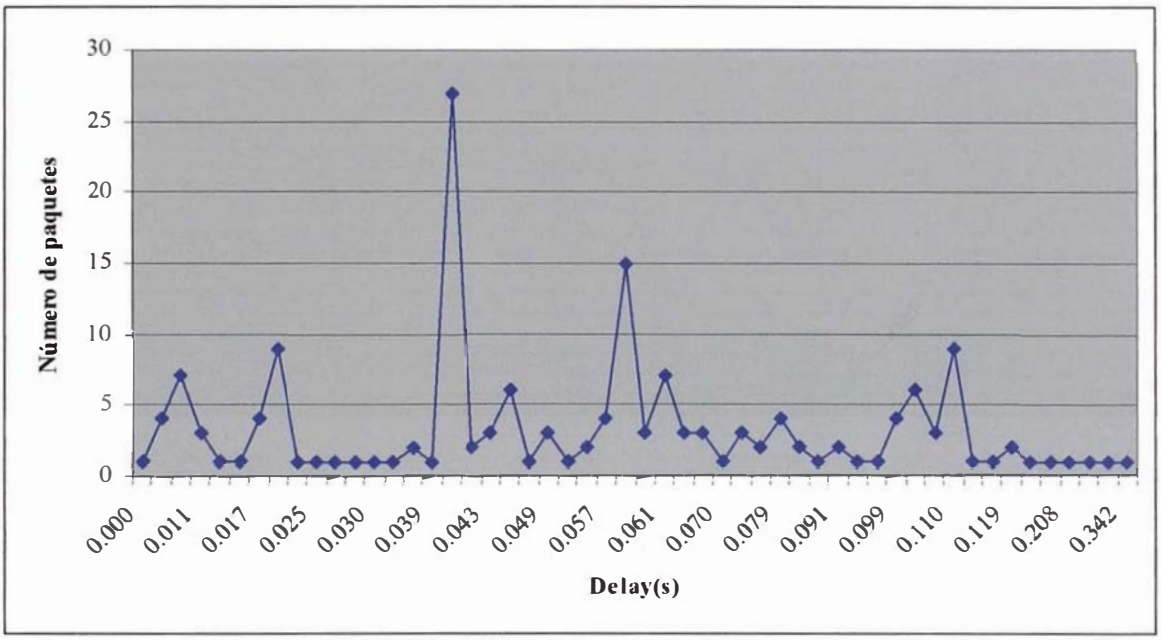

Figura 4.13: distribución de retardo entre el número de paquetes utilizando el codificador G723.1 más el algoritmo de cifrado IDEA en Internet.

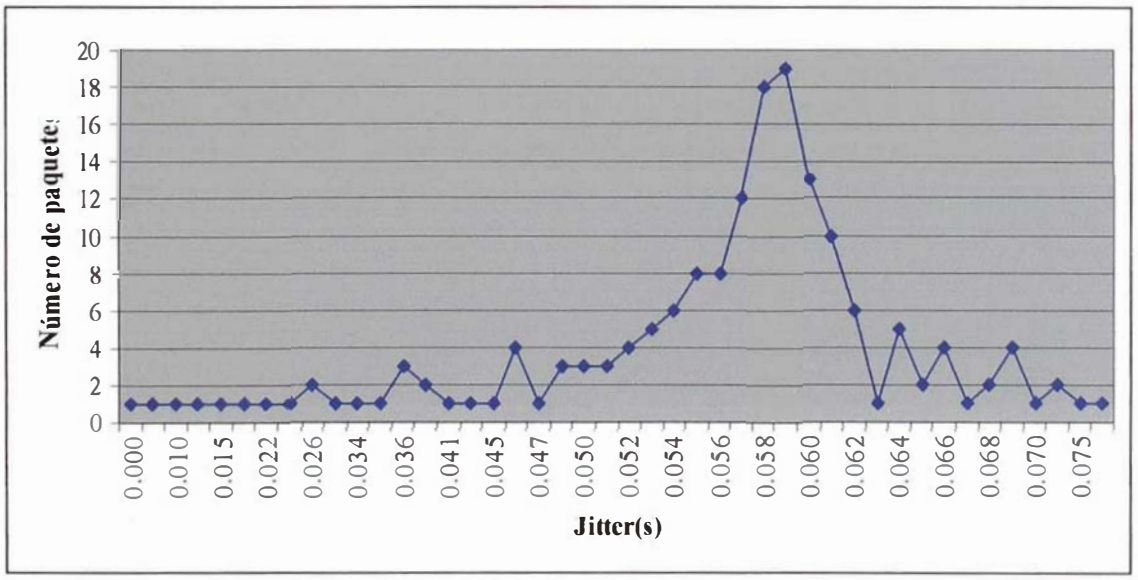

Figura 4.14: distribución del jitter entre el número de paquetes para el codificador G723.1 más el algoritmo de cifrado IDEA en Internet.

Considerando el uso del **codificador G723.1 más el algoritmo de cifrado Blowfish,** se obtuvo 170 paquetes en el lapso de 10 segundos. Asimismo, las distribuciones del retardo y del jitter con respecto al número de paquetes fueron las siguientes:

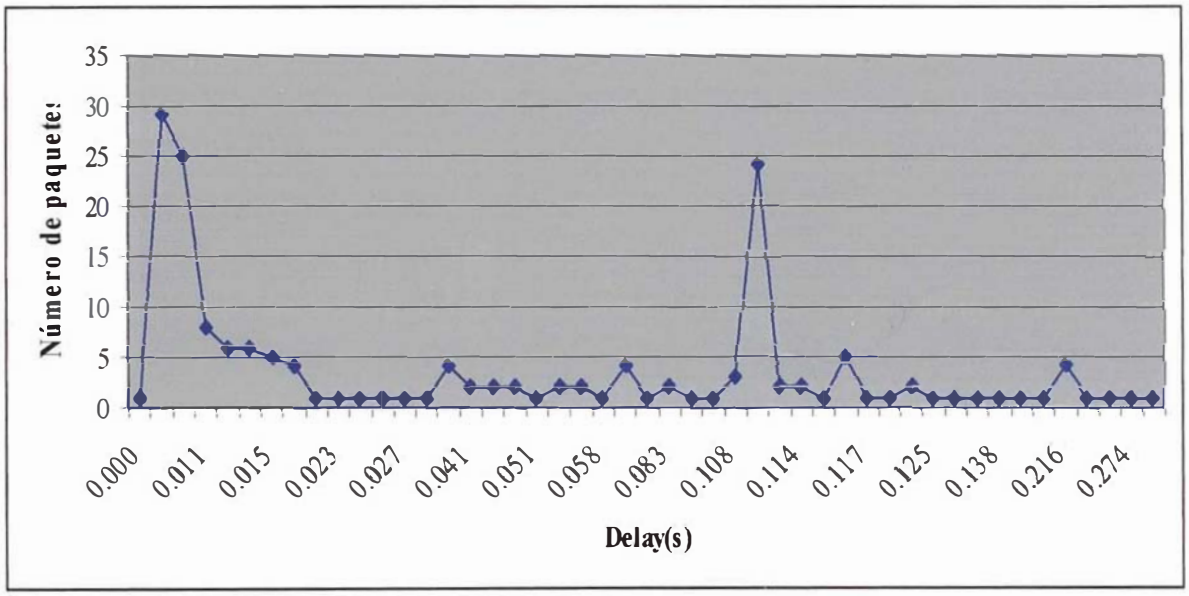

Figura 4.15: distribución de retardo entre el número de paquetes utilizando el codificador G723.1 más el algoritmo de cifrado Blowfish en Internet.

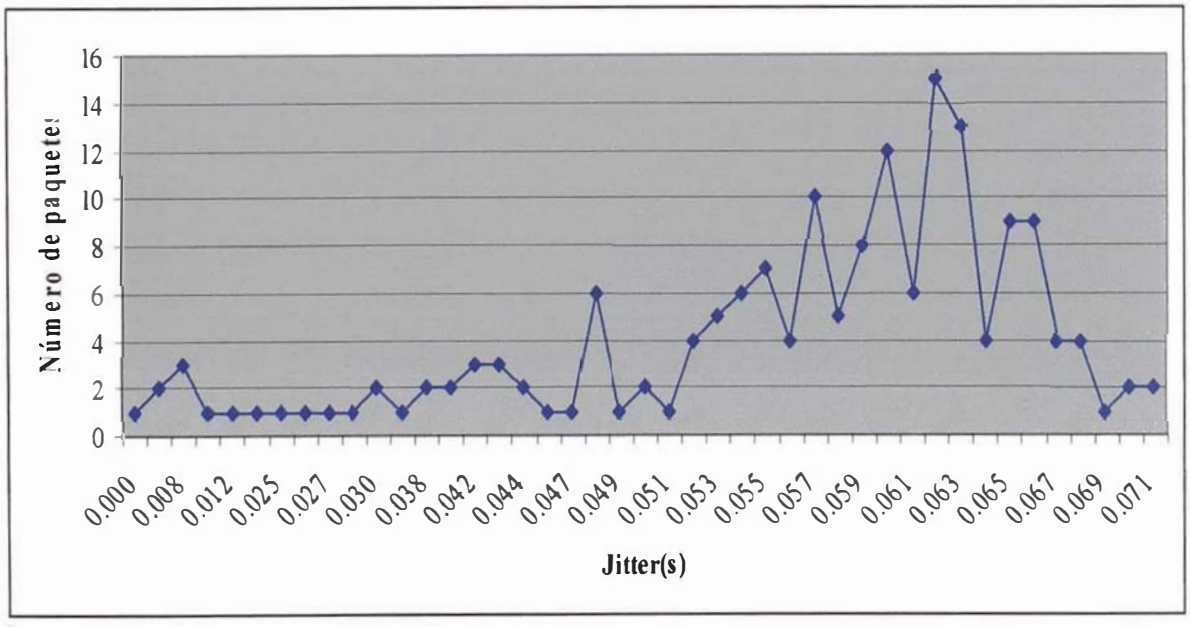

Figura 4.16: distribución del jitter entre el número de paquetes utilizando el codificador G723.1 más el algoritmo de cifrado Blowfish en Internet.

Considerando el uso del **codificador G723.1 más el algoritmo de cifrado AES,** se obtuvo 172 paquetes en el lapso de 10 segundos. Del mismo modo, las distribuciones del retardo y del jitter con respecto al número de paquetes fueron las siguientes:

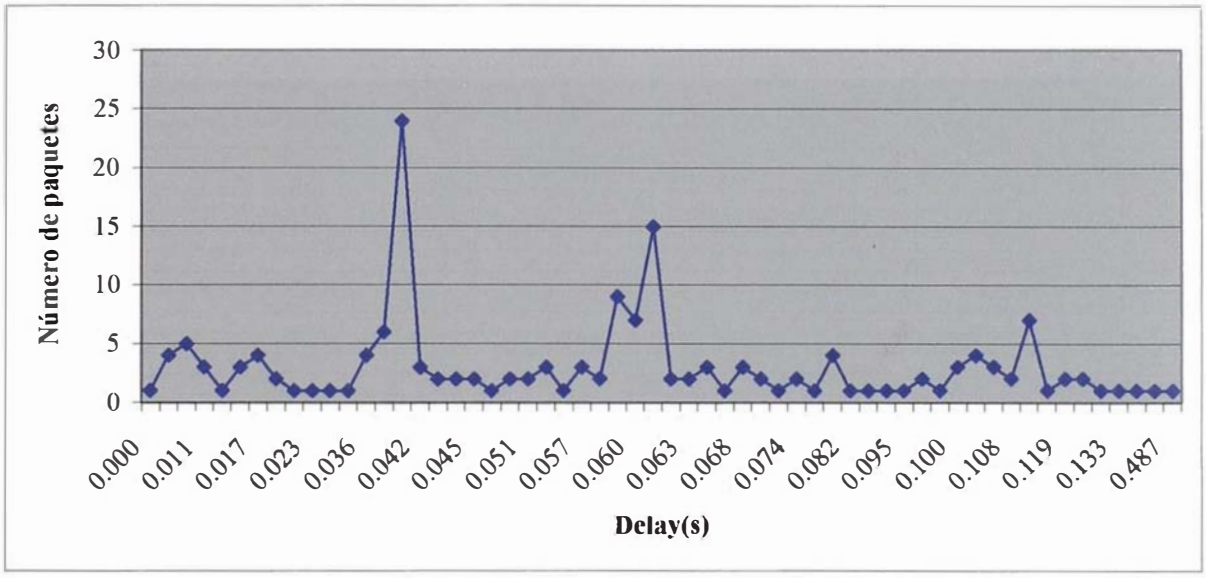

Figura 4.17: distribución de retardo entre el número de paquetes utilizando el codificador G723.1 más el algoritmo de cifrado AES en Internet.

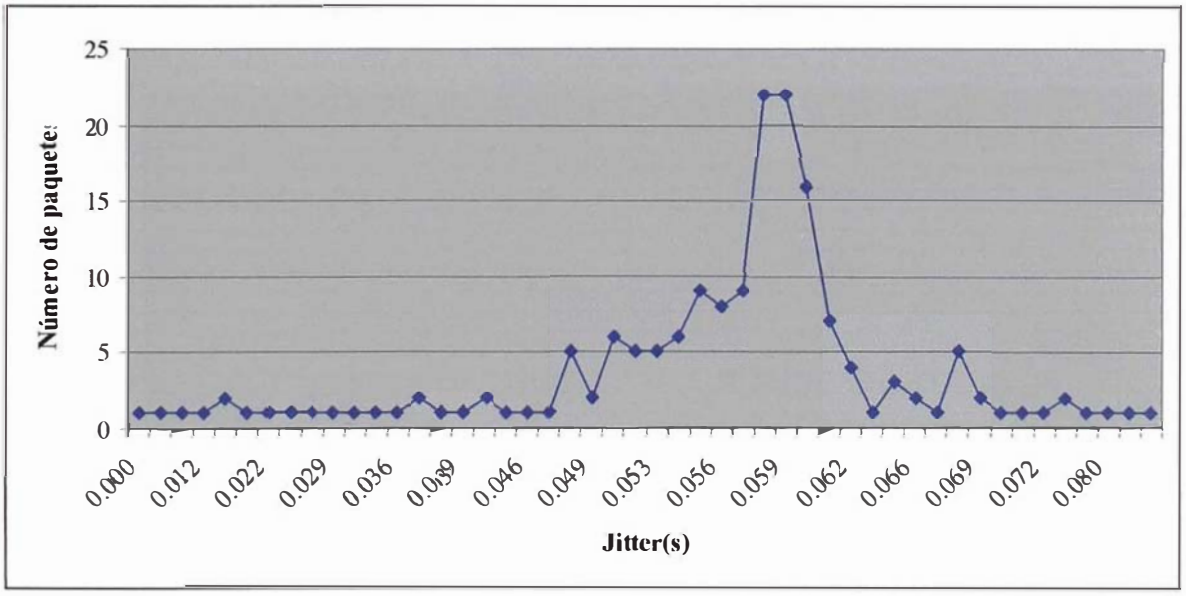

Figura 4.18: distribución del jitter entre el número de paquetes utilizando el codificador G723.1 más el algoritmo de cifrado AES en Internet.

Considerando el uso del **codificador G723.1 más el algoritmo de cifrado CAST5,** se obtuvo 171 paquetes en el lapso de 10 segundos. Asimismo, las distribuciones del retardo y del jitter con respecto al número de paquetes fueron las siguientes:

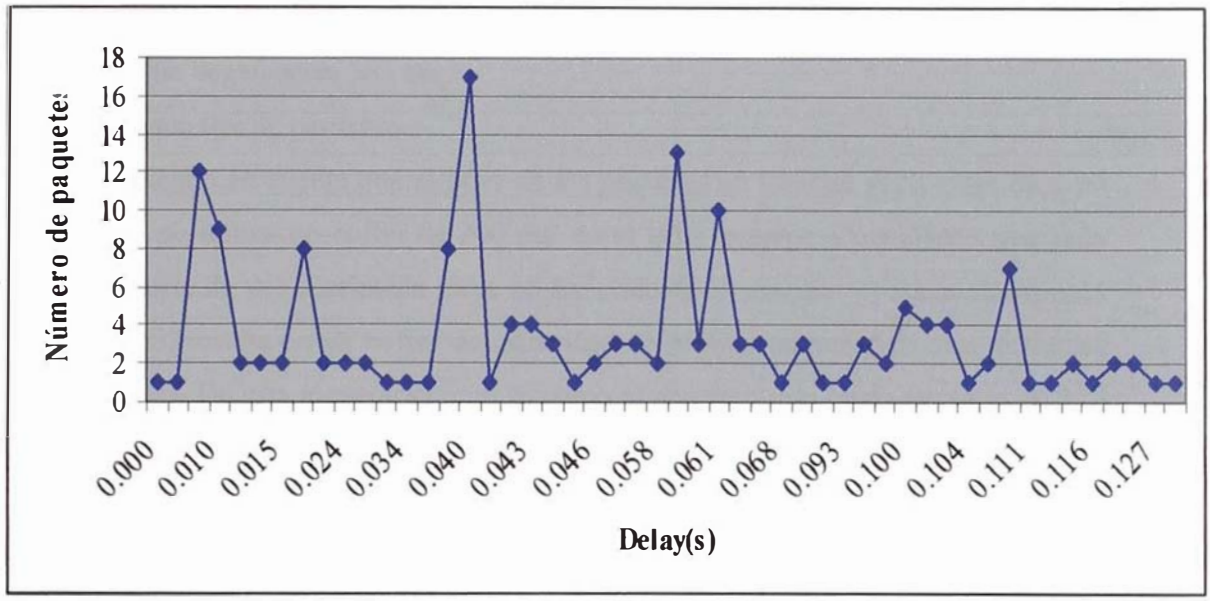

Figura 4.19: distribución de retardo entre el número de paquetes utilizando el codificador G723.1 más el algoritmo de cifrado CAST5 en Internet.

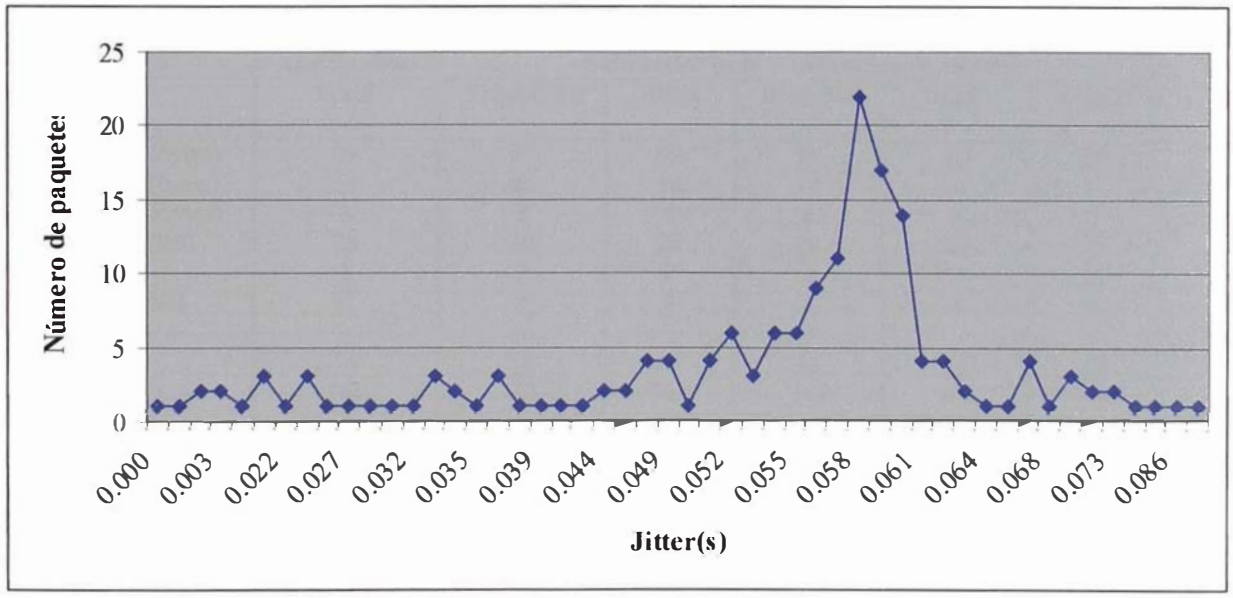

Figura 4.20: distribución del jitter entre el número de paquetes utilizando el codificador G723.1 más el algoritmo de cifrado CAST5 en Internet.

Análisis de los resultados de la tabla 4.16 y de las figuras del 4.9-4.20

1. De acuerdo a los resultados observados en la tabla 4.16 y figuras 4.9, 4.11, 4.13, 4.15, 4.17 y 4.19, observamos que la mayor cantidad de paquetes tienen un delay de hasta 120 ms. Por esta razón, son pocos los que exceden esta cantidad, ya que llegan hasta 563 ms.
2. Existen 0% de pérdidas.
3. Teniendo en cuenta que el jitter de los paquetes en Internet varía entre 60 y 90 ms., se utiliza un buffer de 200 ms. en el lado receptor. Conociendo que cada paquete de voz encriptada tiene un intervalo de 59 ms, que contiene dos tramas G723, resulta en un buffer de capacidad de hasta 3 paquetes de voz (6 tramas G723). De esta manera, el jitter máximo al que se puede hacer frente es de 177 ms., valor que se encuentra por encima de los 90 ms. observados en la tabla 4.16. Además, en las figuras 4.10, 4.12, 4.14, 4.16, 4.18 y 4.20, se observa diferencias en la llegada de paquetes dentro de Internet.

Caso D1: ancho de banda de voz utilizando el codificador GSM con la sobrecarga de cabeceras

	codificador	**codificador gsm + algoritmo de cifrado**				
	GSM	**TripleDES**	**IDEA**	**Blowfish**	**AES**	**CAST**
Voice Payload (bytes)	99	99	99	99	99	99
Cabecera RTP (bytes)	14	14	14	14	14	14
Cabecera UDP (bytes)	8	8	8	8	8	8
Cabecera IP (bytes)	20	20	20	20	20	20
Point-to-Point	2	2	2	2	2	2
PPP-over-Ethernet Session (bytes)	6	6	6	6	6	6
Ethernet II	14	14	14	14	14	14
Total Paquete	161	161	161	161	161	161
Packet count	170	169	171	165	171	164
Avg. Packets/sec	16.875	16.738	16.773	16.769	17.329	16.830
Avg. Packet size (bytes)	161.00	161.00	161.00	161.00	161.00	161.00
Avg. Bytes/sec	2716.853	2694.883	2700.512	2699.881	2790.025	2709.642
BW Half-Duplex (Kbps)	21.734	21.559	21.604	21.599	22.320	21.677

Tabla 4.17: ancho de banda del sistema utilizando el codificador GSM en Internet.

Análisis de la tabla 4.17

Siguiendo las mismas consideraciones que en la tabla 4.6 y considerando que el payload ocupa 99 bytes, entonces tenemos tres tramas GSM dentro de un paquete de voz. Por lo que se necesitará 60 ms para su empaquetamiento.

El tiempo de procesamiento consumido por el empaquetamiento de la voz cifrada utilizando el codificador GSM se muestra a continuación:

	Avg. Packets/sec	s/paquete
TripleDES	16.738	0.059744294 s
IDEA	16.773	0.059619626 s
Blowfish	16.769	0.059633848 s
AES	17.329	0.057706734 s
CAST	16.830	0.059417706 s

Tabla 4.18: tiempo de empaquetamiento de la voz cifrada utilizando el codificador GSM en Internet.

Los resultados correspondientes al tiempo de empaquetamiento utilizando los diferentes algoritmos criptográficos (tabla 4.18) concuerdan con el tiempo consumido por el empaquetamiento de dos tramas GSM equivalentes a 60 ms.

Caso D2: ancho de banda de voz utilizando el codificador GSM con compresión de cabeceras (IP Header + UDP Header + RTP Header)

	codificador	codificador gsm + algoritmo de cifrado				
	GSM	TripleDES	IDEA	Blowfish	AES	CAST
Voice Payload (bytes)	99	99	99	99	99	99
Compresión cabeceras	4	4	4	4	4	4
Point-to-Point	2	2	2	2	2	2
PPP-over-Ethernet Session (bytes)	6	6	6	6	6	6
Ethernet II	14	14	14	14	14	14
Total Paquete	125	125	125	125	125	125
Packet count	170	169	171	165	171	164
Avg. Packets/sec	16.875	16.738	16.773	16.769	17.329	16.830
Avg. Packet size (bytes)	125.00	125.00	125.00	125.00	125.00	125.00
Avg. Bytes/sec	2109.375	2092.25	2096.625	2096.125	2166.125	2103.75
BW Half-Duplex (Kbps)	16.875	16.738	16.773	16.769	17.329	16.830

Tabla 4.19: análisis del ancho de banda del sistema considerando la compresión de cabeceras utilizando el codificador GSM en Internet.

Caso D3: ancho de banda de voz utilizando el codificador GSM considerando el payload

	codificador	codificador gsm + algoritmo de cifrado				
	GSM	**TripleDES**	**IDEA**	**Blowfish**	**AES**	**CAST**
Voice Payload (bytes)	99	99	99	99	99	99
Packet count	170	169	171	165	171	164
Avg. Packets/sec	16.875	16.738	16.773	16.769	17.329	16.830
Avg. Packet size (bytes)	99.00	99.00	99.00	99.00	99.00	99.00
Avg. Bytes/sec	1670.625	1657.062	1660.527	1660.131	1715.571	1666.17
BW Half-Duplex (Kbps)	13.365	13.256	13.284	13.281	13.724	13.329

Tabla 4.20: análisis del ancho de banda que consume el payload utilizando el codificador GSM en Internet.

La tabla 4.21 muestra un resumen del ancho de banda consumido por el empaquetamiento de la voz cifrado con cabeceras, con compresión de cabeceras y sin cabeceras.

	con cabeceras	**compresión cabeceras**	**sólo payload**
TripleDES	21.559	16.738	13.256
IDEA	21.604	16.773	13.284
Blowfish	21.599	16.769	13.281
AES	22.320	17.329	13.724
CAST	21.677	16.830	13.329

Tabla 4.21: utilización del ancho de banda de los diferentes algoritmos criptográficos utilizando el codificador GSM en Internet.

Observamos que sólo el procesamiento del payload ocupa el 61% del ancho de banda total, y que 39% se utiliza para el empaquetamiento de cabeceras. Si utilizáramos algunos de los estándares de compresión de cabeceras (como RFCs 3545 - CRTP Enhanced Compressed RTP - y RFC 3095 – ROHC Robust Header Compression - permiten la compresión de cabeceras IP, UPD y RTP), se lograría disminuir en un 23% el consumo del ancho de banda. Estos mecanismos suprimen la información redundante que se transmite repetidamente en una sesión RTP y consiguen reducir la sobrecarga de cabeceras a 2 ó 4 octetos.

Evaluación del Delay, del Jitter y del retardo de paquetes (voz cifrada) utilizando el codificador GSM

	codificador	codificador gsm + algoritmo de cifrado				
	GSM	TripleDES	IDEA	Blowfish	AES	CAST
Packet received	170	169	171	165	171	164
Packet expected	170	169	171	165	171	164
Lost RTP Packet	0	0	0	0	0	0
Max delay (ms)	176.178	119.254	211.891	465.643	410.719	658.320
Max Jitter (ms)	64.980	62.238	69.814	84.791	81.311	111.744

Tabla 4.22: máximo delay, jitter y número de pérdidas de los paquetes cifrados utilizando el codificador GSM en Internet.

Gráficas del delay y del jitter observados en la transmisión de paquetes

Considerando el uso del **codificador GSM sin ningún tipo de algoritmo de cifrado,** se obtuvo 170 muestras de paquetes en el lapso de 10 segundos. Asimismo, las distribuciones del retardo y del jitter con respecto al número de paquetes fueron las siguientes:

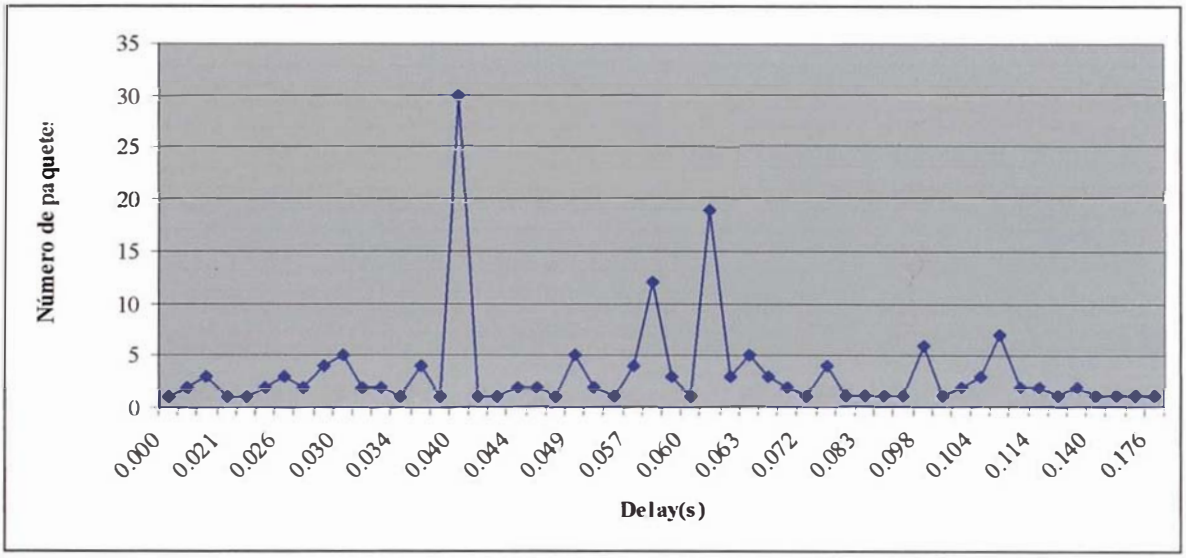

Figura 4.21: distribución de retardo entre el número de paquetes utilizando solamente el algoritmo GSM sin ningún tipo de encriptamiento en Internet.

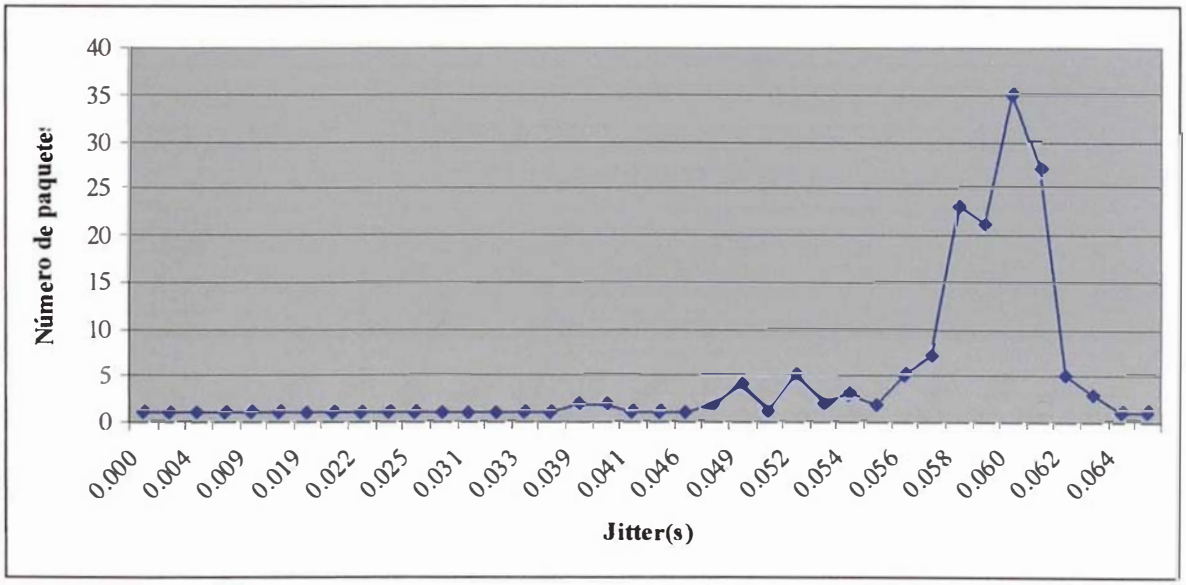

Figura 4.22: distribución del jitter entre el número de paquetes utilizando solamente el algoritmo GSM sin ningún tipo de encriptamiento en Internet.

Considerando el uso del **codificador GSM más el algoritmo de cifrado TripleDES,** se obtuvo 169 paquetes en el lapso de 10 segundos. Del mismo modo, las distribuciones del retardo y del jitter con respecto al número de paquetes fueron las siguientes:

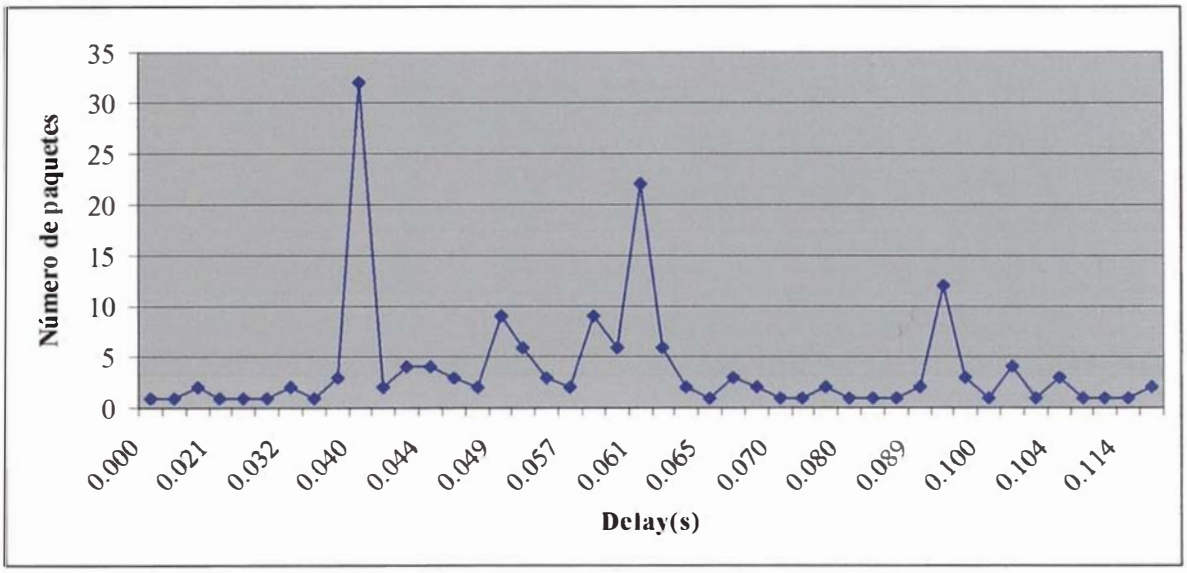

Figura 4.23: distribución del retardo entre el número de paquetes utilizando el codificador GSM más el algoritmo de cifrado TripleDES en Internet.

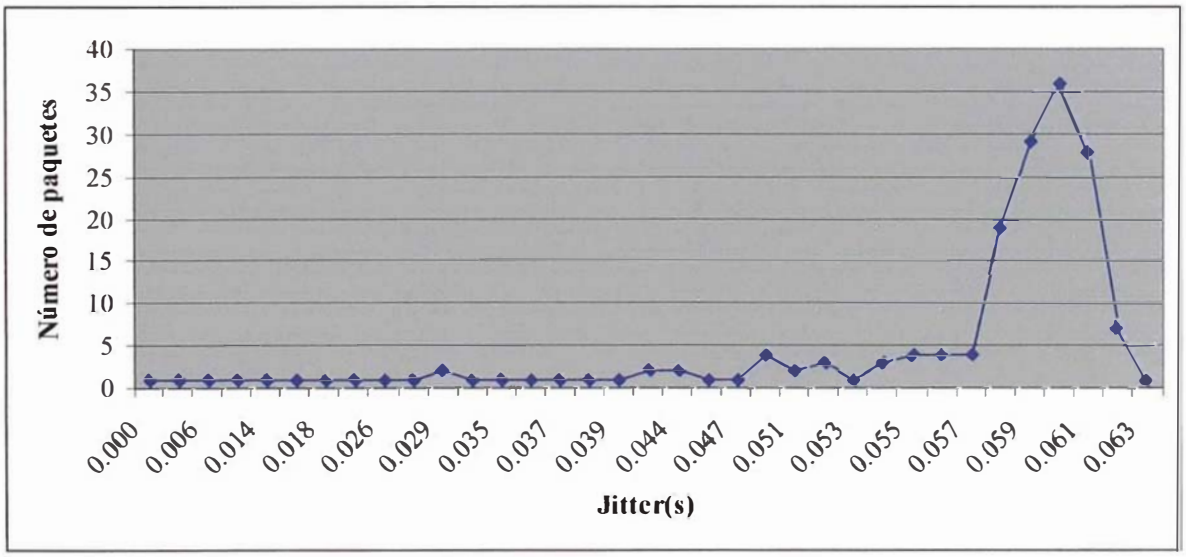

Figura 4.24: distribución del jitter entre el número de paquetes utilizando el codificador GSM más el algoritmo de cifrado TripleDES en Internet.

Considerando el uso del **codificador GSM más el algoritmo de cifrado IDEA,** se obtuvo 171 paquetes en el lapso de 10 segundos. Asimismo, las distribuciones del retardo y del jitter con respecto al número de paquetes fueron las siguientes:

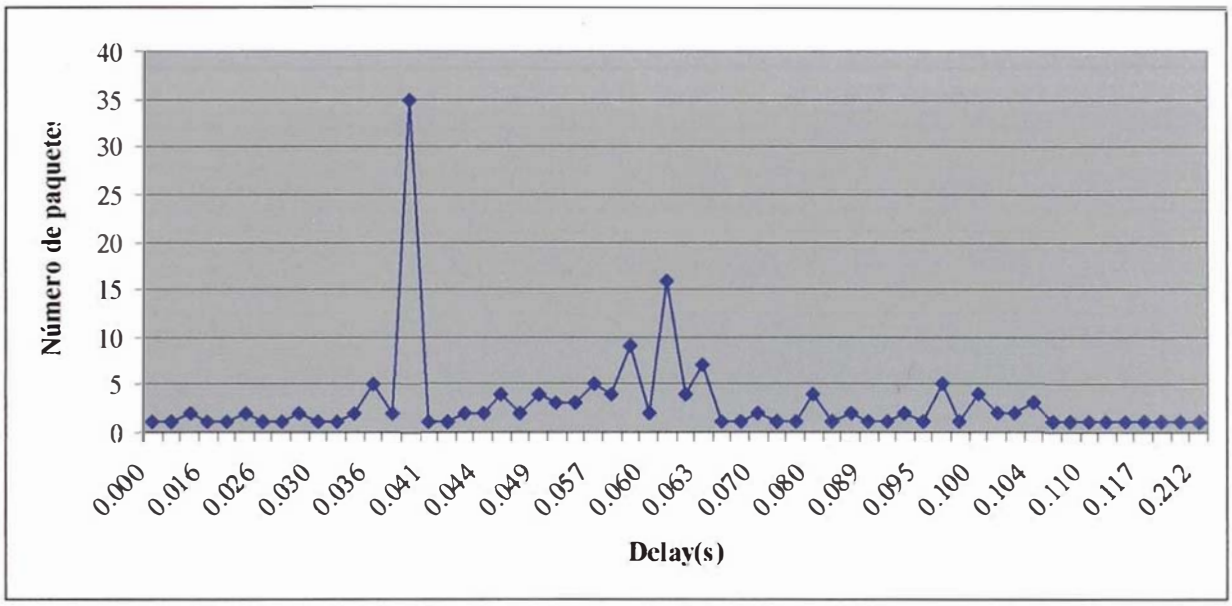

Figura 4.25: distribución del retardo entre el número de paquetes utilizando el codificador GSM más el algoritmo de cifrado IDEA en Internet.

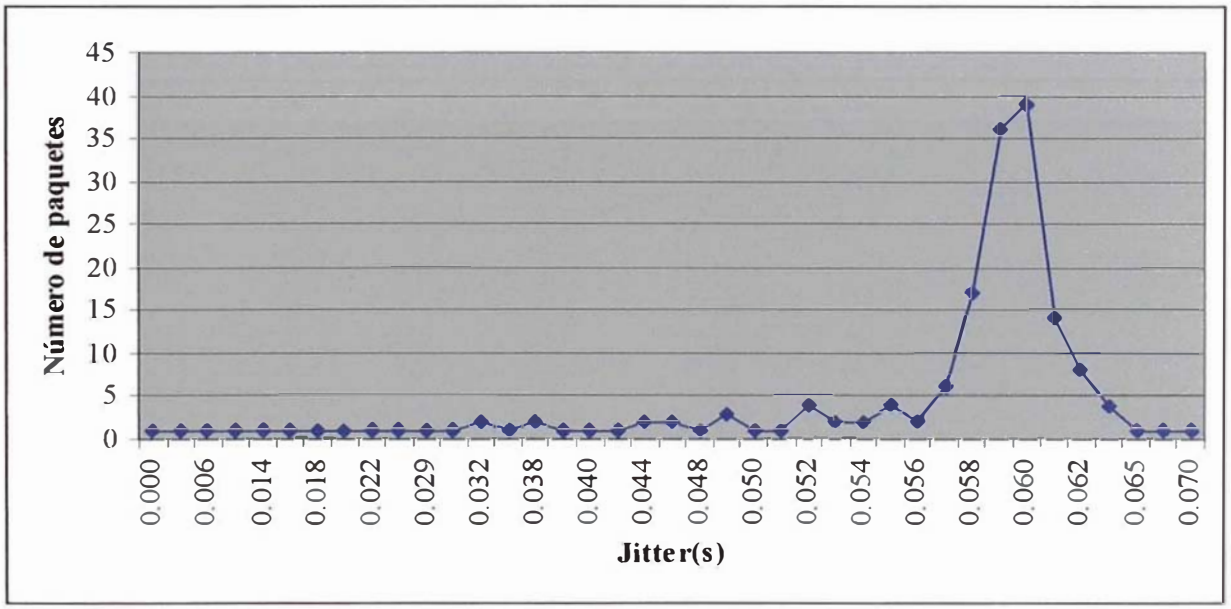

Figura 4.26: distribución del jitter entre el número de paquetes utilizando el codificador GSM más el algoritmo de cifrado IDEA en Internet.

Considerando el uso del **codificador GSM más el algoritmo de cifrado Blowfish,** se obtuvo 165 paquetes en el lapso de 10 segundos. Asimismo, las distribuciones del retardo y del jitter con respecto al número de paquetes fueron las siguientes:

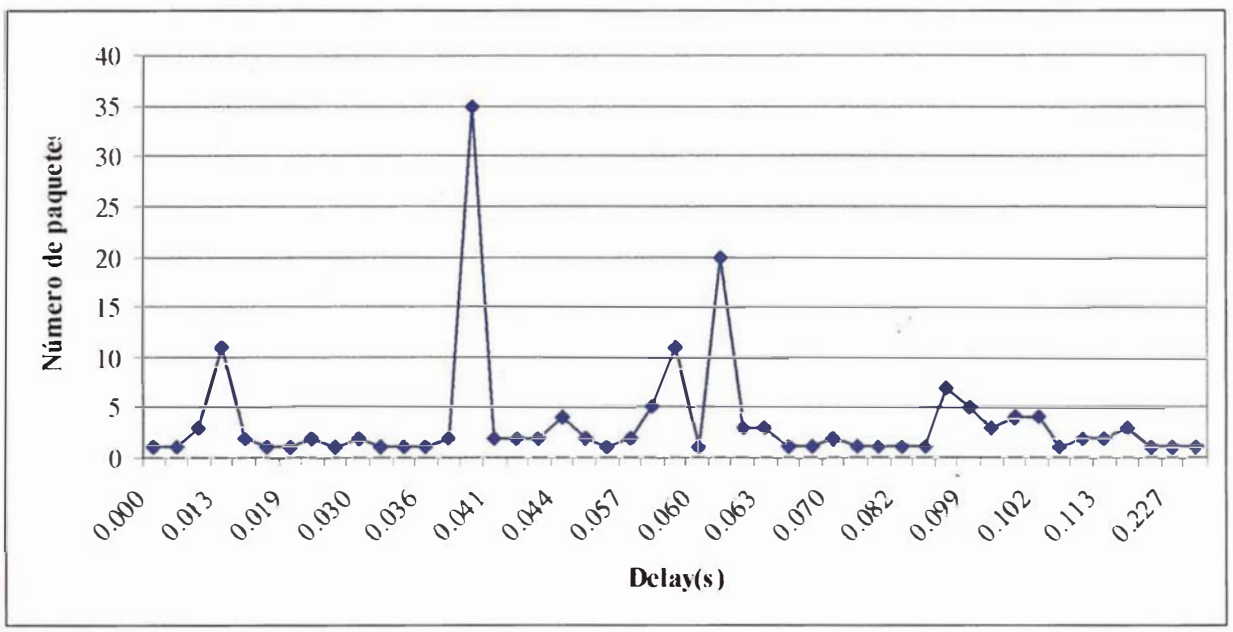

Figura 4.27: distribución del retardo entre el número de paquetes utilizando el codificador GSM más el algoritmo de cifrado Blowfish en Internet.

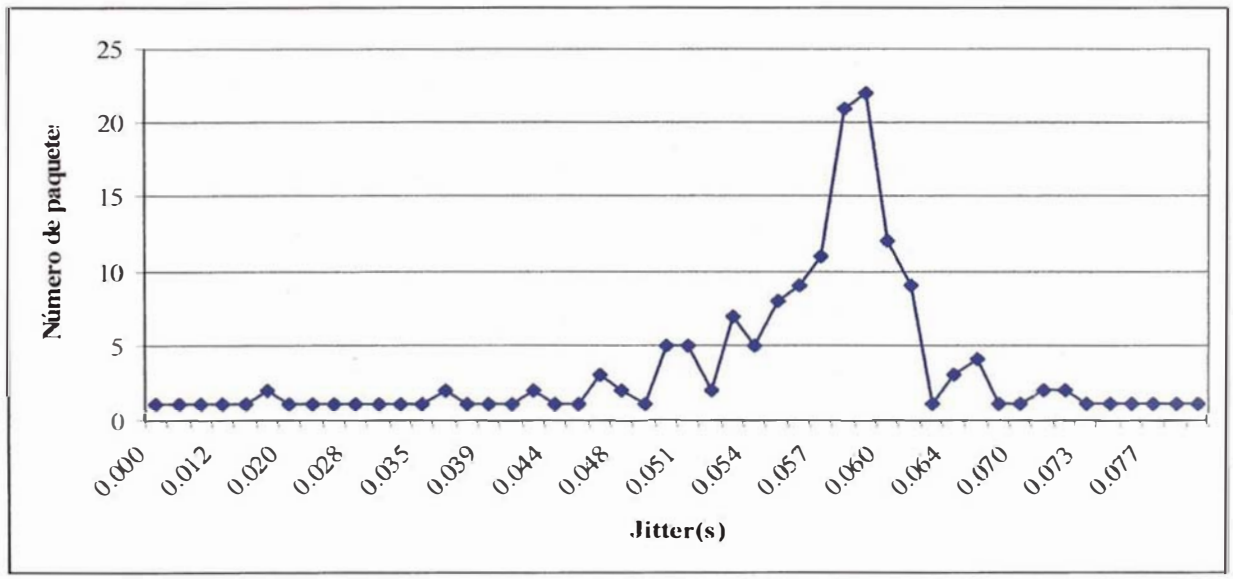

Figura 4.28: distribución del jitter entre el número de paquetes utilizando el codificador GSM más el algoritmo de cifrado Blowfish en Internet.

Considerando el uso del **codificador GSM más el algoritmo de cifrado AES,** se obtuvo 171 paquetes en el lapso de 10 segundos. Por otro lado, las distribuciones del retardo y del jitter con respecto al número de paquetes fueron las siguientes:

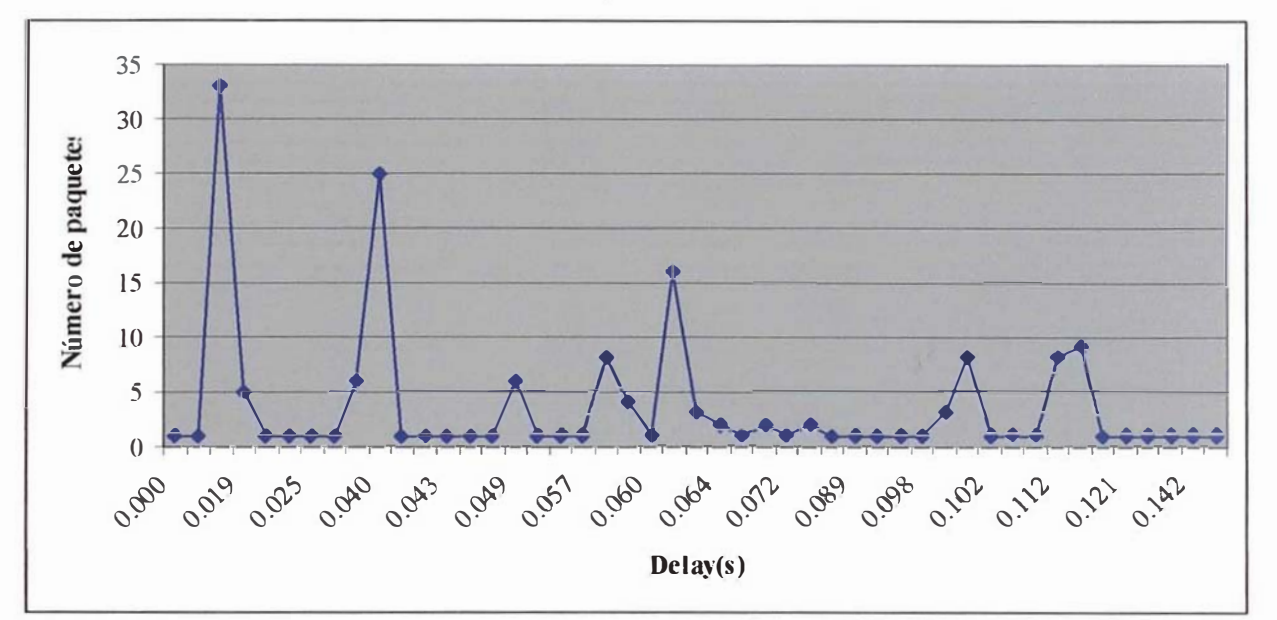

Figura 4.29: distribución del retardo entre el número de paquetes utilizando el codificador GSM más el algoritmo de cifrado AES en Internet.

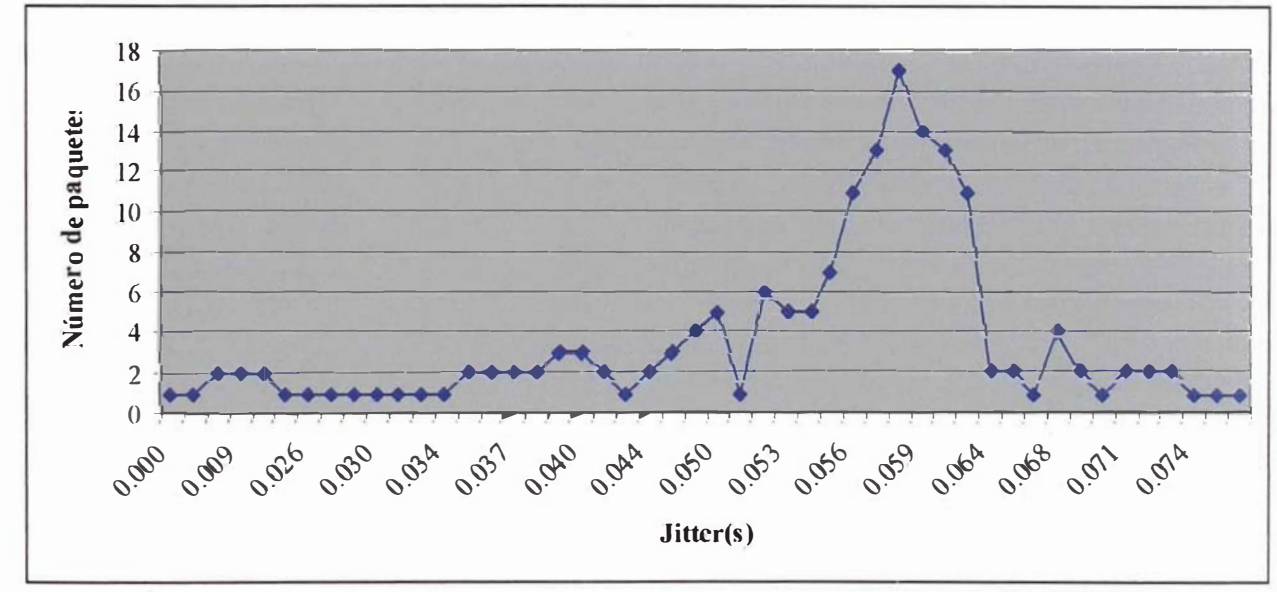

Figura 4.30: distribución del jitter entre el número de paquetes utilizando el codificador GSM más el algoritmo de cifrado AES en Internet.

Considerando el uso del **codificador GSM más el algoritmo de cifrado CAST5,** se obtuvo 164 paquetes en el lapso de 10 segundos. Asimismo, las distribuciones del retardo y del jitter con respecto al número de paquetes fueron las siguientes:

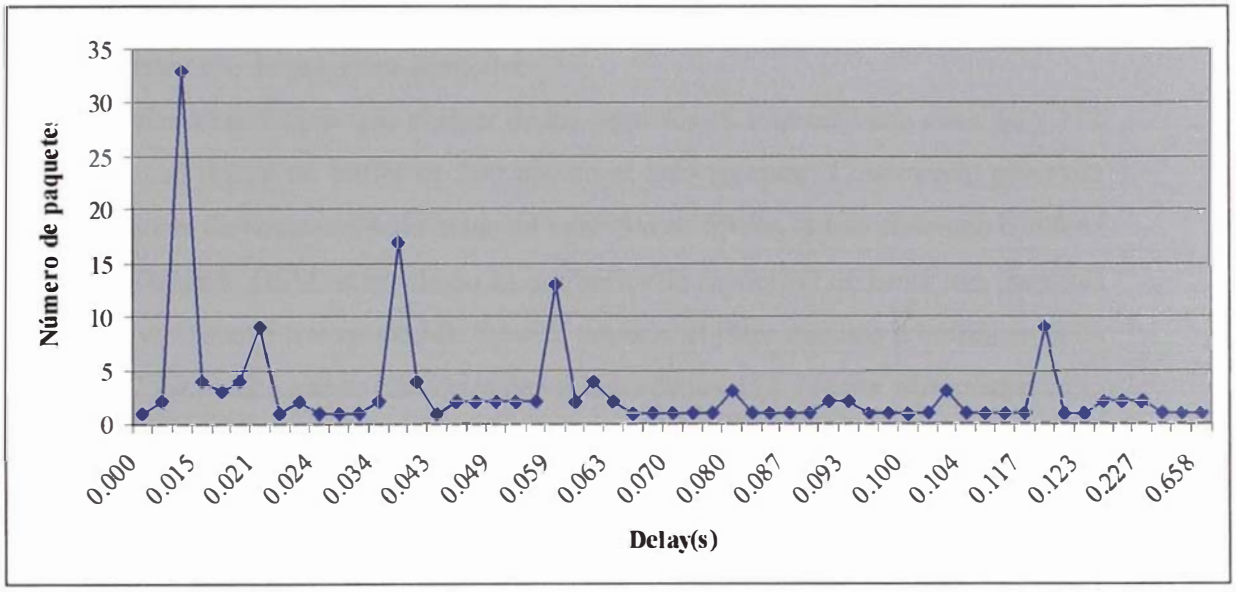

Figura 4.31: distribución del retardo entre el número de paquetes utilizando el codificador GSM más el algoritmo de cifrado CAST5 en Internet.

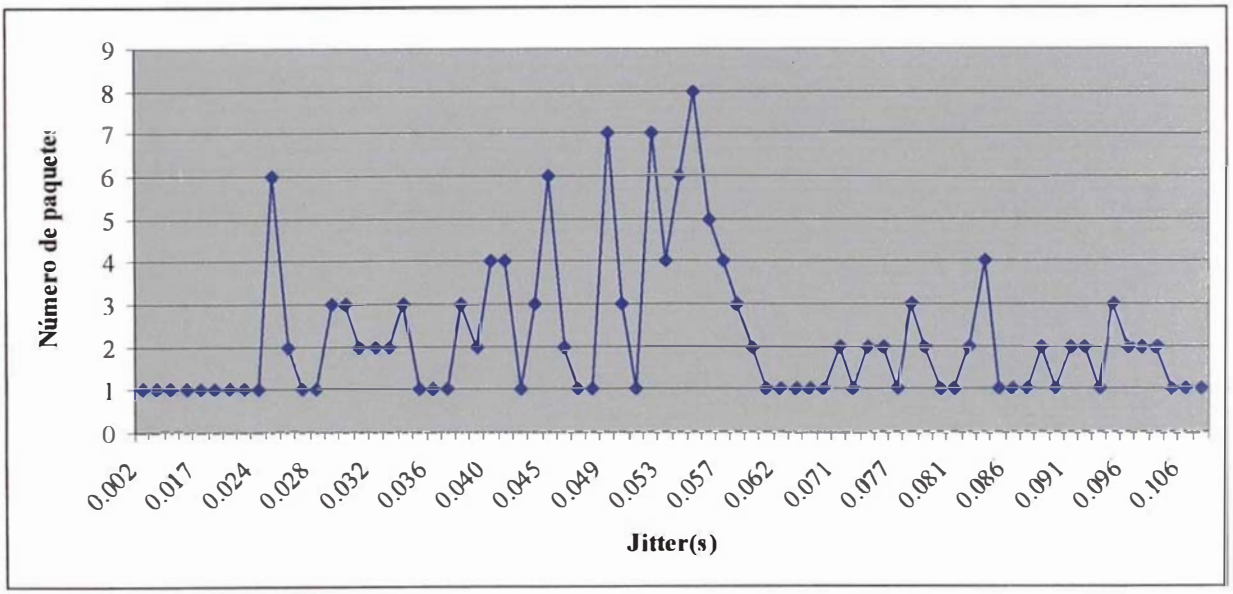

Figura 4.32: distribución del jitter entre el número de paquetes utilizando el codificador GSM más el algoritmo de cifrado CAST5 en Internet.

Análisis de los resultados de la tabla 4.22 y de las figuras del 4.21-4.32

1. De acuerdo a los resultados de la tabla 4.22 y de las figuras 4.21, 4.23, 4.25, 4.27, 4.29 y 4.31, observamos que la mayor cantidad de paquetes tienen un delay de hasta 120 ms. Asimismo, pocos de estos exceden esta cantidad, pues llegan hasta 658.320 ms.
2. Existen 0% de paquetes perdidos.
3. Teniendo en cuenta que el jitter de los paquetes en Internet varía entre 60 y 112 ms., se utiliza un buffer de 200 ms. en el lado receptor. Conociendo que cada paquete de voz encriptada tiene un intervalo de 59 ms, y que cada uno contiene dos tramas GSM, el resultado es un buffer de capacidad de hasta tres paquetes de voz (nueve tramas GSM). De esta manera, el jitter máximo a enfrentar es de 177 ms., valor que se encuentra por encima de los 111.744 ms. observados en la tabla 4.22. Además, se observa (figuras 4.22, 4.24, 4.26, 4.28, 4.30 y 4.32) una gran diferencia en la llegada de paquetes dentro de Internet.

4.3 Conclusiones de las evaluaciones

Después de la evaluación del sistema propuesto en ambos escenarios se ha obtenido las siguientes conclusiones:

- El retardo mínimo de extremo-a-extremo del sistema propuesto es de 118 ms.
- El tiempo de procesamiento de los diferentes algoritmos de encriptamientos es mínimo, ya que comparando los diferentes resultados con y sin encriptamiento son similares.
- El tráfico en Internet es variable de acuerdo a la congestión. De este modo, se ha apreciado variaciones en las mediciones realizadas del retardo y del jitter de acuerdo al día y a la hora en que fueron tomadas.
- Se logra disminuir la pérdida de paquetes con el uso de un buffer que ayuda a solucionar las varianzas del retardo (jitter). Así, se obtuvo 0% de pérdidas, ya que se colocó un buffer de 200 ms. Del mismo modo, si tomamos en cuenta que el intervalo de los paquetes es de 59 ms.; entonces, los que lleguen con un jitter mayor a 177 se desecharán ya que su reproducción estará a destiempo.
- Los anchos de banda necesarios para el sistema propuesto en Internet son:
  30 Kbps con la codificación G723.1; y 44 Kbps con GSM.

Capitulo 5

Conclusiones y recomendaciones

5.1 Conclusiones

Las conclusiones de este trabajo, basados en las evaluaciones y el desarrollo del "Sistema Seguro de Transmisión de Voz mediante Internet", se resumen en los siguientes puntos:

1. La combinación de algoritmos criptográficos permite la creación de una infraestructura de seguridad para cualquier sistema de comunicación. Ejemplo de ello, lo tenemos en el protocolo de comunicación construido en base al algoritmo asimétrico RSA o en el protocolo SSL. Ambos utilizan algoritmos simétricos para el encriptamiento de los datos transmitidos, algoritmos MAC para la autentificación de los mensajes y algoritmos de clave pública para la negociación de las claves de sesión.
2. El uso de codificadores de voz como G723.1 y GSM son recomendables para aplicaciones de voz en Internet por su demanda de poco ancho de banda y porque cuentan con buenos detectores de actividad que disminuiyen el retardo y la pérdida de paquetes (en las transmisiones y, sobretodo, en redes públicas congestionadas).
3. El encriptamiento debe de producirse después de la compresión de voz, ya que produce datos completamente aleatorios que harían que aplicándose en sentido inverso no pudieran encontrar patrones similares de voz.
4. Debe existir un balance entre el número de tramas de los codificadores de voz y la sobrecarga de las cabeceras en el paquete RTP. Esto con la finalidad de transportarlas por la red. Así, si se tiene una sola trama en el paquete, la mayor cantidad de bytes serán representados por la sobrecarga de cabeceras dando como resultado un mayor consumo de ancho de banda; y si, por el contrario, tenemos demasiadas tramas de los codificadores en un paquete RTP, la sobrecarga de cabeceras será menor pero existirá mayor retardo, por lo que, la pérdida de paquetes será mas notoria.

5. El API de JMF (Java Media Framework) para Java ofrece grandes ventajas y beneficios para el desarrollo de aplicaciones multimedia en Java, ya que brinda la facilidad para la inclusión de nuevos codecs, como, en nuestro caso, la inclusión de un codec que realiza las funciones del cifrado de la voz.

5. 2 Recomendaciones para trabajos futuros

Después de haber concluido el "Desarrollo de un Sistema Seguro de Transmisión de Voz mediante Internet" y haber realizado las pruebas en un ambiente real como Internet, enumeraremos algunas recomendaciones:

1. Todo sistema de comunicación que brinde seguridad debe incorporar certificados digitales para verificar la autenticidad de los usuarios antes de realizar algún tipo de negociación. Es decir, debe ir acompañado de una infraestructura de clave pública PKI, que se encarge de distribuir, administrar y denegar certificados a los usuarios.
2. La infraestructura de seguridad propuesta en nuestra tesis se podría mejorar con dos nuevos componentes, el protocolo Secure Real-time Transport Protocol (SRTP) y el Multimedia Internet KEYing (MIKEY). El primero se encargaría del encriptamiento e integridad de la voz digitalizada; mientras que el MIKEY se preocuparía del intercambio de las claves de sesión entre ambos participantes de la comunicación.
3. La incorporación del protocolo SIP para el intercambio de mensajes de señalización entre los participantes.
4. En enlaces de serie de baja velocidad (< 2 Mbps), se debe utilizar la compresión de cabeceras RTP para reducir la sobrecarga de la red. Esta comprimirá el encabezado RTP/UDP/IP de los 40 bytes usuales a 2-4 bytes (RFCs 3545, CRTP). De este modo, el resultado será un considerable ahorro de ancho de banda porque la verdadera "carga útil" dentro del paquete puede ser bastante pequeña, y el encabezado consumirá una porción significativa del ancho de banda.

Referencia bibliográfica

[1] Sun Developer Network. Java Media Framework (JMF). Sun Microsytems, Inc. 2005. [Consulta: Enero 2004]. http://java.sun.com/products/java-media/jmf/

[2] Java™ Media Framework API Guide. Sun Microsystems Inc. November 19, 1999. [Consulta: 22 de Setiembre de 2003]. http://java.sun.com/products/java-media/jmf/index.html

[3] Java ™ Secure Socket Extension (JSSE) Reference Guide for the Java ™ 2 SDK. Standard Edition. v.1.4.2.. Sun Microsystems, Inc. 1998-2003. [Consulta: 07 de Diciembre de 2003]. http://java.sun.com/j2se/1.4.2

[4] Java ™ Cryptography Architecture API Specification & Reference . Sun Microsystems, Inc. 6 Diciembre 1999. [Consulta: 09 de Octubre de 2003]. http://java.sun.com/products/jdk/1.2/docs/guide/security/CryptoSpec.html

[5] Java ™ Cryptography Extension (JCE) Reference Guide for the Java ™ 2 SDK Standard Edition, v 1.4. Sun Microsystems, Inc. Enero 10, 2002. [Consulta: 09 de Octubre de 2003]. http://java.sun.com/products/jce/index.jsp

[6] DeveloperWorks. Java Media Framework Basics. IBM. May 2002.[Consulta: 12 de Setiembre 2003]. http://www-130.ibm.com/developerworks/

[7] DeveloperWorks. Crypto basics. IBM. July, 2002. [Consulta: 12 de Octubre de 2003]. http://www-130.ibm.com/developerworks/.

[8] Schulzrinne, H. et al. RTP: A tranport Protocol for Real-Time Applications, RFC 3550. Columbia University. July 2003. [Consulta: Setiembre 2003]. http://www.rfc-editor.org/

[9] Schulzrinne, H y S. Casner. RTP Profile for Audio and Video Conferences with Minimal Control, RFC 3551. [Consulta: Setiembre 2003]. http://www.rfc-editor.org/

[10] Baugher, M. y D. McGrew.The Secure Real-time Transport Protocol (SRTP), RFC 3711. March 2004. [Consulta: Junio 2004]. http://www.rfc-editor.org/

[11] Arrko, J.et al. MIKEY: Multimedia Internet KEYing. IETF Work in Progress. December 2003. [Consulta: Junio 2004].http://www.ietf.org/internet-drafts/draft-ietf-msec-mikey-06.txt

[12] Cryptix JCE. Cryptix™. 26 de Agosto 2001. [Consulta: 15 de Enero de 2004] .http://www.cryptix.org/products/jce/index.html

[13] Legion of the Bouncy Castle. Bouncy Castle. [Consulta: 30 de Enero de 2004]. http://www.bouncycastle.org/.

[14] Jaworski, Jamie y Paul J. Perrone. Edición especial seguridad en Java. Pearson Educación: Madrid, 2001.

[15] Rodriguez Adolfo et al. TCP/IP Tutorial and Technical Overview. IBM. August 2001. [Consulta: 20 de Setiembre de 2003]. http://www.ibm.redbooks.

[16] Cisco Avvid Network Infraestructure Enterprise Quality of Service Design. Cisco Systems. August 2002.[Consulta: 20 de Setiembre de 2003]. www.cisco.com/application/pdf/en/us/guest/netsol/ns17/c649/ccmigration_09186a00800d67ed.pdf.

[17] Adams, C.RFC 2144: The CAST-128 Encryption Algorithm. Entrust Technoligies .May 1997. [Consulta : Octubre 2003]. http://www.rfc-editor.org/

[18] Dworkin, M.Recommendation for Block Cipher Modes of Operation: Methods and Techniques. NIST Special Publication 800-38A. December 2001. [Consulta: Setiembre 2003]. http://csrc.nist.gov/publications/nistpubs/

[19] Bormann,C. et al. RObust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed, RFC 3095, July 2003.[Consulta: Junio 2004]. http://www.rfc-editor.org/

[20] Koren, T. et al. Enhanced Compressed RTP (CRTP) for Links with High Delay, Packet Loss and Reordering, RFC 3545. July 2003. [Consulta: Julio 2004]. http://www.rfc-editor.org/

[21] Goralski, Walter J y Matthew C. Kolon. IP Telephony. México: McGraw-Hill, 2000.

[22] Davidson, Jonathan y James Peters. Fundamentos de voz sobre IP. Madrid: Pearson Educación, 2001.

[23] Keagy, Scout. Integración de redes de voz y datos. Madrid: Pearson Educación, 2001.

[24] Fúster Sabater, Amparo y Dolores de la Guía Martinez. Técnicas criptográficas de protección de datos. 2da. edición. México: Alfaomega, 2001.

[25] Liesenborgs, Jori. Voice Over IP in network virtual environments.Tesis Doctoral. Universiteit Maastricht, May 2000. [Consulta: 20 de Abril de 2002] http://lumumba.luc.ac.be/jori/thesis/onlinethesis/contents.html

[26] Pais Suaréz, Francisco. Estudio del algoritmo de cifrado Rijndael. Comparativa entre los algoritmos de cifrado DES y Rijndael, Proyecto de Fin de Carrera de Ingeniería Técnica en Informática de Gestión. Universidade da Coruña. Febrero 2003. [Consulta: Octubre 2003]. http://research.tic.udc.es/scg/proyect/rinjdael1/proyecto-rinjdael.pdf

[27] Savards, John. A Cryptographic Compendium. 2000. [Consulta: 20 Abril de 2002] http://www.hypermaths.org/quadibloc/crypto/jscrypt.htm

[28] Specification for the Advanced Encryption Standard (AES). Federal Information Processing Standards Publication 197. November 26, 2001.

[29] Schenier, Bruce. The Blowfish Encryption Algorithm. [Consulta: Octubre 2003].http://www.schneier.com/blowfish.html.

[30] Pons Martorell, Manuel. Criptología. Escola Universitària Politècnica de Mataró. Departament de Telecomunicacions. Febrero 2001. [Consulta: Octubre 2003]. http://www.criptored.upm.es/guiateoria/gt_m013a.htm

[31] Lucena López, Manuel Jose. Criptografía y Seguridad en Computadoras.Tercera Edición (Versión 2.10). Mayo del 2003. [Consulta: Octubre 2003]. http://rinconprog.metropoliglobal.com/Apuntes/Criptografia.pdf

[32] Menezes, A. et al. Handbook of Applied Cryptography. CRC Press. 1997. [Consulta: Setiembre 2002]. http://www.cacr.math.uwaterloo.ca/hac/

[33] Fuica Espinoza, Soledad y Johnny Reveco Arias. Codificación del Canal de Voz. Curso: Comunicaciones de datos.2001.

[34] Kostas, Thomas J. y S. Botella. 3COM. Real-Time Voice Over Packet-Switched Networks. IEEE Networks. January/February 1998.[Consulta: Agosto 2003]. http://www.borella.net/mikc/academics/voip.pdf

[35] Abreu Sernández, Victoria. Codificación multipulso a tasas variables aplicada a la transmisión fiable de voz sobre redes de datos. Resumen de la Tesis. [Consulta: Agosto 2003].

[36] A Matthews, Tim. Suggestions for Random Number Generation in Software. RSA Laboratorios Bulletin. News and advine from RSA Laboratorios. Number 1. January 22, 1996. [Consulta: Octubre 2003]. http://islab.oregonstate.edu/koc/ece575/rsalabs/bulletn1.pdf

[37] Cheol-Lee. Study of Speech Compression and GSM06.10 RPE-LTP. Course: Data Compression. Term Project. April 24, 1997.

[38] Reyes, Benjamin y Kavitha Swaminathan. Secure Sockets Layer Protocol. University of Maryland. Course: Network Security. Project Paper. August, 2003.

[39] Hui, S.C., y S. Foo. Towards a standards-based Internet telephony system. Nanyang Technological Universtiy, School of Applied Science. 1998.

[40] Jeff Schmidt. Guia Avanzada Seguridad en Microsoft Windows 2000. Pearson Educación: Madrid, 2001.

[41] Besacier, L.et al. GSM Speech Coding and Speaker Recognition . University of Neuchâtel, Institute of Microtechnology (Neuchâtel - Switzerland). University Joseph Fourier (Grenoble-France). [Consulta: Octubre 2003]

[42] Gutierrez, J et al. Tema 3 - Cifrado de la información. Universidad del Pais Vasco. Facultad de Informática. Ingeniería en Informática. Seguridad Informática. [Consulta: Setiembre 2003].http://www.sc.ehu.es/jiwibmaj/inicial.htm.

[43] PGP for Personal Privacy Para Macintosh. Guia de Usuario. Pretty Good Privacy Inc. 1997. [Consulta: Agosto 2002].
http://www.mirrors.wiretapped.net/security/cryptography/apps/pgp/PGP/5.5/docs/spanish/pgp553i-macintosh-spanish.pdf

[44] Rinehimer, D. y D. Wilson. Summary of B. Schneier's "Description of New Variable-Length key, 64-Bit Block Cipher (Blowfish). 2003.[Consulta: Noviembre 2003]. http://personal.denison.edu/~bressoud/cs402-f03/summary-question/blowfish-summary.pdf

[45] Mengual, L. et al. Introducción de servicios de seguridad en un sistema de Audioconferencia implementado con Java Media Framework. Dpto. de Lenguajes y sistemas Informáticos e Ingeniería del Software. Facultad de Informática. Madrid.[Consulta: Setiembre 2003]
http://pegaso.ls.fi.upm.es/~lmengual/articulos/art_18.pdf

[46] Mengual, L. et al. Desarrollo de un Codec JPEG seguro implementado con Java Media Framework. [Consulta: Setiembre 2003].
http://pegaso.ls.fi.upm.es/~lmengual/articulos/art_20.pdf

[47] AudioCodecs. Securing Communications over Packet Networks. [Consulta: Setiembre 2003].
http://www.audiocodes.com/objects/Securing_Communications_WP.pdf

[48] Laborda Duque et al. Redes Fijas (VoIP).

[49] 1002. Compresión Vocal, Audio y Sonido. Acerca de los compresores vocales para señales de telefonía, sonido y audio de alta calidad. [Consulta: Noviembre 2003].
http://www.rares.com.ar/interest.htm

[50] IP-Telephony (protocolos). [Consulta: Diciembre 2003]. http://www.rares.com.ar/interest.htm

[51] D. Richard Kuhn et al. Security Considerations for Voice Over IP Systems. Recommendations of the National Institute of Standards and Technology. Abril 2004. [Consulta: Setiembre 2003]. http://csrc.nist.gov/publications/nistpubs/

[52] Mendoza Díaz, María Concepción. Protocolos de Seguridad e Instrumentación de IPSec en Escenarios Experimentales de Internet 2 en México. Centro de Investigación Científica y de Educación Superior de Ensenada.Tesis.2002. [Consulta: Junio 2003]. http://seguridad.internet2.ulsa.mx/publications/2002/tesis_comedi.pdf

[53] Talens-Oliag, Sergio. Introducción a la Criptografía.[Consulta: Octubre 2003] http://www.infocentre.gva.es

[54] García Lobo, Belén. "Algoritmos criptográficos".

[55] Gatliff, Bill. Encrypting data with the Blowfish algorithm. D&R Industry Articles. Julio, 2003.[Consulta: Marzo 2004]. http://www.us.design-reuse.com/articles/article5922.html

[56] WorldHistory.com. Block Ciphers. [Consulta: Agosto 2003]. http://www.worldhistory.com/wiki/B/Block-cipher.htm

[57] Morales Vásquez, José María. SSL, Secure Sockets Layer y Otros Protocolos Seguros para el Comercio Electrónico. [Consulta: Febrero 2004]. http://jo.morales0002.eresmas.net/fencasa.html

[58] Reyes, B et al. Secure Sockets Layer Protocol. University of MaryLand. ENTS-650 Network Security. Agosto,2003

[59] SSL: Theory and Practice. Zeus Technology. June, 2000.

[60] Huidobro Moya, José Manuel. Redes y Servicios de Telecomunicaciones. 3ra ed. Madrid: Paraninfo, 2001.

[61] IEEE 802.20 Working Group on Mobile Broadband Wireless Access. VoIP Traffic Models for 802.20 System Performance Evaluation.IEEE.

[62] Montañana, Rogelio. Seguridad . Asignatura Redes de Ordenadores. [Consulta: Febrero 2004]. http://www.uv.es/~montanan/redes/