Mecanismo Multicast para la Generación y Distribución de Llaves para Sistemas Distribuidos Multimedia en Tiempo Real

por

Carlos Rojas Sánchez

Tesis sometida como requisito parcial para obtener el grado de

MAESTRO EN CIENCIAS EN LA ESPECIALIDAD DE CIENCIAS COMPUTACIONALES

en el

Instituto Nacional de Astrofísica, Óptica y Electrónica

Febrero 2008
Tonantzintla, Puebla

Supervisada por:

Dr. Saúl Eduardo Pomares Hernández, INAOE

Dr. Gustavo Rodríguez Gómez, INAOE

Resumen

La Internet ha experimentado un crecimiento considerable en los últimos años en diversos ámbitos tales como servicios ofrecidos, número de dispositivos así como usuarios interconectados. Algunas de las principales razones de este crecimiento son el incremento de infraestructura en telecomunicaciones, el incremento del ancho de banda, el incremento y a la vez bajo costo en la capacidad de cómputo así como de los dispositivos de almacenamiento en general. Por lo tanto, la Internet actual que conocemos, debido a estos grandes cambios, difiere grandemente de la Internet inicialmente concebido. Dentro de estos cambios está el uso de información multimedia, la aparición del paradigma de comunicación multicast, el empleo de Internet para aplicaciones cooperativas en tiempo real, y por último, la necesidad bajo este ambiente de una comunicación segura. Cada uno de estos cambios representa en sí un área abierta de investigación.

En este trabajo se presenta un mecanismo basado en una comunicación multicast para generar y distribuir llaves criptográficas para la transmisión de multimedia, audio y video, de forma confidencial en sistemas distribuidos en tiempo real. Este mecanismo soporta la pérdida y retraso de paquetes las cuales son características propias de la transmisión de audio y video. Para lograr lo anterior nuestro mecanismo genera tantas llaves como paquetes multimedia se envíen, logrando así cifrar cada paquete con una llave diferente. Dichas llaves tienen las siguientes características: son independientes, difíciles de romper y con un costo computacional aceptable. El mecanismo propuesto en general presenta las siguientes características: es eficiente en la generación y distribución de las llaves utilizadas, es escalable, robusto y es dinámico en el sentido que soporta la entrada y salida de participantes. Según nuestro conocimiento el mecanismo presentado en esta tesis es el primero en trabajar de manera simultánea con restricciones propias de los sistemas distribuidos, de la transmisión confidencial de multimedia en tiempo real, y por último de la comunicación multicast.

Abstract

Internet has experienced a remarkable growing last years in several environments such as offered services, device is numbers and interconnected users, some of the mean reasons are telecommunications infraestructure s increase, bandwidth increase, increase and low cost in computing capacity as storage devices in general. Therefore the present internet, due to these big changes, differs enormously from the firs internet . Inside these changes is the multimedia information usage, the multicast communication paradigm is appearance, the internet is usage for cooperative real time applications, and last, the necessity in this environment for a secure communication. Each one of these changes represents a research is open area.

This work represents a mechanism based in a multicast communication to generate and distribute cryptographic keys to transmit multimedia, audio and video in a confidential manner to distributed systems on real time. This mechanism supports the package is lost and delay, which are features that belong to audio and video transmission. To achieve the mentioned our mechanism generates as many keys as multimedia packages are sent, achieving so to encrypt each package with a different key. These keys have the following features: independent , hard to break and with computing acceptable cost. The general proposed mechanism presents the following features: efficient to generate and distribute the used keys, scalable, robust and dynamic, this means that supports the in and out of participants. Based in our knowledge, the mechanism presented in this thesis is the first to work in simultaneous form with restrictions that belong to distributed systems, real time multimedia confidential transmission, and last the multicast communication.

Agradecimientos

Agradezco de manera especial al Dr. Saúl Eduardo Pomares Hernández y al Dr. Gustavo Rodríguez Gómez por su paciencia, apoyo y enseñanzas al dirigir y supervisar este trabajo.

A mis compañeros y amigos agradezco su amistad, apoyo y alegría durante el tiempo de estudio y convivencia.

Quiero agradecer al Instituto Nacional de Astrofísica,Óptica y Electrónica el brindarme la oportunidad de superación y al Conacyt por el apoyo al otorgarme la beca durante el tiempo que realicé mis estudios de maestría.

Muy especialmente a mis padres agradezco su generosidad al darme la vida, su cariño y enseñanzas.

A mis hermanos por su comprensión y apoyo.

A Yara la motivación para realizar este trabajo.

Finalmente agradezco a Dios el estar aquí.

Índice general

Índice de cuadros

Índice de figuras

Símbolos generales

Símbolo	Significado	Descripción
n	número de *Miembros Organizadores*	Entero
m	número de *Usuarios*	Entero
i	índice de los *Miembros Organizadores* (rango [1,n])	Entero
j	índice de los *Usuarios* (rango [1,m])	Entero
x	índice de potenciales usuarios	Usuario de la internet
$\rightarrow$	Comunicación Unicast	uno a uno
$\leftrightarrow$	Comunicación Multicast	uno a muchos
H	Host/Fuente	Proceso
S_i	*Miembro organizador* i	Proceso
U_j	*Usuario* j	Proceso
U_x	usuario potencial x	Proceso
key	Llave para cifrar un paquete	Número
M	Mensaje	Texto
$U_j \rightarrow U_{j'} : M$	U_j envía un mensaje M para $U_{j'}$, $j \neq j'$	también para $\leftrightarrow$
$\{M\}_{key}$	Mensaje M cifrado con la llave key	
D_i	i^{th} *Dominio Organizador*	Conjunto de procesos
k	índice de *Usuarios* en un D (rango [1,n/m])	Entero
U_{ki}	*Usuario* k de un D_i	Proceso
KP	Llave principal o del *Dominio de Transferencia*	Paquete
KS	Llave del *Dominio de Control*	Entero
KD_i	Llave del D_i	Entero

Símbolos usados para los algoritmos de generación y distribución de llaves, basados en el algoritmo de Diffie-Hellman extendido

Símbolo	Significado	Descripción
α	Número primo aleatorio generado por H	Variables
p	Número primo generado por H	para el
h	Número primo aleatorio generado por H delimitado por p	algoritmo
h_i	Número primo aleatorio generado por S_i delimitado por p	de D-H
h_j	Número primo aleatorio generado por U_j delimitado por p	
h_{ki}	Número primo aleatorio generado por U_{ki} delimitado por p	
h_x	Número primo aleatorio generado por U_x delimitado por p	
$\prod(X)$	Producto de todos los elementos en X	
K_H	Llave compartida grenerada por el $Host$	
K_{S_i}	Llave compartida generada por el S_i	
K_{U_j}	Llave compartida generada por el U_j	
$K_{U_{ki}}$	Llave compartida generada por el U_{ki}	
K_{U_x}	Llave compartida generada por el U_x	

Símbolos usados en el dominio de transferencia

Símbolo	Significado	Descripción
r	índice de paquetes multimedia	Entero
MU_r	paquete multimedia r	Bits
$\{MU_r\}_{key_r}$	paquete multimedia r cifrado con key_r	
bp, bq	números primos bp y bq congruentes con $3\ mod\ 4$	Variables
bn	producto de bp por bq	para el
bs	primo relativo de bn	algoritmo BBS

Capítulo 1

Introducción

1.1. Motivación

La necesidad de comunicación y transmisión confidencial de información en la Internet, ha provocado que se desarrollen y mejoren entre otros los protocolos, algoritmos, mecanismos, que garanticen la confidencialidad, es decir, que la información sólo pueda ser vista por los usuarios autorizados, ver (Stalling, 2002).

Actualmente, el tipo de información que está teniendo mayor uso en la Internet es la multimedia, audio y video según Borko~Furht (2004). Algunas de las aplicaciones que manejan este tipo de información son: las videoconferencias, la distribución de contenido multimedia, y los juegos interactivos. Por lo general, estas aplicaciones se llevan a cabo en tiempo real, los usuarios obtienen la información en el momento en que ésta se genera. A estas aplicaciones se les conoce como sistemas distribuidos multimedia (SDM) debido a que capturan, procesan, comunican, presentan y almacenan información multimedia, según Steinmetz y Nahrstedt (2004).

Los SDM no cuentan con un reloj global, manejan procesos concurrentes (que se ejecutan al mismo tiempo) y tienen tolerancia a fallas independientes (las fallas no se aplica a todos los procesos). Además, el grupo de usuarios que forman al SDM puede ser o no escalable, es decir,se acepta un número no determinado de miembros, y dinámico,

es decir, permite el ingreso y salida de miembros en un momento determinado.

La transmisión de información en un SDM se lleva a cabo aplicando diversas estrategias de transmisión como lo son: *unicast, multicast y broadcast*, como se ve en (Tanenbaum y van Steen, 2002). Tanto *unicast* como *broadcast* tienen desventajas al ser comparándolas con *multicast*. El uso de *multicast* en los SDM permite la entrega de paquetes de uno a muchos y de muchos a muchos, esto facilita la distribución de información.

La estrategia *multicast* dentro de la comunicación en grupo provee una entrega eficiente de datos, lo que reduce los requerimientos de ancho de banda y sobrecarga de datos a transmitir (Hardjono y Dondeti, 2003). Esto hace que *multicast* sea una estrategia ideal para la comunicación en grupo y específicamente para un SDM. Sin embargo, *multicast* no cuenta con servicios de seguridad y por consecuencia se presentan diversos problemas.

Uno de estos problemas es la distribución confidencial de información. La manera en que *multicast* brinda este servicio se trata como en (Chan y Chan, 2003) que incorpora algoritmos criptográficos.

La criptografía tiene cuatro objetivos que son la confidencialidad, autenticación, integridad y el no repudio (Stalling, 2002). El éxito de cumplir estos objetivos reside principalmente en palabras claves (números ó caracteres) llamadas también llaves. Algunos aspectos importantes son la llave y el algoritmo para cifrar y descifrar. El tiempo para cifrar y descifrar depende de la complejidad del algoritmo que se utilice, ver (Schneier, 1995).

En las aplicaciones basadas en grupos como los SDM, por lo general existe un elemento llamado administrador de llaves que se encarga entre otras cosas de generar, distribuir y actualizar llaves (Hardjono y Dondeti, 2003). Dichas llaves son útiles para el control de membresía del grupo y de la seguridad en la transferencia de datos, es decir, para la confidencialidad. Otra tarea importante para estas llaves es la de cifrar y descifrar los datos a transmitir.

Actualmente se han desarrollado trabajos como en (Eskicioglu, 2002; Rafaeli y Hutchison, 2003) sobre los administradores de llaves usando la estrategia de comunicación *mul-*

ticast donde una parte importante es el algoritmo para la generación y distribución de llaves. Sin embargo, estas investigaciones no se han enfocado directamente a los SDM con un envío de paquetes en tiempo real. Para lograr esto es necesario desarrollar un mecanismo (conjunto de algoritmos).

Este mecanismo debe cumplir con características necesarias para ejecutarse en un SDM bajo *multicast* y en tiempo real. Debe ser escalable y dinámico ya que es importante para incrementar el número de miembros y que estos puedan entrar y salir del grupo en un momento determinado.

Proporcionar una transmisión *multicast* ofrece una ventaja considerable de enviar un solo mensaje en lugar de muchos. Debido al bajo costo computacional que ésta ventaja implica al enviar información en el grupo de trabajo. Esta investigación busca específicamente un algoritmo para generar llaves con base en elementos que intervienen en la comunicación y transmisión de datos (identificadores de miembros, números aleatorios, etc.).

Actualmente, una propuesta importante que se sugiere en (hua chu y otros, 2002) es el desarrollo de administradores de llaves que generen una llave por paquete para dar una mayor seguridad en los datos. Tomando en cuenta que los sistemas actuales utilizan solo una llave por cada cierto período, sesión, la entrada y la salida de participantes. El tener muchas llaves complica el proceso de conocerlas en tiempo real, dado que en los SDM se transmite mucha información, es decir, muchos paquetes.

Una llave por paquete daría hasta cierto punto una independencia entre los paquetes, porque no es necesario llevar un orden en el descifrado, dando como resultado que no se afecte el funcionamiento del SDM dadas las restricciones en tiempo y la fiabilidad parcial en el medio de comunicación. Sin embargo, no se ha desarrollado un algoritmo de generación y distribución de llaves en el que se tenga una llave por paquete.

Se han desarrollado diversos métodos de generación y distribución de llaves que bajo ciertas condiciones funcionan adecuadamente como lo hacen (Padmavathi y Annadurai, 2006; Banerjee y Bhattacharjee, 2002; Rodeh y otros, 2000). Lo que se espera del

mecanismo propuesto en esta investigación es que genere una llave por paquete y pueda interactuar en tiempo real con información de un sistema distribuido multimedia que transmita mediante *multicast*.

1.2. Seguridad en *Multicast* en SDM en Tiempo Real

Muchas de las aplicaciones en Internet se basan en la comunicación en grupo, aspectos importantes dentro de este tipo de aplicaciones son la estrategia de comunicación entre los participantes y la seguridad de los datos. Un factor importante que actualmente se está incorporando a estas aplicaciones es la comunicación en tiempo real, lo que implica restricciones de tiempo, para más detalle consultar (Borko~Furht, 2004).

La estrategia de comunicación usada actualmente es *multicast* debido a la eficiencia que proporciona para la entrega de paquetes. Para garantizar la seguridad se han propuesto algoritmos de criptografía pero aún no se ha podido definir uno en particular para la seguridad en *multicast*. Tanto *multicast* como los algoritmos de criptografía se han combinado para dar solución de manera parcial al problema de tener una comunicación confidencial ya sea en tiempo real o bajo demanda. Lo anteriormente dicho puede verse de manera esquemática en la figura 1.1.

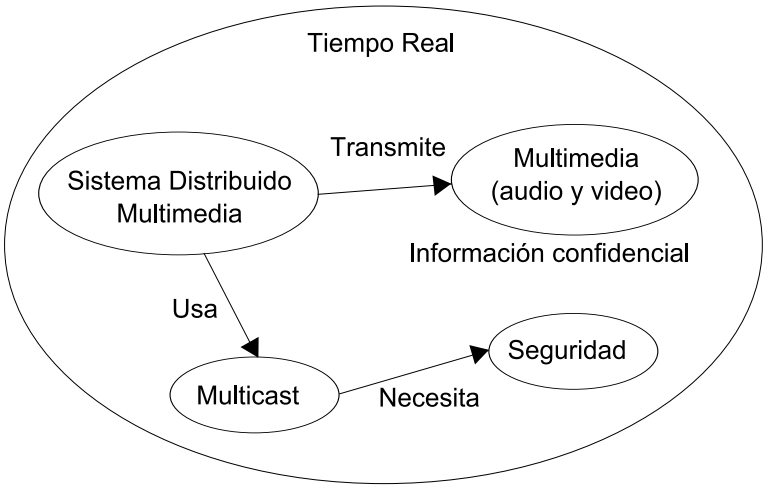

Figura 1.1: Representación esquemática del problema

Debido a la necesidad del intercambio y distribución de información multimedia (audio y video) en tiempo real de forma confidencial se han desarrollado varias soluciones, es decir, administradores de llaves pero en el caso de los SDM se presenta un problema de retraso para ejecutarse en tiempo real, esto debido a los grandes volúmenes de información que manipulan.

Al cifrar los paquetes se presenta un problema en la generación y actualización de llaves, esto es indispensable para cifrar los paquetes y garantizar su confidencialidad. Actualmente se trata de resolver el problema de la generación y actualización de llaves estableciendo restricciones a los administradores de llaves usado en sistemas distribuidos multimedia que usan *multicast*, tales restricciones van desde el establecer llaves fijas hasta el renovar las llaves cada vez que haya cambios en el grupo (que salgan o entren usuarios).

Existe una propuesta hecha en (hua chu y otros, 2002), para garantizar la independencia de los paquetes para que el retraso y la pérdida de información no afecten al sistema, como lo hacen los administradores actuales. Sin embargo, esta propuesta no está dirigida a la confidencialidad de información.

Los SDM en tiempo real con comunicación *multicast* que distribuyen información

confidencial usando una llave por paquete tendrán más confiabilidad en todos los ámbitos donde la prioridad sea la transmisión confidencial de multimedia entre varias personas, por ejemplo, las videoconferencias y la televisión por paga entre otras aplicaciones.

Para realizar nuestro mecanismo propuesto es necesario desarrollar un modelo para el administrador de llaves, así como también un algoritmo para la generación de llaves y un algoritmo para la distribución de llaves. Tanto el modelo como los algoritmos deben permitir la escalabilidad en el sistema. Lo anterior logrará que el sistema distribuido multimedia ofrezca un servicio de envío de información de forma confidencial, es decir, que sólo los usuarios autorizados vean la información y además lo hagan en tiempo real. Finalmente, unir el algoritmo de generación y el algoritmo de distribución de llaves para desarrollar el mecanismo y hacer la comparación con los ya existentes.

1.3. Objetivos

EL objetivo general de este trabajo es desarrollar un mecanismo *multicast* para generación y distribución de llaves que provea confidencialidad, en un sistema distribuido multimedia en tiempo real (el intervalo de tiempo que se presenta entre la entrada y la salida debe ser muy pequeño para que la respuesta temporal del sistema sea aceptable, según Borko~Furht (2004)). Por confidencialidad se entiende que se asegure que el contenido de los mensajes no pueda ser conocido por usuarios no autorizados.

Para llevar a cabo este objetivo es necesario realizar lo siguiente:

1. Desarrollar un algoritmo de generación y actualización de llaves, que provea una llave por paquete.

2. Desarrollar un algoritmo para la distribución de llaves.

3. Analizar el algoritmo de generación y actualización de llaves con base en las dos principales métricas que son el número de llaves almacenadas y la cantidad de mensajes para la actualización de llaves.

4. Implementar el funcionamiento real de los algoritmos de generación y distribución de llaves, con el lenguaje java.

1.4. Propuesta de Solución

Dado el problema antes mencionado sobre la confidencialidad en los SDM en tiempo real, la solución que proponemos es un mecanismo para la generación y distribución de llaves. Se propone que el mecanismo cuente con tres algoritmos, un primer algoritmo para la distribución de llaves, dado que éste es uno de los elementos principales para garantizar que la entrega sea eficiente. Un segundo algoritmo para la generación y la actualización de llaves que permita a los usuarios ser parte del sistema. Éste segundo algoritmo se basará en la extensión del algoritmo de Diffie-Hellman orientado a la comunicación en grupo, ver (Steiner y otros, 1996). El tercer algoritmo generará la llave por paquete, estas llaves deben tener una secuencia impredecible para no ser conocidos durante el tiempo de vida de los paquetes.

Para ejecutar los algoritmos de generación y distribución de llaves es necesario que el SDM sea estructurado jerárquicamente en tres niveles, para que el mecanismo propuesto pueda funcionar adecuadamente.

En el primer nivel estará un proceso llamado *Host principal* que generará variables que tendrán dos tareas, la primera será el permitir el ingreso al SDM a los procesos que lo deseen, y la segunda será la generación de una llave por paquete.

El segundo nivel alojará procesos llamados *miembros organizadores*, estos procesos atenderán a un número no definido de procesos qué quieran entrar al sistema, el *Host* indicará qué procesos atenderán a cada miembro organizador.

Finalmente, en el tercer nivel estarán los procesos llamados usuarios, quienes harán la petición de ingreso al sistema por medio del *Host principal* para luego ser atendidos por un miembro organizador.

El algoritmo elegido para cifrar y descifrar el audio y video es el AES (Advanced

Encryption Standard) debido a que es un estándar, además de tener ventajas como su fácil implementación así como también su ejecución rápida comparándolo con otros algoritmos de cifrado, para más información ver (Stalling, 2002). Esta última característica es muy importante dado que nuestro sistema se ejecutará en tiempo real.

Para la generación de las llaves que serán usadas por el algoritmo AES seleccionamos el algoritmo BBS (Blum Blum Shub). Este algoritmo genera números pseudo aleatorios robustos, ver (Blum y otros, 1986). La secuencia que genera es impredecible comparada con otros algoritmos generadores de números pseudo aleatorios. Una ventaja del BBS es su capacidad de recuperar cualquier número de la secuencia pseudo aleatoria con sólo introducir la posición que tiene dicho número dentro de la secuencia.

Se estima que nuestra propuesta dará buenos resultados, tomando en cuenta las ventajas de los algoritmos de generación y distribución de llaves y principalmente el diseño de nuestro mecanismo.

1.5. Organización de la Tesis

Este trabajo se encuentra dividido en seis capítulos. El capítulo uno contiene la introducción, así como también la problemática, los objetivos y un esbozo de la propuesta de solución. En el capítulo dos dará un poco de los conceptos básicos que son requeridos para entender los capítulos siguientes. Posteriormente, el capítulo tres incluye el estado del arte donde se describen los principales trabajos que han sido desarrollados para la seguridad *multicast* en SDM en tiempo real. Después, el capítulo cuadro contiene la descripción de nuestro mecanismo y de los algoritmos que en él se encuentran. En el capítulo cinco describimos la implementación, así como también las pruebas y resultados obtenidos dadas las métricas que se proponen. Por último, en el capítulo seis se exponen las conclusiones y el trabajo futuro de la tesis desarrollada.

Capítulo 2

Conceptos Básicos

En este capítulo se abordarán los conceptos teóricos para el desarrollo del mecanismo en esta investigación. Se describen las funciones, las propiedades y los servicios de las partes que integran al mecanismo y que proporcionan en un momento determinado, así como también las restricciones de los mismos.

2.1. Modelos de Comunicación en Grupo

Un grupo es un conjunto de usuarios o procesos en una red de computadoras. La comunicación en grupo en una red LAN, WAN o la Internet exige un modelo de comunicación dependiendo de las necesidades de las aplicaciones, ver (Hardjono y Dondeti, 2003). Dado un grupo de trabajo con n-nodos (elementos del grupo, usuarios, procesos)existen básicamente tres modelos de comunicación que son:

- *Unicast.*- La comunicación es uno a uno, es decir, para que un nodo envíe un mensaje a todos los nodos en el grupo es necesario hacer tantas copias del mensaje como el número de nodos en el grupo, y enviar cada copia por separado.
- *Multicast.*- La comunicación es uno a muchos, es decir, sólo se necesita hacer una copia del mensaje, los enrutadores serán los encargados de hacer las copias necesarias para hacer llegar dicho mensaje a todos los nodos en el grupo.

- *Broadcast.*- La comunicación es uno a todos, es decir, se envía un mensaje que llegará a todos los nodos que estén en la red, estén o no en el grupo de trabajo.

También existe una clasificación respecto al tipo de grupo, a continuación se presenta:

- Grupo cerrado o abierto.- Se dice que un grupo es cerrado cuando sólo los miembros de dicho grupo pueden enviar mensajes al grupo. En caso contrario se considera como grupo abierto.
- Grupo centralizado o distribuido.- En un grupo centralizado existe un miembro encargado de procesar todas las peticiones de ingreso. En cambio, para un grupo distribuido todos los miembros conocerán todas las peticiones de ingreso al sistema.

A este tipo de grupo se le puede implantar cualquiera de los modelos de comunicación antes mencionados, dependiendo del tipo de aplicación que se desee.

2.2. La Comunicación con *Multicast*

Muchas de las aplicaciones que se ejecutan en grupos de trabajo usan *multicast* dada la eficiencia en la entrega de paquetes que ésta proporciona, ver (Borko~Furht, 2004) . *Multicast* inicialmente trabaja bajo un tipo de grupo abierto, donde no hay restricción para quien envía mensajes al grupo, es decir, el que envía el mensaje puede o no pertenerser al grupo, de aquí los problemas de seguridad que afectan a multicast y que se ven en la figura 2.1.

La comunicación con *multicast* tiene tres características sobresalientes las cuales son:

1. El paquete para un miembro es recibido por todos miembros en el grupo.
2. La posibilidad de ingreso al grupo se mantiene abierta para todo usuario.
3. El usuario que no pertenezca al grupo puede enviar información al grupo.

Estas dos últimas características permiten que cualquier usuario envíe y reciba información del grupo, esto causa problemas de seguridad como lo son: la denegación de servicios y la suplentacia de identidad, ver figura 2.1. Para estos problemas de seguridad se proponen los administradores de llaves, pueden ser de tres tipos centralizados, descentralizados y distribuidos, según Judge y Ammar (2003).

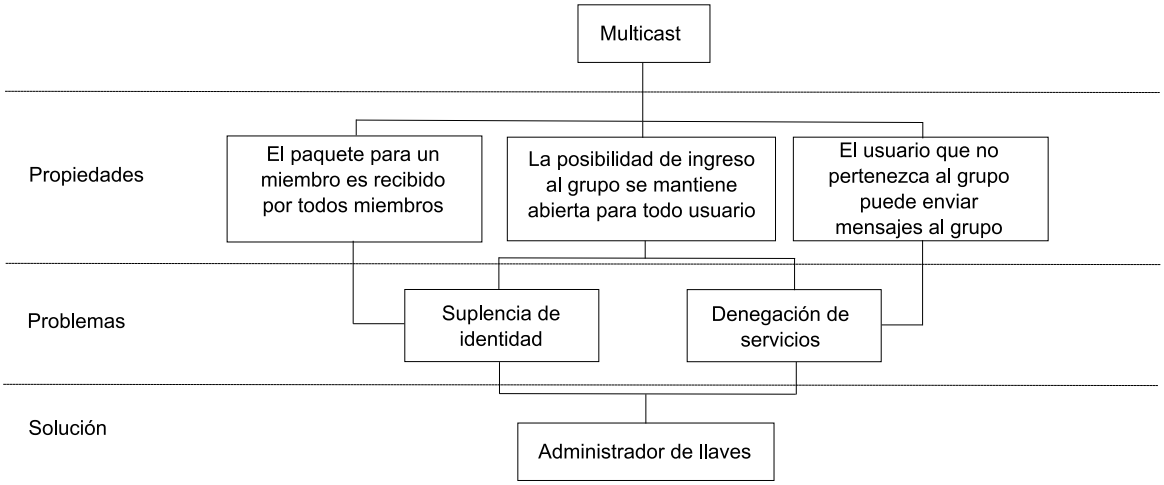

Figura 2.1: Propiedades, problemas y solución en multicast

2.3. Administradores de Llaves

Para garantizar que el acceso a la información de un determinado grupo lo realicen únicamente los usuarios que tengan autorización es utilizada la criptografía. Existen muchos algortimos de cifrado con diferentes características como se ve en (Stalling, 2002). Sin embargo, para todos ellos es de vital importancia que se seleccionen llaves las cuales son muy complicadas de conocer (Rafaeli y Hutchison, 2003), para garantizar la seguridad de acceso. En los grupos de trabajo debe existir un administrador de llaves encargado de realizar las siguientes tareas:

- Generar y distribuir llaves.

- Controlar de acceso.
- Identificar y autentificar de miembros.

Las llaves que genera el administrador de llaves son utilizadas en el control de acceso, así como para la identificación y autenticación de los miembros del grupo. Estos administradores de llaves pueden clasificarse de acuerdo al número de nodos que realizan la generación y distribución de llaves de la siguiente manera:

1. Centralizados. En este caso, un nodo dentro del grupo toma el papel del administrador de llaves.
2. Descentralizado. El grupo es dividido en subgrupos y para cada subgrupo se toma un miembro que realiza las tareas de un administrador de llaves, es decir, se tiene más de un administrador de llaves.
3. Distribuido. Todos los miembros del grupo colaboran para llevar a cabo las tareas del administrador de llaves.

En las siguientes secciones describimos los tipos de administradores de llaves antes mencionados.

2.3.1. Administrador de Llaves Centralizado

En este tipo de administrador una entidad llamada Key Distribution Center (KDC) es la que genera y distribuye las llaves para todo el grupo. Algunas de las características de este control es la escalabilidad en un grupo grande con frecuentes entradas y salidas de usuarios, ver (Rafaeli y Hutchison, 2003).

La eficiencia de los modelos centralizados se mide tomando en cuenta las características siguientes:

- El número de llaves que necesitan los miembros y el administrador del grupo para tener comunicación segura.

- El número de mensajes que se necesita para actualizar el grupo dada la entrada o salida de usuarios.
- La capacidad de mantener en secreto las llaves una vez que salgan o entren usuarios al grupo.
- La notificación a los usuarios que causen colisiones.

Entre las ventajas que tiene este control se encuentran:

- Permitir la entrada y salida de miembros de manera fácil, dado que los nuevos nodos no llevarán a cabo ningún proceso para crear la llave de acceso.
- Conservar la sincronía del sistema.
- Entregar la nueva llave de manera directa.

Algunas de las desventajas del administrador centralizado son las siguientes:

- Depender de un nodo, si falla el KDC fallará todo el sistema.
- Permitir la asincronía en el sistema.
- Perder una llave causa que todo el sistema se detenga hasta que todos los miembros tengan la misma llave.

2.3.2. Administrador de Llaves Descentralizado

Bajo este administrador el grupo de usuarios se divide en subgrupos. A cada subgrupo se le asigna un tipo de control diferente para administrarlo. Por lo tanto, si algunos miembros fallan, el grupo completo no se ve afectado (Rafaeli y Hutchison, 2003).

Las características para evaluar la eficiencia de los modelos descentralizados son las siguientes:

- Independencia de las llaves: las nuevas llaves no dependen de las anteriores, como es el caso de algunos modelos centralizados.
- Controlador descentralizado.
- Generación de nuevas llaves de forma local. Los cambios locales se hacen en cada subgrupo y se involucra a todos los miembros de todos los grupos.
- Llaves vs. datos: la ruta de los datos deben ser diferentes a la ruta de las llaves, para no retrasar la comunicación.
- Creación de nuevas llaves dependiendo de la membresía: cambiar las llaves cada vez que sale o entra un usuario.
- Tipo de comunicación: depende de cuántos miembros transmitan información.

Al usar este tipo de administradores se tienen las siguientes ventajas:

- La generación y la distribución de llaves es realizada entre varios miembros.
- La entrada y salida puede ser atendida por varios nodos.
- El nodo encargado de un grupo no debe fallar, porque si lo hace los usuarios conectados a este son desconectados del sistema.

Sus desventajas se enuncian a continuación:

- El problema de sincronización al conocer la nueva llave del grupo.
- EL tiempo para ponerse de acuerdo para usar una misma llave en el grupo.
- La falla en la comunicación entre los nodos que organizan los subgrupos causa que no esa posible la comunicación entre todos los usuarios del sistema.

2.3.3. Administrador de Llaves Distribuido

Este control se caracteriza por no tener un administrador de grupo. En este esquema todos los miembros colaboran para la generación de la llave del grupo haciendo este tipo de control muy robusto. Sin embargo, muchas de las propuestas presentadas en (Judge y Ammar, 2003), que se han realizado tienen la desventaja de incrementar linealmente el tiempo de procesamiento y comunicación en el grupo respecto al número de usuarios, ver (Rafaeli y Hutchison, 2003).

Por lo general, este tipo de control se basa en el algoritmo de intercambio de llaves propuesto por Whitfield Diffie y Martin Hellman, ver (Steiner y otros, 1996).

Para evaluar el desempeño de los modelos propuestos se toman en consideración las siguientes características:

- El número de ciclos que necesitan los usuarios para realizar el procesamiento y comunicación en el grupo.
- El número de mensajes que se necesitan para llevar a cabo la comunicación entre miembros del grupo.
- El procesamiento durante la configuración, es decir el tiempo para sincronizar a los usuarios y ponerse de acuerdo para el establecimiento de las llaves.
- La llave generada por Diffie-Hellman (D-H) para el intercambio de llaves, el cual, todos los usuarios deben fusionar sus llaves para generar una única llave para el grupo.

Para estos administradores se tienen las ventajas siguientes.

- Las tareas de generar y distribuir la llave del grupo son realizadas por todos los miembros en el sistema.
- Los canales seguros se construyen para la transmisión de información.

- Los mensajes se envían de manera segura una vez establecida la llave.

Las desventajas que presentan estos administradores son:

- El tiempo de sincronización aumenta si un usuario falla.
- El proceso para la entrada y salida de un nuevo usuario depende del tiempo de sincronización de los usuarios en el grupo.
- La falla de un nodo causa que todo el sistema se detenga, dada la colaboración de este en los procesos de actualización del sistema.

Una vez visto las ventajas y desventajas de los diferentes administradores de llaves se da un repaso sobre los algoritmos de cifrado.

2.4. Algoritmos de Cifrado

Los algoritmos de criptografía se clasifican en dos tipos, los de llave privada, también llamados simétricos tales como DES, 3DES y AES, y los de llave pública, también llamados asimétricos tales como RSA o DH, consultar (Koblitz y Menezes, 2003). Los algoritmos de llave privada a su vez se clasifican por el tipo de cifrado, por bloque o flujo.

Los algoritmos simétricos usan una llave para el cifrado y descifrado. En los algoritmos asimétricos se usan dos llaves una para cifrar y otra para descifrar. El tamaño de las llaves simétricas por lo general es más pequeño que el de las llaves asimétricas, ver (Alfred J.˜Menezes y Vanstone, 1996). Debido a esta característica se opta por el uso de algoritmos simétricos para la tarea del cifrado multimedia.

La mayoría de los algoritmos simétricos actuales se apoyan en los conceptos de confusión y difusión introducidos por Claude Shannon a la Teoría de la Información a finales de los años cuarenta (Alfred J.˜Menezes y Vanstone, 1996).

- Confusión, consiste en ocultar la relación entre el texto plano, el texto cifrado y la clave.

- Difusión, consiste en repartir la influencia de cada bit del mensaje original entre el mensaje cifrado.

2.4.1. Características de los Algoritmos Simétricos

Las características principales de los algoritmos de cifrado de llave privada son:

- La clave para cifrar es la misma para descifrar.
- La ejecución al implementarse en hardware se realiza en menor tiempo que al implementarse en software.
- La clave única debe distribuirse por un medio seguro.

2.4.2. Características de los Algoritmos Asimétricos

Los algoritmos de llave pública tiene las siguientes características:

- Utilizan un par de claves, una para cifrar y la otra para descifrar.
- Están basados en operaciones matemáticas de un solo sentido (Koblitz y Menezes, 2003). Se dice que f es una función de un solo sentido si:
  - Dado cualquier y es computacionalmente imposible hallar x tal que $f(x) = y$.
  - Existe una función h y una información secreta s tal que conocidos s e y, es fácil de calcular $h(s, y)$ dado que $f(h(s, y)) = y$.
- Son más lentos que los algoritmos simétricos.
- Necesitan claves más largas para ser igual de seguros.
- Utilizan una clave pública y distribuirse sin problemas por cualquier medio.
- Usan algoritmos asimétricos duales, es decir, cada clave sirve para cifrar o descifrar lo que la otra descifra o cifra.

2.4.3. Algoritmo AES

El algoritmo para cifrar AES (Advanced Encryption Standard) (Stalling, 2002) es el estándar de cifrado simétrico propuesto por NIST (National Institute of Standards and Technology). Este algoritmo fue elegido después de un período de competencia con 14 candidatos. Sus principales características se presentan a continuación:

- Tiene diseño simple.
- Garantiza la seguridad.
- Es inmune a la mayoría de ataques.

En general, AES consiste de dos partes, la primera es el proceso de cifrado y la segunda el proceso de generación de subclaves (Key Schedule), ver figura 2.2. El tamaño del bloque de datos a cifrar es de 128 bits, pero el tamaño de las llaves varía entre 128, 192 y 256 bits. El proceso de cifrado consta de 10, 12 ó 14 vueltas respectivamente, cada vuelta consiste en la aplicación de una ronda estándar de cuatro transformaciones básicas las cuales son:

- AddRoundKey. Esta transformación toma una matriz de información a cifrar y simplemente hace un XOR byte a byte con la correspondiente matriz de la llave depende de la ronda en la que se esté.
- SubByte. En este caso a cada elemento byte de la matriz estado (que se obtiene de la operación AddRoundKey) se le sustituye por otro byte depende del primero.
- ShiftRows.- Aplica corrimientos a las columnas de la matriz estado.
- MixColumns.- Cada columna de la matriz estado se multiplica por una columna constante en la matriz de la llave.

Para nuestra implementación el tamaño de llave propuesto es de 128 bits debido a que garantiza que no sea posible conocer la llave en al menos 10 años. Por otra parte,

dedido al cambio constante de llaves sería muy complicado conocer todas las llaves en un tiempo considerable, ver (Blum y otros, 1986). Además dichas llaves son robustas, es decir, al conocer una no se tiene la información necesaria para conocer las demás.

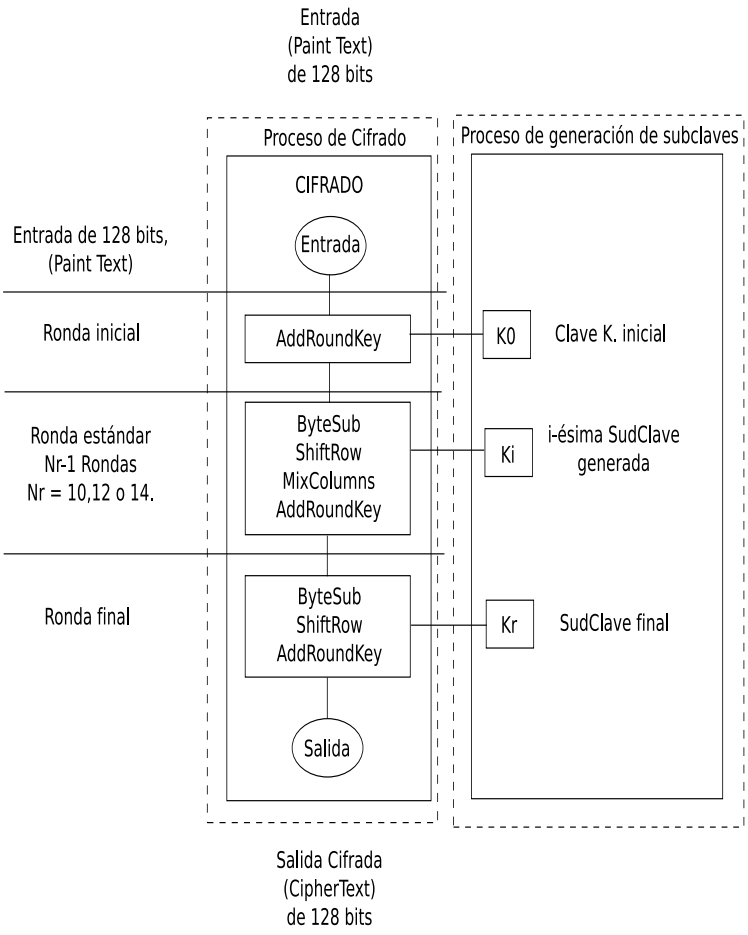

Figura 2.2: Arquitectura de AES. Las fases a la izquierda representan el proceso completo de cifrado. Las fases del centro representan las operaciones asociadas a cada parte del proceso de cifrado. Finalmente, las fases a la derecha representan el proceso de generación de subcalves que son utilizadas en las operaciones del proceso de cifrado.

2.5. Algoritmos Generadores de Números Pseudoaleatorios

El uso de los números pseudoaleatorios en los algoritmos criptográficos es muy importante por que generalmente estos números son usados como llaves.

Los generadores de números pseudoaleatorios utilizan un valor x_0, denominado *semilla*, y a partir de él se generan x_1, x_2, ..., x_n. Donde n es la cantidad de números distintos que se genera, y es conocida como longitud de ciclo. El conjunto de números $[x_1, ..., x_n]$ es llamado período o ciclo, ver (Gentle, 2004).

Es deseable que estos generadores tengan dos principales características que son:

- Es imposible predecir la salida del generador de números aleatorios incluso conociendo las salidas extraídas en generaciones anteriores.
- No existen patrones entre los números generados, ádemas el período de los números generados debe tener una longitud larga.

Algunos de los algoritmos más conocidos son: BBS (Blum Blum Shub), DSA(Digital Signature Standard), ANSI X9.17, RSAREF, Generador lineal congruencial, Lagged-Fibonnaci. De estos algoritmos el que ha superado un número mayor de pruebas de aleatoriedad de manera satisfactoria es el BBS, ver en (Alfred J.˜Menezes y Vanstone, 1996). Otro punto a favor de este algoritmo es que puede calcular cualquier valor sin necesidad del anterior.

2.5.1. Algoritmo Blum Blum Shub

El algoritmo Blum Blum Shub (BBS) fue propuesto por Lenore Blum, Manuel Blum y Michael Shub. Es el algoritmo de generación de números pseudo-aleatorios que más pruebas estadísticas ha superado tales como las que se describen en (Alfred J.˜Menezes y Vanstone, 1996), es también muy simple y fácil de implementar. Se basa en residuos cuadráticos, ver (Blum y otros, 1986). Otro detalle muy interesante es que se puede

calcular directamente cualquier valor de la serie de números generados sin tener que calcular los anteriores con la ecuación 2.1.

$$x_i \ = \ (x_0^{(2^i (mod((p-1)*(q-1))))}) \ mod \ n \tag{2.1}$$

Donde x_i es el *i-ésimo* número generado, x_0 es la semilla para el generador, p y q son números congruestes con $4 \ mod \ 3$ y n es el producto de $p * q$.

El algoritmo BBS pueden generar varios números sin necesidad de almacenarlos todos, cada número puede ser reconstruido a partir de su índice y del primer número de la serie. El período máximo para este generador está dado por n, dado que B.B.S. es un generador congruencial.

Una aplicación del BBS es la generación de secuencias de bits pseudo aleatorios. A continuación se describe el funcionamiento de algoritmos B.B.S. para la generación de secuencias de bits pseudo aleatorios:

1. Se inicia eligiendo dos números primos grandes p y q, cuyo residuo sea 3 al ser divididos entre 4 en otras palabras que sean congruentes($\equiv$) con $4 \ mod \ 3$, es decir:

$$p \equiv 3 \ (mod \ 4) \qquad q \equiv 3 \ (mod \ 4) \tag{2.2}$$

2. Una vez elegidos p y q se obtiene $n = pq$.

3. Se elige entonces un número aleatorio x primo relativo con n, es decir, el mínimo común multiple de x y n es 1, que será la semilla inicial.

4. Las ecuaciones para calcular los valores de la secuencia estan expresadas en 2.3 y 2.4.

$$s_0 = x^2 \ mod \ n \tag{2.3}$$

$$s_{i+1} = x_i^2 \ mod \ n \tag{2.4}$$

5. Por regla se debe emplear sólo unos pocos bits t_i del final de cada s_i. Como se usan no más de $log_2(log_2\ s_i)$ bits, el cálculo para conocerlos es tan difícil como factorizar n, ver (Eastlake y otros, 2005).

6. La secuencia de bits será $t_1, t_2, t_3, ...$ tantos como se requiera.

Dadas las ventajas que este algoritmo nos proporciona se implementará en el presente trabajo. A continuación se presenta la descripción de los sistemas distribuidos, medio en que el algoritmo BBS se usará.

2.6. Sistemas Distribuidos

Un sistema distribuido (SD) es *un conjunto de procesos que se ejecutan en una o más computadoras que colaboran y comunican intercambiando mensajes*. Otra definición es la hecha por Leslie Lamport la cual dice que *un sistema distribuido es aquél en el que no puedes trabajar con tu máquina por el fallo de otra máquina que ni siquiera sabías que existía*.

Algunos ejemplos de aplicaciones de los sistemas distribuidos son las siguientes:

- Correo electrónico, transferencia de archivos.
- Servicios de News.
- World Wide Web.
- Sistemas de control de tráfico aéreo.
- Aplicaciones bancarias.
- Comercio electrónico.
- Aplicaciones multimedia (videoconferencias, video bajo demanda, etc.).

    - El ancho de banda en estas aplicaciones es un orden de magnitud mayor que en otras.
    - La calidad de servicio (QoS) es primordial.

- Aplicaciones médicas (transferencia de imágenes).

Algunas de las principales ventajas de los SD, son las siguientes: pueden compartir recursos (HW, SW, datos), capacidad de escalabilidad, tolerancia a fallas, concurrencia y ejecución paralela de una aplicación.

Las desventajas de los sistemas distribuidos son entre otras: la pérdida de mensajes, la saturación, inseguridad en las comunicaciones.

2.6.1. Sistemas Distribuidos Multimedia

Se les conoce como sistemas distribuidos multimedia (SDM) a los sistemas que capturan, procesan, comunican, presentan y almacenan grandes volumenes de información multimedia. El término multimedia se refiere a la manipulación integrada de información representada como datos continuos, audio y video, en conjunto con datos discretos, texto y gráficos.

Los SDM al igual que los SD no cuentan con un reloj global, manejan procesos concurrentes y tienen tolerancia a fallas independientes, ver (Tanenbaum y van Steen, 2002).

Algunos ejemplos de estos sistemas son los juegos en línea, televisión por Internet, videoconferencias, videos en demanda en la Internet, entre otros.

Capítulo 3

Estado del Arte

En este capítulo presentamos la revisión del estado del arte. Mencionamos los trabajos más importantes que han sido desarrollados sobre los administradores de llaves que garantizan la seguridad en *multicast* en SDM. Este capítulo se divide en cinco secciones. En las primeras tres secciones, presentamos los trabajos hechos sobre los diferentes administradores de llaves como son los basados en un control centralizado, descentralizado y distribuido. En la sección cuatro presentamos la ubicación de nuestro trabajo dados los trabajos presentados en las tres primeras secciones. Finalmente, la sección cinco presentamos conclusiones del presente capítulo.

3.1. Administrador de Llaves Basado en un Control Centralizado

Algunos de los trabajos más destacados en este tipo de control son descritos a continuación.

Uno de los trabajos pioneros en este control es el Group Key Management Protocol (GKMP) propuesto por Herney y Muckenhirn en 1997 (Harney y Muckenhirn, 1997). En este protocolo el trabajo del KDC (Key Distribution Center) al iniciar el grupo es crear y distribuir un paquete llamado GKP (Group Key Packet) que sirve para identidicar si

los usuarios están o no dentro del SDM. El GKP contiene dos llaves, una para cifrar los datos (GTEK, Group Transport Encryption Key) y otra para descifrar (GKEK, Group Key Encryption Key) los posteriores GKP's. Dicho paquete es enviado a todos los usuarios que ingresen, para que puedan comunicarse de forma confidencial en el SDM. Para hacer una actualización de llaves el KDC debe crear un nuevo GKP y cifrarlo con la llave anterior GKEK.

Debido a la actualización de GKP es posible que un usuario entre, conozca la llave GKEK y después salga, no obstante puede seguir conociendo todo lo que se distribuya en el grupo, debido a que conoce la llave GKEK que sirve para descifrar los posteriores GKP's.

Una contribución importante en este tipo de control fue hecho por Wong y Wallner en 1998 (kei Wong y otros, 2000), al proponer el Logical Key Hierarchy (LKH). Este modelo crea una estructura de árbol binario para todo el SDM como se ve en la figura 3.1, donde U_1, U_2, U_3, U_4 son los usuarios y K_x es la llave asociada a x. El KDC distribuye la llave KEK (Key Encryption Key) a todos los nodos, de esta manera todos conocen la llave de la raíz que se usa para cifrar y descifrar los mensajes. El modelo LKH junto con el GKMP tienen la misma desventaja previamente mencionada dado que un usuario que haya pertenecido al grupo puede conocer la llave para cifrar y descifrar aún cuando éste ya no pertenezca al SDM.

En el mismo año surge un nuevo modelo basado en el LKH llamado One-way Function Tree (OFT), el cual incorpora una función para cifrar las llaves de los nodos hoja (que ya no tienen asociados otros nodos) y con esto garantizar que los usuarios que hayan salido no conozcan la nueva llave. No obstante, estos usuarios siempre tendrán datos suficientes para llegar a conocer la nueva llave debido a que ésta es calculada con base en la anterior.

En 1999 el modelo LHK de (Waldvogel y otros, 1999) es modificado por Canetti, creando así el Hierarchical a-ary Tree with Clustering (HTC), el cual crea subgrupos. El número de subgrupos está dado por la formula n/m, donde n es el número de miembros y m es la cantidad de miembros por grupo. Para cada grupo se crea un KEK, así también

se les asigna una llave K a cada miembro. El KDC genera las llaves K basándose en un generador de números pseudoaleatorios. En este esquema un generador congruencial lineal (basado en el cálculo de residuos de potencias sucesivas) es el encargado de generar las llaves K. Desgraciadamente una desventaja de este tipo de generador es que resulta fácil conocer todas las llaves generadas con sólo conocer una.

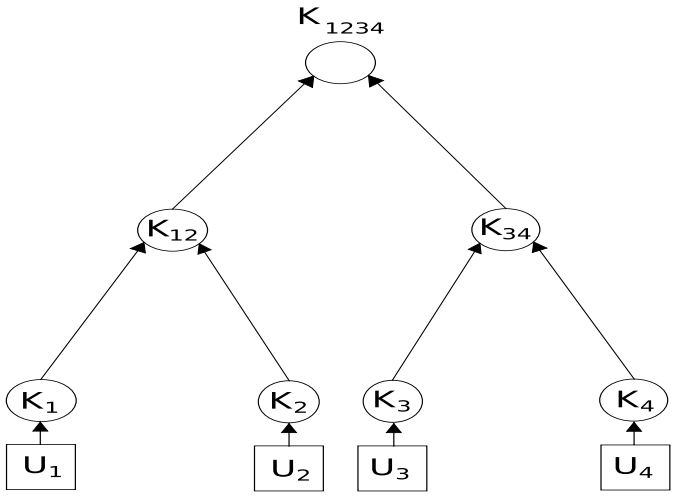

Figura 3.1: Ejemplo de la estructura del LKH, donde U_i representan los usuarios, K_i representan las llaves asociadas a los usuarios x. La generación de las llaves K_{ij} y K_{ijkl} es de forma ascendente.

Waldvogel por su parte propone un modelo llamado Flat Table (FT) cuyo propósito es disminuir el número de llaves que se almacenan en el KDC. El método para lograr su objetivo es crear una tabla que almacena todas las llaves del SDM. Dicha tabla está estructurada por llaves para cifrar datos TEK (Transport Encryption Key) y las llaves para cifrar llaves KEK. Cada vez que se necesite cambiar llaves el KDC crea las llaves y las manda a los usuarios para luego colocar estas nuevas llaves en la FT. La organización de la tabla (ver cuadro 3.1.) se realiza tomando en cuenta las llaves KEK de los usuarios que representan las columnas y el tamaño de identificación en bits que son las filas, consiguiendo con esto que cada usuario conozca únicamente las llaves que le son necesarias

y no todas como en algunos modelos anteriormente mencionados, como lo son GKMP y LKH.

	TEK	
ID Bit#0	KEK 0.0	KEK 0.1
ID Bit#1	KEK 1.0	KEK 1.1
ID Bit#2	KEK 2.0	KEK 2.1
ID Bit#3	KEK 3.0	KEK 3.1
	Valor del Bit = 0	Valor del Bit = 1

Cuadro 3.1: Ejemplo de una FT, donde ID BIT#X es el identificador de un usuario el cual esta asociado con dos grupos de llaves una vertical y otra horizontal.

Finalmente en 2006 Ganapathi y Samukutty , ver (Padmavathi y Annadurai, 2006), proponen el modelo crossbreed que disminuye el número de llaves que son almacenadas respecto a modelos anteriores como el GKMP y LKH, entre otros. En este modelo se combinan las características de tener subgrupos y que cada usuario tenga una llave. Se propone un esquema con usuarios estáticos y dinámicos con el propósito de facilitar la distribución de nuevas llaves cada vez que se necesiten. Los miembros estáticos son de gran utilidad debido a que ellos almacenan las llaves de los subgrupos, tomando el papel de KDC, por lo tanto se crean tantos miembros estáticos como subgrupos. Estos miembros están coordinados con un Host principal para realizar el ingreso o salida de usuarios, gracias a este tipo de organización es posible disminuir el paso de mensajes para distribuir las llaves, ver figura 3.2. En este modelo el tamaño de los subgrupos está dado por el número de usuarios que se encuentren a una misma distancia r de un determinado miembro estático. Donde r se refiere al número de enrutadores.

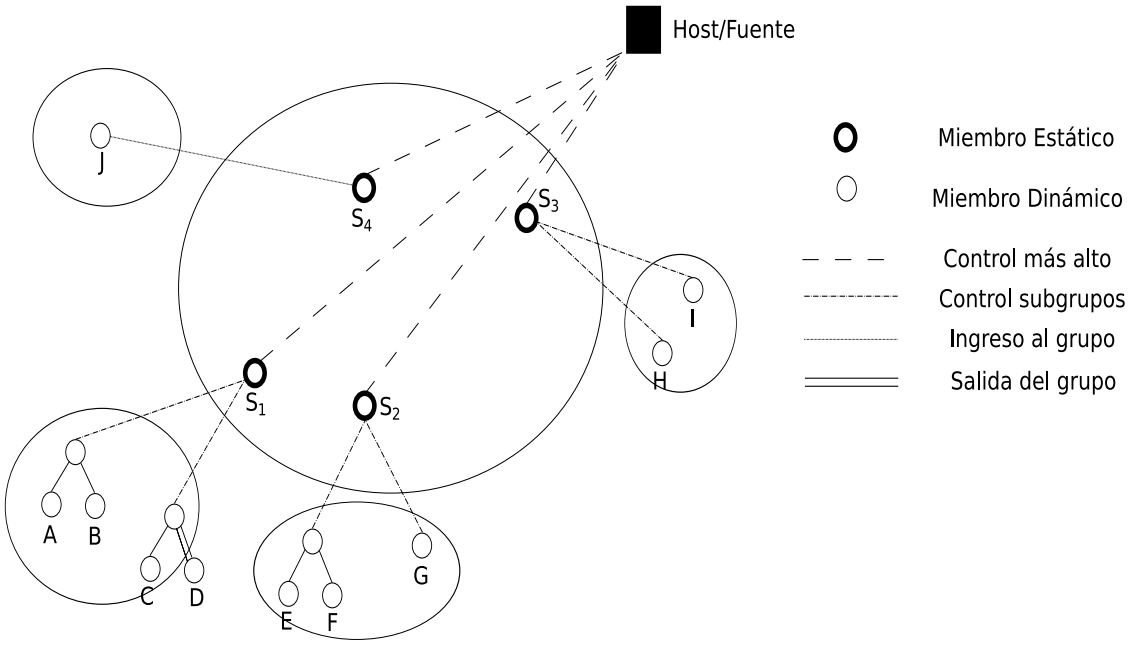

Figura 3.2: Esquema del Crossbreed, donde S_i reprentan los miembros estáticos, y los A, B,...,I representan los miembros dinámicos, las líneas indican asociación entre ellos. En esta figura se ilustra la entrada J y la salida de D en un momento dado.

Actualmente, estos modelos de administradores de llaves sólo son utilizados para sistemas de video bajo demanda, pago por evento y para actualización de software.

En el cuadro 3.2 se presenta un comparativo entre los modelos más usados en el control centralizado tomando en cuenta el número de llaves almacenadas en el KDC. El trabajo propuesto por G. Padmavathi y S. Annadurai en (Padmavathi y Annadurai, 2006) (Croosbeed) es el que mejor se comporta, dado que usa menos llaves comparado con los demás.

Criterio	GKMP	LKH	HTC	Crossbeed
Total de llaves en el KDC	$2 * n$	$(1 + log_d n) * n$	$n*[2+log_d(n/m)]$	$2*n/(1+r)+$ $[n/(1+r) *$ $\{1 + log_d r\}]$

Cuadro 3.2: Comparación de la cantidad de llaves que se almacenan en cada modelo, donde n representa el número de usuarios, d el tamaño de cada subgrupo y r el número de enrutadores entre un usuario y un miembro estático

3.2. Administrador de Llaves Basado en un Control Descentralizado

A continuación se presentan algunos de los trabajos más destacados en este tipo de control.

El RFC1946 (Requests for Comments) de Ballardie , ver (Ballardie, 1996), propone un esquema de árboles llamado Scalable Multicast Key Distribution (distribución *multicast* escalable de llaves). Este control está basado en el CBT (Core Based Tree) para la entrega de llaves vía *multicast*, ver figura 3.3. En el CBT los enrutadores son los encargados de la comunicación entre usuarios, es decir, son los encargados entre otras cosas de llevar a cabo la autentificación, verificar que los usuarios que manden información pertenezcan al grupo.

En este esquema también se considera el IGMP (Internet Group Management Protocol) pero con modificaciones; sin embargo, aún con un problema, una vez que alguien entra de ahí en adelante se considera parte del grupo aún cuando ya haya salido de algún subgrupo.

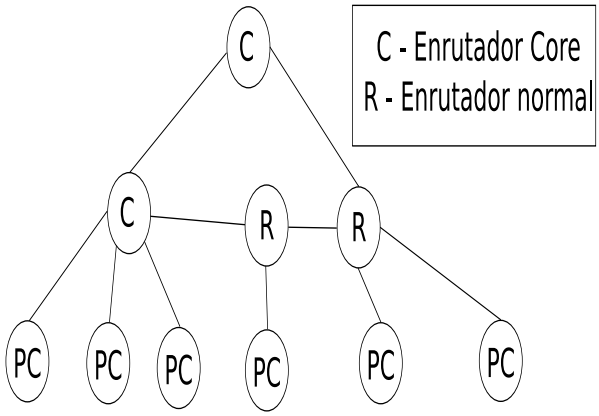

Figura 3.3: Ejemplo de la distribución *multicast* de llaves usando CBT, donde PC representa una computadora, las líneas indican la relación jerárquica que existe entre los enrutadores representados por C (core) y R (normal). Si PC desea ingresar debe solicitarlo con un enrutador C, por otro lado C se encarga de entregarle una llave a PC. Tanto la solicitud y la llave se envian por deferentes rutas.

El esquema Iolus propuesto por Mitra en 1997 (Mittra, 1997), utiliza el concepto de GSA (Group Security Agent). Son básicamente KDC que controlan a cada subgrupo. Los GSA son igualmente agrupados para ser controlados por el GSC (Group Security Controller), ver figura 3.4. Los GSC están encargados de reportar la salida o entrada de usuarios entre los GSA.

En Iolus se usan llaves independientes para cada subgrupo y no se cuenta con una llave general. Este esquema se distingue por ser escalable debido al manejo distribuido de la entrada y salida de los usuarios por parte de los GSA's, desgraciadamente al momento de reportar salidas o entradas se pueden ocasionar cuellos de botella en el GSC, debido a la falta de sincronía. Otra característica del Iolus es que transmite por separado los datos y las llaves.

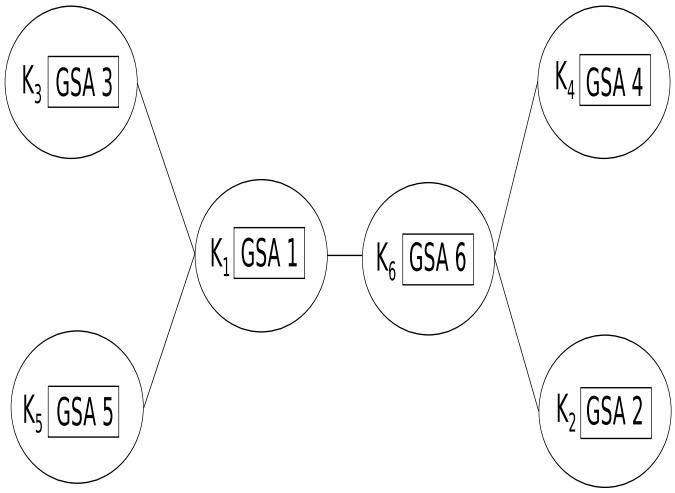

Figura 3.4: Jerarquía de Iolus, donde K_i es la llave del usuario i y los circulos indican el alcance de los GSA (Group Security Agent).

En el esquema MARKS propuesto por Briscoe (Briscoe, 1999) se realiza una división de la información multimedia en secciones (conjunto de paquetes) y a cada una de estás se le asigna una llave diferente. Las llaves son hechas con un árbol binario generado con base en una semilla (ver figura 3.5), para crear el árbol se llevan a cabo los siguientes pasos:

1. Primero se elige la profundidad D del árbol, ésta define el número de llaves N, esto es $N = 2^D$.

2. La semilla $S_{0,0}$ se elige aleatoriamente y se coloca en la raíz del árbol. Donde $S_{i,j}$, i es la profundidad de la llave en el árbol y j es el número de llave en el nivel i.

3. Dado $S_{0,0}$ el nodo raíz para generar los nodos descendientes izquierdo $(S1,0 = b(ls(S_{0,0})))$ y derecho $(S1,1 = b(rs(S_{0,0})))$ se utiliza una función basada en $S_{0,0}$.

4. Se aplica el paso anterior para los siguientes niveles.

A los usuarios que deseen ingresar al grupo se les envía una semilla para obtener la

llave. Desgraciadamente, una vez repartidas las semillas no puede ser aceptado ningún otro usuario y tampoco puede salir ninguno.

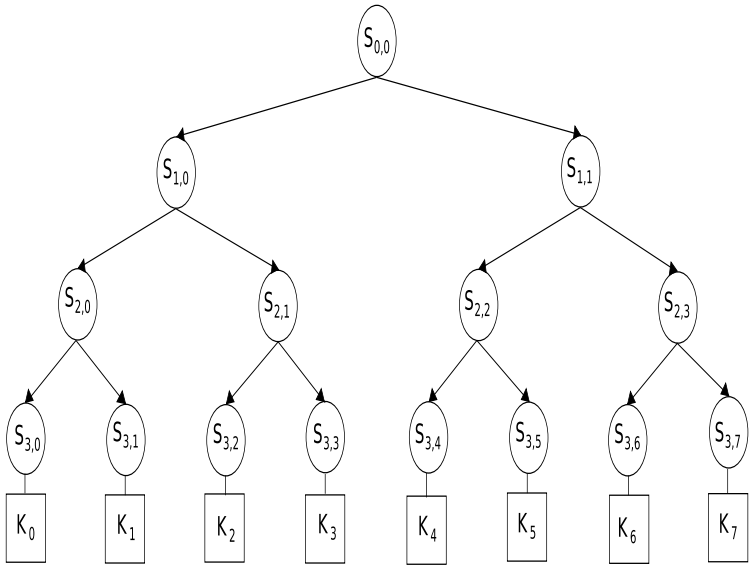

Figura 3.5: Esquema del modelo MARKS, donde $S_{x,y}$ representa la semilla asociada a la llave K_y y la jerárquia de las semillas indica derivación entre ellas.

El modelo Kronos propuesto por Setia (Setia y otros, 2000), se basa en un intercambio de llaves durante el periodo en el que entran o salen usuarios. El Network Time Protocol (NTP) es usado para sincronizar los relojes (relojes internos de la computadoras) y asegurar que el ingreso o salida de usuarios no afecte al sistema.

El envío de las llaves se realiza mediante un canal seguro a todos los administradores de subgrupos, la generación de todas las llaves basada en un algoritmo de cifrado E, una llave maestra K y una semilla R_0, es decir, $R_1 = E_K(R_0)$. La principal desventaja de éste es que si alguien conociera una llave podría conocer todas las demás fácilmente.

En el cuadro 3.3 se presenta un comparativo de las características principales de los controles descentralizados.

Esquema	Llave independiente	Generación local de llaves	Llaves y Datos en un mismo paquete	Actualización de llaves
RFC1946	Si	No	Si	No
Iolus	Si	Si	No	Si
MARKS	No	No	Si	Si
KRONOS	No	No	Si	No

Cuadro 3.3: Comparación de los modelos descentralizados

3.3. Administrador de Llaves Basado en un Control Distribuido

Algunos de los modelos propuestos usando este tipo de control son descritos a continuación.

El intercambio de llaves de D-H en un grupo fue propuesto por Steiner (Steiner y otros, 1996). Este modelo es una extensión del algoritmo original de D-H (ver figura 3.6) sólo que en éste es capaz de soportar operaciones en grupo. En este caso los miembros del grupo intercambian numeros por medio de mensajes, para seleccionar dos números primos q y α. Posteriormente con la ecuación $\alpha x_1 x_2 x_3 ... x_i \ mod \ q$, se fusionan las llaves de todos los usuarios $(x_1 \ x_2 \ x_3 ... x_i)$para formar la llave del grupo. Para minimizar la ejecución de la fusión el grupo es dividido en subgrupos los cuales se encargan de la selección de los números primos q,α y de distribuir la llave del grupo.

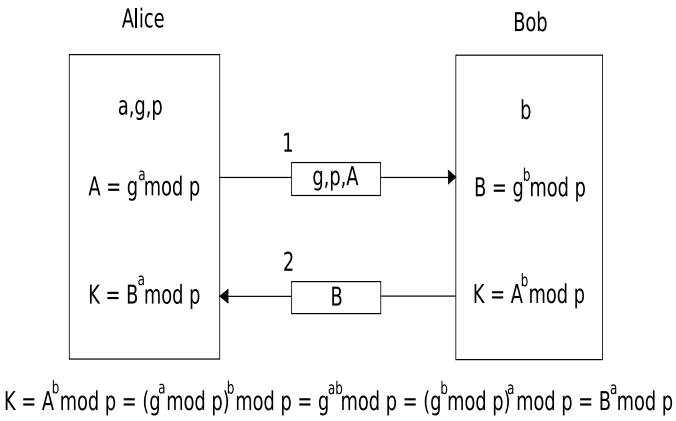

Figura 3.6: Intercambio de llaves de D-H, donde a,g,p y b representan números primos. A,B y K representan valores cálculados como se indica en la figura.

Otro esquema, el Distributed Logical Key Hierarchy (DLKH) es presentado por Rodeh (Rodeh y otros, 2000) y está basado, como su nombre lo indica, en el modelo LKH. En DLKH se utiliza un esquema jerárquico para los usuarios y gracias a esto se reduce el tiempo de procesamiento con un total de $log_2 n$, así como también la cantidad de llaves que almacena cada usuario con sólo $log_2 n$ llaves. La jerarquía está hecha por árboles binarios, en la que se agrupan todos los usuarios tal como en el modelo LKH.

Sin embargo, el trabajo de generación de llaves es realizado por todos los miembros del grupo, cada uno en su subgrupo correspondiente y posteriormente se realiza la combinación de las llaves de cada subgrupo para finalmente formar la llave general de todos los subgrupos, ver figura 3.7.

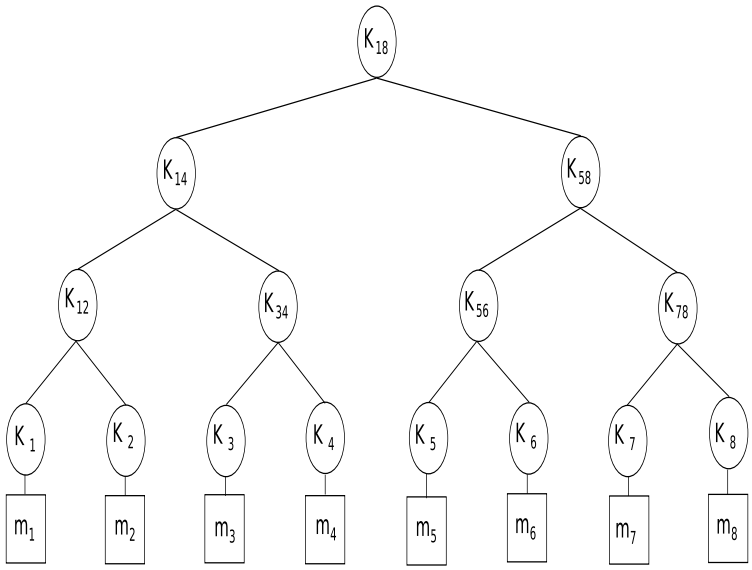

Figura 3.7: Árbol del modelo DLKH, donde m_x representa un usuario, K_x representa la llave asociada a un usuario x, la jerarquía indica una derivación ascendente entre las llaves. Las llaves $K_{x,y}$ representan la llve de un subgrupo.

Perrig y Kim (Kim y otros, 2000) proponen un modelo que usa el LKH combinándolo con el algoritmo de D-H; el objetivo es reducir el número de llaves usadas por cada usuario. La diferencia con los modelos antes mencionados es que la llave del grupo se genera a partir del algoritmo de D-H aplicado a cada nivel del árbol binario. Cada vez que un nuevo grupo es creado se realiza una operación *merge*, es decir, se realiza la fusión de los nuevos miembros con el árbol actual, ver figura 3.8.

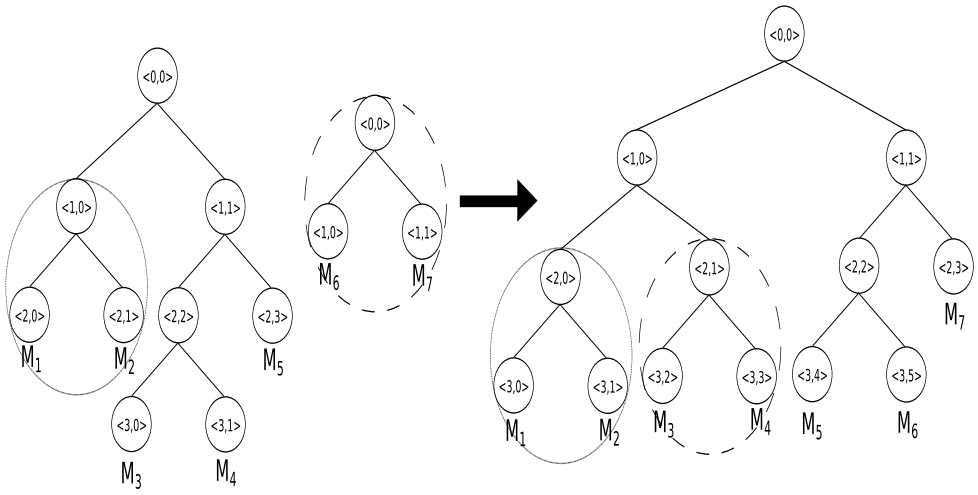

Figura 3.8: Operación merge. En la parte izquierda se muestra un grupo al que se le aplica la operación merge con un subgrupo (líneas discontinuas). En la parte derecha se muestra el resultado de la operacion merge al grupo. En esta figura M_x representa un usuario y $< w, z >$ es la llave asociada a este.

Finalmente en el cuadro comparativo 3.4 se comparan las características de los modelos distribuidos presentados.

Esquema	Número de ciclos	Mensajes con *Multicast*	Mensajes con Unicast	Llave basada en D-H
D-H en grupos	n	n	$n-1$	Si
DLKH	3	1	n	No
D-H LKH	$log_2 n$	$log_2 n$	0	Si

Cuadro 3.4: Comparación de los modelos distribuidos

3.4. Comparación

Este mecanismo propuesto se encuentra ubicado entre los modelos centralizados tomando en cuenta que la generación de la llave que comunica a todos los usuarios en un SDM es realizadas por un nodo. Sin embargo, la distribución de las llaves se realiza con un modelo descentralizado, debido a esto nuestro trabajo es dinámico y escalable.

Para nuestro mecanismo el proceso de ingreso está definido con base en un modelo distribuido, proporcinando con esto una robustez en el acceso, característica principal en los esquemas distribuidos.

En general, podemos decir que nuestro mecanismo es un híbrido, debido a que contiene características de los tres tipos de control descritos en secciones anteriores.

En el cuadro 3.5 se presenta una tabla comparativa de los trabajos hechos y nuestra propuesta, señalando su ubicación con base en los diferentes tipos de control que usan los administradores de llaves.

Trabajos	Dinámico	Escalable	# llaves para cifrar	Tiempo real
Wong	Si	Si	1 en total	No
Canetti	Si	No	1 en total	Si
Ganapathi	Si	No	1 en total	No
Briscoe	No	Si	1 en total	No
Setia	Si	Si	1 en total	Si
Rojas	Si	Si	1 por paquete	Si

Cuadro 3.5: Ubicación de nuestro trabajo

Del cuadro 3.5, la característica que hace notar nuestra aportación es el *número de llaves para cifrar* teniendo una llave por paquete de información. Esta característica se logra debido al modelo de nuestro mecanismo que sugiere una generación local de las llaves asignadas a los paquetes.

3.5. Conclusión

En el presente capítulo se describieron los principales trabajos hechos sobre los administradores de llaves en los diferentes tipos de control, también se presentaron tablas de comparación dependiendo de las características a evaluar de los diferentes administradores de llaves. Nuestra propuesta no puede ser clasificada específicamente en alguno de estos grupos debido a las características con las que cuenta. En el capítulo siguiente se describirá a detalle estas características.

Capítulo 4

Desarrollo del Mecanismo Propuesto

En este capítulo se presenta nuestro modelo del sistema del Mecanismo Multicast para la Generación y Distribución de Llaves para SDM en Tiempo Real (MUGENEDIS), se describe cada algoritmo y la forma en que éste funciona e interactúa con los otros. Como se mencionó en los capítulos anteriores, la importancia de los administradores de llaves se debe a la seguridad que proporcionan por medio de las llaves que estos generan y distribuyen. Dichos administradores tienen la responsabilidad de garantizar una eficiente generación de llaves y una distribución de llaves sólo a los usuarios que tienen derecho a ello. Existen diversas características necesarias y algunas otras deseables para la generación y la distribución de llaves en un SDM en tiempo real.

4.1. Modelo del Sistema

El modelo del sistema que consideramos en este trabajo tiene las características de un SDM que se ejecuta en tiempo real bajo un tipo de comunicación *multicast*. Un SDM está formado de n procesos que se comunican por medio de mensajes con la finalidad de transmitir multimedia de forma segura. Para nuestro trabajo multimedia la integran el

audio y el video únicamente. Al decir que el SDM se ejecuta en tiempo real se refiere a que los paquetes de información tienen un tiempo de vida(cierto tiempo después de su llegada, antes de ser destruidos). En este trabajo *multicast* indica la entrega de mensajes de uno a muchos, ver figura 4.1.

Dentro del SDM no se cuenta con un reloj global, se manejan procesos concurrentes y se tiene tolerancia a fallas independientes. La transmisión de multimedia entre los procesos está propensa a la pérdida o el retraso de paquetes de información, porque dicha transmisión se realiza en un medio de comunicación parcialmente fiable. También existe la posibilidad de que alguien escuche el medio de transmisión debido a que no es posible garantizar una seguridad total.

4.2. Descripción General del MUGENEDIS

Los procesos en el SDM están organizados en grupos llamados *dominios*, ver figura 4.1. Dentro de cada *dominio* la comunicación *multicast* es confidencial entre los miembros que lo forman. El domino de control contiene al proceso denominado *Host* y a procesos llamados *miembros organizadores*. Los dominios organizadores contienen a un miembro organizador y a varios procesos denominados *usuarios*. Finalmente, el dominio de transmisión es donde están todos los procesos llamados usuarios compartiendo multimedia.

El proceso *Host* tiene una capacidad de procesamiento mayor comparada con los demás procesos en el SDM, debido a que genera constantemente variables robustas que son utilizadas por todos los procesos. Está al pendiente de la entrada y salida de los miembros organizadores y de la solicitud de entrada de nuevos usuarios.

Los *miembros organizadores* no necesitan tantos recursos de procesamiento como el *Host*, inicialmente están inactivos hasta que sean invocados por el *Host* para dar servicio a usuarios. Los *miembros organizadores* transmiten mensajes del *Host* hacia los usuarios.

Las personas que están conectadas en alguna red tienen que solicitar el ingreso al SDM y una vez que estos ingresan se convierten en *usuarios*. Para este trabajo esto se

lleva a cabo en una intranet y la solicitud debe ser enviada al *Host*. Las computadoras de los usuarios deben tener los recursos mínimos para el manejo de multimedia.

Cada vez que exista un cambio, que entren o salgan usuarios, en algún dominio organizador el miembro organizador debe encargarse de esta tarea. El miembro organizador debe hacer llegar esta información a todos los demás miembros organizadores y éstos a su vez al resto de los usuarios.

Los tres elementos principales del mecanismo propuesto son la estructura organizacional, la distribución de llaves y la transmisión segura. Al inicializar el funcionamiento del SDM, el *Hos*t se activa esperando la petición de las personas que quieran pertenecer al SDM, con esto se inicia la formación de la estructura organizacional. Durante esta formación y en el caso que entren o salgan procesos del SDM, sea miembro organizador o usuarios se realiza la distribución de llaves.

Dependiendo de la forma en que se generan y distribuyen las llaves, de manera distribuida o centralizada, éstas son agrupadas en llaves de dominio y llave principal.

Las llaves de dominio son utilizadas para la comunicación confidencial dentro de los dominios de control y de organización. Las llaves de dominio son: La llave KS, es usada en el dominio de control. Las KD_i llaves (con $i = 1, ..., n$, donde n es el número de miembros organizadores) para cada dominio organizador i.

La llave principal sirve para compartir multimedia de forma segura en el dominio de transmisión, es denotada por KP. En nuestro trabajo la transmisión segura indica garantizar la confidencialidad en el envío de paquetes de información.

4.2.1. Descripción de los Procesos que Intervienen en MUGENEDIS

En la sección del modelo del sistema se mencionaron tres distintos procesos: el *Host*, los *miembros organizadores* y los *usuarios*. En los siguientes párrafos describimos las tareas que cada uno cumple.

El *Host* es un proceso que inicia el MUGENEDIS y lleva a cabo las siguientes tareas:

- Coordinar a los miembros organizadores que a su vez coordinan a los usuarios del SDM.
- Generación de variables para el algoritmo BBS que generará las llaves para los paquetes.
- Dar a conocer el algoritmo de cifrado y el algoritmo para la generación de llaves.
- Asignar a los usuarios un subgrupo para inscribirse.

Los *Miembros Organizadores* son procesos que llevan a cabo las siguientes tareas:

- Dar de alta y baja a los usuarios.
- Enviar mensajes entre el *Host* y los *usuarios*.

Los *Usuarios* en el SDM son procesos que solicitarán el ingreso al sistema mediante un mensaje al *Host* principal, quien se encargará de atender su petición.

4.3. Estructura Organizacional del MUGENEDIS

La organización de los procesos en MUGENEDIS se basa en dominios. En los dominios los procesos tienen una comunicación confidencial que esta basada en el algoritmo de Diffie-Hellman. Existen niveles que sirven para identificar a los procesos con mayor importancia en el mecanismo. Estos niveles indican la dependencia de los procesos, ver figura 4.1.

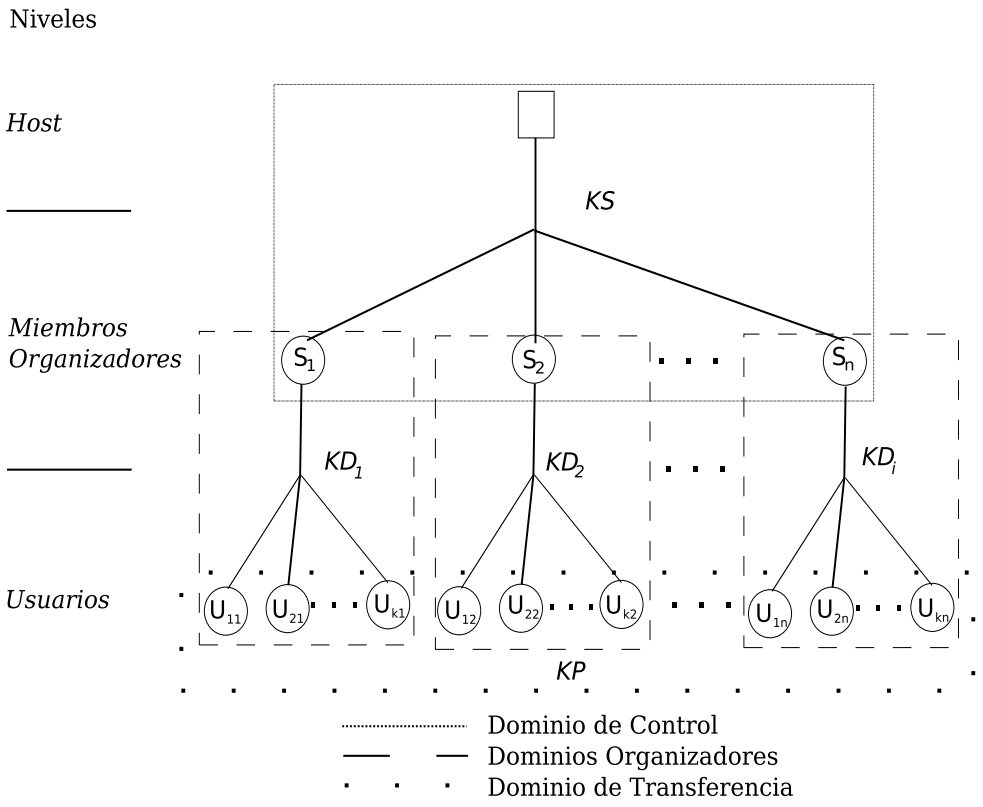

Figura 4.1: Estructura Organizacional del Mecanismo Propuesto, donde S_x representan los miembros organizadores, $U_{w,z}$ representan los usuarios, KS representa la llave para el dominio de control, los KD_y representan las llaves para los dominios organizadores y KP representa la llave para el dominio de transferencia. La columna de la izquierda indica los niveles de la jerarquía de nuestro mecanismo

El primer nivel es ocupado por el proceso *Host* que se encarga de recibir las peticiones de entrada a los *usuarios*. Así como de generar la llave KP.

En el segundo nivel están los *miembros organizadores*. Estos juegan un papel muy importante, ya que minimizan la carga de trabajo al *Host* para la organización del SDM. También apoyan al *Host* para distribuir la llave KP.

En el último nivel se encuentran los usuarios, estos se comunican con apoyo de la llave KP enviada por el miembro organizador del dominio al que pertenezcan.

Esta estructura hace que nuestro mecanismo sea escalable, acepta tantos usuarios como sea posible sin afectar su funcionamiento, debido al manejo eficiente de llaves.

4.4. Generación y Distribución de las Llaves en el MUGENEDIS

Como se mencionó anteriormente las llaves de dominio se utilizan para la comunicación confidencial entre los procesos en un mismo dominio. Estas son creadas de manera distribuida con base en el algoritmo extendido de Diffie-Hellman hecho por Steiner, ver (Steiner y otros, 1996).

Para el dominio de control se utiliza la llave KS y para cada dominio organizacional se utiliza una llave KD_i, ver figura 4.1.

La llave principal KP es utilizada en el dominio de transferencia para la transmisión de multimedia entre los usuarios y es creada de forma centralizada. El *Host* es el encargado de esta tarea. Los miembros organizadores son los encargados de distribuirla.

A continuación se describen los métodos para la generación y distribución de las llaves utilizadas en el mecanismo propuesto.

4.4.1. Generación y Distribución de la Llave KS

Para generar y distribuir esta llave es necesaria la interacción del *Host* con los miembros organizadores por medio del algoritmo extendido de Diffie-Hellman, ver (Steiner y otros, 1996).En el algoritmo 1 se presenta la generación y distribución de la llave KS.

Algoritmo 1 Generación y Distribución de la Llave KS

Función: La generacion y distribucion de la llave KS, usada en el dominio de control.

Datos de entrada: Las llaves K_{S_i} de los *miembro organizador*

Datos de salida: La llave KS.

Inicio

{ El *Host* H define h, α y p números primos grandes}

1.- H genera h, α y p;

{ El *Host* crea $K_H = \alpha^h \ mod \ p$}

2.- $K_H = \alpha^h \ mod \ p$;

{ El *Host* envía α, p y K_H a los *miembros organizadores*}

3.- $H \ \leftrightarrow \ S_i$: { α, p, K_H };

{Todos los S_i generan su h_i y crean su llave $K_{S_i} = \alpha^{h_i} \ mod \ p$}

{Todo *miembro organizador* generará un h_i para crear su $K_{S_i} = \alpha^{h_i} \ mod \ p$, y lo envía al *Host* y al resto de *miembros organizadores*}

4.- $S_i \ \leftrightarrow \ [H, D_i]$: { K_{S_i} };

{Tanto H como los S_i generan un mismo *KS*, donde i' representa el resto de los usuarios en D_i, $i \neq i'$}

5.- H genera $KS = \prod (K_{S_i})^h \ mod \ p$;

6.- S_i genera $KS = (\prod (K_{S_{i'}}) * K_H)^{h_i} \ mod \ p$;

Fin

En la figura 4.2 se ilustra la relación que tiene el *Host* y los miembros organizadores.

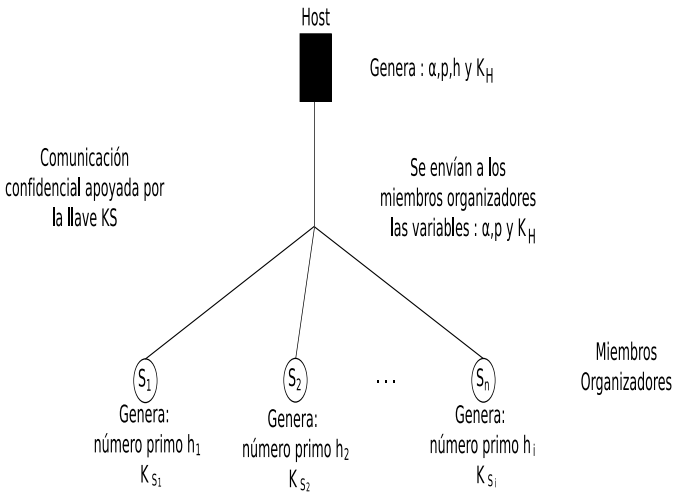

Figura 4.2: Relación del *Host* con los miembros organizadores, donde α,p,h y K_H son generadas por el *Host*, S_i representan los miembros organizadores y K_{S_i} la llave asociada a cada usuario i. La llave K_{S_i} es formada a partir de α,p y K_H enviadas por el *Host* y un número primo h_i generado por S_i.

4.4.2. Generación y Distribución de las Llaves KD_i

Las llaves KD_i se utilizan para cifrar mensajes entre los miembros organizadores y los usuarios. Para crear estas llaves los miembros organizadores y los usuarios usan las variables α y p generadas por el *Host*. Estas llaves son generadas una vez que el *Host* y los miembros organizadores estén activos. El *Host* recibirá todas las peticiones de ingreso al sistema y asignará a un miembro organizador para que sea el encargado de dar de alta al usuario, es decir, de agregar a dicho usuario a un subgrupo, y por consecuencia al sistema.

La generación y distribución de una KD_i es consecuencia del proceso de ingreso y salida de un D_i de algún usuario dado un miembro organizador. En el algoritmo 2 se muestra los dos casos en que el KD_i es creado, dado un usuario y un miembro organizador, y en el algoritmo 3 se muestra el proceso de salida de usuarios.

Algoritmo 2 Ingreso de usuarios

Función: Permite el ingreso de los usuarios o los miembros organizadores a un domino organizador o al domino de control respectivamente.

Datos de entrada: La llave K_{U_x} del usuario o miembro organizador que desee ingresar.

Datos de salida: La nueva llave KD_i para el dominio organizador o la nueva llave KS para el dominio de control.

Inicio

{Dado un S_i miembro organizador y su llave $K_{S_i} = \alpha^{h_i} \ mod \ p$, donde α y p enviados previamente por el $Host$}

1.- S_i tiene su h_i y $K_{S_i} = \alpha^{h_i} \ mod \ p$;

{$Host$ envía un mensaje a algún miembro organizador para atender al usuario que desea entrar al sistema}

2.- $H \ \rightarrow \ S_i$: { Petición de ingreso al sistema del usuario U_x};

{El miembro organizador envía sus datos en un mensaje *M* al usuario interesado, dicho paquete contiene K_{S_i},α y p}

3.- $S_i \ \rightarrow \ U_x$: {α, p, K_{S_i} };

{Para todo usuario U_x interesado en ingresar al grupo es necesario que se cree un número primo grande h_x para generar su $K_{U_x} = \alpha^{h_x} \ mod \ p$}

4.- U_x genera su h_x y crea su $K_{U_x} = \alpha^{x} \ mod \ p$;

{El usuario U_x envía su K_{U_x} al miembro organizador y a los otros usuarios en el subgrupo}

5.- $U_x \ \leftrightarrow \ [D_i, S_i]$: { K_{U_x} };

{Tanto S_i como los usuarios en el subgrupo generan un KD_i}

6.- S_n genera $KD_i = \prod (K_{U_{ki}})^{h_i} \ mod \ p$;

7.- U_{ki} genera $KD_i = (\prod (k_{U_{ki}}) * K_{U_x})^{h_x} \ mod \ p$;

Fin

Algoritmo 3 Salida de usuarios

Función: Permite la salida de los usuarios o los miembros organizadores de un domino organizador o del domino de control respectivamente.

Datos de entrada: El número del usuario o miembro organizador que desee salir.

Datos de salida: La nueva llave KD_i para el dominio organizador o la nueva llave KS para el dominio de control.

Inicio

{Algún usuario $U_{ki'}$ en algún D_i solicita su salida al miembro organizador S_i al que está asociado}

1.- $U_{ki'} \rightarrow S_i$: { Solicitud de salida };

{El miembro organizador solicita a los usuarios restantes en el grupo una nueva llave local, es decir, la generación de un nuevo número primo y con base en este formar la nueva KD_i del dominio}

2.- $S_i \leftrightarrow U_{ki}$: { nueva llave local };

{Todos los usuarios del D_i generan un nuevo número primo h_{ki} y crean su $K_{U_{ki}} = \alpha^{h_{ki}} \ mod \ p$}

3.- Todos los U_{ki} generan un nuevo h_{ki} y crean su $K_{U_{ki}} = \alpha^{h_{ki}} \ mod \ p$;

{Los usuarios que continuarán en el D_i se envían y le envían al miembro organizador su nueva llave $K_{U_{ki}}$}

4.- $U_{ki} \leftrightarrow [D_i, S_i]$: { $K_{U_{ki}}$ };

{Tanto el miembro organizador como los usuarios que siguen en el subgrupo generan la nueva KD_i}

{Tanto S_i como los usuarios en el subgrupo generan un KD_i}

5.- S_i genera $KD_i = \prod(K_{U_{ki}})^{h_i} \ mod \ p$;

6.- U_{ki} genera $KD_i = (\prod(K_{U_{ki}}) * (K_{S_i}))^{h_{ki}} \ mod \ p$;

Fin

En la figura 4.3 se observa la relación entre los miembros organizadores y los usuarios

asociados a ellos.

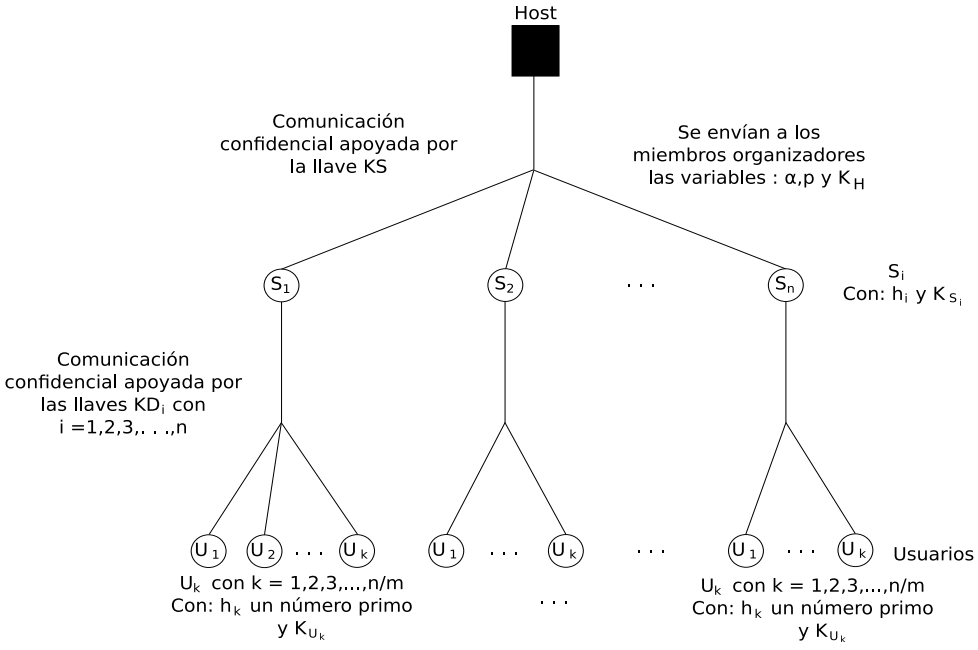

Figura 4.3: Relación de los usuarios con los miembros organizadores, donde α y p son generadas por el *Host*, S_i representan los miembros organizadores, K_{S_i} la llave asociada a cada S_i, U_j representan los usuarios. La llave K_{U_j} es formada a partir de α,p y K_{S_i} enviadas por cada S_i y un número primo h_j generado por U_j.

4.4.3. Generación y Distribución de la Llave KP

En esta sección se define el protocolo para generar la llave principal KP. Dicha llave es un paquete con variables necesarias para generar números pseudoaleatorios con el algoritmo BBS. Para esto es necesario definir variables necesarios para el algoritmo BBS. Con base en este algoritmo se generarán secuencias de bits pseudo aleatorios (llaves) que le serán proporcionadas al algoritmo de cifrado AES el cual cifrará los paquetes multimedia, también ver figura 4.4.

1. El *Host* generará las variables para el algoritmo BBS, dos números primos bp y bq

congruentes con $3\ mod\ 4$, también generará un número primo bs que será la semilla del generador BBS, bs tiene que ser primo relativo con $bn = bp * bq$. Una vez hecho esto se formará un paquete con (bp, bq, bn, bs) llamado KP.

2. El Host envía el paquete KP cifrado con la llave KS; $H \leftrightarrow S_i : \{KP\}_{KS}$; a todos los miembros organizadores.

3. Los miembros organizadores se encargán de hacer llegar este paquete a los usuarios por medio de un mensaje cifrado con la respectiva llave K_{S_i} de cada subgrupo $S_i \leftrightarrow U_{ki}$ en el D_i:$\{KP\}_{KD_i}$;

4. Dado que todos los usuarios en el sistema tienen los elementos necesarios para generar las mismas llaves con las que se cifran los paquetes, es decir, el paquete KP que descifrarán los paquetes enviados, todos conocerán el contenido de éstos.

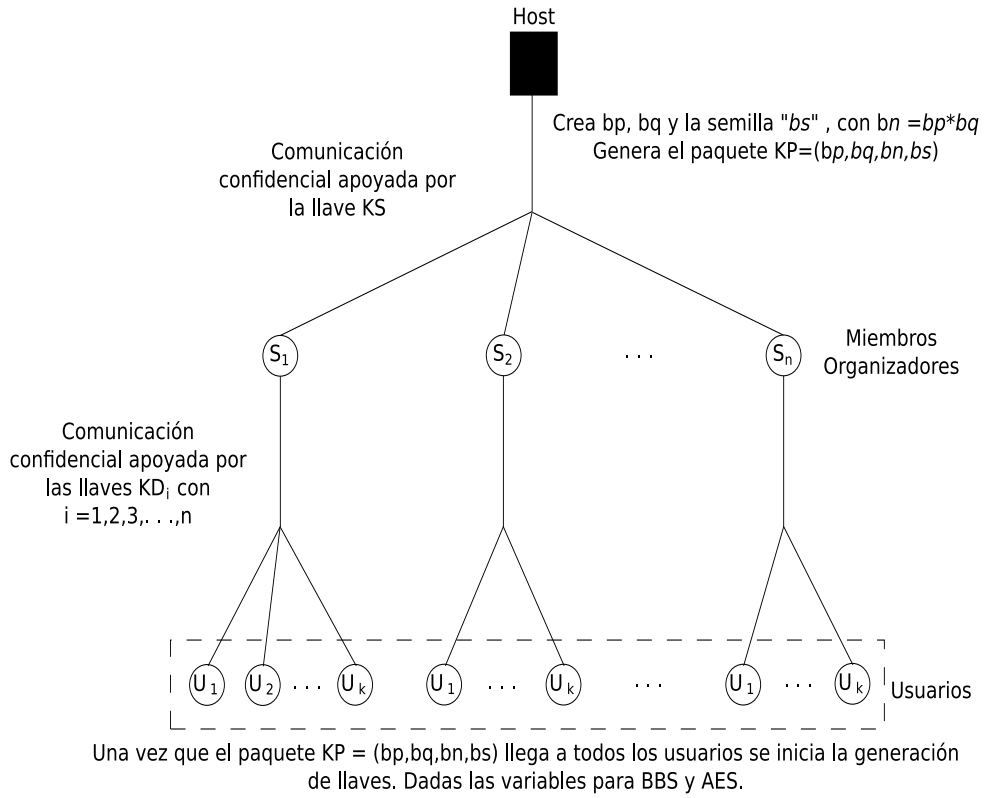

Figura 4.4: Esquema de la comunicación entre los usuarios, esta se lleva por medio de la llave KP que es generado por el *Host* y es distribuida por los miembros organizadores S_i para llegar finalmente con todos los usuarios U_j

4.5. Transmisión Segura

A continuación, se describe la forma en que se realiza la transmisión segura en nuestro mecanismo. Esta transmisión es realizada una vez que todos los *usuarios* tienen la llave principal KP, ver figura 4.4.

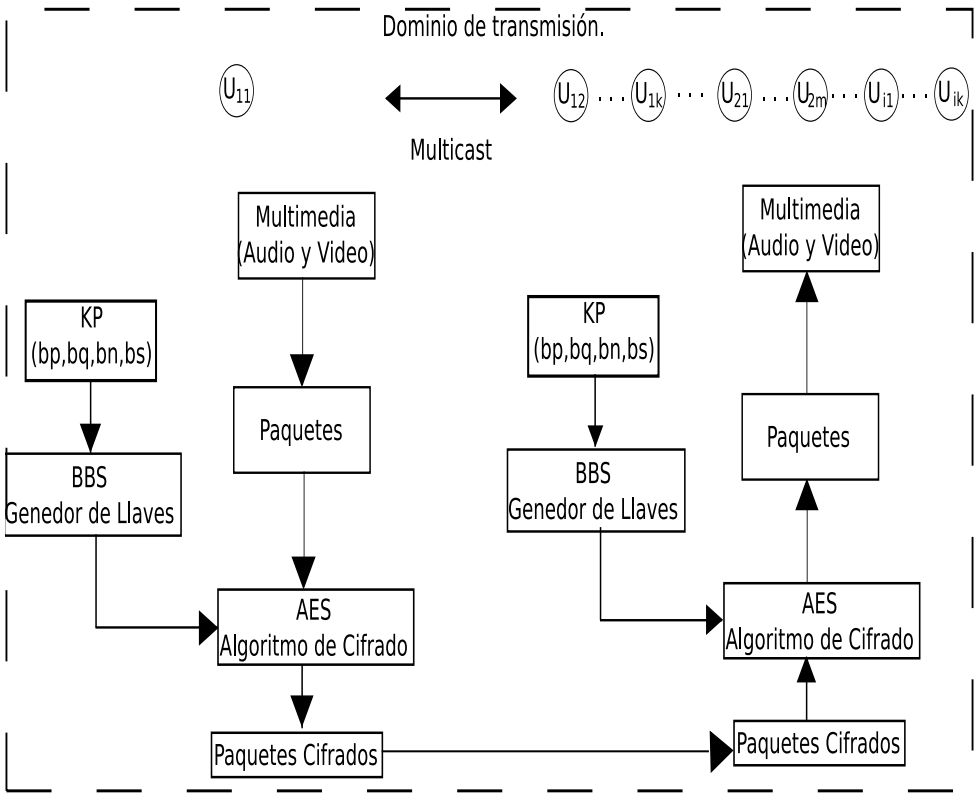

Figura 4.5: Esquema de transmisión segura, donde $U_{x,y}$ representa un usuario, $U_{1,1}$ envía información al resto de los usuarios.

Dada la figura 4.5 se tiene el siguiente procedimiento:

1. Generar paquetes $MU_1, MU_2, ..., MU_r$ de información multimedia a compartir por parte de algún usuario U_j.

2. Para cada paquete MU_r, se crea una llave key_r (secuencia de bits pseudoaleatorios) utilizando la extensión del algoritmo BBS que genera secuencias de bits pseudo aleatorios, ver (Blum y Micali, 1984).

3. Se cifran los paquetes MU_r con las llaves key_r usando el algoritmo AES.

4. Se envían los paquetes cifrados usando *multicast*.

5. Para cada usuario receptor $U_{j'}$, éste recupera la llave con la que fue cifrado el paquete MU_r.

6. Una vez que $U_{j'}$ recupera la llave key_r que le corresponde a dicho paquete, descifra el paquete MU_r utilizando el algoritmo AES.

La propiedad del algoritmo BBS para acceder a cualquier número de una secuencia dando el índice de éste, permite que se tenga tolerancia a la pérdida o el retraso de paquetes.

4.5.1. Generación de Llaves para Cifrar Paquetes Multimedia

La generación es sencilla y robusta debido a las características del algoritmo BBS, ver sección 2.5.1. Como se ha comentado en secciones anteriores, AES es el algoritmo utilizado para cifrado y descifrado.

Para la generación de llaves para cifrar paquetes multimedia por un usuario es necesario hacer lo siguiente:

- Crear paquetes de la información multimedia.
- Iniciar la generación de números (llaves) con el algoritmo BBS dado el paquete $KP = (bp, bq, bn, bs)$.
- Formar secuencias de bits pseudo aleatorios (llaves) tomando los $log_2(log_2\ L)$ bits menos significativos por cada número generado (recomendado en (Blum y Micali, 1984)) por el algoritmo BBS donde L es la longitud en bits de los números pseudo aleatorios.
- Cifrar los paquetes con el algoritmo AES y una de las secuencias de bits pseudo aleatorios (llaves).
- Enviar los paquetes cifrados a todos los usuarios en el SDM por medio del protocolo de transferencia en tiempo real (RTP).

En la figura 4.6 se muestra un esquema de cómo se generan las llaves y se cifran los paquetes.

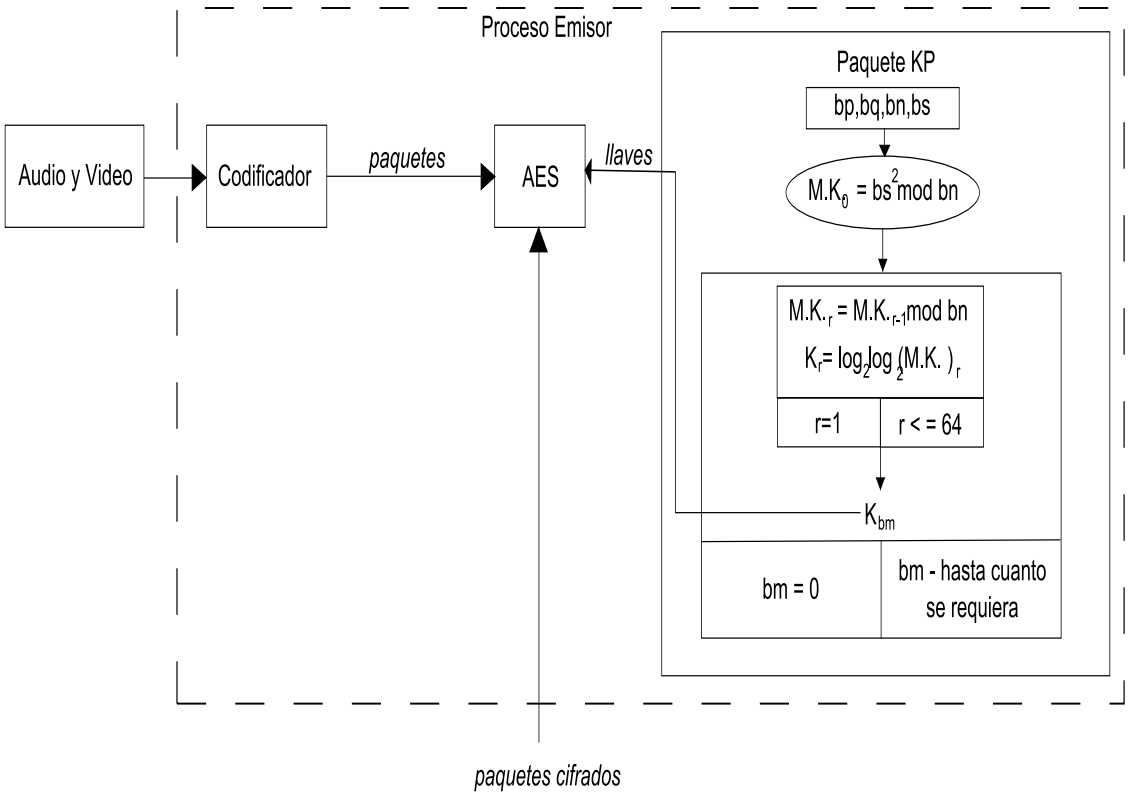

Figura 4.6: Generación de llaves para cifrar,Donde, M.K. representa el miembro de la llave, es decir, algunos de los bits que conforman la llave, K_r representa la llave de cifrardo de 128 bits de tamaño.

El algoritmo BBS es usado para la generación de llaves para el cifrado de paquetes multimedia. El tamaño de las llaves que se propone es de 128 bits. Esta medida se toma con base en los requerimientos del algoritmo AES.

El algoritmo AES toma como llave una secuencia de 128 bits. El algoritmo BBS generará esta secuencia de bits, por lo tanto es necesario 64 números pseudo aleatorios y tomar de estos algunos bits, a lo más $log_2(log_2\ L)$ (donde L es la longitud en bits de los

números pseudo aleatorios) para formar la llave de 128 bits.

Un factor importante que se considera es que el número de llaves diferentes debe ser mayor o igual al de los paquetes a enviar, es decir, a cada paquete se le asigna una llave única. El período máximo t del algoritmo BBS está dado por $t \geq n^{3/4+\delta}$, con $\delta > 0$, ver (Friedlander y Shparlinski, 2001).

Tomando en cuenta que se eligen 64 números para generar una llave, por lo tanto el número mínimo de llaves es aproximadamente de $t/64$.

Esto se debe a que cuando los generadores llegan a cumplir con su período máximo vuelven a repetir valores, por esta razón es posible tener diferentes combinaciones, es decir, el valor de 64 puede cambiar pero siempre se formarán las mismas llaves.

4.5.2. Recuperación de Llaves para el Descifrado de Paquetes Cifrados Multimedia

El proceso para la generación de llaves que realizan los procesos receptores de los paquetes cifrados se muestra en la figura 4.7.

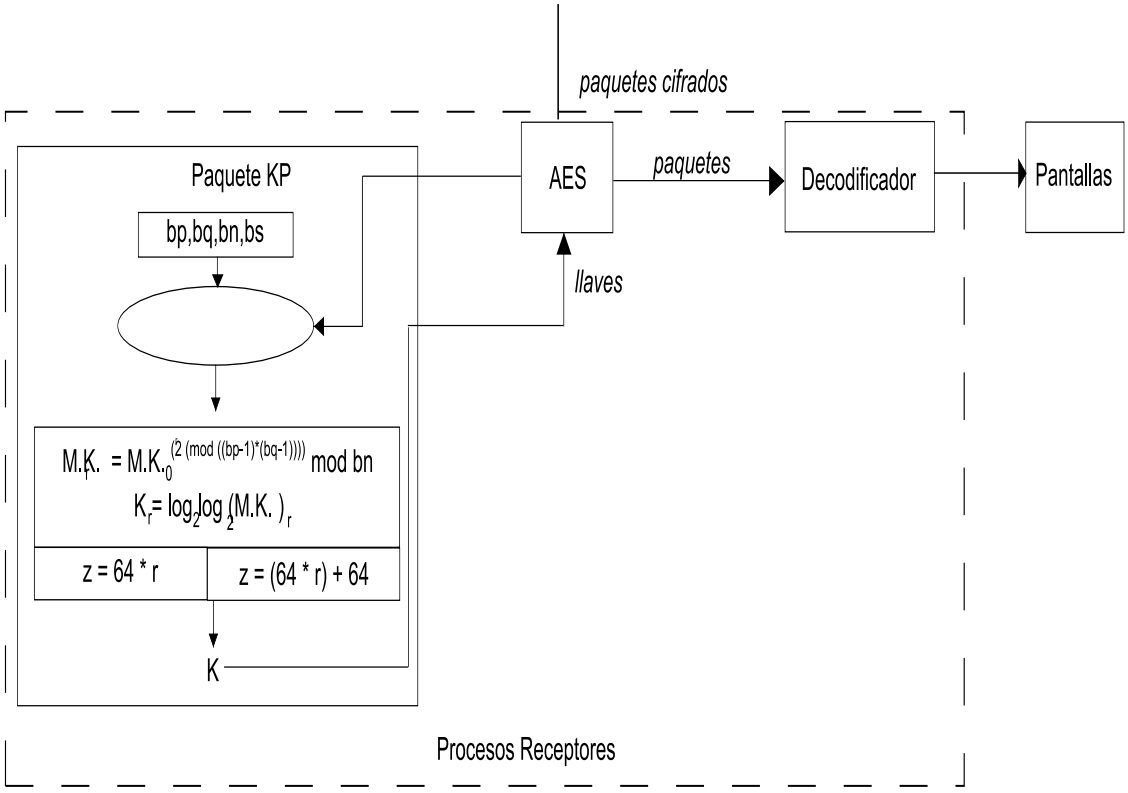

Figura 4.7: Generación de llave para el descifrado de paquetes cifrados, donde M.K. es el miembro de la llave, es decir, bits que conforman la llave. K_r es la llave para descifrar.

A continuación, se describe el proceso de recuperación de llaves:

- Se reciben los paquetes y se lee de éstos el índice de la llave con que fueron cifrados, el índice es el número de paquete recibido.
- Este índice es enviado al algoritmo BBS, que genera la llave correspondiente.
- Se forma la secuencia de bits pseudo aleatorios (llaves) tomando los $log_2(log_2\ L)$ bits menos significativos por cada número generado por el algoritmo BBS, donde L es la longitud en bits de los números pseudo aleatorios.
- Se descifran los paquetes con el algoritmo AES y las secuencias de bits pseudo aleatorios (llaves).

4.5.3. Ejemplo de la Transmisión Segura

Para este ejemplo, los dominios están formados y todos los usuarios tienen la llave principal $KP = (bp, bq, bn, bs)$. Con los siguientes valores:

$bp = 9993843791755596287$

$bq = 12516003245446525127$

$bn = 125082981332098649743333780500013403449$

$bs = 23971744642465329911631708053282913930$

Definimos el tamaño de la llave para cada paquete que será de 128 bits. Pero en este ejemplo sólo tomaremos 4 números pseudoaleatorios para generar cada llave, para tener un ejemplo claro. Es decir, tomaremos 32 bits de cada número dado que $32 * 4 = 128$.

Supongamos que el usuario U_j desea transmitir el texto *Mecanismo Multicast*. El primer paso es dividir el texto en paquetes, para nuestro ejemplo tenemos que $MU_1 = Mecanismo$ y $MU_2 = Multicast$.

Ahora tenemos que generar una llave para cada paquete key_1 y key_2. Se generan 8 números pseudoaleatorios con BBS con las ecuaciones 4.1 y 4.2.

$$num_0 = bs^2 \ mod \ bn \tag{4.1}$$

$$num_{i+1} = (num_i)^2 \ mod \ bn \tag{4.2}$$

Los cuadros 4.1 y 4.2, muestran los números generados.

	Número en decimal
1	5256958920322527981
2	6834522368856372840
3	1339670386506889651
4	17895753748262096694
5	17150014081811665482
6	2286061860784088893
7	643115294829314357
8	4564412208686005206

Cuadro 4.1: Números generados por BBS expresados en decimal

De estos números se toman los 32 bits menos significativos (los que estan más a la derecha) para formar las llaves que usará AES para el cifrado, ver Cuadro 4.3.

	Número en binario
1	10010001111010001111000010010010001010010100101001100101110110 1
2	10111101101100100011001111000100011010100100101001101100110100 0
3	10010100101110111011100011110010111111101101001111001101100 11
4	11111000010110100111110100110111111000010000011100001111001101 10
5	11101110000000010001011011110001010101110101010011010010010010 10
6	11111101110011011100100100110011110101001101010100111001111 01
7	10001110110011001101111101000000100111101100000000010011010 1
8	11111101011000000011001011100001110011001011110111011110101 10

Cuadro 4.2: Números generados por BBS expresados en binario

	Llave
key_1	00010100101001010011001011101101
	00110101001001010011011001101000
	01011111110110100111100110110011
	11100001000001110000111100110110
key_2	01010111010101001101001001001010
	01111010100110101010011100111101
	00001001111011000000000100110101
	01110011001011110111101111010110

Cuadro 4.3: Llaves generados por el mecanismo propuesto para cifrar los paquetes

Se cifran los paquetes MU_1 y MU_2 dando como resultado lo siguiente:

Paquete MU_1 cifrado, $\{MU_1\}_1$ = 000cc035baca96382d6af8c5a6d3072c

Paquete MU_2 cifrado, $\{MU_2\}_2$ = daf51d32b1ca185f8d5df7289462aa4f

Estos paquetes cifrados (expresados en hexadecimal) son enviados a todos los *usuarios*, usando el protocolo RTP. Dentro de este paquete hay un identificador (número de paquete) que nos sirve para recuperar la llave del paquete.

Una vez que algun usuario $U_{j'}$ quiere ver el contenido, necesita obtener de cada paquete el identificador e iniciar el proceso de recuperación. Dado que tenemos KP y con la ayuda de la ecuación 4.3:

$$num_i = (num_0^{(2^i (mod((bp-1)*(bq-1))))}) \ mod \ bn \tag{4.3}$$

Si llega primero $\{MU_2\}_2$, i tomará los valores de $(5, 6, 7, 8)$ y utilizando la fórmula 4.3 se recobrará los siguientes números:

$5 = 17150014081811665482$

$6 = 2286061860784088893$

$7 = 643115294829314357$

$8 = 4564412208686005206$

Entonces se formará la llave del paquete 2:

$key_2 = 0101011101010100110100100100101001111010100110101010011100111101$
$0000100111101100000000010011010101110011001011110111101111010110$

Finalmente, introducimos el $\{MU_2\}_2$ y su llave key_2 al algoritmo AES y obtenemos el texto enviado.

$MU_2 = Multicast$

Este mismo procedimiento de recuperación de llaves es aplicado para el paquete $\{MU_1\}_1$ para obtener $MU_1 = Mecanismo$

4.6. Conclusión

Dado el tipo de generador de llaves BBS y la forma de distribución en nuestro mecanismo, este es capaz de proporcionar una llave diferente para cada paquete, además el almacenamiento de las llaves no será necesario dada la capacidad de BBS para generar una llave key_l en un momento dado, proporcionando la independencia de paquetes, atributo deseable en un sistema parcialmente fiable.

Capítulo 5

Implementación y Análisis

En el presente capítulo se describen los análisis realizados al mecanismo *multicast* para la generación y distribución de llaves para SDM en tiempo real. Mostramos los resultados obtenidos y finalmente se realiza una comparación con los modelos más importantes existentes.

5.1. Análisis de la Generación y Distribución de Llaves

Las principales métricas para evaluar la eficiencia de los administradores de llaves están definidas en función de la cantidad de llaves y de los mensajes que estos utilizan. El número de llaves usadas en el sistema repercute en los requerimientos de almacenamiento de cada miembro dentro del mismo. El número de mensajes para la actualización del sistema cada vez que alguien entra o sale es un factor importante causante de los retrasos en el funcionamiento del sistema.

Para llevar a cabo el análisis del almacenamiento de llaves y del número de mensajes, se toman en cuenta las cuatro arquitecturas más importantes en los tipos de control, ver (Padmavathi y Annadurai, 2006).

5.1.1. Modelos a Evaluar

Arquitectura de nodos **estrella**. Debido a que todos los modelos de administradores de llaves dependen de un nodo que generará y distribuida llaves. Todos los modelos basados en el control centralizado se basan en esta arquitectura.

Arquitectura de **árbol** binario, donde los nodos que controlan a los subgrupos están entre el segundo y penúltimo nivel del árbol. La ubicación de estos nodos también refleja el número de subgrupos formados. Algunos de los modelos basados en el control descentralizado se basan en esta arquitectura.

Arquitectura de **árbol con agrupamiento** muy parecida al del árbol pero este no es necesariamente binario, es decir, puede tener más de dos nodos asociados, otra característica es que no existen administradores de subgrupos. Algunos de los modelos basados en el control descentralizado se basan en esta arquitectura, así como también algunos modelos de control distribuido.

Arquitectura **híbrida** basada en un árbol, donde el segundo nivel contiene a nodos llamados miembros estáticos que controlan a subgrupos que están organizados por medio de arboles binarios, es decir, los usuarios están agrupados en arboles binarios y controlados por los miembros estáticos. Los modelos basados en un control descentralizado y de control distribuido se basan en esta arquitectura.

El modelo de nuestra propuesta está basado en una arquitectura de árbol con tres niveles, donde el primer nivel corresponde al *Host* que controla a todo el grupo, en el segundo nivel se encuentran los organizadores de los subgrupos, finalmente en el último nivel se encuentran los usuarios del sistema.

5.1.2. Evaluación sobre el Número de Llaves usadas en el SDM

Dados n usuarios en el sistema con comunicación *multicast* y tomando como las llaves necesarias para establecer una comunicación confidencial en los diferentes modelos, se obtuvo lo siguiente:

1. Arquitectura tipo estrella (control centralizado). Tomando en cuenta la estructura en la que todos están conectados entre sí y que se necesitan 2 llaves por miembro, por lo tanto se requiere $2 * n$ llaves.

2. Arquitectura tipo árbol binario (control descentralizado). Tomando los nodos auxiliares que agrupan a los usuarios en árboles binarios se tiene que para usuario, éste debe contener $[1 + log_d n]$ llaves, por lo tanto se requieren $[(1 + log_d n) * (n)]$ llaves.

3. Arquitectura de árbol con i-aristas (control distribuido). Dada la estructura en que los n usuarios son divididos en subgrupos de tamaño m distribuidos en d niveles, se tienen $[2 + log_d(n/m)]$ llaves por usuario. Por lo tanto para todos los usuarios es necesario tener $[n * (2 + log_d(n/m))]$ llaves en el sistema.

4. Arquitectura híbrida (crossbreed). Para esta arquitectura se toma en cuenta primeramente las llaves usadas por los miembros estáticos. El número de miembros estáticos está en función de la variable r que denota el número de enrutadores entre el usuario y el Host. El número de miembros estáticos es $n/(1 + r)$ y dado que requieren 2 llaves por miembro estático, entonces se necesitan $[2 * (n/(1 + r))]$. Para cada usuario es necesario tener $(1 + log_d r)$ llaves y como existen $(n/(1 + r))$ miembros estáticos, por lo tanto para todos los usuarios es necesario tener $[(n/(1 + r)) * (1 + log_d r)]$ llaves. Finalmente, para conocer el total llaves en el sistema se suman las llaves de los miembros estáticos y las llave de los usuarios que dan un total de $[2 * (n/(1 + r)) + (n/(1 + r)) * (1 + log_d r)]$ llaves usadas en el sistema.

5. Propuesta. En la propuesta es necesario tomar en cuenta el número de dominios organizadores que está definido como (n/m), también al *Host* y el dominio de transferencia. Finalmente, decimos que se necesitan $[2+(n/m)]$ llaves en el sistema, esto es (n/m) de los dominios organizadores, uno para el dominio de control y otro para el dominio de transmisión.

En el cuadro 5.1 se presentan los valores evaluados así como también los resultados obtenidos . Dichos resultados están también representados por medio de una gráfica en la figura 5.1.

Arquitecturas Evaluadas	Parámetros				**Resultados**
	d	n	m	r	
Estrella		100			200
		200			400
		300			600
		400			800
		500			1000
		600			1200
		700			1400
		800			1600
		900			1800
		1000			2000
Árbol	2	100			764.3
	2	200			1728.7
	2	300			2768.6
	2	400			3857.5
	2	500			4982.8
	2	600			6137.2
	2	700			7315.8
	2	800			8515.0
	2	900			9732.4
	2	1000			10965.7
Árbol con agrupamiento	2	100	4		764.3
	2	200	4		1728.7

Cuadro 5.1 – Continua en la siguiente página...

Arquitecturas Evaluadas	Parámetros				**Resultados**
	d	n	m	r	
Árbol con agrupamiento	2	300	4		2768.6
	2	400	4		3857.5
	2	500	4		4982.8
	2	600	4		6137.2
	2	700	4		7315.8
	2	800	4		8515.0
	2	900	4		9732.4
	2	1000	4		10965.7
Híbrido	2	100	4	3	114.6
	2	200	4	3	229.2
	2	300	4	3	343.8
	2	400	4	3	458.4
	2	500	4	3	573.1
	2	600	4	3	687.7
	2	700	4	3	802.3
	2	800	4	3	916.9
	2	900	4	3	1031.6
	2	1000	4	3	1146.2
Rojas		100	4		27
		200	4		52
		300	4		77
		400	4		102
		500	4		127
		600	4		152
		700	4		177

Cuadro 5.1 – Continua en la siguiente página...

Arquitecturas Evaluadas	Parámetros				**Resultados**
	d	n	m	r	
Rojas		800	4		202
		900	4		227
		1000	4		257

Cuadro 5.1: Datos de la evaluación respecto al número de llaves, donde, d indica que los árboles son binarios, n el número de usuarios, m el tamaño de los subgrupos y r el numero de enrutadores entre los usuarios y el *Host* principal.

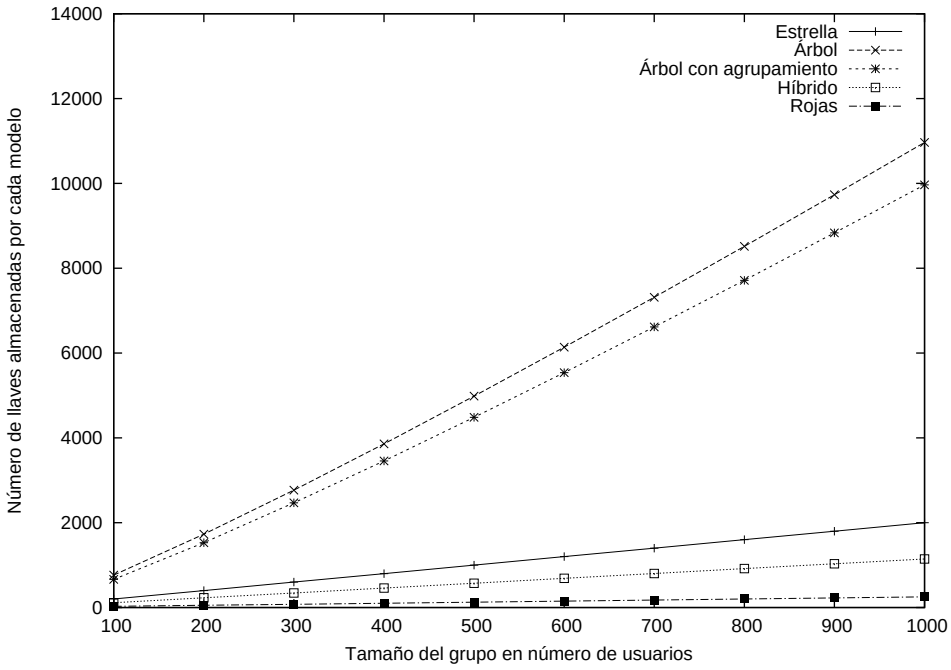

Figura 5.1: Evaluación respecto al número de llaves de diferentes arquitecturas

5.1.3. Evaluación sobre el Número de Mensajes para la Actualización de Llaves

Se realizó un análisis basándose en la cantidad de mensajes para la actualización de llaves bajo las arquitecturas presentadas en la sección 5.1.1. En el cuadro 5.2 se presentan los resultados, los cuales son graficados en la figura 5.2. El cálculo de la cantidad de mensajes se describe a continuación.

1. Arquitectura tipo estrella (control centralizado). Dada su estructura se necesitan n mensajes para que el *Host* haga llegar la nueva llave principal a todos los usuarios.

2. Arquitectura tipo árbol binario (control descentralizado). Para este modelo son necesarios $d * (n - 1)$ mensajes debido a la formación del arbol binario.

3. Arquitectura de árbol con i-aristas (control distribuido). Este esquema necesita $(n/m) + [(n/m) * d * (m - 1)]$ mensajes para repartir la llave.

4. Arquitectura híbrida (crossbreed). Dada esta arquitectura es necesario mandar $n/(1+r) + [n/(1+r) * (d * r)]$ mensaje.

5. Propuesta. Es necesario envíar $(n/m) + n$ mensajes para distribuir la nueva KP

Arquitecturas Evaluadas	Parámetros				**Resultados**
	d	n	m	r	
Estrella		100			100
		200			200
		300			300
		400			400
		500			500
		600			600

Cuadro 5.2 – Continua en la siguiente página...

Arquitecturas Evaluadas	Parámetros				**Resultados**
	d	n	m	r	
Estrella		700			700
		800			800
		900			900
		1000			1000
Árbol	2	100			198
	2	200			398
	2	300			598
	2	400			798
	2	500			998
	2	600			1198
	2	700			1398
	2	800			1598
	2	900			1798
	2	1000			1998
Árbol con agrupamiento	2	100	25		196
	2	200	25		392
	2	300	25		588
	2	400	25		784
	2	500	25		980
	2	600	25		1176
	2	700	25		1372
	2	800	25		1568
	2	900	25		1764
	2	1000	25		1960
Híbrido	2	100	25	3	175

Cuadro 5.2 – Continua en la siguiente página...

Arquitecturas Evaluadas	Parámetros				**Resultados**
	d	n	m	r	
Híbrido	2	200	25	3	350
	2	300	25	3	525
	2	400	25	3	700
	2	500	25	3	875
	2	600	25	3	1050
	2	700	25	3	1225
	2	800	25	3	1400
	2	900	25	3	1575
	2	1000	25	3	1750
Rojas		100	25		104
		200	25		208
		300	25		312
		400	25		416
		500	25		520
		600	25		624
		700	25		728
		800	25		832
		900	25		936
		1000	25		1040

Cuadro 5.2: Datos de la evaluación respecto al número de mensajes, donde d indica que los árboles son binarios, n el número de usuarios, m el tamaño de los subgrupos y r el número de enrutadores entre los usuarios y el Host principal.

Como puede verse en el cuadro 5.2 nuestro mecanismo supera a la mayoría de las arquitecturas pero no a la arquitectura de Estrella en cuanto al criterio del número de mensajes. Sin embargo, en el esquema de estrella los mensajes son enviados por un solo nodo a diferencia del MUGENEDIS donde la distribución es realizada por el *Host* con apoyo de los miembros organizadores dando con esto escalabilidad a nuestro mecanismo.

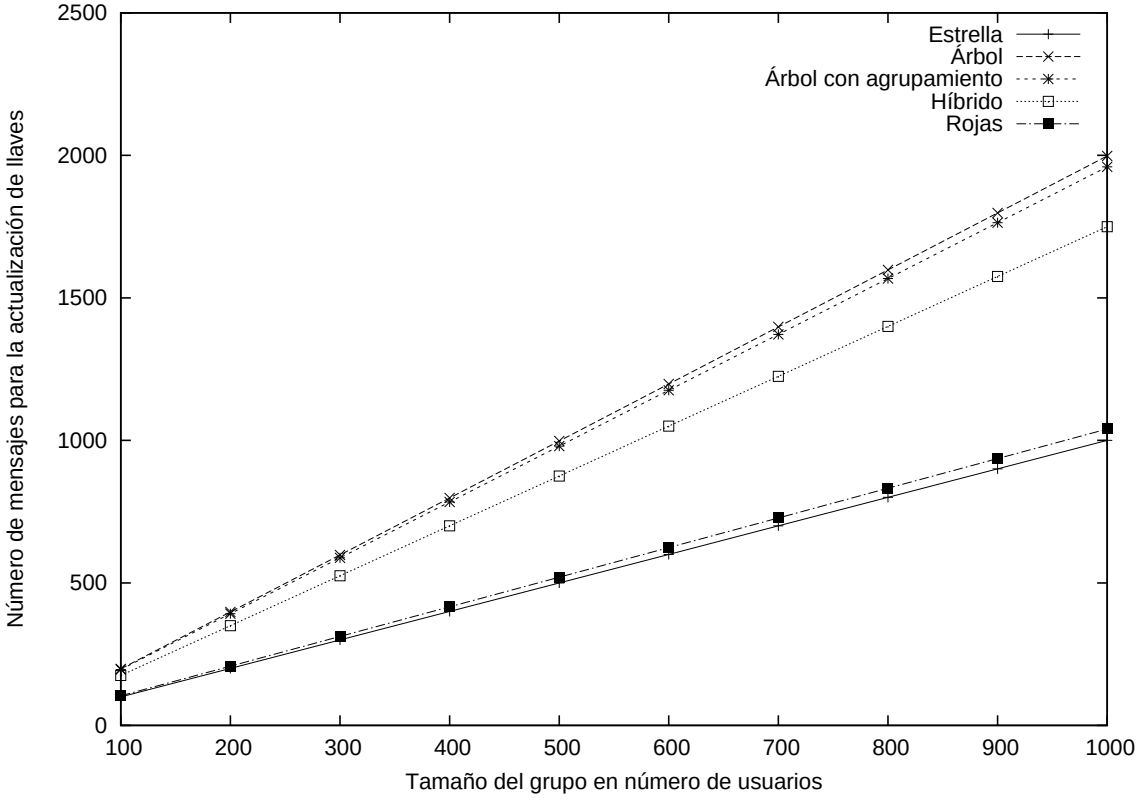

Figura 5.2: Evaluación respecto al número de mensajes de diferentes arquitecturas

5.2. Simulación del mecanismo

El ambiente donde se implementó el mecanismo propuesto fue en una red local. Los procesos *Host* y *Miembros Organizadores* se ejecutaron en una misma PC. Los *usuarios*

fueron ejecutados en otra PC. Este escenario se observa en la figura 5.3.

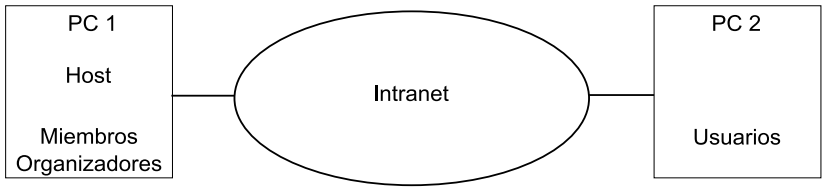

Figura 5.3: Escenario de la implementación, donde PC1 contiene los procesos *Host* y los *miembros organizadores*, PC2 contiene a los procesos llamados *usuarios*

5.2.1. Herramientas para la Simulación

Los programas para la simulación fueron hechos en el lenguaje Java dado que ofrece las ventajas de un adecuado manejo de multimedia. El API JMF (Java Media Framework) proporciona los codecs y protocolos de transporte en tiempo real necesarios para la simulación de nuestro mecanismo, ver (Sun~Microsystems, 2003). Los codecs que se utilizaron para el video fueron el MPEG y el H.323 para el audio. El protocolo RTP (Real Time Protocol) se utilizó para la transmisión. La implementación fue hecha en una PC con un procesador Intel Core 2 Duo a 1.86 MHz con memoria RAM de 2 GB. Se utilizó una cámara web Philips SPC 900NC con una resolución de 1280x960 pixeles y una captura de 30 frames por segundo.

5.2.2. Resultados de la Simulación

La simulación se realizó en dos escenarios. EL primero, con el proceso completo que consiste en la codificación, cifrado, envío, recepción, descifrado y decodificación, ver figura 5.4. El segundo, con un proceso que excluía el cifrado y descifrado con el objeto de ver si existia un retraso considerable para que nuestro mecanismo no fuera factible en un sistema de tiempo real. Por último, los resultados de ambos escenarios fueron comparados.

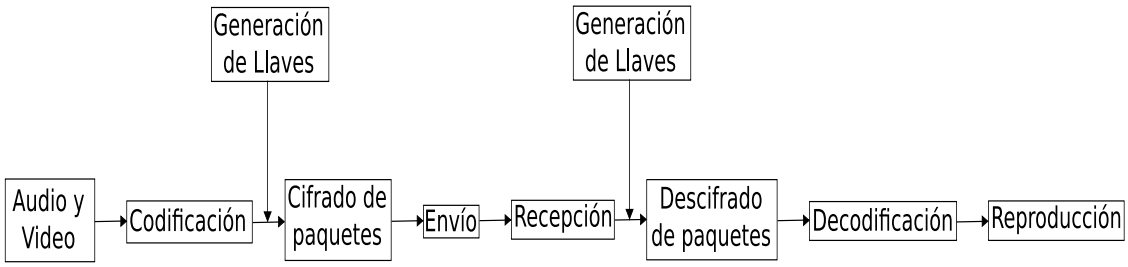

Figura 5.4: Proceso de Envío-Recepción de Multimedia

La simulación que realizamos fue una sesión de 60 segundos de transmisión en tiempo real. Los resultados del tiempo promedio de cifrado y descifrado se presentan en el cuadro 5.3.

	Tiempo de cifrado	Tiempo de descifrado
Una llave para todos los paquetes	98	105
Una llave por paquete	110	135

Cuadro 5.3: Comparación de tiempos en milisegundos

El tiempo en que se expresan los resultados del cuadro 5.3 está dado en milisegundos. Como puede verse, el tiempo extra que necesita nuestro mecanismo que utiliza una llave por paquete comparado con otro que sólo usa una llave en toda la sesión, la diferencia es mínima.

5.3. Conclusiones

Hechas las pruebas podemos decir que nuestros resultados son satisfactorios. Se alcanzó el objetivo de crear una llave por paquete además de que en la comparación del número de llaves usadas en el sistema, nuestro mecanismo se comportó mejor que los modelos de estrella, árbol, árbol con agrupamiento e híbrido, ver Cuadro 5.1. Finalmente, la simulación realizada aporta que nuestro mecanismo puede ser ejecutado en tiempo real.

Capítulo 6

Conclusiones y trabajo futuro

En este capítulo se presentan las conclusiones a las que se llegó después de haber realizado este trabajo, así mismo se dan algunas propuestas para mejorar el mecanismo propuesto.

6.1. Conclusiones Finales

Analizando el funcionamiento del mecanismo propuesto dentro del SDM en que se probó podemos enunciar las siguientes conclusiones:

1. Con la generación de una llave independiente por paquete se alcanza el objetivo de soportar la transmision de audio y video en tiempo real, la cual se caracteriza por trabajar bajo condiciones de fiabilidad y orden parcial.

2. Hacemos notar que la generación de llaves independientes que se logran con el algoritmo BBS, conserva la complejidad computacional de acceso a la información.

3. Podemos asegurar a partir de los resultados mostrados en el presente documento que se logró la escalabilidad del mecanismo propuesto. Dicha escalabilidad es un resultado directo de la estructura del mecanismo, el cual permite que la distribución

de las llaves sea realizada por el *Host* y los miembros organizadores, con el propósito de disminuir el trabajo del *Host*.

4. Hacemos notar también que el número de llaves en nuestro mecanismo para tener una comunicación confidencial en el SDM es menor comparado con otros modelos existentes, como estrella, árbol e híbrido.

6.2. Trabajos Futuros

A continuación se mencionan algunas de las propuestas de trabajo a futuro para el mecanismo propuesto en esta tesis.

La aplicación de nuestro mecanismo hacia arquitecturas móviles. Visualizamos que los miembros organizadores pueden ser mapeados dentro de la arquitectura de telefonía celular en estaciones base con dispositivos móviles como usuarios.

Otro trabajo a futuro es el desarrollo de un mecanismo de autenticación que pueda ser utilizado en SDMs con características similares a las del presente trabajo. Para garantizar que los usuarios sean quienes dicen ser al momento de adherirse al grupo de participantes.

Determinar un tamaño ideal de los subgrupos. Esto sería una contribución muy importante debido a que se tendría una mayor escalabilidad en el sistema. Los algoritmos distribuidos de generación de llaves, tal como el usado en el presente trabajo consideran la participación de cada uno de los integrantes del grupo para la generación de las respectivas llaves. Un problema abierto es el de determinar cuál es el número de miembros adecuados por subgrupo, ya que si el número de miembros es pequeño de hasta diez, se incrementa el número de mensajes, pero si éste número es grande del orden de cientos, el costo computacional es prohibitivamente alto.

Apéndice A

Algoritmo Blum Blum Shub en Java

```
package com.modp.random;
import java.util.Random;
import java.security.SecureRandom;
import java.math.BigInteger;

public class BlumBlumShub implements RandomGenerator {

    /* Variables auxiliares*/
    private static final BigInteger two = BigInteger.valueOf(2L);
    private static final BigInteger three = BigInteger.valueOf(3L);
    private static final BigInteger four = BigInteger.valueOf(4L);
    /* Variables auxiliares*/
    private BigInteger n;
    private BigInteger state;

    /* Método para obtener los números p y q*/
```

```
private static BigInteger getPrime(int bits, Random rand) {
BigInteger p;
while (true) {
    p = new BigInteger(bits, 100, rand);
    if (p.mod(four).equals(three))
    break;
}
return p;
}

/* Método para obtener los números p y q*/
public static BigInteger generateN(int bits, Random rand) {
BigInteger p = getPrime(bits/2, rand);
BigInteger q = getPrime(bits/2, rand);
while (p.equals(q)) {
    q = getPrime(bits, rand);
}
return p.multiply(q);
}

/* Método para definir el número de bits para los números*/
public BlumBlumShub(int bits) {
this(bits, new Random());
}

/* Método para elegir aleatoriamente bits*/
public BlumBlumShub(int bits, Random rand) {
this(generateN(bits, rand));
```

```
}

/* Método para obtener los números pseudoaleatorios*/
public BlumBlumShub(BigInteger n) {
this(n, SecureRandom.getSeed(n.bitLength() / 8));
}

/* Método para obtener los números pseudoaleatorios*/
/* dado una semilla*/
public BlumBlumShub(BigInteger n, byte[] seed) {
this.n = n;
setSeed(seed);
}

/* Método para obtener una semilla*/
public void setSeed(byte[] seedBytes) {
BigInteger seed = new BigInteger(1, seedBytes);
state = seed.mod(n);
}

/* Método para obtener los bits menos significativos*/
public int next(int numBits) {
int result = 0;
for (int i = numBits; i != 0; --i) {
    state = state.modPow(two, n);
    result = (result << 1) | (state.testBit(0) == true ? 1 : 0);
}
return result;
```

```
    }

    public void main(String[] args) {
    SecureRandom r = new SecureRandom();
    System.out.println("Generando una semilla alearotia");
    r.nextInt();

    System.out.println("Generando n");
    int bitsize = 128;
    BigInteger nval = BlumBlumShub.generateN(bitsize, r);

    byte[] seed = new byte[bitsize/8];
    r.nextBytes(seed);

    BlumBlumShub bbs = new BlumBlumShub(nval, seed);

    System.out.println("Generando una llave de 128 bytes");
    for (int i = 0; i < 128; ++i) {
        System.out.println(bbs.next(8));
    }

    BlumBlumShub bbs2 = new BlumBlumShub(bitsize);

    BlumBlumShub bbs3 = new BlumBlumShub(nval);
    }
}
```

Bibliografía

Alfred J. Menezes, Paul C. Van Oorschot y Vanstone, Scout A.: *Handbook of Applied Cryptography*. CRC Press, 1996.

Ballardie, A.: «Scalable Multicast Key Distribution». *Request For Comments 1949, Internet Engineering Task Force*, 1996.

Banerjee, S. y Bhattacharjee, B.: «Scalable secure group communication over IP multicast». *JSAC Special Issue on Network Support for Group Communication*, 2002, **20(8)**, pp. 1511–1527.

Blum, Lenore; Blum, Manuel y Shub, Michael: «A Simple Unpredictable Pseudo-Random Number Generator». *SIAM Journal on Computing*, 1986, **15**, pp. 364–383.

Blum, M. y Micali, Silvio: «How to Generate Cryptographically Strong Sequences of Pseudo Random Bits». *SIAM Journal on Computing*, 1984, **13**, pp. 850 – 864.

Borko Furht, Darko Kirovski: *Multimedia Security Handbook*. CRC, 2004.

Briscoe, B.: «MARKS: Multicast key management using arbitrarily revealed key sequences». *In Proceedings of the 1st InternationalWorkshop on Networked Group Communication*, 1999.

Chan, Kin-Ching y Chan, S.-H. Gary: «Key Management Approaches to Offer Data Confidentiality for Secure Multicast». *IEEE Network*, 2003, **17**, pp. 30 – 39.

EASTLAKE, D.; SCHILLER, J. y CROCKER, S.: «Randomness Requirements for Security». *Network Working Group*, 2005, pp. 26–27.

ESKICIOGLU, AHMET M.: «Multimedia Security in Group Communications: Recent Progress in Key Management, Authentication and watermarking». *International Conference on Communications and Computer Networks*, 2002, pp. 125–133.

FRIEDLANDER, JOHN B. y SHPARLINSKI, IGOR E.: «On the Distribution of the Power Generator». *Mathematics of Computation*, 2001, **70(236)**, pp. 1575–1589.

GENTLE, JAMES E.: *Random Number Generation and Monte Carlo Methods*. Statistics and Computing. Springer, 2nd[a] edición, 2004.

HARDJONO, THOMAS y DONDETI, LAKSHMINATH R.: *Multicast and Group Security*. Artech House Publishers, 2003.

HARNEY, H. y MUCKENHIRN, C.: «Group Key Management Protocol (GKMP) Architecture». *Request For Comments 2094, Internet Engineering Task Force*, 1997.

HUA CHU, HAO; QIAO, LINTIAN y NAHRSTEDT, KLARA: «A Secure Multicast Protocol with Copyright Protection». *ACM SIGCOMM Computer Communications Review*, 2002, **32(2)**.

JUDGE, PAUL y AMMAR, MOSTAFA: «Security Issues and Solutions in Multicast Content Distribution: A Survey». *IEEE Network*, 2003, **17**, pp. 30–36.

KEI WONG, CHUNG; GOUDA, MOHAMED y LAM, SIMON S.: «Secure Group Communications Using Key Graphs». *IEEE/ACM Transactions on Networking*, 2000, **8(1)**, pp. 16 – 30.

KIM, Y.; PERRIG, A. y TSUDIK, G.: «Simple and fault-tolerant key agreement for dynamic collaborative groups». *In Proceedings of the 7th ACM Conference in Computer and Communication Security*, 2000, pp. 235–241.

KOBLITZ, N. y MENEZES, ALFRED J.: «A Survey of Public-Key Cryptosystems». *SIAM REVIEW*, 2003, **46(4)**, pp. 599–634.

MITTRA, S.: «Iolus: A Framework for Scalable Secure Multicasting». *Proceedings of ACM SIGCOMM'97*, 1997, pp. 277–288.

PADMAVATHI, GANAPATHI y ANNADURAI, SAMUKUTTY: «Storage Efficient Key Management Technique for Secure Multicasting». *Journal of Information Science and Engineering*, 2006, **22(3)**, pp. 675–689.

RAFAELI, SANDRO y HUTCHISON, DAVID: «A Survey of Key Management for Secure Group Communication». *ACM Computing Surveys*, 2003, **35(3)**, pp. 309–329.

RODEH, O.; BIRMAN, K. y DOLEV, D.: «Optimized group rekey for group communication systems». *Proceedings of ISOC Network and Distributed Systems Security Symposium*, 2000.

SCHNEIER, BRUCE: *Applied Cryptography: Protocols, Algorithms, and source code in C*. John Wiley and Sons, Inc., 2nd[a] edición, 1995.

SETIA, SANJEEV; KOUSSIH, SAMIR; JAJODIA, SUSHIL y HARDER, ERIC: «Kronos: A Scalable Group Re-Keying Approach for Secure Multicast». *IEEE Symposium on Security and Privacy*, 2000, pp. 215–228.

STALLING, WILLIAM: *Cryptography and Network Security: Principies and Practices*. Prentice Hall, 3ra[a] edición, 2002.

STEINER, M.; TSUDIK, G. y WAIDNER, M.: «Diffie-Hellman key distribution extended to group communication». *In SIGSAC Proceedings of the 3rd ACM Conference on Computer and Communications Security*, 1996, pp. 31–37.

STEINMETZ, RALF y NAHRSTEDT, KLARA: *Multimedia Systems*. Springer, 1ra[a] edición, 2004.

Sun Microsystems, Inc.: «Java Media Framework API», 2003. http://java.sun.com/products/java-media/jmf/

Tanenbaum, Andrew S. y van Steen, Maarten: *Distributed Systems: Principles and Paradigms*. Prentice Hall, 1ra^{a} edición, 2002.

Waldvogel, M.; Caronni, G.; Sun, D.; Weiler, N. y Plattner, B.: «The VersaKey framework: Versatile group key management». *IEEE Journal on Selected Areas in Communications*, 1999, **17**, pp. 1614–1631.