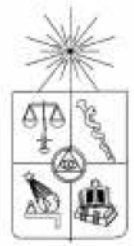

UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

SEGURIDAD DE DATOS EN SISTEMAS COLABORATIVOS MÓVILES

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL EN COMPUTACIÓN

FABIÁN GONZALO SOTO FERNÁNDEZ

PROFESOR GUÍA:
SERGIO OCHOA DELORENZI

MIEMBROS DE LA COMISIÓN:
ALEJANDRO HEVIA ANGULO
JENS HARDINGS PERL

SANTIAGO DE CHILE
Enero 2010

RESUMEN DE LA MEMORIA PARA
OPTAR AL TITULO DE INGENIERO
CIVIL EN COMPUTACION
POR FABIAN SOTO F.
PROF. GUIA: SERGIO OCHOA

SEGURIDAD DE DATOS EN SISTEMAS COLABORATIVOS MÓVILES

El objetivo general del trabajo realizado fue desarrollar un sistema genérico de seguridad y privacidad de datos, que pudiera ser aplicable a un amplio espectro de aplicaciones colaborativas móviles.

Se logró desarrollar un sistema de control de acceso a archivos basado en roles, usando herramientas de criptografía, tomando en cuenta los objetivos de seguridad, así como las características y restricciones propias de los sistemas colaborativos móviles. El sistema es capaz de mantener la privacidad, integridad y autenticación de los documentos (en realidad de cualquier archivo) usando un sistema de permisos de escritura y lectura. La privacidad de los documentos se asegura haciendo uso de la criptografía simétrica; la integridad y autenticación se obtiene mediante el uso de códigos de autenticación de mensajes basados en hash.

Como resultado final se obtuvo una interfaz de servicios que puede ser reutilizada por distintas aplicaciones móviles, así como una aplicación gráfica que permite que los usuarios de un sistema colaborativo protejan y compartan sus archivos. El sistema funciona sobre PDA's y Notebooks, con un desempeño adecuado para los usuarios y acorde a la capacidad de los dispositivos empleados.

Agradecimientos

Deseo dar las gracias a todas las personas que me han brindado su apoyo durante mis estudios especialmente a mi familia: a mis padres David y Silvia, y a mi hermano David.

También quiero agradecer a mi profesor guía Sergio Ochoa por su dedicación y consejo a lo largo del desarrollo de este trabajo.

Este trabajo ha sido parcialmente financiado por el proyecto Fondecyt (Chile), Nº 11060467 y por el proyecto LACCIR No. R0308LAC005.

Glosario

ACL: Access control lists

AES: Advanced Encryption Standard

ARS: Application role server

CBC: Cipher block chaining

CTR: Counter

DES: Data Encryption Standard

DLL: Dynamic Link Library

EBC: Electronic Codebook

HMAC: Hash-based Message authentication code

MAC: Message authentication code

MANET: Mobile ad-hoc network

MD5: Message-Digest Algorithm 5

P2P: Peer-to- peer

PDA: Personal Digital Assistant.

RBAC: Role based access control

RNG: Random number generator

RPP: Requesting peer-pull

SHA: Secure Hash Algorithm

N.S.S.C: Namespace System.Security.Cryptography

TIC: Tecnologías de Información y Comunicaciones.

UPP: Ultrapeer-pull

VI: Vector de inicialización.

Tabla de contenido

1. Introducción

El vertiginoso avance de las tecnologías de la información y las comunicaciones (TICs) ha provocado que la utilización de distintos dispositivos computacionales (tales como notebooks, celulares, PDA's, etc.) se haya masificado, así como la cantidad de actividades que las personas realizan con éstos. Por lo tanto, han surgido nuevos escenarios en los cuales éstos pueden ser usados como herramientas de gran ayuda. Uno de estos escenarios son los sistemas colaborativos, los cuales permiten que un grupo de personas trabajen en forma conjunta para lograr una meta común.

El desarrollo de las tecnologías móviles y de las comunicaciones inalámbricas dan lugar a aplicaciones colaborativas que funcionan en distintos dispositivos móviles. Éstas hacen uso de las redes inalámbricas para comunicar y coordinar los dispositivos de un equipo de trabajo. En esta situación surge una nueva infraestructura de red, puesto que no hay servidores centralizados, sino que se utiliza una topología especialmente adecuada para usuarios móviles. A este tipo de redes se las conoce como MANETs (Mobile Ad-hoc Networks) [Andriessen, 2006; Brugnolli, 2005]. Dichas redes poseen características especiales, como por ejemplo: el bajo alcance de la comunicación, y las frecuentes desconexiones de los dispositivos de la red [Neyem, 2005].

Un aspecto importante a considerar en todo sistema informático es la seguridad, que incluye aspectos como la confidencialidad e integridad de la información, además del control y auditoría de las acciones que los usuarios pueden realizar. Esto es particularmente relevante en sistemas colaborativos móviles, puesto que para que dos o más usuarios puedan colaborar, usualmente es necesario que utilicen "de puente" a usuarios intermedios de la red, los cuales no deberían tener acceso a los datos que transportan.

En este escenario de trabajo están presentes dispositivos con escasos recursos de hardware, como por ejemplo celulares y PDAs, por tanto los mecanismos de seguridad que se propongan deberán estar adaptados para funcionar en este tipo de dispositivos. Esta propuesta de memoria busca determinar cuáles son los objetivos de seguridad que surgen en ambientes de trabajo colaborativo móvil, específicamente para la protección de documentos y archivos en general, cuáles son los esquemas de seguridad que se deben aplicar para satisfacer dichos objetivos y finalmente diseñar e implementar un sistema que permita aplicar la solución diseñada sobre una herramienta colaborativa móvil.

1.1. Justificación

La privacidad de la información es importante en el trabajo diario de las personas, sobretodo en situaciones de colaboración, puesto que muchas veces los agentes que colaboran, no necesariamente se conocen y confían completamente entre sí. Por ejemplo, supongamos que dos compañías trabajan conjuntamente en la construcción de un edificio, donde la primera se encarga de las instalaciones eléctricas, y la otra de la estructura física. Ambas deberán compartir información de planos y probablemente estado de avance del trabajo en algunos sectores del proyecto. Sin embargo, si la compañía de instalaciones eléctricas es además competencia de la segunda compañía (ofreciendo también servicios de estructuras), entonces la primera compañía desea mantener la privacidad de información sensible y buscará compartir sólo la información necesaria para realizar las instalaciones eléctricas. Estos problemas de privacidad pueden tener graves consecuencias si no son manejados en forma adecuada.

Por otra parte, la integridad de la información puede llegar a ser un aspecto crítico en ambientes en los cuales la información debe ser confiable. Por ejemplo, en un ambiente hospitalario distintos agentes (cirujanos, enfermeras, etc.) interactúan a través de protocolos médicos, y comparten información entre ellos. La información en estos casos debe ser confiable, puesto que en base a ella se tomarán decisiones acerca de cómo proseguir con el tratamiento de un paciente. Bajo este mismo contexto la autenticación de los agentes involucrados puede ser útil, puesto que permite identificar a las personas que realizaron acciones para fines de auditoría y/o contacto.

En muchas organizaciones las acciones que las distintas personas pueden llevar a cabo en diferentes situaciones, son restrictivas respecto a quien puede realizar ciertas acciones (por política interna de la organización por ejemplo). Esto también sucede en los equipos de personas que trabajan colaborativamente, y por tanto, es necesario definir cuál es el sistema de roles adecuado en cada caso.

Todos estos problemas han sido estudiados desde hace años, y han sido solucionados a través de diversos mecanismos como la criptografía (respecto a la autenticación, confidencialidad e integridad), y los sistemas de control de acceso (para establecer el control de las acciones que llevan a cabo los usuarios). Sin embargo, la aplicación de estos conceptos a ambientes móviles y colaborativos dan lugar a nuevos desafíos y oportunidades, puesto que surgen restricciones y características especiales como la escasez de recursos por parte de los dispositivos móviles, las restricciones de comunicación, el soporte para la comunicación no prevista, etc. Hasta el momento no se encontraron soluciones para abordar este desafío, por lo tanto esta propuesta de memoria podría realizar un aporte en el ámbito del trabajo colaborativo móvil.

1.2. Objetivos de la Memoria

El objetivo general de esta memoria fue desarrollar un sistema genérico de seguridad y privacidad de datos, que pueda ser aplicable a un amplio espectro de aplicaciones colaborativas móviles. Este sistema debía ser capaz de funcionar en diversos dispositivos móviles y en una red ad-hoc, además de cumplir con los aspectos de seguridad y privacidad pertinentes. Los objetivos específicos que se derivaron del objetivo general, fueron los siguientes:

1. Abordar los siguientes aspectos importantes de seguridad y privacidad en sistemas colaborativos móviles: control de acceso (de los usuarios) a documentos y archivos en general; e integridad y autenticación de documentos sensibles.
2. Seleccionar las herramientas de seguridad y privacidad que pueden ser usadas para satisfacer tales aspectos.
3. Implementar/adaptar dichas herramientas de forma que puedan funcionar en una amplia gama de dispositivos móviles.

1.3. Estructura de la memoria

La estructura de la memoria es la siguiente. La sección 2 presenta la revisión bibliográfica relacionada al ámbito del trabajo realizado. La sección 3 describe los requisitos con los que la solución desarrollada debía cumplir. La sección 4 presenta el diseño del esquema de seguridad de la solución. La sección 5 explica y justifica la implementación del esquema diseñado. La sección 6 muestra las pruebas realizadas. La sección 7 presenta las conclusiones finales y el trabajo a futuro.

2. Revisión Bibliográfica

2.1. Conceptos previos

A continuación se introducen los conceptos de criptografía simétrica, checksum, y funciones de hash criptográficas. Estos conceptos serán utilizados en las explicaciones presentadas en los capítulos venideros.

2.1.1 Criptografía simétrica

La criptografía simétrica consiste en técnicas que permiten encriptar y desencriptar archivos usando la misma clave, de esta forma las partes que interactúan en la comunicación deben compartir la misma clave.

Los algoritmos de cifrado simétricos usan primitivas llamadas cifradores de bloque, que son algoritmos de encriptación que cifran cadenas de bits de un largo fijo y que el resultado es otra cadena del mismo largo. Los cifradores de bloque más ampliamente usados son AES y DES (además de sus variantes 3DES y DESX) [Bellare 2006, Chapter 3].

Los cifradores de bloque se usan para cifrar cadenas de cualquier largo haciendo uso de los modos de operación [Bellare 2006, Chapter 5], los modos de operación considerados más seguros son CTR y CBC, puesto que mezclan los bloques a diferencia del modo ECB que cifra los bloques individualmente, por tanto patrones en el texto plano pueden repetirse en el texto cifrado. En la solución propuesta en esta memoria se usa DES con el modo de operación CBC para cifrar archivos.

2.1.2 Checksum y funciones de hash criptográficas

El checksum es un método para verificar la integridad de los datos, detectando de este modo posibles corrupciones. Las funciones de hash criptográficas son funciones de hash resistentes a colisiones, estas funciones proporcionan un nivel de seguridad mayor frente a ataques de usuarios mal intencionados que otros mecanismos usados para los checksum (como la suma de los bits o CRC (comprobación de redundancia cíclica) [Peterson 1961]).

Ejemplos de funciones de hash criptográficas son MD5 y la familia de funciones SHA [Bellare 2006, Chapter 6] (entre ellas SHA-1, SHA-256 y SHA-512). De ellas la familia SHA es considerada más segura [Secure Hashing 2009], aunque es probable que sea reemplazada en un futuro cercano debido a recientes ataques. En la solución propuesta en esta memoria se usa SHA-1 para verificar la integridad de los archivos. Las funciones de hash pueden ser usadas para crear códigos de autenticación de mensajes (más conocidos como MAC) haciendo uso de claves simétricas (tal como en el caso de la criptografía simétrica) para la creación y verificación de los códigos [RFC-2104 2009]. Las técnicas que implementan MAC basadas en funciones de hash reciben el nombre de HMAC.

2.2. Revisión de trabajos relacionados

En el área de la seguridad en sistemas colaborativos móviles se han desarrollado varias propuestas tomando en cuenta un subconjunto de las características propias de estos sistemas, como lo son la arquitectura de red (MANETs y P2P) y las propiedades subyacentes de los ambientes colaborativos y móviles. A continuación se exponen los trabajos más relevantes categorizados según los problemas o limitaciones que buscan resolver:

MOTION: Es un proyecto que busca desarrollar una plataforma peer-to-peer para el trabajo en equipo móvil [Kirda 2002], la que incluye una componente encargada del control de acceso, DUMAS [Fenkam 2002]. Aquí el control de acceso se establece mediante el uso de certificados digitales y ACL's (Access control lists) que indican los permisos de los usuarios en un determinado periodo de tiempo. Los certificados de autorización son representados usando XML. Además este es el único trabajo de control de acceso en el que se superponen las características de un ambiente p2p, colaborativo y móvil. Además aquí se identifican las principales características que debería tener un sistema de control de acceso en las condiciones antes mencionadas.

Red de a pares (P2P) y colaboración: La autonomía de los peers en una red p2p puede lograrse a través del uso de web services en el control de acceso. En el trabajo de Park [Park 2003] se hace uso de este enfoque para proteger los archivos en un ambiente colaborativo. También han surgido nuevos modelos de control de acceso (además del usual RBAC), que proveen de nuevas funcionalidades a los usuarios como lo son la delegación y revocación de permisos a otros usuarios [Wang 2008] , (en el modelo RBAC el administrador asigna los permisos a los roles y estos quedan estáticos mientras el no los cambie), o enfoques basados en la gestión de la confianza que se produce entre los agentes [Adams 2006], esto basado en la reputación (en base al comportamiento observado) de cada nodo.

Red de a pares (P2P): En [Park 2007] se proponen 2 arquitecturas RPP (Requesting peer-pull) y UPP (Ultrapeer-pull) para implementar RBAC en redes p2p. La autenticación se hace en 2 niveles y se hace uso de un ARS (Application role server) y de ciertos peers especiales denominados ultrapeers. En la arquitectura RPP la comunicación entre los peers se hace por medio de ultrapeers que son los encargados de hacer la gestión correspondientes de los recursos y son en el enlace entre los peers que se están tratando de comunicar. Además la autenticación del primer nivel se hace contra el ARS y la de segundo nivel contra los ultrapeers. En la arquitectura UPP para el primer nivel de autenticación los ultrapeers hacen de enlace y son ellos (y no el peer que se está incorporando) los encargados de recuperar la información desde el ARS para autenticar al peer regular. Además al final se hace una comparativa de ambas arquitecturas y se introduce un nuevo tipo de certificados LWPCs (lightweight peer certificates) en contraposición de los certificados X.509 usados para el uso de criptografía asimétrica. En [palomar 2006, 2008] se usan certificados en redes p2p para control de acceso y protección de contenidos. Los certificados son de clave pública, pero no se requiere una infraestructura de clave pública (PKI) para las tareas de autenticación y control de acceso.

Colaboración: En el trabajo de Traoré [Traoré 2003] se establece un modelo basado en reticulados (lattice model) y se discute que un modelo de seguridad adecuado para un ambiente colaborativo móvil debe considerar varios participantes con diferentes requerimientos de seguridad y operacionales. Con lo anterior un modelo debe acomodar y forzar los variados y a veces conflictivos requerimientos de seguridad que se dan en este tipo de ambientes. En el trabajo de Shen [Shen 1994] se usa un modelo simple para la edición colaborativa de documentos. Además también se señalan los requerimientos que un sistema de control de acceso debiera poseer en un ambiente colaborativo, estos son: roles de usuario (múltiples y dinámicos), permisos colaborativos, flexibilidad, especificación simple de los permisos de acceso, almacenamiento y evaluación eficiente de las especificaciones y automatización, que se refiere a la implementación en aplicaciones multiusuario. Además, en el trabajo de Tolone [Tolone 2005] se hace una comparativa de los distintos modelos de control de acceso en sistemas colaborativos. Estos modelos son matriz de acceso y RBAC, los 2 modelos más ampliamente utilizados, y otros no tan populares pero menos rígidos que los anteriores (sobre todo por su dinamismo). TBAC (Task based access control), establece el control de acceso tomando en cuenta las actividades o tareas que los participantes están llevando a cabo en un determinado proceso, un caso claro de aplicación de esto sería un sistema de permisos para algún workflow. TMAC (Team based access control), define 2 aspectos importantes, el contexto de los usuarios (los individuos desempeñan un papel en un equipo en un momento dado) y el contexto de los objetos (se identifican objetos específicos que son requeridos para el propósito de colaboración), C-TMAC, extiende TMAC incluyendo información contextual adicional como tiempo y espacio por ejemplo. SAC (Spacial access control), regula el acceso basado en condiciones espaciales puesto que divide el ambiente en regiones y las políticas tienen relación con la capacidad de los movimientos entre regiones, Context-Aware, que regula el acceso basado en roles del ambiente, esto es, que los individuos asume roles basados en el contexto en el cuál se está llevando a cabo la colaboración

Redes ad-hoc móviles (MANETs): En el trabajo de Memon [Memon 2007b] se propone un framework para el control de acceso y la transferencia segura de archivos. Este framework hace uso de varias herramientas de seguridad provistas por XML como lo son XML Encryption y XML Signature. En otro trabajo [Memon 2007a] se describe la arquitectura usada en el framework anterior y los fundamentos de ésta, además de un diseño de los roles basado en redes neuronales.

Colaboración: En el trabajo de [Kim 2008] se presenta un esquema de control de acceso para computación ubicua basado en XACML (un lenguaje declarativo de control de acceso basado en XML). Un aspecto importante de este esquema es que toma en cuenta la comunicación repentina entre distintos objetos del ambiente.

Acceso basado en roles (RBAC): Un tratamiento más acabado del modelo RBAC se hace en el trabajo de Ferraiolo [Ferraiolo 1999] y una estandarización (el modelo del NIST) de éste en el trabajo de Sandhu [Sandhu 2000], donde se definen 4 niveles que se pueden observar en todo modelo RBAC. En general los esquemas más detallados hacen uso de RBAC (o variaciones de este) como modelo de control de acceso. En los trabajos que hacen uso de otros modelos tratan más el aspecto conceptual de los modelos y los requerimientos distintivos de los ambientes móviles sin entrar en detalles de implementación.

Criptografía diferencial o basada en atributos: En el trabajo de [Bethencourt 2007] y en el de [Goyal 2006] se presentan métodos de cifrado que permiten controlar el acceso a datos cifrados de distintos individuos, el enfoque usado en estos métodos se basa en definir una serie de atributos o condiciones que los usuarios deben cumplir para tener acceso a los datos cifrados. Para representar dichas condiciones se hacen uso de estructuras de acceso para definir cuáles son los usuarios autorizados a acceder a los datos. De este modo se pueden definir una política de seguridad basada en condiciones que deben cumplir los usuarios para acceder a los datos, está condición podría ser por ejemplo el rol que un usuario posee dentro de un sistema colaborativo.

Como conclusión respecto de la bibliografía revisada, se tiene que son pocos los trabajos que toman en cuenta la combinación de los factores: trabajo móvil y colaborativo. Sin embargo, estos aspectos sí son tomados en cuenta en forma independiente en varios trabajos de investigación.

3. Requisitos de la Solución

La solución propuesta usa el enfoque RBAC para proteger los archivos. De este modo se establecen permisos sobre los archivos, y estos permisos afectan de la misma forma a todos los usuarios que comparten el mismo rol. Para proteger los archivos éstos se almacenan cifrados, y se hace el uso de un checksum para verificar la integridad de datos.

A continuación se especifican los requisitos funcionales y no funcionales básicos (o principales) de un sistema de seguridad para sistemas colaborativos móviles.

3.1. Requisitos funcionales

1. Los usuarios dueños de los archivos deben poder establecer los permisos sobre sus archivos y estos deben mantenerse durante toda la "vida" del archivo (a menos que el usuario permita a otro usuario cambiar los permisos).
2. Los archivos deben poder ser protegidos con permisos de escritura y lectura.
3. Los usuarios deben de tener acceso sobre los archivos según los permisos establecidos sobre dichos archivos. De este modo los usuarios que tienen permisos de lectura sobre un archivo, deben ser capaces de ver su contenido, así como los que no tienen permisos no deben ser capaces de verlo. Del mismo modo los usuarios que tienen permisos de escritura sobre un archivo deben ser capaces de modificar su contenido, así como los que no tienen permisos no deben ser capaces de modificarlo.
4. Los usuarios deben de ser capaces de poder compartir los archivos con otros usuarios dentro de la misma red (MANET).
5. Los usuarios deben ser capaces de verificar la integridad de los datos/archivos.

3.2 Requisitos no funcionales y restricciones

1. La protección de los archivos debe ser rápida para archivos de cierto tamaño. La protección debe demorarse menos de 20 segundos en archivos con tamaño inferior a 5MB. Este es un tiempo límite de espera para el usuario, el cual fue obtenido en base a experiencias prácticas.
2. El acceso a los archivos debe ser rápido para archivos de cierto tamaño. El acceso debe demorarse menos de 20 segundos en archivos con tamaño inferior a 5MB. Este límite fue determinado siguiendo el mismo procedimiento anterior.
3. El sistema debe correr sobre PDAs con Windows Mobile 5.1 o superior y sobre Notebooks con Windows XP Service Pack 3 o superior. Estas restricciones surgieron a partir de que el sistema propuesto debía acoplarse a otros que están actualmente funcionando, y que son los que imponen estas restricciones.
4. El sistema debe ser P2P (peer-to-peer); esto es, debe ser simétrico para cualquier nodo dentro de las MANET's que forman los usuarios, de este modo ningún nodo posee una posición privilegiada por sobre otro (aparte de los privilegios que tenga asociado su rol). En este trabajo esta simetría es alcanzada parcialmente, puesto que

de todos modos los usuarios deben acudir a un administrador centralizado para “inscribirse” en el sistema.

4. Diseño de la Solución

A continuación se presenta el diseño de la solución implementada. En primer lugar se describe el sistema de permisos sobre los archivos que se comparten. Luego se describe el mecanismo de administración de roles. Finalmente se presentan los mecanismos de seguridad que se utilizaron para implementar la solución.

4.1. Permisos sobre archivos

4.1.1 Representación de los permisos

Se establecen dos tipos de permisos dentro del sistema, de escritura y de lectura, estos permisos afectan a los usuarios a nivel de rol. Cabe mencionar que el permiso de lectura indica que un usuario puede ver un archivo, y que el permiso de escritura indica que un usuario puede ver y modificar un archivo, por tanto los permisos de escritura incluyen los de lectura. Esto se estableció así, puesto que son permisos ideados para personas y no para usuarios de sistema (propios de aplicaciones y servicios del sistema operativo). Además, los usuarios trabajan con los documentos, y por tanto es operativamente imposible que un usuario modifique un archivo sin tener acceso a su contenido.

Si hay n roles en el sistema, los permisos son representados a través de un string de largo igual a n, donde cada carácter del string representa el permiso que tiene el rol correspondiente a ese carácter. Le damos un número identificador a cada rol, partiendo desde 1, hasta n. Los permisos sobre un archivo se pueden describir de la siguiente forma:

Si s es un string de permiso de largo n, $s[i]$ representa el permiso del rol con identificador i $(1 < i < n)$. $s[i]$ puede tomar los valores 0, w y r; que indican permisos nulos (sin permisos), y permisos de escritura (write) y lectura (read) respectivamente. Por ejemplo el string "*rw00*", indica que el rol con identificador 1 tiene permisos de lectura, el con identificador 2 tiene permisos de escritura, y los 2 roles restantes no tienen ningún permiso.

4.1.2 Dinámica de los roles y claves

El administrador del sistema colaborativo debe generar los roles del sistema (en la sección 4.2 se explica como), en ese momento se genera el archivo con todas las claves que se usarán dentro del sistema. Este archivo tiene el siguiente formato:

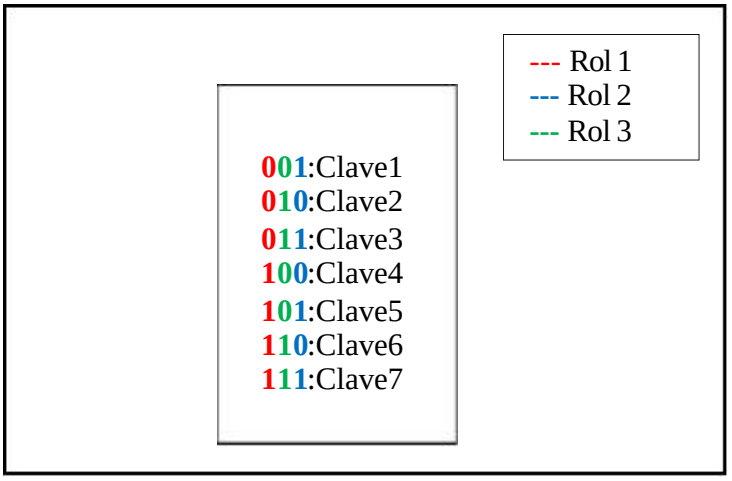

Figura 1: Archivo de claves

En este caso hay 3 roles dentro del sistema, por eso las cadenas son de 3 bits. Los bits indican si el rol correspondiente al color tiene permisos para descifrar el archivo, los roles tienen un identificador que es un número entre 1 y n, donde n es la cantidad de roles. Este identificador coincide con la posición en la cadena de bits. De este modo, el rol con identificador 1 está representado por los bits rojos, el rol con identificador 2 por los bits verdes y el rol con identificador 3 por los bits azules. Un bit con valor 1, indica que el rol correspondiente tiene permisos (de lectura), y un bit con valor 0 indica que el rol no tiene permisos. Así por ejemplo la *Clave1* se usa cuando el primer y segundo rol (según los identificadores) no tienen permisos y el tercero sí tiene permisos. Las claves son usadas como semillas para generar 2 claves pseudoaleatoriamente: una para la encriptación y otra para el uso de MAC. El algoritmo de encriptación es DES, y se usa HMAC haciendo uso de SHA-1.

Aquí se puede observar que si hay n roles en el sistema se deben generar 2^n -1 claves (las cadenas de bits de largo n excepto la 0^n). De este modo también se puede observar que si se agrega un nuevo rol al sistema, la cantidad total de claves se duplica. Además, las claves de los usuarios de los roles anteriores deben de ser actualizadas, puesto que las claves de cada usuario aumentan (al doble, como se muestra en las figuras 2 y 3) dado que deben agregarse las claves para considerar los permisos del nuevo rol a los archivos protegidos, esto debido a que el largo de las máscaras de bits usadas aumentará en 1. Por ejemplo en el caso ilustrado en las figuras 2 y 3 las claves correspondientes al rol “Rol 1” pasan de ser 4 a 8, y las máscaras de bits pasan de tener un largo de 3 a un largo de 4 (se agrega un bit adicional indicando si el nuevo rol tiene permisos o no).

En las figuras 2 y 3 se muestran los cambios que ocurren en la generación de claves al pasar de un sistema con 3 roles (en la figura 2), a uno con 4 roles (en la figura 3). Comparando ambas figuras se puede observar cómo el archivo de claves se duplica, y cómo los archivos de claves de cada rol deben actualizar las claves debido a que se ha agregado un rol al sistema. La actualización se ve reflejada en el hecho de que ahora hay más claves (las cadenas de bits pasan de ser de largo 4 a largo 5).

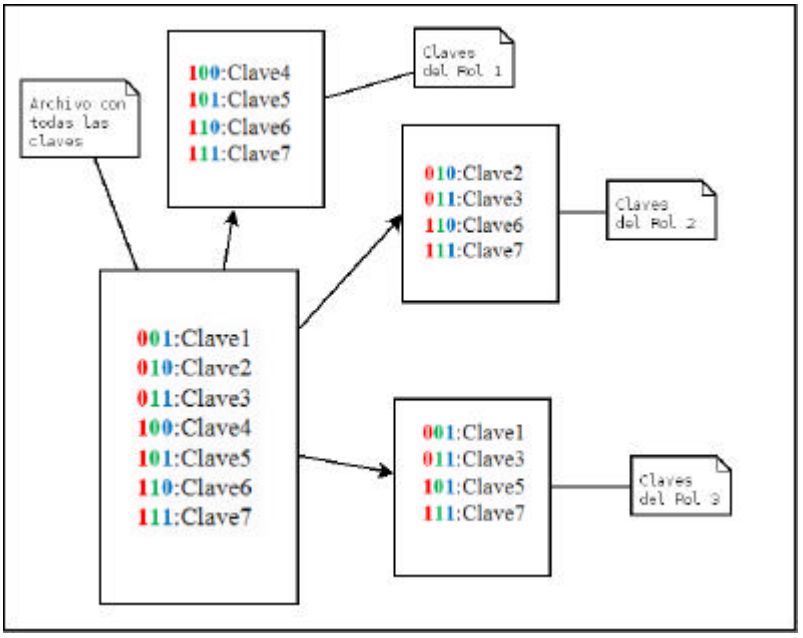

Figura 2: Generación de archivo de claves para 3 roles

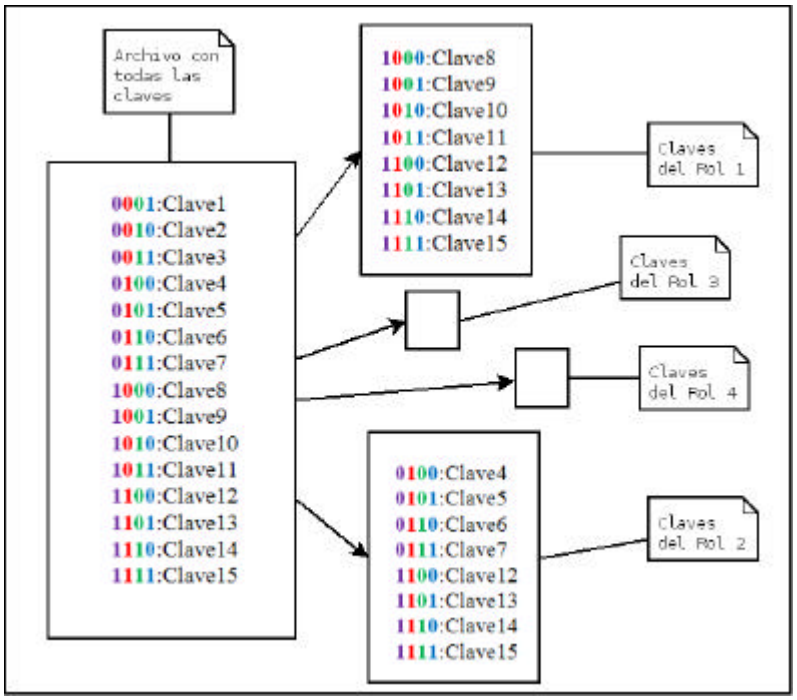

Figura 3: Generación de archivo de claves para 4 roles

Debido a la forma en que se manejan las claves dentro del sistema, los roles deben de ser estáticos, puesto que agregar o eliminar roles afecta las claves de todos los usuarios del sistema colaborativo. Esta situación provoca que los roles antiguos deben de considerar la participación del nuevo rol, y por tanto, deben actualizar las claves. Este aspecto lleva a una complicación respecto a la flexibilidad del sistema; sin embargo en la práctica los roles no cambian con frecuencia, de hecho muchas veces éstos son definidos antes de que un sistema sea puesto en marcha, y no cambian en el futuro.

4.1.3 Estableciendo permisos sobre un archivo

Un usuario puede establecer permisos sobre cualquier archivo que esté en su dispositivo móvil, para ello selecciona el archivo que quiere proteger y establece los permisos sobre él a través de una interfaz gráfica. Al establecer los permisos sobre un archivo se genera un string de permisos y se encripta el archivo que se está protegiendo. La llave que se usa para el cifrado queda determinada por el string de permisos. Cada usuario posee un archivo de claves, que tiene la estructura que se muestra en la figura 4:

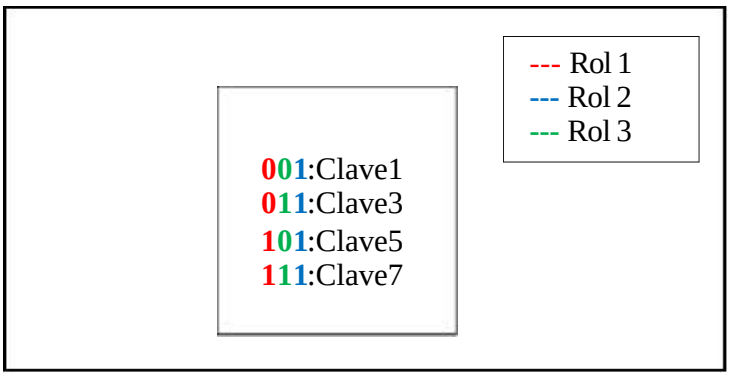

Figura 4: Archivo de claves de un usuario

Este archivo se entiende del mismo modo que el archivo presentado en la figura 1. La diferencia radica en que aquí corresponde al archivo de claves del tercer rol, puesto que este rol debe poseer sólo las claves para cifrar/descifrar los archivos en que tiene permisos, o visto en la cadena de bits, todas las claves en que el tercer bit de la cadena es 1.

Cuando un usuario establece los permisos, se generan 2 archivos temporales, un archivo con el string de permisos y el archivo encriptado, luego estos 2 se concatenan en un archivo que contiene el string de permisos como encabezado y el archivo cifrado como contenido, al resultado de la concatenación se le denomina el archivo protegido.

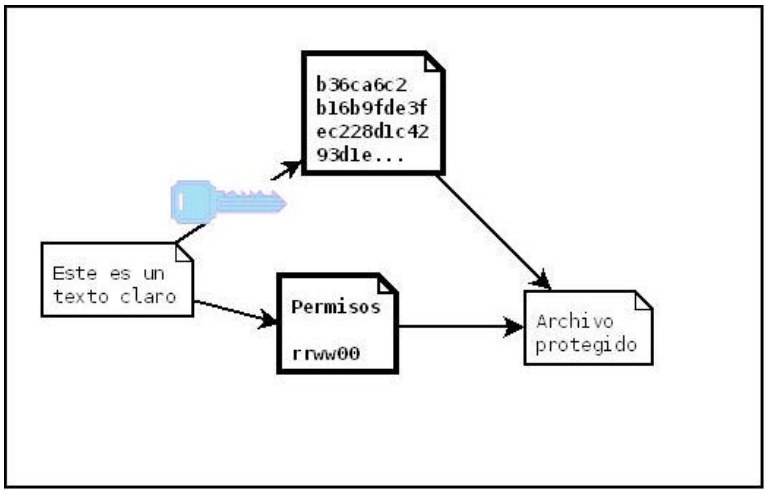

Figura 5: Protección de archivos

En la figura 5 se marcan con negritas los archivos temporales (el de permisos y el encriptado) que se generan, para luego mezclarse con en el archivo protegido que contiene el archivo cifrado así como la información de los permisos. El archivo protegido es el archivo que se comparte con los demás usuarios del sistema. Este archivo se creó por las siguientes razones:

1) Los permisos sobre el archivo deben ser conocidos por cualquier usuario del sistema, y debido a esta restricción, este archivo no puede estar centralizado. Debido a la arquitectura P2P de la red donde se usan, se decidió incluir los permisos dentro del archivo que se comparte.

2) Para compartir el archivo se decidió encriptarlo por las siguientes razones:

   a) La carencia de sistemas de permisos del sistema operativo en dispositivos móviles (Windows Mobile [Security in Windows Mobile 2009]), puesto que los usuarios siempre pueden acceder a través del explorador de archivos y ver el contenido de éstos.

   b) Para permitir la homogeneidad de distintos equipos móviles. De este modo, no se tiene la restricción de tener que usar el sistema de permisos de un sistema operativo específico. Además, aún si se usara esta vía de solución, se debería establecer un esquema de sincronización de permisos entre los usuarios correspondientes al sistema colaborativo y al sistema operativo, y los permisos deberían estar replicados en cada dispositivo. Esto llevaría a que en cada dispositivo estuvieran presentes todos los usuarios del sistema colaborativo y que cada equipo sincronizara los permisos de todos los usuarios.

   c) Para evitar que intromisiones en la red (al transferir archivos) tanto de sniffers o intrusos externos, así como de usuarios no autorizados dentro del sistema colaborativo que estén siendo usados como “puente” durante la transferencia de archivos.

4.2 Administración de roles

Para la operación del sistema es necesario establecer configuraciones previas a su uso, las configuraciones necesarias son: definir los roles de los participantes en el sistema colaborativo subyacente, e instalar las configuraciones individuales en cada equipo de los usuarios. Está configuración en general debiese hacerse en el momento en que se le facilita un dispositivo móvil a algún usuario, o como una actualización (en el caso de que el usuario ya posea un dispositivo) al implantar nuevas aplicaciones colaborativas.

4.2.1 Definición de los roles

El administrador del sistema colaborativo debe definir los roles del sistema, un ejemplo de roles podría ser: enfermero/a, médico, paramédico. A través de una herramienta vía consola el administrador genera los roles del sistema. El administrador del sistema es un nodo especial dentro del sistema colaborativo subyacente, sus características especiales son las siguientes:

1. No forma parte de las MANET's que forman los participantes del sistema colaborativo subyacente, ni tampoco tiene asociado ningún rol.

2. Es un computador de escritorio o un notebook, y usa la herramienta de consola de Windows XP.

3. Se comunica con las PDA's (para definir el rol correspondiente) vía un cable USB.

4. Se asume que este nodo está protegido de alguna forma (aislado de la red para evitar intromisiones, zona con acceso restringido, etc.). Esto principalmente porque posee las todas las claves del sistema.

Así el administrador del sistema cumple sólo la tarea de establecer la configuración inicial de los participantes del sistema colaborativo, y no participa de la operación cotidiana en el sistema. El administrador del sistema genera los roles del sistema escribiendo el siguiente comando:

```
rolesAdmin rol_1 rol_2 … rol_n
```

Aquí además de establecer los roles del sistema, se generan todas las claves que van a ser usadas por los usuarios para proteger y acceder a los archivos. En este momento se generan 2^n-1 claves de 128 bits, donde n es la cantidad de roles definida. Estas claves corresponden a todas las combinaciones de permisos entre los distintos roles, la razón de estos valores se explicaron en la sección 4.1.3.

Dependiendo del rol de cada usuario, se seleccionarán un subconjunto de las claves generadas (exactamente la mitad, esto también se explicó en la sección 4.1.3), que serán las claves que los usuarios de ese rol utilizarán para proteger y acceder al contenido de los archivos. A través de la herramienta vía consola, el administrador genera las claves para un rol establecido anteriormente:

```
rolesAdmin rol_x
```

Aquí se seleccionan las claves correspondientes al rol "rol_x". Las claves seleccionadas se instalan en los equipos de los usuarios según corresponda; esto es, se le otorgan las claves correspondientes a su rol. Estas son las claves que necesitarán los usuarios del rol correspondiente para poder descifrar los archivos a los que tiene acceso, o para cifrar los archivos al momento de establecer los permisos sobre éste. Cabe mencionar que las claves no se usan directamente, sino como semilla para generar la clave de encriptación.

El primer paso de la configuración debe ser establecer los roles del sistema, luego cada vez que un nuevo usuario se incorpore al sistema se deben seleccionar las claves correspondientes a su rol e instalarlas en su equipo. Cabe mencionar que debido al sistema de generación de claves, una vez definidos los roles del sistema, no se pueden agregar nuevos roles, excepto si se redefinen los roles y se actualizan las claves de todos los usuarios del sistema. Las razones de esto se explicaron en la sección 4.1.3.

La información de los roles de un usuario en un dispositivo personal es obtenida a través de unos archivos de configuración ubicados en dicho dispositivo. Estos archivos deben ser ubicados en los directorios predefinidos por el administrador del sistema colaborativo móvil.

Los archivos de configuración son dos. Un primer archivo que contiene la información correspondiente a cuáles son los roles del sistema. Este archivo es igual para todos los usuarios del sistema colaborativo, puesto que cada usuario debe conocer todos los roles dentro del sistema para poder establecer permisos a cada rol. El segundo archivo contiene las claves de un rol particular y su nombre (el nombre del rol, por ejemplo: médico). Este archivo es el mismo para todos los usuarios que comparten un rol, pero diferente entre usuarios que poseen distintos roles.

4.3 Criptografía

Dado que para proteger los archivos éstos se encriptan, para poder llevar esto a cabo se usa una pequeña biblioteca que provee los servicios de cifrado y descifrado. La biblioteca permite cifrar/descifrar archivos con algoritmos de clave simétrica basados en los cifradores de bloques AES, DES y 3DES [Block Ciphers 2009]. Los algoritmos usan el modo de operación Cipher Block Chaining (CBC) [Cipher Modes 2009]. La biblioteca ofrece los 3 cifradores de bloque, pero la solución implementada usa sólo DES, puesto que es el cifrador que mostró el mejor desempeño en las pruebas (ver sección 6.1). Esta biblioteca se construye usando como base el espacio de nombres System.Security.Criptography de .NET [Criptography .NET 2009].

4.4 Checksum

Para lograr conservar la integridad de datos de los archivos, y además, como una forma de implementar los permisos de escritura, se usan MAC de los archivos a través de funciones de hash criptográficas. Para ello, se usa una pequeña biblioteca que permite calcular el checksum de un archivo vía HMAC [RFC-2104 2009], usando el algoritmo SHA-1. Se escogió este algoritmo puesto que en el compact framework 3.5 de .NET sólo se dispone de SHA-1 (no hay ningún otro de la familia SHA) y MD5; y además SHA-1 es considerado más seguro que MD5 (esto fue aprobado por el NIST) [Secure Hashing 2009].

Al igual que en el caso anterior, la biblioteca se construye usando como base el espacio de nombres System.Security.Criptography de .Net.

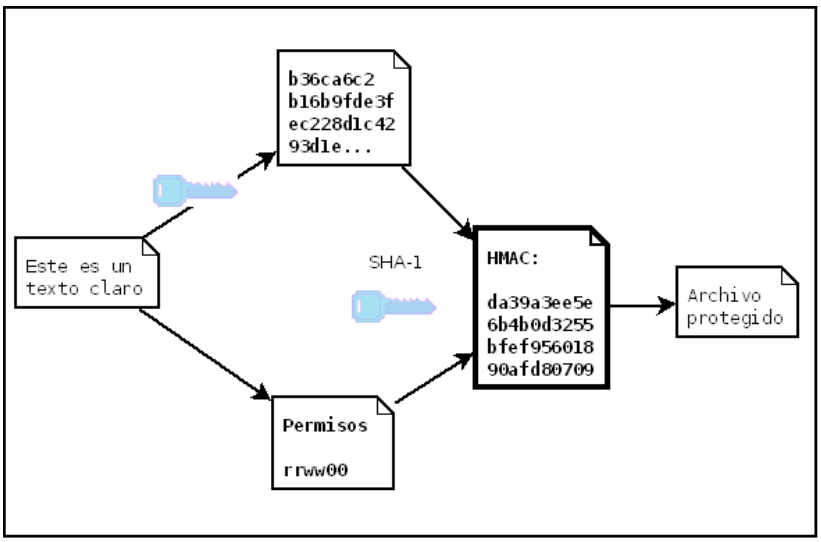

Figura 6: Protección de archivos

Con esta nueva funcionalidad los usuarios del sistema colaborativo pueden, al proteger sus archivos, elegir agregar una autenticación del mensaje (es decir el checksum del archivo) al archivo protegido. El checksum es calculado sobre el archivo cifrado y el string de permisos (se calcula sobre la concatenación de ambos). De este modo las personas con las cuales se comparten los archivos pueden verificar la integridad y autenticidad de los datos de éstos, lo que consiste en calcular el checksum del archivo cifrado concatenado al string de permisos y compararlo con la huella o MAC. Si ambos checksum son iguales (la huella y el calculado) significa que el archivo no ha sido alterado y si difieren significa que el archivo ha sido alterado (o los permisos). Los checksum se agregan a los archivos como encabezado, del mismo modo que el string de permisos. Además para calcular el checksum se usa la llave generada a partir de la clave del archivo de claves, por tanto sólo los usuarios con permisos de lectura pueden verificar el MAC de los archivos, todo esto se puede observar en la figura 6. El MAC se calcula sobre los permisos para evitar su modificación.

4.4.1 Uso para la protección contra escritura

Para implementar los permisos de escritura, se usan checksum para dejar huellas cuando los archivos son modificados. Esto se realiza permitiendo a los usuarios que tienen permisos de escritura sobre un archivo, agregar checksum y evitando que los usuarios sin permisos de escritura lo hagan. Este control se hace sólo a través de la interfaz del sistema (tanto la GUI del prototipo como de la biblioteca misma) en el caso de los usuarios que sí tienen permisos de lectura sobre un archivo, pero no tienen permisos de lectura, puesto que de todos modos poseen la clave para descifrar el archivo (que es la misma que clave para calcular el MAC). En el caso de los usuarios que no tienen permisos de lectura sobre un archivo, tampoco pueden verificar el checksum (dado que no tienen la clave). Toda esta rigidez se debe principalmente a que el uso de MAC, es simétrico y se debe usar la misma llave para autenticar un mensaje, así como para verificarlo.

El uso de las huellas permite identificar modificaciones a un archivo, hechas por un usuario sin permisos de escritura. Sin embargo, esto no evita que un usuario escriba sobre un archivo si tiene permisos de lectura (y por tanto puede la clave para descifrar, así como la clave para calcular el checksum). Esto no se puede evitar debido a la carencia de permisos en Windows Mobile [Security in Windows Mobile 2009], que fue el sistema operativo tomado como base para este trabajo. De esta forma para verificar que no se han modificado archivos sin autorización, los usuarios deben verificar manualmente la integridad de los archivos compartidos por los demás usuarios. Además, no se puede evitar que un usuario no autorizado modifique un archivo, sólo se puede evitar que recalcule el checksum (y sólo a través de la interfaz). De este modo al compartir el archivo modificado, los demás usuarios obtendrán una verificación fallida.

4.5 Creación de la MANET y transferencia de archivos

Para que el sistema colaborativo subyacente funcione, los usuarios deben de ser capaces de crear una MANET para comenzar a trabajar en línea, además los usuarios necesitan transferir los archivos a través de esta red. El protocolo HLMP (High Level MANET Protocol) permite formar MANETs entre usuarios móviles, así como transferir archivos dentro de esta red [Rodriguez 2009].

Para la transferencia de archivos a través de la MANET se usa una aplicación llamada FileTransfer (desarrollada por Juan Francisco Rodríguez como parte de su tesis de Magister en Ciencias, mención computación), que usa una estructura simple de directorios para indicar cuáles son los archivos que los usuarios comparten en la MANET.

Un usuario X de la aplicación debe definir 2 directorios dentro de su dispositivo móvil: un primer directorio de descargas, que es el directorio donde se descargan los archivos que los demás usuarios comparten; y un segundo directorio compartido, que es el directorio donde el usuario X debe tener todos los archivos que desea compartir con cualquier usuario del sistema. Los archivos dentro de este directorio compartido son visibles para cualquier usuario conectado a la MANET.

Los usuarios se conectan a la MANET usando un nombre de usuario que ellos definen. Cuando un usuario examina los archivos compartidos en la red, puede identificar cuál es el usuario que está compartiendo cada uno de los archivos. Cabe decir que la información de los roles se obtiene a través de archivos de configuración en los dispositivos y no a través del nombre de usuario que se usa para conectarse a la MANET. Si bien el usuario puede cambiar su nombre de usuario cada vez que se conecta a la red, no puede cambiar su rol dentro del sistema (esto sólo lo puede hacer el administrador), puesto que en su dispositivo tiene sólo las claves correspondientes a su rol, las demás claves están en el nodo correspondiente al administrador del sistema. Además las claves no pueden ser monitoreadas a través de la red, puesto que siempre están en los dispositivos y la única situación en que se transmiten es cuando el administrador las instala en un dispositivo móvil (esto se hace a través de un cable USB).

5. Solución Implementada

5.1 Módulos del sistema

Para la construcción de un prototipo que permita compartir archivos protegidos entre usuarios de un sistema colaborativo móvil, se desarrollaron siete módulos que proveen diferentes servicios:

1. **CryptoServices:** Este módulo es el encargado de proveer los servicios para cifrar y descifrar archivos.

2. **HashServices:** Este módulo es el encargado de proveer servicios para calcular marcas de hash de archivos.

3. **GrantsManager:** Este módulo es el encargado de representar los permisos de los usuarios sobre los archivos. A través de él, se establecen y verifican los permisos.

4. **Hexadecimal:** Este módulo provee conversiones de tipo y formato de cadenas de números hexadecimales. Es usado principalmente para el manejo de las claves de encriptación.

5. **Helpers:** Este módulo es el resultado de refactorizaciones del código y ofrece diversas funcionalidades comunes a los demás módulos. (Escribir un string a un archivo, leer una línea, concatenar archivos, separarlos, etc.)

6. **ManetSec:** Este módulo es la aplicación gráfica que usan los participantes del sistema colaborativo para proteger y transferir archivos. Se construyó usando como base la aplicación FileTransfer.

7. **RolesAdmin:** Este módulo es una aplicación de consola que permite al administrador del sistema administrar los roles y sus respectivas claves.

Todos los módulos anteriores exceptuando ManetSec y RolesAdmin son librerías (DLL's). ManetSec es una aplicación que usa todas las librerías para llevar a cabo su función. RolesAdmin es una aplicación que se usa para establecer configuraciones previas a la instalación de ManetSec en los dispositivos móviles.

Todos los módulos fueron implementados en 2 versiones (para Notebook y para PDA), excepto RolesAdmin que sólo fue implementado para Notebook. La razón para esto fue que el administrador del sistema colaborativo subyacente no es considerado un usuario móvil, y por tanto, no participa activamente en la protección y transferencia de archivos. El módulo RolesAdmin cumple con la tarea de hacer un setup centralizado de los roles, para que los usuarios puedan usar el sistema de permisos en cualquier momento y lugar (dentro de una MANET).

En la figura 7 se pueden apreciar las dependencias entre los distintos módulos, las flechas indican una relación de uso, por ejemplo el módulo GrantsManager usa a los módulos Hexadecimal y Helpers.

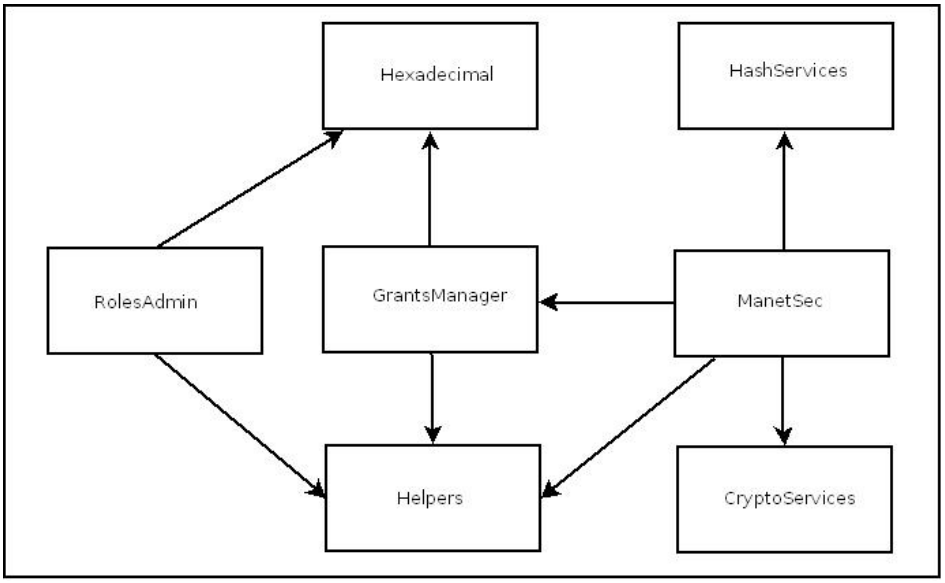

Figura 7: Relación de uso de los módulos

5.1.1 CryptoServices

Este módulo está compuesto por 2 clases, una interna CryptoServiceProvider y una externa Crypto.

5.1.1.1 La clase CryptoServiceProvider

La clase CryptoServiceProvider es la encargada de hacer de enlace con los proveedores de cifrado de .NET (del espacio de nombres System.Security.Cryptography). Esta clase posee los siguientes atributos:

1. *CryptoProvider* **algorithm**: Es el algoritmo que se usa para llevar a cabo el cifrado/descifrado, puede ser *TripleDES*, *DES* o *Rijndael (AES).*

2. *CryptoAction* **cAction**: Representa la acción que se ejecuta (cifrar o descifrar), puede ser *Encrypt* (encriptar) o *Desencrypt* (desencriptar).

Los métodos de la clase son los siguientes:

1. **CryptoServiceProvider**(*CryptoProvider alg, CryptoAction action*): Es el constructor de la clase y simplemente se encarga de inicializar los atributos *algorithm* y *cAction* con los parámetros *alg* y *action* respectivamente.

2. *ICryptoTransform* **GetServiceProvider**(*byte[] Key, byte[] VI*): Se encarga de devolver un objeto que implementa la interfaz *ICryptoTransform* (del U.S.S.C) que es la que define las operaciones básicas de las transformaciones criptográficas (En este caso cifrar y descifrar). Key es la llave usada para el cifrado/descifrado, e VI es el Vector de Inicialización usado en el modo de operación CBC.

5.1.1.2 La clase Crypto

La clase Crypto es la que provee los servicios de cifrado/descifrado. Permite cifrar/descifrar strings o archivos. Esta clase posee los siguientes atributos:

1. *CryptoProvider* **algorithm**: Es el algoritmo que se usa para llevar a cabo el cifrado/descifrado, puede ser *TripleDES*, *DES* o *Rijndael (AES).*

2. *byte []* **byteKey**: Es un arreglo de 16 bytes, y representa la clave con que se realizan las operaciones de cifrado/descifrado.

3. *byte []* **byteVI**: Es un arreglo de 16 bytes, y representa el Vector de Inicialización con que se realizan las operaciones de cifrado/descifrado.

La clase crypto posee las propiedades predeterminadas para establecer sus atributos:

1. *byte [ ]* **Key**: Lee (***get***) y escribe (***set***) el valor de la clave de cifrado/descifrado.

2. *byte [ ]* **VI**: Lee (***get***) y escribe (***set***) el valor del vector de inicialización de cifrado/descifrado.

Las claves y VI que se usan son de 16 bytes, esto es porque éstos son los tamaños para el cifrador AES. Sin embargo para 3DES el largo del VI es de 8 bytes y para DES el largo tanto de la clave como del VI es de 8 bytes. En el caso de 3DES se usa como VI la primera mitad del atributo *VI.* Para el caso de DES se usa como clave la primera mitad del atributo *Key* y la segunda mitad del atributo *VI*. Los métodos de la clase son los siguientes:

1. *string* **CifrarCadena**(*string CadenaOriginal*)**:** Cifra el string *CadenaOriginal* de acuerdo al objeto *Crypto* (este objeto ya contiene toda la información del cifrado/descifrado, como la clave, algoritmo y Vector de Inicialización).

2. *string* **DescifrarCadena**(*string CadenaCifrada*)**:** Descifra el string *CadenaCifrada* de acuerdo al objeto *Crypto* (este objeto ya contiene toda la información del cifrado/descifrado, como la clave, algoritmo y Vector de Inicialización).

3. *void* **CifrarDescifrarArchivo**(*string InFileName, string OutFileName, CryptoAction action*): Cifra o descifra el archivo indicado por la ruta *InFileName,* usando las propiedades del objeto Crypto, el resultado queda en el archivo indicado por la ruta *OutFileName,* si el archivo no existe: se crea, si ya existe: se sobrescribe; *action* indica si se cifra (*Encrypt*) o descifra (*Desencrypt).*

5.1.2 HashServices

Este módulo está compuesto por 1 clase (la clase Hash), que se encarga proveer los métodos (estáticos) necesarios para calcular los hash de archivos. Los métodos de la clase son los siguientes:

1. *String* **GetMd5Hmac**(*FileStream fs, byte [] key*): Retorna el HMAC MD5 del archivo representado por el objeto *fs* usando la clave *key* .

2. *String* **GetSHA1Hmac**(*FileStream fs, byte [] key*): Retorna el HMAC SHA-1 del archivo representado por el objeto *fs* usando la clave *key*.

5.1.3 GrantsManager

Este módulo está compuesto por una clase (la clase Grants). Los permisos sobre los archivos se representan con objetos de la clase Grants. Esta clase permite establecer y consultar la información relacionada con los permisos de los archivos. La clase Grants posee los siguientes atributos:

1. *char[]* **grants:** Este arreglo representa la cadena de permisos explicada en la sección 3.2.2.1, tal como se explican dicha sección los valores que toman los elementos del arreglo son 'r', 'w' y '0', para representar los permisos de lectura, escritura y permisos nulos respectivamente.

2. *string[]* **rols**: Este arreglo contiene cada uno de los roles dentro del sistema.

3. *string[]* **grantsNames**: Este arreglo contiene posibles alias para los permisos de lectura y escritura, así como para los permisos nulos. Estos alias se mapean a los valores 'r', 'w' y '0', respectivamente.

4. *string* **file**: Es el nombre del archivo al cual se le están otorgando los permisos (debe ser la ruta completa al archivo).

5. *string* **extGrants**: es la extensión que se usa para el archivo de permisos temporal, su valor por defecto es "*.grt.txt*".

6. *string* **extSec**: es la extensión que se usa para el archivo encriptado temporal, su valor por defecto es "*.crp.txt*".

7. *string* **extMerge**: es la extensión que se usa para el archivo protegido, su valor por defecto es "*.mrg.txt*".

8. *string* **extSum**: es la extensión que se usa para archivos con checksum, pero sin su contenido cifrado, su valor por defecto es "*.sum.txt*".

9. *bool* **hashed**: indica si el archivo ha sido marcado con su respectiva checksum.

10. *bool* **secured**: indica si el archivo ha sido protegido (chekcum y cifrado).

Los constructores de la clase son:

1. **Grants**(*int numRoles , string file*): Inicializa el valor del atributo file y se encarga de inicializar y definir el largo del atributo grants (inicialmente todos los elementos son '0').

2. **Grants**(*string[] rols, string file*): Inicializa los atributos rols, file y grantsNames con los valores de los parámetros homónimos. También se inicializa el atributo *grants* con el largo de *rols*.

3. **Grants**(*string[] rols, string file, string[] grantsNames*): Inicializa los atributos rols, file y grantsNames con los valores de los parámetros homónimos. También se inicaliza el atributo *grants* con el largo de *rols*.

4. **Grants**(*string file*): El archivo indicado por la cadena file, debe ser un archivo protegido. Este constructor crea un objeto Grants a partir de la cadena de permisos contenida en la cabecera del archivo file.

También hay 4 constructores análogos a los anteriores que además permiten setear las extensiones de los archivos, todos tienen la forma:

Grants(*, *string extGrants, string extSec, string extMerge, string extSum*).

De este modo los cuatro constructores son:

- **Grants**(*int numRoles , string file, string extGrants, string extSec, string extMerge, string extSum*)
- **Grants**(*string[] rols, string file, string extGrants, string extSec, string extMerge, string extSum*)
- **Grants**(*string[] rols, string file, string[] grantsNames, string extGrants, string extSec, string extMerge, string extSum*)
- **Grants**(*string file, string extGrants, string extSec, string extMerge, string extSum*)

Los métodos de esta clase son:

1. *void* **setGrantsRol**(*int RolId, char grant*): establece con el valor *grant* el elemento del atributo *grants* con índice RolId. (*grants[RolId]=grant*).

2. *void* **writeGrants2File**(*string FileName, string extGrants, string extSec*): escribe los permisos (representados por el atributo *grants*) al archivo indicado por la ruta *FileName*. *extGrants* y *extSec* son las extensiones de archivo de los archivos temporales que se mencionan en la sección 4.1.3. El archivo de permisos usa la extensión *extGrants* y el encriptado usa la extensión *extSec*.

3. *bool* **hasWritePermissions** (*string Rol, string FileNameMergeFile*): retorna *true* si el rol con nombre *Rol* tiene permisos de escritura sobre el archivo *FileNameMergeFile*. Retorna *false* si el archivo no existe o bien si el rol con nombre Rol no tiene permisos de lectura sobre el archivo *FileNameMergeFile*.

4. *void* **setGrantsRol2File**(*string rol, string grant, string file*): establece sobre el archivo *file* el permiso indicado por *grant* para el rol *rol.* El valor de *grant* debe ser uno de los alias que tienen los permisos ('r', 'w' y '0'). Aquí también se crea el archivo temporal de permisos. Después de utilizar este método (1 o varias veces) se debe ejecutar el método **save()** para crear el archivo protegido definitivo.

5. *int* **save**(): Crea el archivo protegido a partir de los valores de permisos del objeto. Si el usuario no tiene permisos de escritura se devuelve el valor 1 (indicando que no hay permisos) y no se realiza ninguna acción. Si el usuario sí tiene permisos de escritura (esto ocurre también cuando el archivo es un archivo nuevo) se crea el archivo temporal que contiene el texto cifrado y se calcula el checksum del archivo, luego se unen el archivo cifrado con el de permisos para generar el archivo protegido. Los archivos temporales se borran al finalizar la operación, se retorna el valor 0 (indicando éxito).

6. *int* **MAC**(): Establece una marca de checksum sobre el archivo correspondiente al atributo *file.* Si el usuario no tiene permisos de escritura se devuelve el valor 1 (indicando que no hay permisos) y no se realiza ninguna acción. Si el usuario sí tiene permisos de escritura escribe el checksum al archivo de permisos. Después de utilizar este método (1 o varias veces) se debe ejecutar el método **save()** para crear el archivo protegido definitivo. Este método se debe utilizar cuando se están usando permisos sobre los archivos, hay otro método llamado **onlyMAC()** que se usa para marcar con checksum los archivos sin realizar el cifrado.

7. *int* **onlyMAC**(): Establece una marca de checksum sobre el archivo correspondiente al atributo *file*. Aquí se genera un archivo con la extensión *extSum*, este archivo es similar al archivo protegido, la diferencia radica en que el contenido no se guarda encriptado y sólo se usa para verificar los checksum de los archivos. Este método se debe usar cuando se desea sólo establecer marcas de hash sobre los archivos.

8. *int* **access**(): Desencripta el archivo protegido de acuerdo al atributo *file*. En el caso que el archivo protegido no exista devuelve el valor -2 sin realizar ninguna acción; Si el usuario no tiene permisos de lectura devuelve el valor 1 y no realiza ninguna acción, si el usuario sí tiene permisos de lectura desencripta el archivo y devuelve el valor 0. Para ello divide el archivo protegido en el archivo de permisos y el archivo encriptado (estos archivos son temporales), luego desencripta el archivo encriptado según los permisos del archivo de permisos. Al finalizar se borran los archivos temporales y se conservan el archivo protegido (este puede compartirse manteniendo los permisos si se desea) así como el archivo desencriptado (para ver su contenido y trabajar sobre el en el equipo local).

9. *int* **check** (): Verifica el checksum del archivo protegido de acuerdo al atributo *file.* En el caso que el archivo protegido no exista devuelve el valor -2 y no realiza ninguna acción. Si el archivo protegido existe, realiza la verificación de cheksum (entre el checksum de este archivo y el indicado en el archivo

protegido), si los valores coinciden devuelve el valor 0, y si difieren devuelve el valor 1.

10. *int* **accessOnlyMAC**(): Extrae el contenido de un archivo con una marca de checksum. Se usa sobre los archivos marcados con el método **onlyMAC()**(archivos con extensión *extSum*)**.** En el caso que el archivo no exista devuelve el valor -2 y no realiza ninguna acción. Si el archivo existe, extrae su contenido y lo deja en un archivo sin la extensión *extSum*.

11. *int* **checkOnlyMAC**():Verifica el checksum del archivo con una marca de checksum de acuerdo al atributo *file*. Se usa sobre los archivos marcados con el método **onlyMAC()**(archivos con extensión *extSum*)**.** En el caso que el archivo no exista devuelve el valor -2 y no realiza ninguna acción. Si el archivo existe, realiza la verificación de cheksum (entre la marca de hash y el checksum del archivo), si los valores coinciden devuelve el valor 0, y si difieren devuelve el valor 1.

También se provee los siguientes métodos estáticos:

1. *string[]* **getRoles**(*string rolsPath*): Retorna un arreglo de strings con los roles del sistema. Estos están ordenados ascendentemente según el identificador de rol. La información de los roles se obtiene del archivo de configuración representado por la ruta *rolsPath*.

2. *byte[]* **getKey**(*string FileNameClaves, string FileNameTargetFile*): Retorna la llave del archivo de claves de acuerdo a los permisos del archivo *FileNameTargetFile*. La clave se obtiene buscando en el archivo *FileNameClaves* la clave correspondiente al permiso del descrito en la cabecera del archivo *FileNameTargetFile.*

3. *string* **getMAC**(*string FileNameTargetFile*): Retorna la marca de HMAC SHA-1 del archivo *FileNameTargetFile.*

4. *string* **getMyRol**(): Retorna el nombre del rol correspondiente al usuario del equipo. Esta función es dependiente del equipo, puesto que retorna el valor leyendo los archivos de configuración locales dentro del equipo.

5.1.4 Los módulos Hexadecimal y Helpers

Los módulos Hexadecimal y Helpers prestan servicios de apoyo a los demás módulos y no proveen de funcionalidad extra ni mejorada al sistema. Estos módulos surgen de una refactorización del código, al identificar trozos de código comunes en los demás módulos.

El módulo Hexadecimal es usado por los módulos GrantsManager y RolesAdmin para manejar las llaves de los roles del sistema; principalmente para transformar las llaves escritas en los archivos de configuración (el archivo de claves) a las llaves usadas por los objetos de la clase Grants.

El módulo Helpers es usado por los módulos RolesAdmin, ManetSec y GrantsManager, y realiza tareas como escribir un string a un archivo, leer una línea específica de un archivo, concatenar archivos, separarlos, etc.

5.1.5 ManetSec

Este módulo es la aplicación que permite a los usuarios proteger y compartir archivos sobre una MANET. El módulo no provee ningún tipo de servicio, simplemente provee la GUI para que los usuario lleven a cabo sus tareas. A través de la interacción se hacen las llamadas pertinentes a los servicios que provee el módulo GrantsManager. El funcionamiento de este módulo se explica en la sección 6.2.

5.1.6 RolesAdmin

El módulo RolesAdmin es un programa de consola que realiza las acciones que se indican en la sección 3.2.1.1. El programa tiene el método main y varios métodos que son los que se usan para definir los roles y sus claves. Estos métodos (todos estáticos) se describen a continuación:

1. *String* **CreateKey**(*int numBytes*): Genera una clave pseudoaleatoria de *numBytes* bytes, la clave se representa como una cadena de números hexadecimales. La cadena de números hexadecimales se retorna como un string. La clave se genera usando la clase RNGCryptoServiceProvider del N.S.S.C. que es un generador de números (o bytes) pseudoaleatorios.

2. *int* **getRolId**(*string rol, string rolesPath*): Retorna el identificador correspondiente al rol con nombre *rol*. La información es leída del archivo de roles indicado por *rolesPath*.

3. *void* **CreateRoles**(*string[] roles, string rolsFile*): Crea el archivo de configuración de roles del sistema. El archivo *rolesFile* es creado y en él se escriben los roles indicados por el arreglo roles. Se escribe un rol en cada línea.

4. *void* **CreateKeysForRoles**(*int numBytes, string[] Rols, string keysPath*): Crea el archivo de configuración con todas las claves del sistema. Dicho archivo es keysPath, los roles que se usan son los del arreglo Rols, y el largo de las claves es de numBytes bytes.

5. *void* **CreateRoleKeysFile**(*int idRol, string[] Rols, string allRolesKeysFile*): Crea el archivo con las claves correspondientes al rol con identificador *idRol* según los roles en *Rols*; *allRolsKeysFile* es el archivo con todas las claves. El archivo de claves para el rol correspondiente se crea bajo el mismo directorio que el archivo con todas las claves, además se usa como prefijo el nombre de este archivo y como sufijo el nombre del rol.

6. Pruebas de la solución

En esta sección se describen las pruebas de desempeños usadas para seleccionar el archivo de cifrado, así como las pruebas de la aplicación ManetSec.

6.1 Pruebas de desempeño

En esta sección se describen las pruebas de desempeño usadas para seleccionar el algoritmo usado para encriptar los archivos. Se realizaron pruebas con los algoritmos DES, 3DES y AES, también se evaluó el desempeño del algoritmo de hash SHA-1 para observar su desempeño comparado con los algoritmos de encriptación. Se usó encriptación solamente en las pruebas debido a que para realizar la desencriptación en los algoritmos usados corresponde a una encriptación con una llave diferente (derivada de la clave de encriptación) [Bellare 2006, Chapter 3].

Para las pruebas se usaron 2 PDA's:

- Un modelo "HP-ipaq h4150", con un procesador Intel PXA255 de 400MHz, con 64MB de RAM y 32 MB ROM. Con el SO Windows CE 4.2.

- Un modelo "DELL-axim" X50V, con un procesador Intel PXA270 de 642 MHz, con 64MB de RAM y 128 MB ROM. Con el SO Windows Mobile 5.0.

Se hicieron 4 pruebas usando los 4 algoritmos (AES, DES, 3 DES y SHA-1):

- Se usaron archivos pequeños, de 100 KB hasta 900KB. Aumentando el tamaño en 100 KB cada vez (100, 200,…, 800, 900). Esta secuencia se probó en la PDA "HP-ipaq" y en la PDA "DELL-axim".

- Se usaron archivos grandes, de 1 MB hasta 8,5MB. Aumentando el tamaño en 0.5 MB cada vez (1, 1.5, 2,…, 8, 8.5). Esta secuencia se probó en la PDA "HP-ipaq" y en la PDA "DELL-axim".

A continuación se muestran gráficos de línea indicando el desempeño de cada uno de los algoritmos:

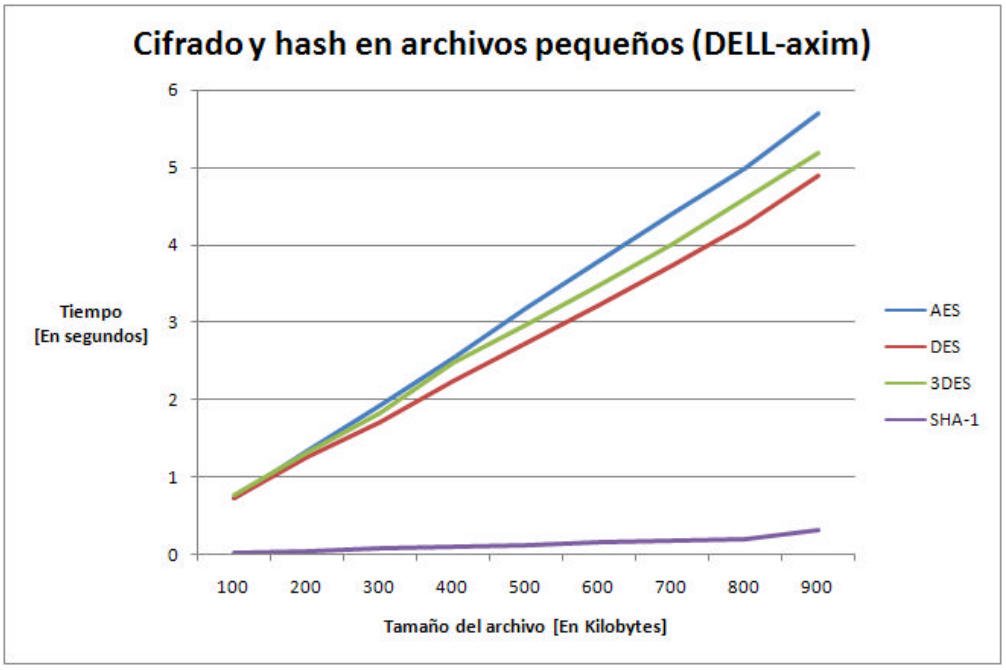

Figura 8: Cifrado y hash en archivos pequeños para la PDA “DELL-axim”

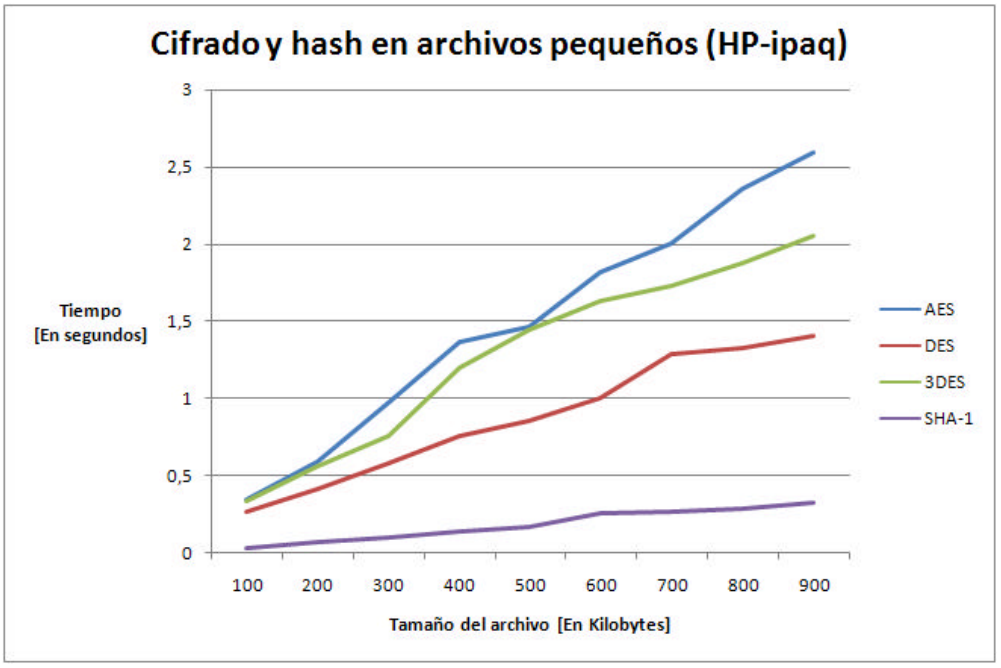

Figura 9: Cifrado y hash en archivos pequeños para la PDA “HP-ipaq”

En las pruebas realizadas la PDA “HP-ipaq” tuvo un mejor desempeño que la PDA “DELL-axim”, a pesar de tener características de hardware más reducidas. Esto seguramente se debe al sistema operativo, puesto Windows Mobile tiene una mejor GUI que Windows CE, además de activar más servicios. Vale decir que estas pruebas se realizaron con el fin de comparar el desempeño de los algoritmos y no el de los dispositivos. Por esta razón lo más importante es el desempeño relativo de los algoritmos en cada dispositivo.

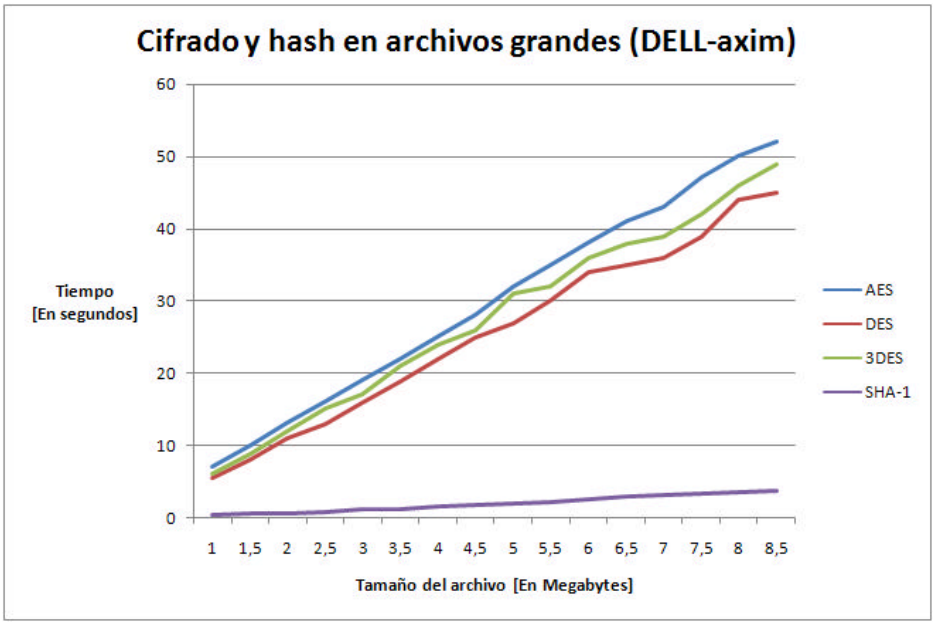

Figura 10: Cifrado y hash en archivos grandes para la PDA “DELL-axim”

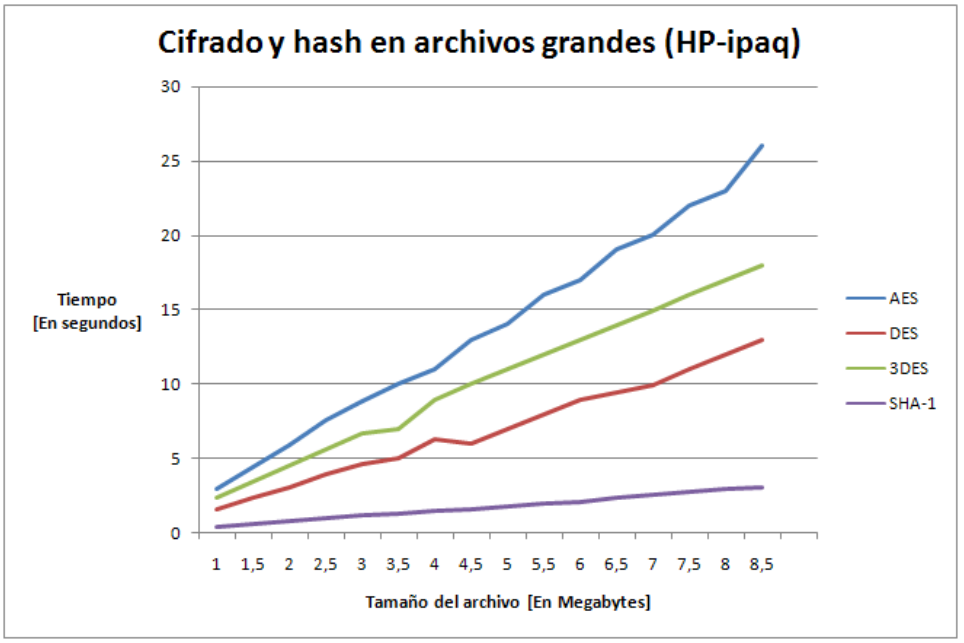

Figura 11: Cifrado y hash en archivos grandes para la PDA “HP-ipaq”

En las pruebas realizadas se pudo observar que el algoritmo de DES es el más eficiente de los 3. Además la prueba revela que DES es significativamente más rápido que 3DES y AES, y por tanto su uso permite sin duda una mejor experiencia para los usuarios del sistema colaborativo subyacente.

Debido a esta diferencia significativa en el desempeño se optó por usar DES en vez de los otros algoritmos para la solución implementada, esto a pesar de que DES es considerado el menos seguro de los 3 algoritmos. Sin embargo si se quisiera mejorar la seguridad de la encriptación (en detrimento del desempeño) el impacto de un cambio de

algoritmo es menor (sólo se debe modificar la clase *GrantsManager* en los métodos que crean objetos de la clase Crypto).

Respecto del algoritmo SHA-1 se pudo observar que el tiempo usado en calcular el checksum de los archivos es despreciable respecto del tiempo usado en las tareas de encriptación.

6.2 Pruebas de ManetSec

Las pruebas de la aplicación ManetSec consisten en: establecer la configuración inicial por parte del administrador del sistema colaborativo (generación de roles e instalación de claves); definir permisos por parte de los usuarios móviles sobre los archivos; compartir archivos dentro de una MANET; y acceso a los archivos de acuerdo a los permisos de los distintos roles del sistema.

6.2.1. Estableciendo la configuración inicial

Un paso preliminar para que los usuarios móviles del sistema colaborativo subyacente puedan proteger y compartir sus archivos es establecer la configuración inicial respecto de los roles del sistema, así como para compartir archivos dentro de la MANET.

El administrador del sistema debe crear el directorio "C:\Manet" dentro de su estación de trabajo, luego debe generar los roles del sistema (como se explicó en la sección 4.1.1), para eso usa la interfaz por consola:

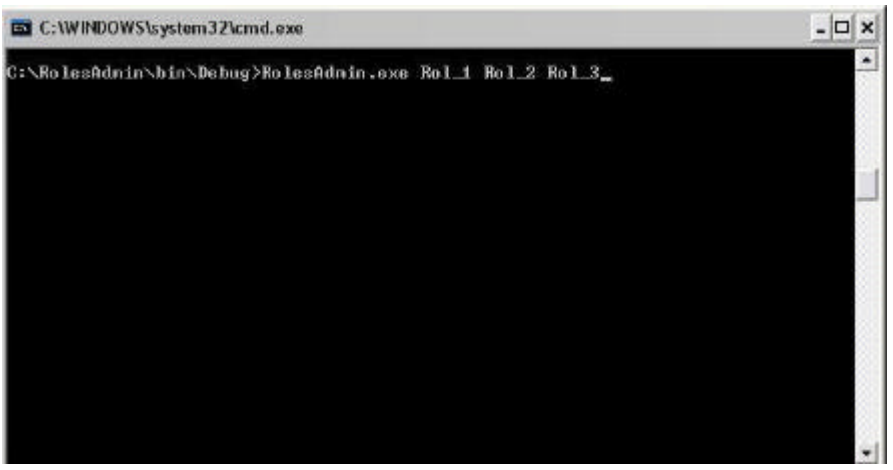

Figura 12: Generación de roles por consola

En este caso se generan 3 roles, los que llevan por nombre "Rol_1", "Rol_2" y "Rol_3". Una vez ejecutado el comando se generan 2 archivos en el directorio "C:\Manet", el archivo keys, que contiene todas las claves del sistema y el archivo roles, que contiene todos los roles del sistema.

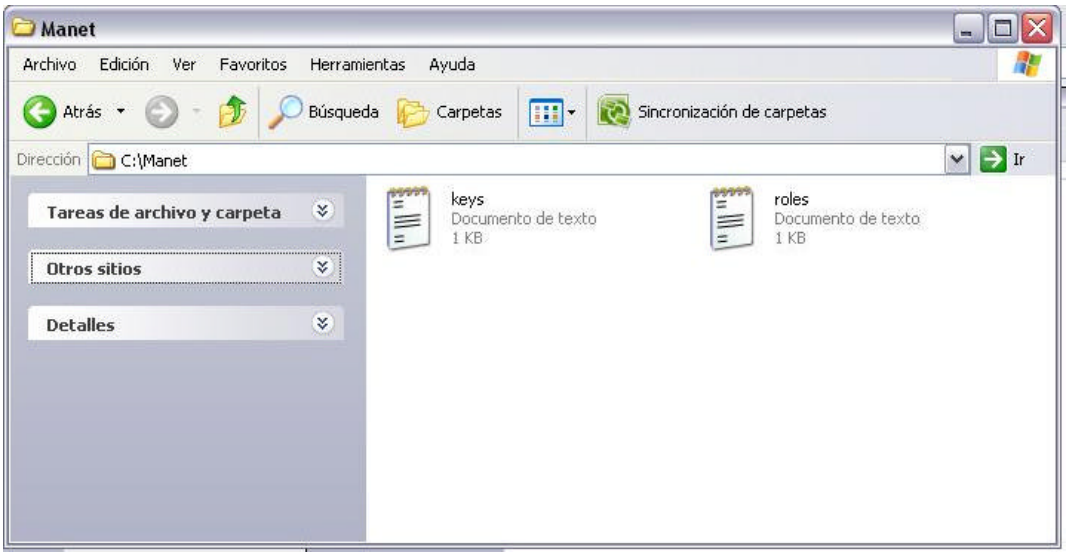

Figura 13: Archivo de claves y roles

El contenido de los archivos es el nombre de todos los roles para el caso del archivo “roles.txt”, y todas claves y sus respectivas cadenas de bits para el caso del archivo “keys.txt”.

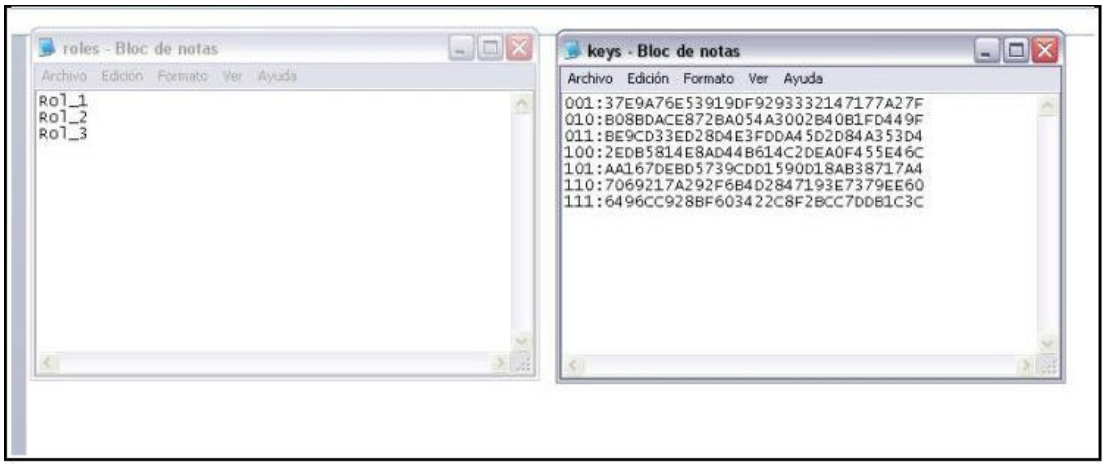

Figura 14: Contenido de los archivos de claves y roles

El siguiente paso es seleccionar las claves para cada uno de los roles del sistema:

Figura 15: Generación de archivo de claves para 4 roles

Una vez ejecutado el comando para cada rol se generan un archivo de claves por cada uno, en la figura 13 se marcan los archivos de claves generados para cada rol.

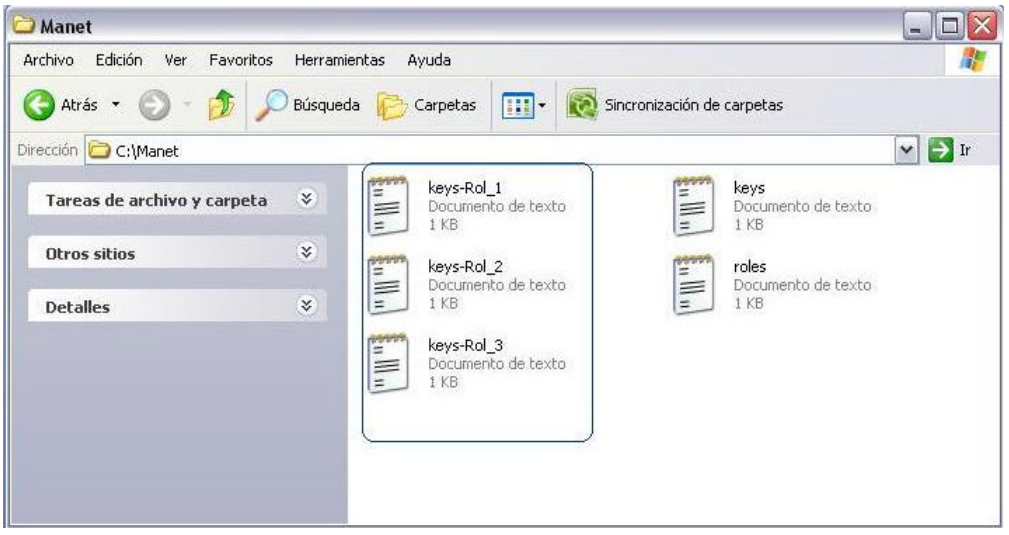

Figura 16: Archivos de claves para 4 roles

Cuando un nuevo usuario pasa a formar parte del sistema colaborativo subyacente, el administrador debe copiar el archivo “roles.txt” y el archivo correspondiente al rol del nuevo usuario al directorio “/My Documents/Personal/ manet/” en el dispositivo móvil.

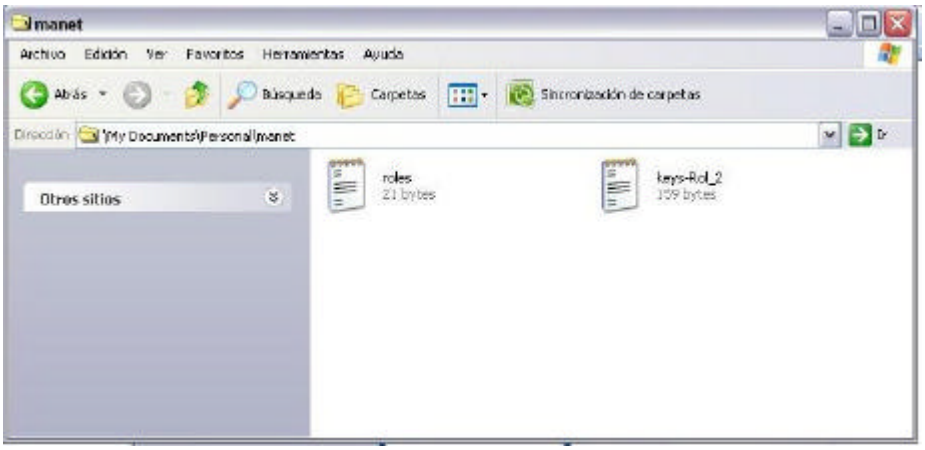

Figura 17: Instalación de claves para un usuario

En la figura 14 se muestran los archivos que deben ser copiados al dispositivo de los usuarios móviles pertenecientes al rol “Rol_2”. El archivo de claves copiado al dispositivo móvil debe ser renombrado a “keys.txt”, esto es debido a que se usa la convención de que este es el archivo de llaves de un usuario móvil (las librerías van a ese archivo a buscar las claves).

Figura 18: Renombrando el archivo de claves de un usuario

Por parte de los usuarios móviles, ellos deben crear el directorio “/My Documents/download/” y el directorio “/My Documents/share/” para usarlos como directorios para compartir archivos sobre la MANET como se explica en la sección 3.2.5. Se definieron estas rutas debido a que los cuadros de diálogo para abrir archivos en el .NET Compact Framework 3.5 permiten ver sólo los directorios que están bajo el directorio “/My Documents/”, de este modo los directorios (y los archivos que contienen) que están en otra ubicación no son accesibles vía cuadros de diálogo para abrir archivos (sólo por el explorador). Debido a la ubicación definida para los directorios compartidos y de descarga de archivos, esto debe de configurarse también en la aplicación ManetSec.

Figura 19: Configuración de usuario de ManetSec

Una vez realizados todos los pasos descritos se puede comenzar a utilizar la aplicación MantSec.

6.2.2. Compartiendo archivos protegidos

Una vez establecida la configuración inicial los usuarios pueden comenzar a proteger y compartir archivos dentro de la MANET. En esta sección se muestra un ejemplo en el cual 3 usuarios móviles forman una MANET para compartir archivos protegidos. Las características de los usuarios son las siguientes:

- Se usan dos PDA's y un notebook.
- El usuario de la primera PDA es "DELL-axim", este usuario pertenece al "Rol_1".
- El usuario de la segunda PDA es "HP-ipaq", este usuario pertenece al "Rol_2".
- El usuario del notebook es "Acer", este usuario pertenece al "Rol_2".

La prueba consiste en lo siguiente:

- El usuario "HP-ipaq" establece permisos sobre un archivo.
- Los usuarios "DELL-axim" y "Acer" descargan el archivo desde el dispositivo del usuario "HP-ipaq".
- Se comprueba en los dispositivos que los usuarios pueden realizar acciones según los permisos establecidos.
- El usuario "HP-ipaq" verifica la integridad de los datos de un archivo marcado de checksum (con una huella de hash y sin encriptar).

6.2.2.1. Estableciendo permisos sobre un archivo

El usuario “HP-ipaq” establece los siguientes permisos sobre el archivo mensaje.txt:

- Permisos de escritura para el rol “Rol_1”
- Permisos de lectura para el rol “Rol_2”
- Ningún permiso para el rol “Rol_3”

Primero el usuario selecciona el archivo que desea proteger presionando el botón “Abrir”.

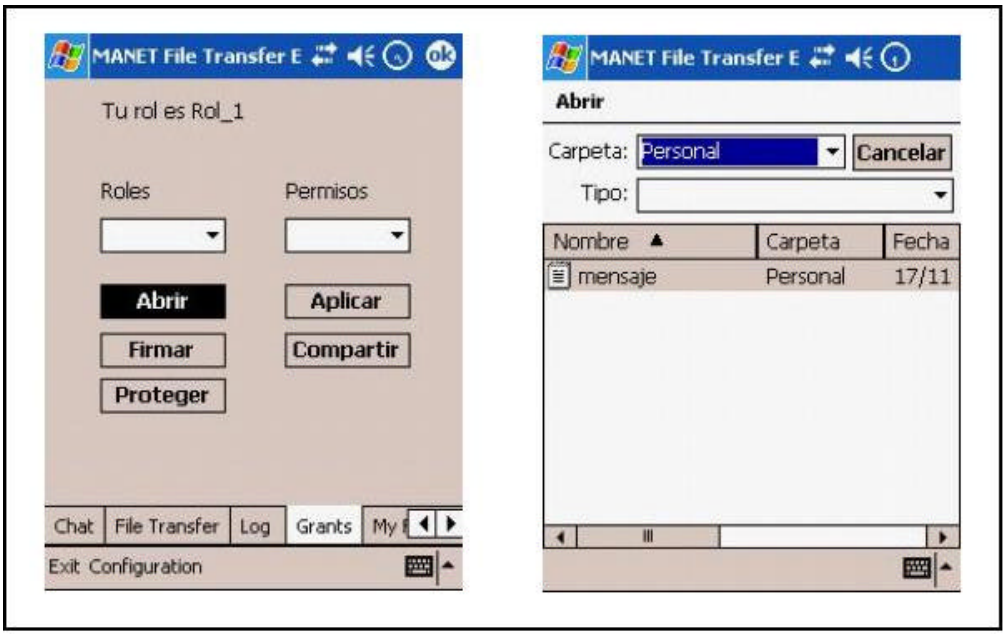

Figura 20: Selección del archivo a proteger

Una vez seleccionado el archivo se comienzan a establecer los permisos, se selecciona el rol y el permiso asignado para ese rol, presionando el botón “Aplicar” se establecen los permisos. Primero se establecen permisos de escritura para el rol “Rol_1”.

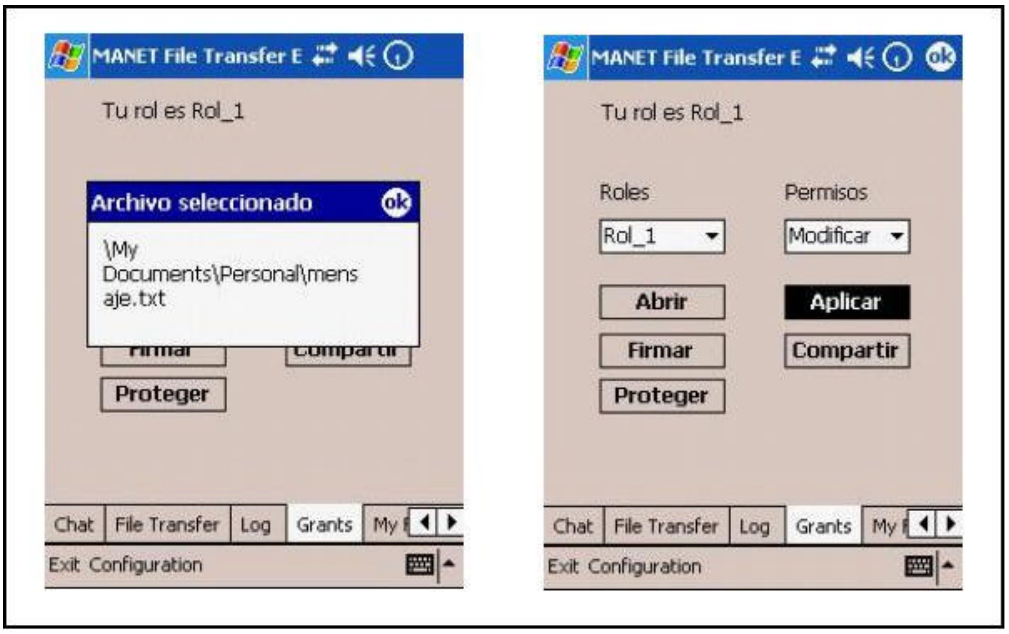

Figura 21: Estableciendo permisos de escritura para el rol “Rol_1” sobre el archivo “mensaje.txt”

Luego se establecen permisos de lectura para el rol “Rol_2” y permisos nulos para el rol “Rol_3”.

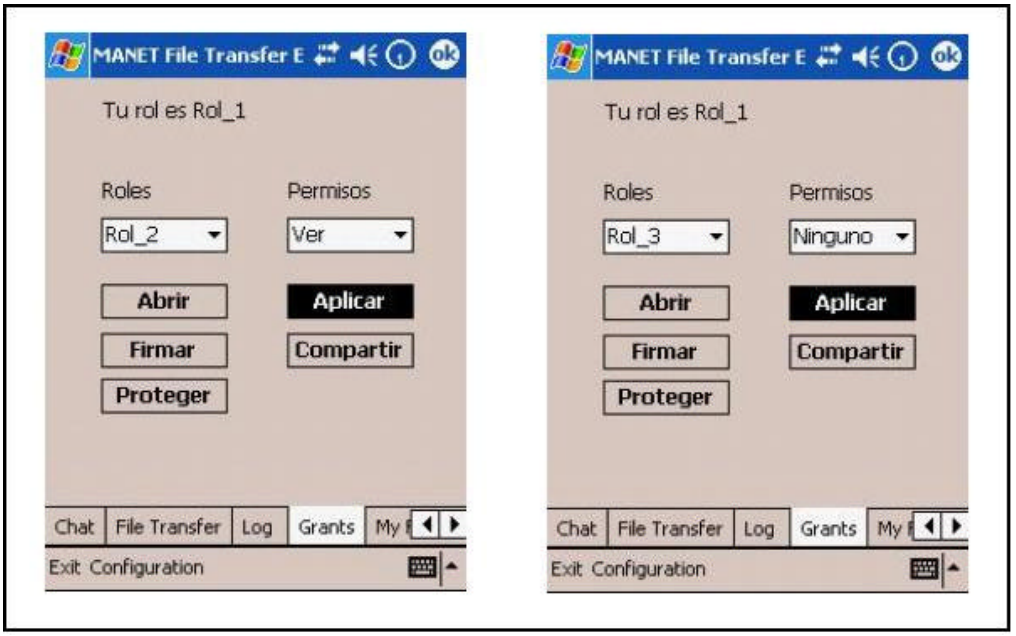

Figura 22: Estableciendo permisos de lectura para el rol “Rol_2” y permisos nulos para el rol “Rol_3” sobre el archivo “mensaje.txt”

Una vez establecidos todos los permisos se deben guardar pulsando el botón “Proteger”.

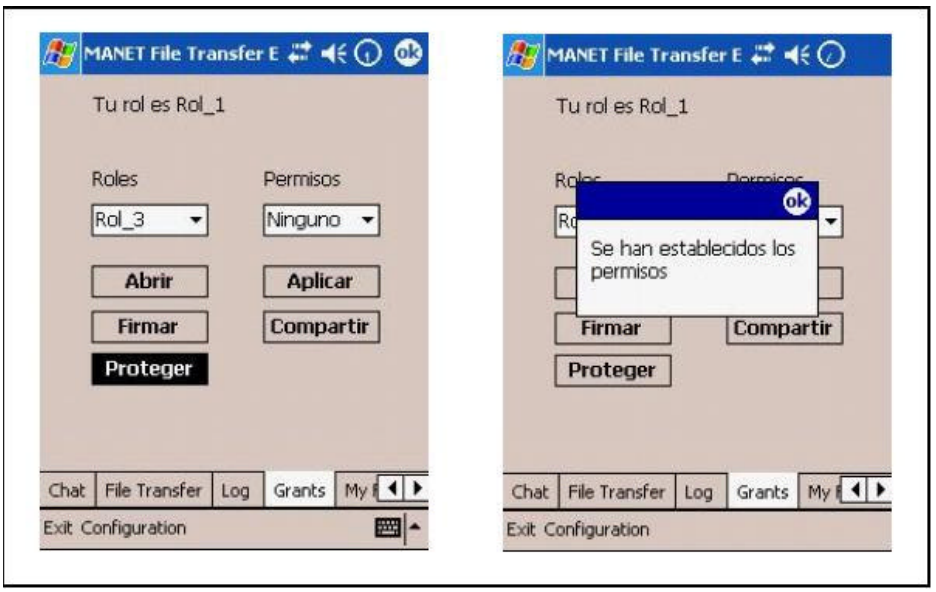

Figura 23: Guardando los permisos sobre un archivo

Una vez protegido el archivo, éste se puede compartir (mover al directorio compartido) a través de la MANET pulsando el botón "Compartir". Cabe mencionar que es en éste momento cuando se lleva a cabo la encriptación del archivo y el cálculo del checksum, tal como se muestra en la Figura 6. El archivo protegido queda con la extensión indicada por el atributo *extMrg* del objeto *GrantsManager* usado (por defecto es ".sec.txt").

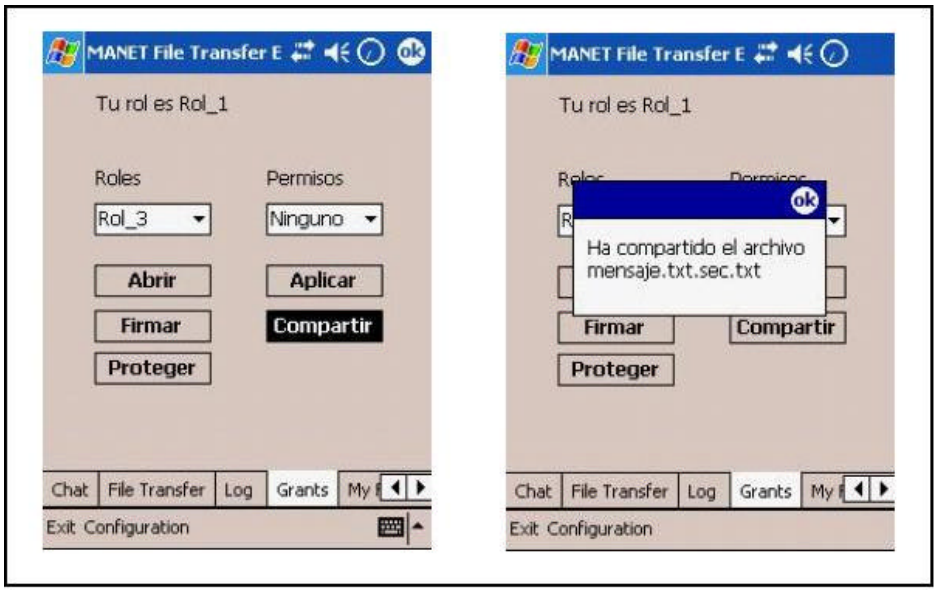

Figura 24: Compartiendo el archivo protegido

El archivo que se comparte es el archivo protegido, el archivo claro permanece intacto y puede ser usado para trabajar localmente. Sin embargo si se desea compartir una nueva versión del archivo "mensaje.txt", se deben establecer nuevamente los permisos sobre la nueva versión. Una vez compartido el archivo, éste puede ser descargado por cualquier usuario que se encuentre en la misma MANET que el usuario "HP-ipaq".

6.2.2.1. Compartiendo un archivo protegido

Los 3 usuarios (“HP-ipaq”, “DELL-axim” y “Acer”) forman una MANET y una vez que todos están conectados pueden transferir archivos entre ellos.

El usuario “HP-ipaq” comparte el archivo “mensaje.txt.sec.txt” con los demás usuarios de la MANET. Este archivo es descargado por los usuarios “Acer” y “DELL-axim”. Una vez conectados, cada uno de los usuarios puede ver a los demás usuarios dentro de la MANET. El ícono del usuario propio se ve en color azul y los demás en verde.

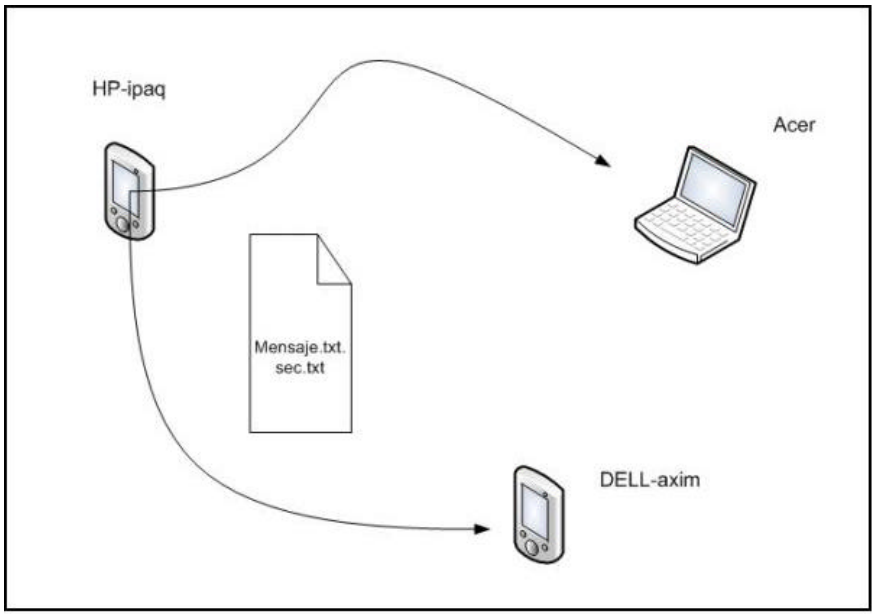

Figura 25: “Hp-ipaq” compartiendo archivos con los demás usuarios de la MANET

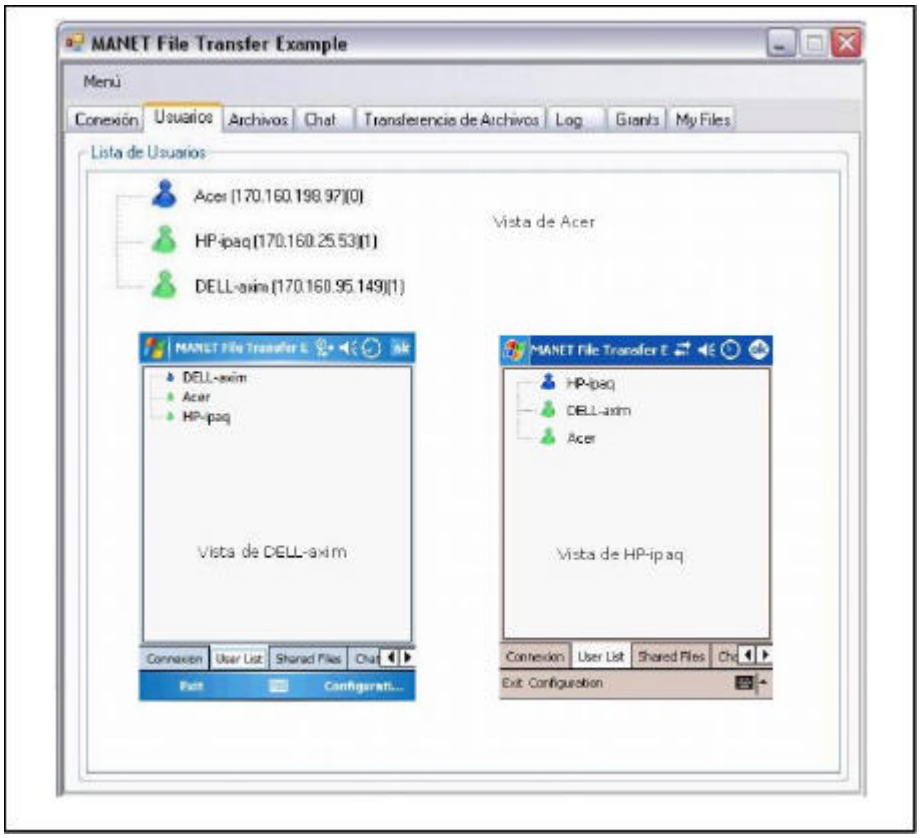

Figura 26: Vista de la lista de usuarios para cada uno de los dispositivos de la MANET

El usuario “Acer” puede ver el archivo “mensaje.txt.sec.txt” en la lista de archivos compartidos. El usuario “Acer” descarga el archivo desde el dispositivo del usuario “HP-ipaq”.

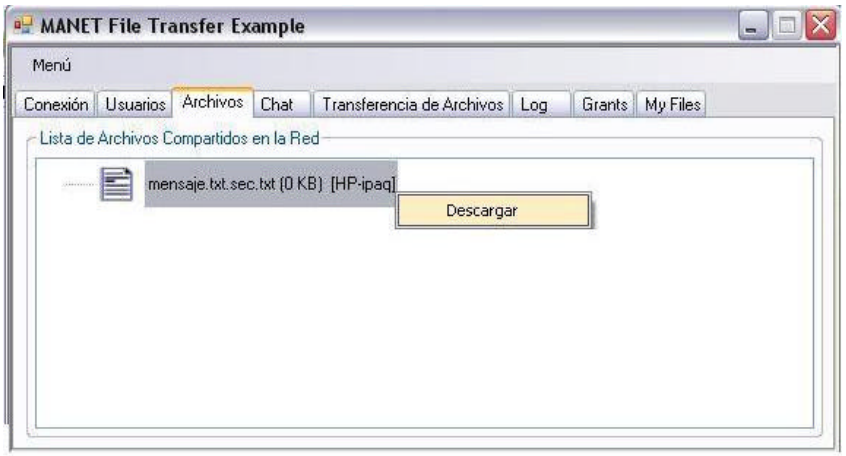

Figura 27: Descarga de un archivo compartido desde la PDA del usuario “HP -ipaq” al notebook del usuario “Acer”

En la pestaña transferencia de archivos el usuario “Acer” puede ver el progreso de la descarga.

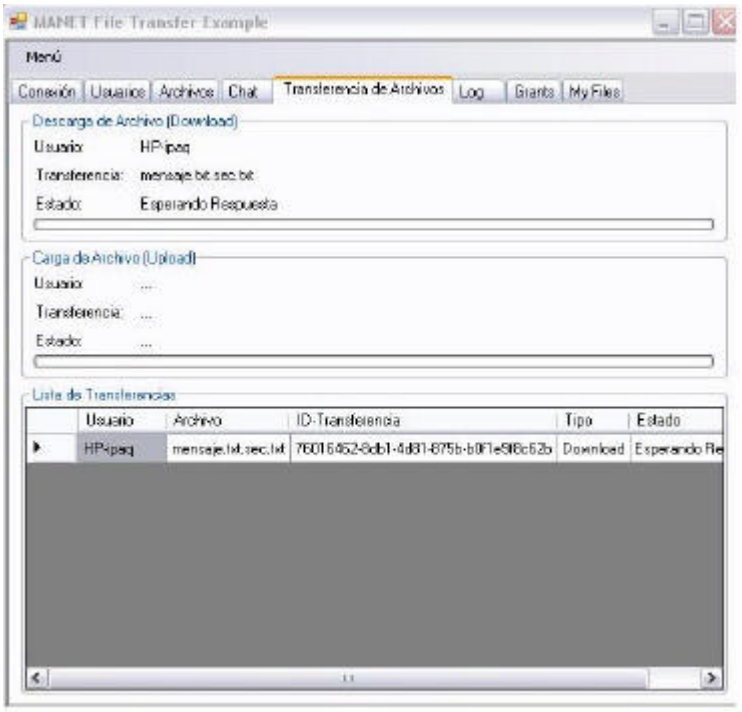

Figura 28: Progreso inicial de la descarga de un archivo compartido

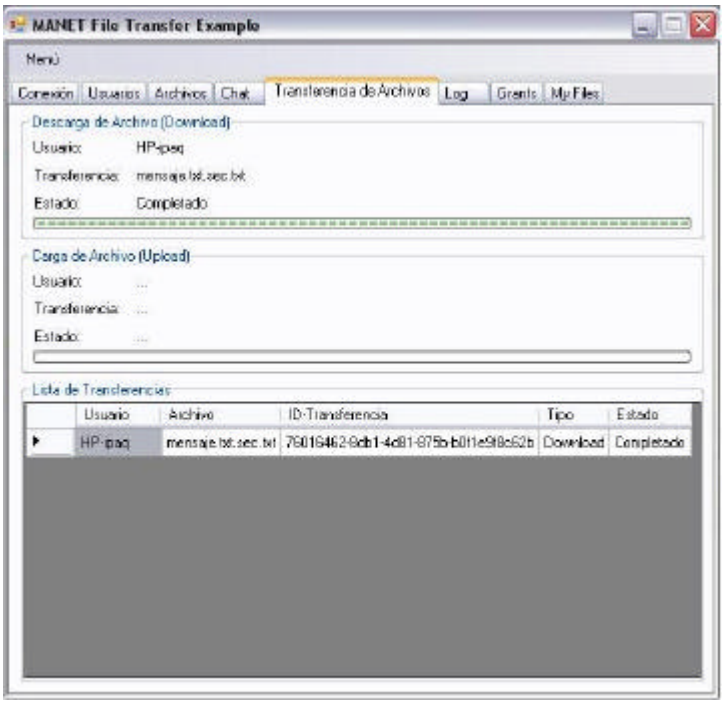

Figura 29: Progreso final de la descarga de un archivo compartido

En la pestaña “MyFiles” el usuario “Acer” puede ver el archivo protegido descargado” (puede ver la lista de todos los archivos protegidos descargados). El archivo descargado tiene como prefijo el nombre de usuario desde donde se descargó el archivo y un número indicando la “versión” del archivo (la primera vez que se descarga es un ‘0’, la segunda vez un ‘1’ y así sucesivamente). A través de un diálogo contextual el usuario puede descifrar el archivo marcando la opción acceder (sólo si tuviera los permisos pertinentes).

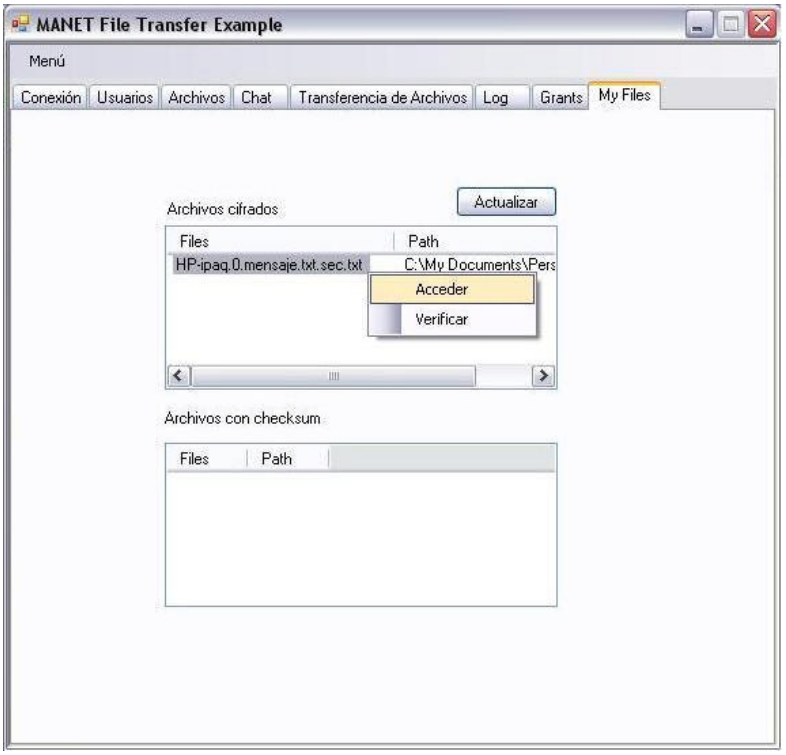

Figura 30: Accediendo a un archivo protegido

El usuario “Acer” no puede ver el archivo “mensaje.txt.sec.txt”, debido a que su rol (el “Rol_3”) no tiene permisos de lectura

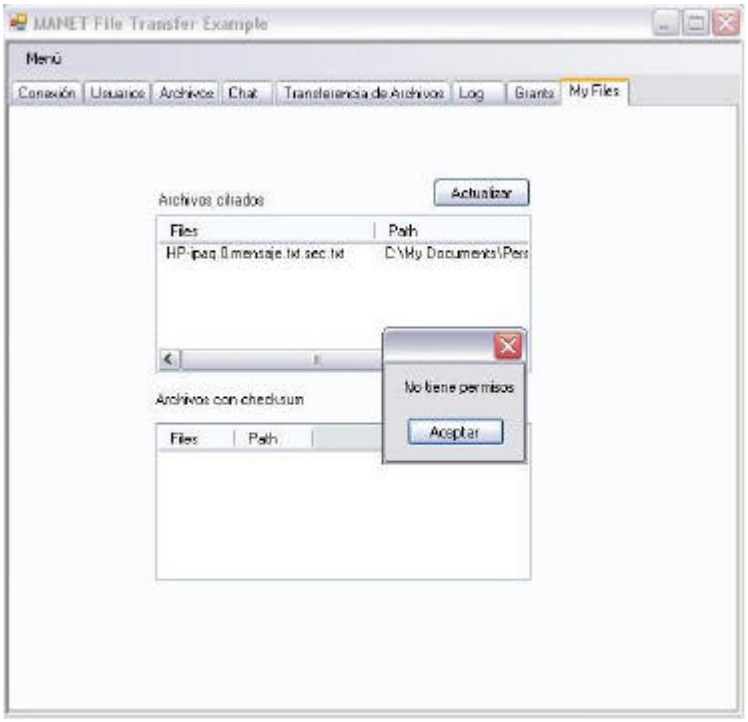

Figura 31: Mensaje indicando que el usuario no tiene acceso al contenido del archivo protegido

Sólo los usuarios con permisos de lectura pueden verificar la integridad de los datos de un archivo protegido. Debido a que deben poseer la llave correspondiente para poder llevar a cabo la verificación.

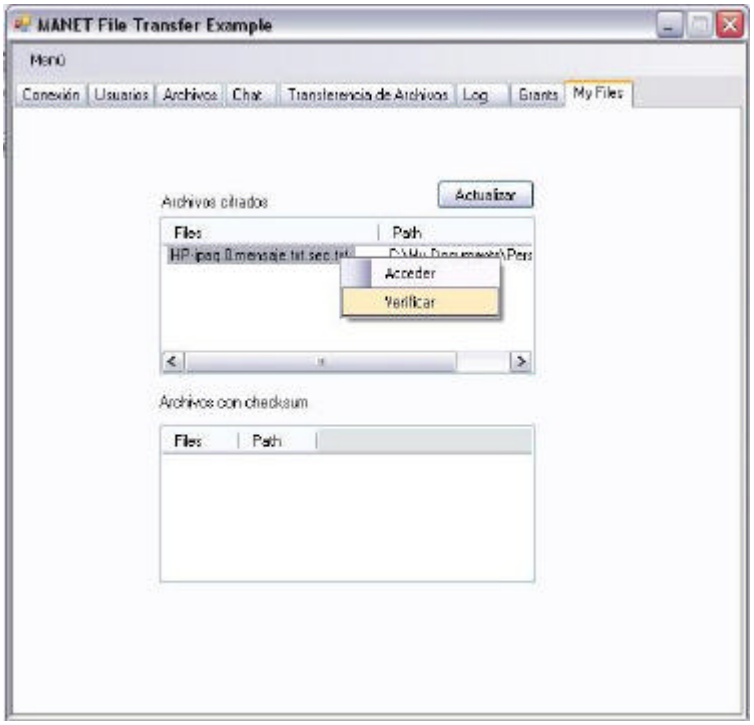

Figura 32: Verificando un archivo protegido

El usuario “Acer” no puede verificar el archivo “mensaje.txt.sec.txt” debido a que no posee permisos de lectura y por tanto no posee la llave necesaria para calcular el checksum (vía HMAC).

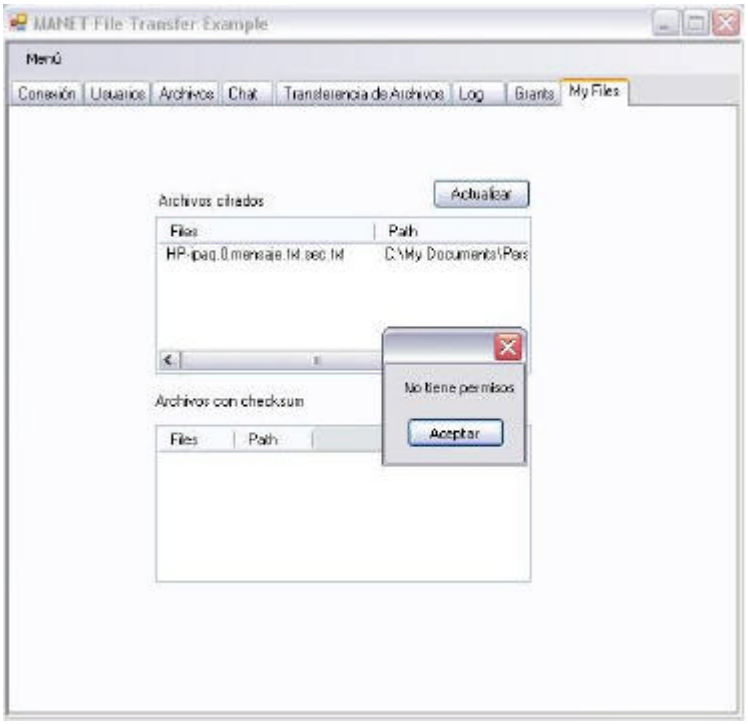

Figura 33: Mensaje indicando que el usuario no puede verificar un archivo sin tener permisos de lectura

Del mismo modo que el usuario “Acer” puede descargar el archivo, “mensaje.txt.sec.txt” el usuario “DELL-axim” también puede hacerlo.

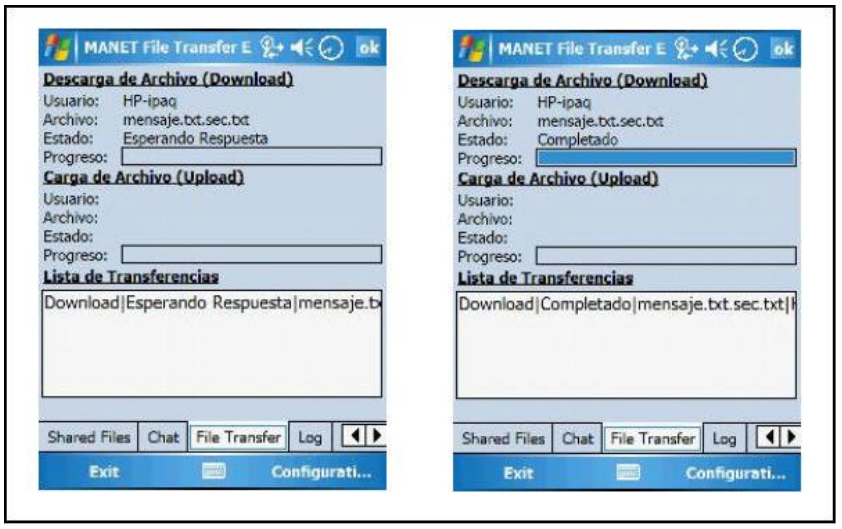

Figura 34: Progreso de la descarga de un archivo compartido

El usuario “DELL-axim” puede descifrar el archivo “mensaje.txt.sec.txt”, debido a que posee permisos de lectura.

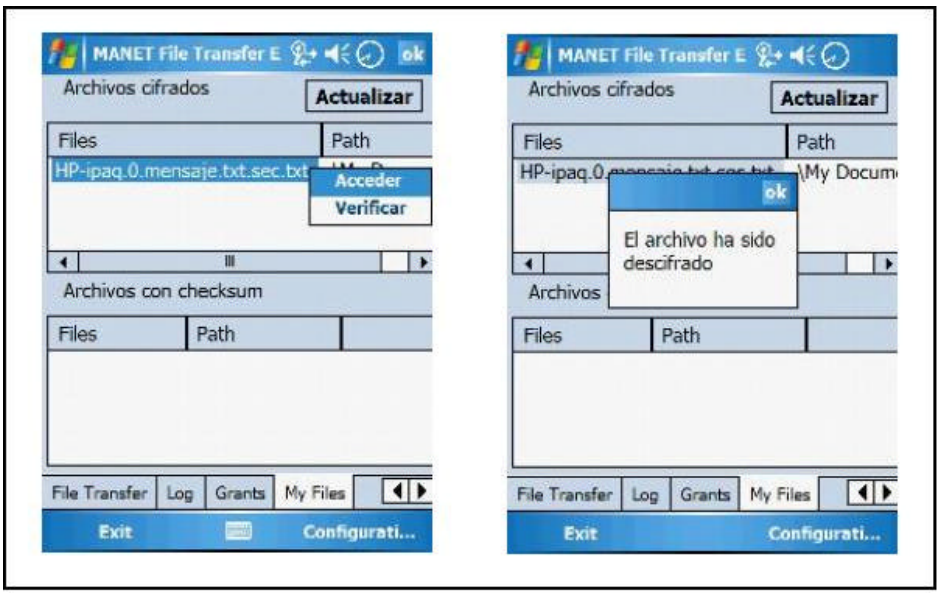

Figura 35: Descifrando un archivo protegido

El usuario “DELL-axim” puede verificar la integridad del archivo protegido debido a que tiene permisos de lectura sobre éste.

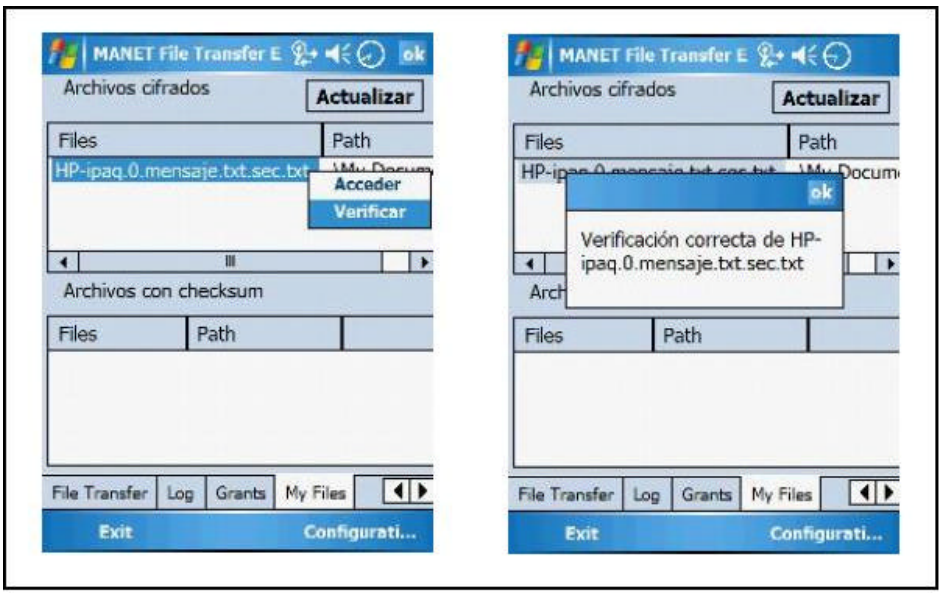

Figura 36: Verificando el checksum de un archivo protegido

Al descifrar un archivo protegido se conserva la versión protegida (con el contenido cifrado) y además se genera el archivo original (texto claro).

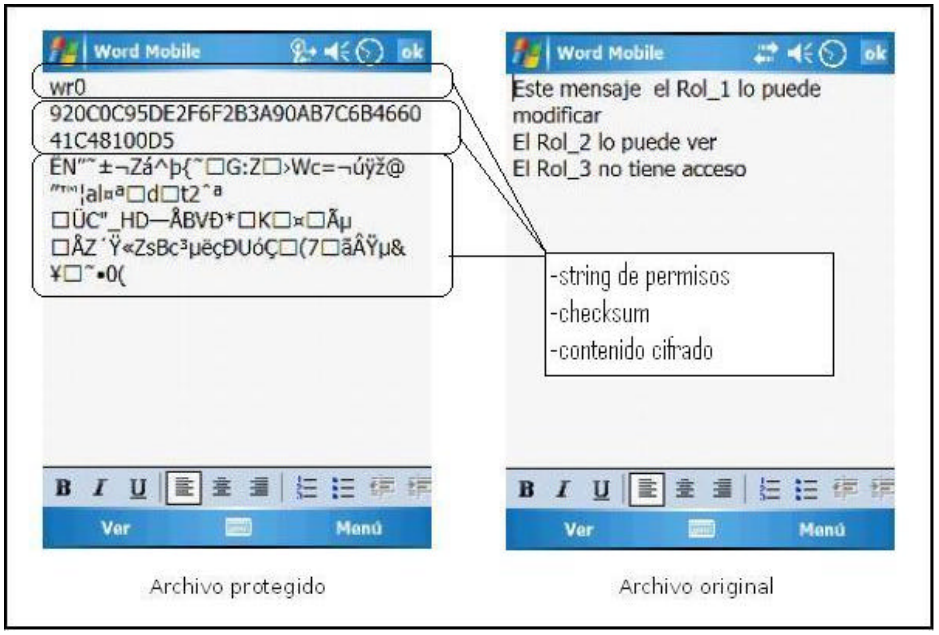

Figura 37: Archivo protegido y archivo original

Los usuarios tienen la opción de “firmar” los archivos, que en realidad es calcular el checksum del archivo y dejar la huella en él, esto no es una firma bajo el concepto de firmas digitales, es un MAC, pero se usa este término debido a que otros términos como “marcar”, “dejar huella” o “calcular hash” pueden ser confusos para los usuarios.

Si el usuario “DELL-axim” intenta firmar o establecer permisos sobre el archivo protegido, no puede debido a que no tiene permisos de escritura.

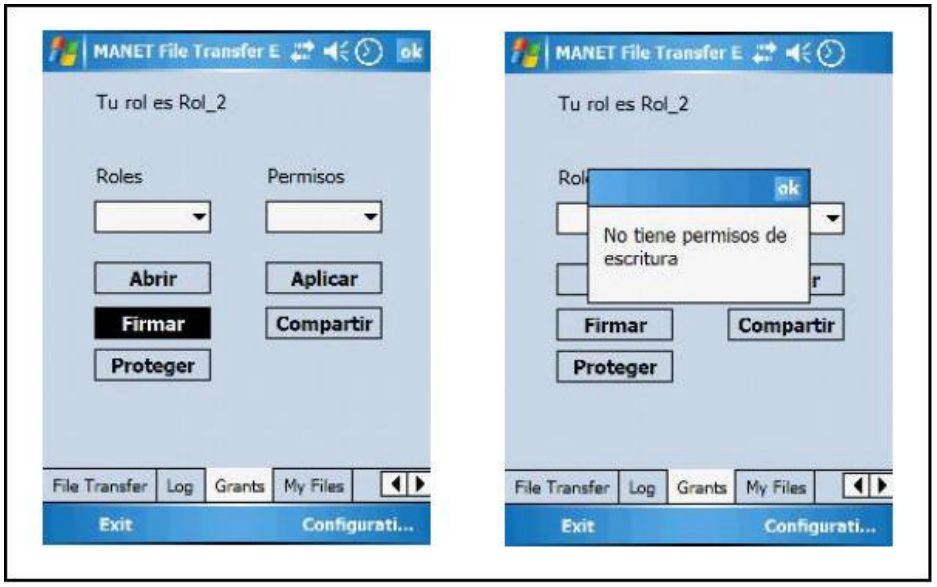

Figura 38: Intento de firmar un archivo por un usuario sin permisos de escritura

El usuario “HP-ipaq” puede descifrar el archivo “mensaje.txt.sec.txt”, debido a que posee permisos de lectura.

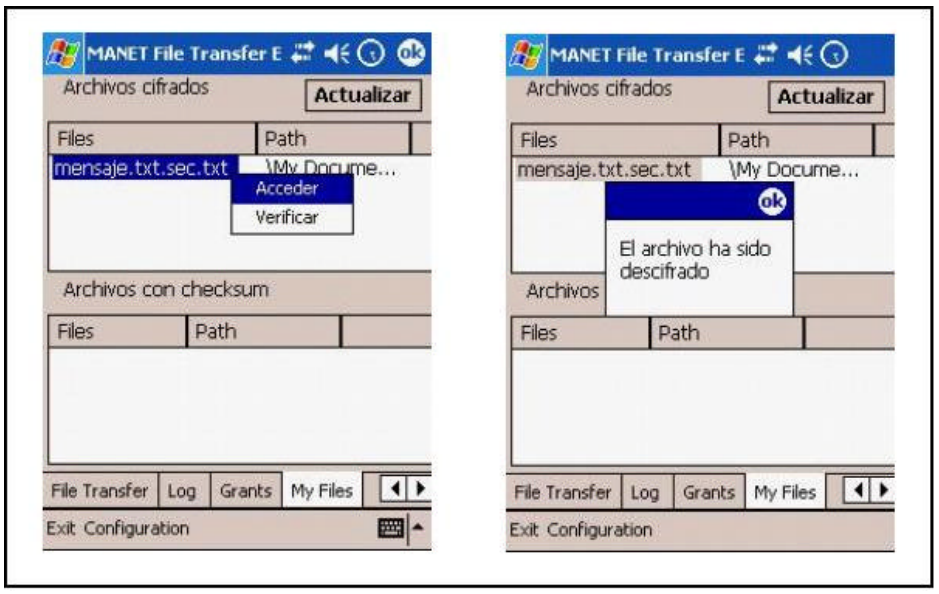

Figura 39: Descifrando un archivo protegido

El usuario “HP-ipaq” puede verificar la integridad del archivo protegido debido a que tiene permisos de lectura sobre éste.

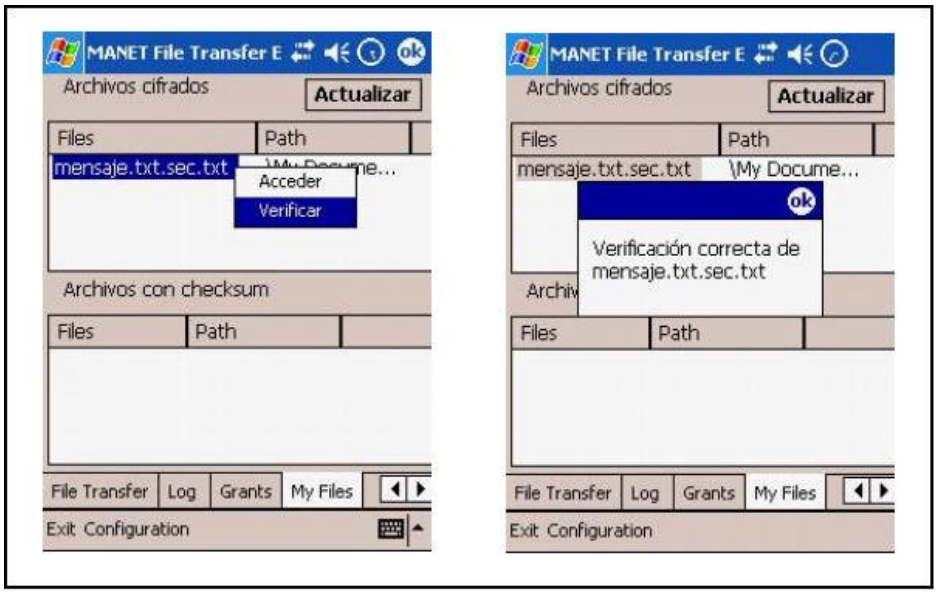

Figura 40: Verificando el checksum de un archivo protegido

El usuario “HP-ipaq” puede cambiar el archivo “mensaje.txt” y volver a firmarlo o establecer permisos. Antes, el archivo original cambia como se muestra en la figura 43.

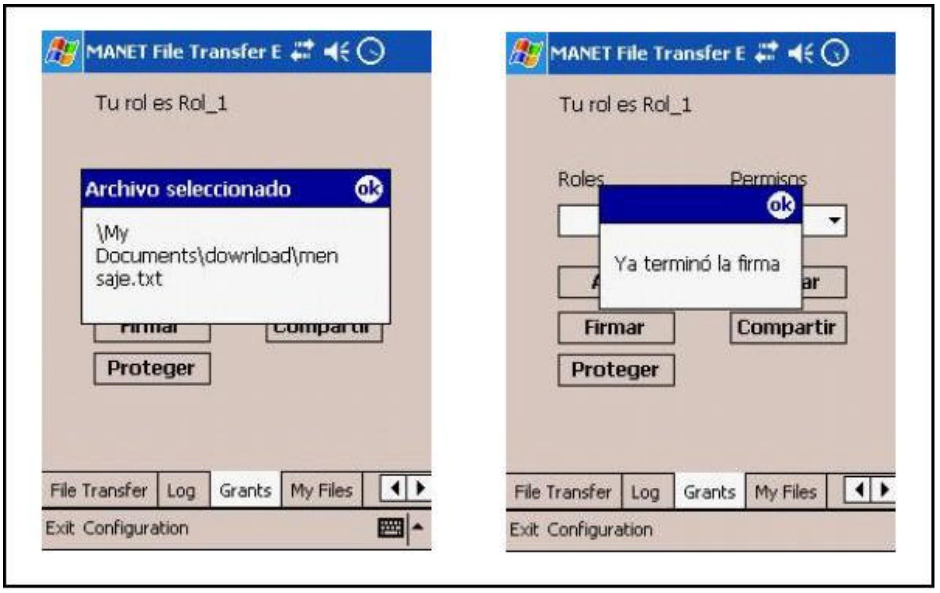

Figura 41: Recalculando el checksum de un archivo modificado

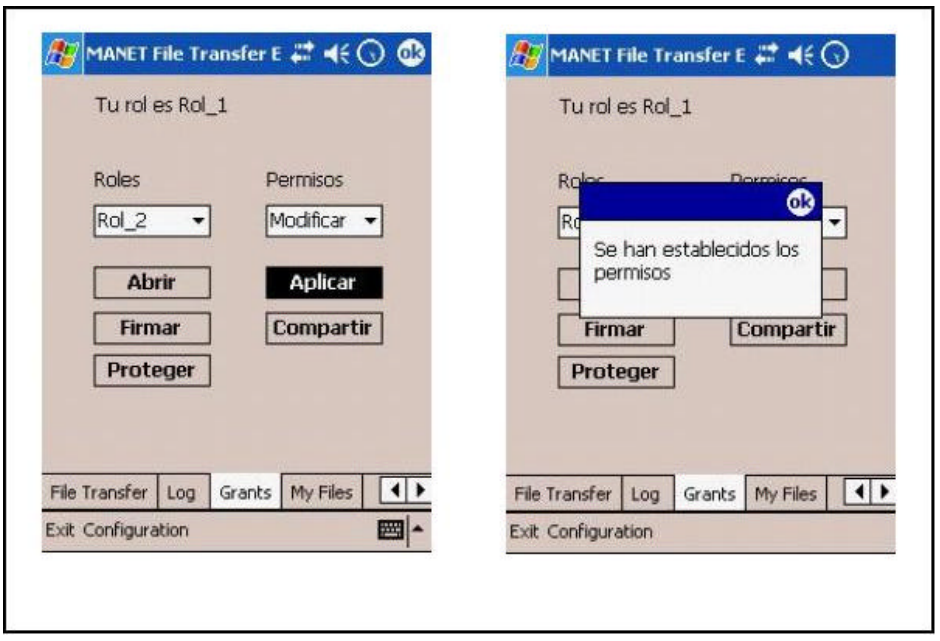

Figura 42: Reestableciendo permisos de un archivo protegido

Al proteger el archivo modificado, el archivo protegido también cambia.

Figura 43: Cambios producidos por la modificación de un archivo

Los usuarios también tienen la opción de sólo registrar el checksum de los archivos sin encriptar, en este caso al compartir un archivo los demás usuarios con permisos de lectura pueden verificar la integridad de los datos. Cabe mencionar que en este caso no hay contenido cifrado, pero de todos modos influyen los permisos de lectura (que en este caso serían permisos de verificación). Para usar esta funcionalidad basta con firmar un archivo (que no haya sido protegido) estableciendo permisos de lectura sobre él, en este caso la diferencia entre permisos de lectura y escritura radica sólo en la capacidad de refirmar el archivo (para el caso de los usuarios con permisos de escritura). Al firmar los archivos se genera un archivo con extensión *extSum* en vez de *extMrg*.

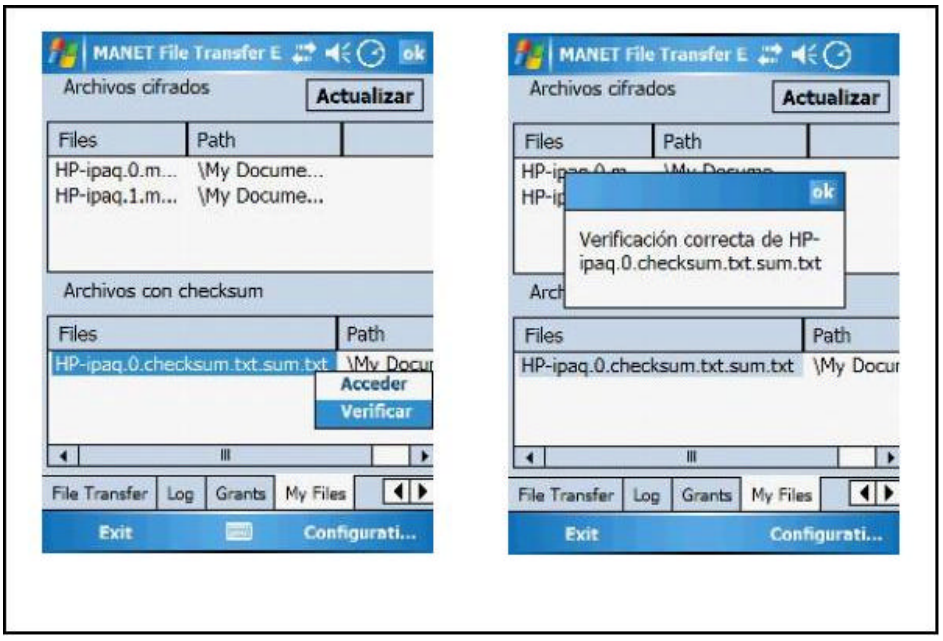

Figura 44: Verificación de un archivo (no cifrado) por medio de checksum

Al firmar los archivos sin cifrar, el usuario receptor (el que desea verificar la integridad de los datos) puede verificarlo directamente usando la opción “Verificar” del menú contextual o simplemente puede extraer su contenido para trabajar con el archivo original usando la opción “acceder”. En ambos casos se genera un archivo (quitando la extensión *extSum*) cuyo contenido es el mismo que el del archivo original. Al firmar los archivos sin cifrar el contenido del archivo original se puede leer viendo el archivo con la extensión *extSum*.

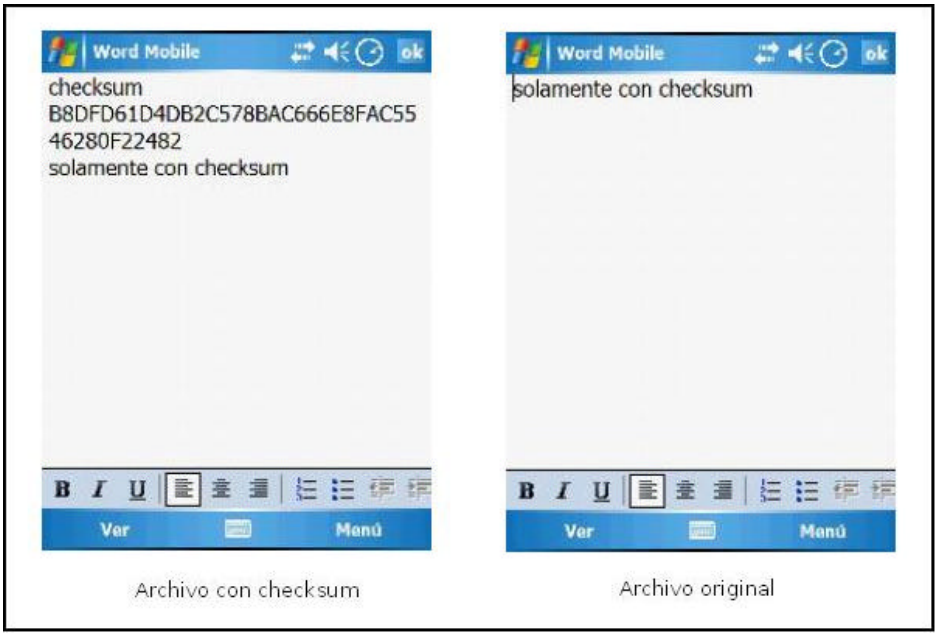

Figura 45: Archivo marcado con checksum (sin cifrar) y archivo original

7. Conclusiones y trabajo a futuro

En este trabajo de memoria se implementó un sistema genérico de seguridad para sistemas colaborativos móviles. Se logró establecer un sistema de control de acceso a archivos, gracias al esquema de permisos diseñado. Además, se logró establecer una forma de verificar la integridad de los archivos. Todo esto fue integrado en un sistema de seguridad que funciona sobre PDA's y Notebooks, con un desempeño adecuado para los usuarios, y según las capacidades de los dispositivos empleados.

Respecto de la programación, se consiguió desarrollar una interfaz de servicios que puede ser reutilizada por distintas aplicaciones colaborativas móviles. Esto se hace por medio de la clase *GrantsManager* (previa definición de los roles vía la utilidad *RolesAdmin*), puesto que los desarrolladores pueden aprender a usar esta clase y proteger cualquier tipo de archivo, desde planos de construcción hasta fichas médicas electrónicas.

El objetivo general de la memoria se cumplió a cabalidad. Sin embargo, no se abordó el aspecto de la autenticación (que el receptor de un documento pueda verificar la identidad del emisor) de documentos, debido a que esto requiere el uso de firmas digitales. Cabe mencionar que sí se abordó el aspecto de verificación de documentos, en el sentido de la integridad de la información que contiene.

El sistema de control de acceso conserva la privacidad de los archivos gracias a los permisos de lectura. Sin embargo, el control sobre la modificación de los archivos es un poco débil, puesto que realmente no se evita que usuarios no autorizados escriban sobre los archivos, además la diferencia entre permisos de sólo lectura y de escritura se hace sólo a través de la interfaz. Esto es algo que debiera ser mejorado en un futuro, usando por ejemplo los sistemas de permisos propios del sistema operativo (en el caso que sea aplicable), y estableciendo algún mecanismo de firmas digitales para que el control entre escritura y lectura sea más fuerte que sólo a través de la interfaz.

Respecto de la integridad de los datos, se logró establecer un esquema simple y conocido de verificación. En un futuro se podrían integrar nuevos algoritmos de hash criptográficos de acuerdo al concurso organizado por el NIST [Hash Project 2009].

El uso de identidades digitales (además de roles) permitiría a los usuarios del sistema, tener más posibilidades respecto de la protección de documentos, entre ellas: los usuarios podrían cifrar los documentos para que sólo un usuario pudiera accederlo, o se podrían realmente firmar (no sólo usar MAC) los documentos sensibles usando firmas digitales. Sin embargo, la implementación de estas características es difícil en sistemas colaborativos móviles, debido a que los esquemas se basan en una infraestructura PKI, la cual rompe la arquitectura de red de propia de estos sistemas (P2P y MANET).

Los tipos de permisos usados así como los contenidos protegidos podrían ser de granularidad más fina. Los permisos podrían permitir que un usuario agregara información y evitar que modificara la información previa de un documento. Del mismo modo, se podrían elegir proteger partes de un documento en vez de un documento completo. Sin embargo, esto sólo es aplicable a documentos que basan sus contenidos en alguna estructura, por ejemplo en base a XML como los documentos ODF (open document format).

Bibliografía

[Adams 2006] W. Adams, N. Davis, Tms: a trust management system for access control in dynamic collaborative environments, in: Proceedings of the 25th International Conference on Performance, Computing, and Communications, IEEE, Arizona, USA, April 2006.

[Aissi 2006] Aissi, S. Dabbous, N., and Prasad, A. R. Security for Mobile Networks and Platforms (Artech House Universal Personal Communications). Artech House Inc, 2006.

[Andriessen 2006] Andriessen, J. H. E., Vartiainen, M. Mobile virtual work: A new paradigm? Springer, 2006.

[Bethencourt 2007] Bethencourt, J., Sahai, A., and Waters, B. Ciphertext-Policy Attribute-Based Encryption. In Proceedings of the 2007 IEEE Symposium on Security and Privacy, May 2007.

[Brugnolli, 2005] Brugnoli, M.C., Davide, F. Slagter, R. The Future of Mobility and of Mobile Services. In P. Cunningham and M. Cunningham (eds.), Innovation and the Knowledge Economy: Issues, Applications, Case Studies, pp 1043-1055, IOS Press. 2005.

[Fenkam 2002] Fenkam, P., Dustdar, S., Kirda, E., Reif, G., and Gall, H. Towards an Access Control System for Mobile Peer-to-Peer Collaborative Environments. In Proceedings of the 11th IEEE international Workshops on Enabling Technologies: Nfrastructure For Collaborative Enterprises. WETICE. IEEE Computer Society, Washington, DC, 95-102. June 2002.

[Ferraiolo 1999] Ferraiolo, D. F., Barkley, J. F., and Kuhn, D. R. A role-based access control model and reference implementation within a corporate intranet. ACM Transactions on Information Systems Security. 2, 1, 34-64. Feb. 1999.

[Goyal 2006] Goyal, V., Pandey, O., Sahai, A., and Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Proceedings of the 13th ACM Conference on Computer and Communications Security ,Alexandria, Virginia, USA, November 2006.

[Keoh 2002] Keoh, S. L. and Lupu, E. Towards flexible credential verification in mobile ad-hoc networks. In Proceedings of the Second ACM international Workshop on Principles of Mobile Computing. POMC '02. ACM, New York, NY, 58-65. October 2002.

[Kim 2008] Kim, K. I., Ko, H. J., Choi, W. G., Lee, E. J., and Kim, U. M. A Collaborative Access Control Based on XACML in Pervasive Environments. In Proceedings of the 2008 international Conference on Convergence and Hybrid information Technology - Volume 00. ICHIT. IEEE Computer Society, Washington, DC, 7-13. August 28 - 29, 2008.

[Kirda 2002] Kirda, E.; Gall, H.; Fenkam, P.; Reif, G., MOTION: a peer-to-peer platform for mobile teamwork support, Computer Software and Applications Conference, COMPSAC 2002. Proceedings. 26th Annual International, vol., no., pp. 1115-1117, 2002.

[Memon 2007a] Memon, Q., Akhtar, S., and Aly, A. A. Role management in adhoc networks. In Proceedings of the 2007 Spring Simulaiton Multiconference - Volume 1. Spring Simulation Multiconference. Society for Computer Simulation International, San Diego, CA, 131-137. March 2007.

[Memon 2007b] Memon, Q.; Khoja, S., RBAC and XML security in adhoc networks, Signal Processing and Its Applications, 2007. ISSPA 2007. 9th International Symposium on , vol., no., pp.1-4, 12-15 Feb. 2007.

[Neyem, 2005] Neyem, A., Ochoa, S.F., Guerrero, L.A., Pino, J.A. Sharing Information Resources in Mobile Ad-hoc Networks. 11th Int. Workshop on Groupware (CRIWG 2005). Lecture Notes in Computer Science. Springer. Porto do Galinhas, Brazil. Sept. 2005.

[Palomar 2006] Palomar, E., Estevez-Tapiador, J. M., Hernandez-Castro, J. C., and Ribagorda, A. Certificate-based Access Control in Pure P2P Networks. In Proceedings of the Sixth IEEE international Conference on Peer-To-Peer Computing P2P. IEEE Computer Society, Washington, DC, 177-184, 2006.

[Palomar 2008] Palomar, E., Tapiador, J. M., Hernandez-Castro, J. C., and Ribagorda, A. 2008. Secure content access and replication in pure P2P networks. Comput. Commun. 31, 2, 266-279. Feb. 2008.

[Park 2003] Park, J. S. and Hwang, J. 2003. Role-based access control for collaborative enterprise in peer-to-peer computing environments. In Proceedings of the Eighth ACM Symposium on Access Control Models and Technologies. SACMAT '03. ACM, New York, NY, 93-99, June 2003.

[Park 2007] Park, J.S.; An, G.; Chandra, D., "Trusted P2P computing environments with role-based access control," Information Security, IET , vol.1, no.1, pp.27-35, March 2007.

[Peterson 1961] Peterson, W. W. y Brown, D. T. “Cyclic Codes for Error Detection” Proceedings of the IRE, January 1961.

[Puzar 2008] Puzar, M.; Plagemann, T.; Roudier, Y., "Security and privacy issues in middleware for emergency and rescue applications," Pervasive Computing Technologies for Healthcare, 2008. PervasiveHealth 2008. Second International Conference on , vol., no., pp.89-92, Jan. 30 2008-Feb. 1 2008.

[Rodriguez 2009] Rodríguez-Covili, J., Ochoa, S.F., Pino, J.A. “HLMP: High Level MANET Protocol”. Technical Report TR/DCC-2009-11, Departmento de Ciencias de la Computación, Universidad de Chile. Nov. 2009.

[Sandhu 2000] Ferraiolo, D. F., Sandhu, R., Gavrila, S., Kuhn, D. R., and Chandramouli, R. 2001. Proposed NIST standard for role-based access control. ACM Trans. Inf. Syst. Secur. 4, 3, 224-274. Aug. 2001.

[Shen 1994] Shen, H. Access Control for Collaborative Environments. Doctoral Thesis. UMI Order Number: UMI Order No. GAX95-13064., Purdue University. 1994.

[Tolone 2005] Tolone, W., Ahn, G., Pai, T., and Hong, S. Access control in collaborative systems. ACM Comput. Surv. 37, 1 March 2005.

[Traoré 2003] Traoré, I. and Khan, S. 2003. A protection scheme for collaborative environments. In Proceedings of the 2003 ACM Symposium on Applied Computing. SAC '03. ACM, New York, NY, 331-337. March 2003.

[Wang 2008] Li, M. and Wang, H. 2008. Protecting Information Sharing in Distributed Collaborative Environment. In Advanced Web and Network technologies, and Applications: Apweb 2008 international Workshops: Bidm, Iwhdm, and Deweb Shenyang, China, April 2008.

Sitios Web:

[AES 2009] Advanced Encryption Standard.

http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf. Visto por última vez en Noviembre, 2009.

[Bellare 2006] Mihir Bellare y Phil Rogaway, "Introduction to Cryptography, Lecture Notes", University of California San Diego, 2006. Disponible en versiones del curso para:

pregado: http://cseweb.ucsd.edu/users/mihir/cse107/classnotes.html.

postgrado: http://cseweb.ucsd.edu/users/mihir/cse207/classnotes.html. Visto por última vez en Noviembre, 2009.

[Block Ciphers 2009] Cifradores de bloque.

http://csrc.nist.gov/groups/ST/toolkit/block_ciphers.html. Visto por última vez en Noviembre, 2009.

[Cipher Modes 2009] Modos de operación.

http://msdn.microsoft.com/enus/library/system.security.cryptography.ciphermode.aspx. Visto por última vez en Noviembre, 2009.

[Criptography .NET 2009] Criptografía en .NET.

http://msdn.microsoft.com/es-es/library/system.security.cryptography.aspx

http://www.elguille.info/NET/library/System.Security.Cryptography.aspx. Visto por última vez en Noviembre, 2009.

[DES 2009] Data Encryption Standard.

http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf.

http://csrc.nist.gov/publications/nistpubs/800-67/SP800-67.pdf. Visto por última vez en Noviembre, 2009.

[Hash Project 2009] Concurso del NIST de hash criptográfico.

http://csrc.nist.gov/groups/ST/hash/index.html. Visto por última vez en Noviembre, 2009.

[RFC-2104 2009] RFC de HMAC.

http://tools.ietf.org/html/rfc2104. Visto por última vez en Noviembre, 2009.

[Secure Hashing 2009] Hash criptográfico.

http://csrc.nist.gov/groups/ST/toolkit/secure_hashing.html. Visto por última vez en Noviembre, 2009.

[Security in Windows Mobile 2009] Modelo de seguridad en Windows Mobile.

http://technet.microsoft.com/es-es/library/cc512651%28en-us%29.aspx. Visto por última vez en Noviembre, 2009.