TABLA DE CONTENIDO

ANEXO A. HERRAMIENTAS SOFTWARE EMPLEADAS EN EL PROTOTIPO Y CONFIGURACIONES

1. Configuración de Apache Tomcat para usar SSL dentro del Servidor de Aplicaciones JBoss

Las versiones más recientes de JBoss, desde la 3.2.2 hasta la serie 4 del mismo, incorporan como servidor Web a Tomcat 4.1 o superior, como es el caso de JBoss 4.0 que incorpora a Tomcat 5, para brindar soporte SSL con Tomcat es necesario tener en cuenta los siguientes aspectos descritos a continuación.

Existen dos configuraciones posibles: la más sencilla consiste en usar la herramienta keytool, distribuida con el kit de desarrollo de Java (JDK), para generar un certificado autofirmado del servidor.
Es necesario cambiar la configuración del fichero server.xml para activar un conector que permitirá acceder a Tomcat a través de https. Esta configuración tiene una limitación importante: no es posible aprovechar el mecanismo de autoridades de certificación X509.
La segunda alternativa es un poco más compleja, porque requiere crear una autoridad de certificación (CA) con la cual se pueden generar certificados para Tomcat y para los clientes que acceden a Tomcat. Como la herramienta keytool no permite realizar las operaciones necesarias, se requiere otra herramienta para crear la autoridad de certificación y firmar solicitudes de certificado. En esta descripción se emplea OpenSSL [OpenSSL].

1.1. Software Necesario

La herramienta keytool se encuentra en el directorio bin de la instalación del j2sdk (kit de desarrollo estándar de Java).
Si se emplea una versión anterior a la 1.4 del kit de desarrollo de Java, será necesario instalar la extensión de Java JSSE, que se puede obtener en: http://java.sun.com/products/jsse/. Los ficheros jar se deben copiar al directorio de instalación de extensiones:
%JAVA_HOME%\jre\lib\ext.

OpenSSL se puede obtener del sitio web http://www.openssl.org. Puede ser compilada en entornos Linux o Windows. Para este caso descargamos una distribución de instalación para Win32: El software que incluye la distribución OpenSSL para la gestión de autoridades de certificación requiere la instalación previa de Perl. Para sistemas Windows, existe una distribución gratuita llamada ActivePerl [ActivePerl].

1.2. Configuración de Tomcat

Tanto si se emplea un certificado autofirmado como si se crea una autoridad de certificación, los cambios en el fichero server.xml de Tomcat son los mismos. Hay que buscar el siguiente elemento Connector que aparece comentado:

```
<Connector className="org.apache.coyote.tomcat4.CoyoteConnector"
    port="8080" redirectPort="8443"
    minProcessors="5" maxProcessors="100"
    enableLookups="true" acceptCount="10" debug="0"
    connectionTimeout="20000" useURIValidationHack="false"/>

<Connector className="org.apache.coyote.tomcat4.CoyoteConnector"
    port="8443" scheme="https" secure="true" >
  <Factory className =
"org.apache.coyote.tomcat4.CoyoteServerSocketFactory"
    keystoreFile="/home/neal/.keystore"
    keystorePass="changeit"
    clientAuth="false"
    protocol="TLS"/>
</Connector>
```

Para activar el uso de SSL, hay que dejar sin comentarios el elemento Connector. Los aspectos más destacables de la configuración que se activa son:
• Se habilita el acceso al servidor de forma segura a través del puerto 8443. Por tanto, las URL que se tienen que emplear son de la forma: https://servidor:8443/aplicacionweb/recurso.
• No se solicita certificado al cliente durante el establecimiento de la conexión SSL. Por tanto, no es necesario cambiar nada en el navegador Web que usemos.

1.3. Generación del certificado autofirmado para el servidor

Para completar la instalación sencilla de Tomcat en modo seguro es necesario generar un certificado autofirmado y almacenarlo en un repositorio de claves (keystore) al alcance de Tomcat. Los certificados se almacenan en los repositorios asociándoles un alias. En el caso de Tomcat, la configuración por defecto busca el certificado con alias tomcat. Los keystores se protegen con una clave. Por defecto, dicha clave es changeit, y en este tutorial no se modifica.
El uso de la herramienta keytool para generar un keystore con un certificado autofirmado es este:

JAVA_HOME\bin\keytool –genkey –alias tomcat –keyalg RSA

Donde JAVA_HOME obedece a la ubicación de instalación del JDK en el P.C de trabajo

Al introducir los datos relativos al certificado, es importante emplear el nombre del servidor como CN (common name). Otros datos solicitados son: unidad organizativa, organización, provincia, ciudad y país.
La ejecución del comando anterior genera un fichero llamado .keystore. En los sistemas Windows, el fichero va a parar a un directorio diferente, de acuerdo con la versión:

C:\Documents and Settings\user en sistemas Windows XP o 2000
C:\Winnt\Profiles\user en sistemas multi-usuario en Windows NT

En adelante, se considera que tanto la herramienta keystore como el propio Tomcat emplean el fichero .keystore correspondiente al usuario que está ejecutando dichos programas.

1.4. Generación de una autoridad de certificación

Se considera que no se han realizado las operaciones referidas en el apartado anterior.
El primer paso para establecer la segunda configuración de Tomcat en modo seguro es la generación de un certificado autofirmado de una autoridad de certificación.
En la distribución OpenSSL existe un directorio apps que contiene una herramienta escrita en Perl, llamada CA.pl. Esta herramienta funciona basándose en la configuración del fichero openssl.cnf que se encuentra en el mismo directorio. Para su correcto funcionamiento, debe estar definida la variable de entorno OPENSSL_CONF cuyo valor debe ser la localización del fichero openssl.cnf. Únicamente puede resultar conveniente modificar una línea del fichero de configuración de OpenSSL, que es la siguiente:

 # and for everything including object signing:
ncCertType= client, email, objsign

nsCertType aparece, por defecto, comentado con un signo #. Al descomentarlo permitimos que los certificados firmados por la CA puedan ser empleados para clientes SSL, para correo SMIME y para firma de código.
El primer paso para crear la autoridad de certificación es ejecutar la herramienta con el siguiente
parámetro:

perl CA.pl -newca

El comando anterior genera un subdirectorio dentro de apps, llamado demoCA. Los ficheros y carpetas más importantes de demoCA son:
• cacert.pem: contiene el certificado autofirmado de la CA
• index.txt: contiene un resumen de los certificados que se han firmado con esta CA
• serial: contiene el número de serie que se asignará al siguiente certificado firmado con esta CA
• carpeta private: contiene el fichero cakey.pem, que es la clave privada de la CA,

protegida por contraseña
• carpeta newcerts: contiene ficheros XX.pem con los certificados firmados por la CA (XX = número de serie)
Una vez generado el certificado de la CA, es necesario añadirlo a un almacén de certificados de autoridades de certificación del JRE que use Tomcat, cuya localización es: %JAVA_HOME%\jre\lib\security\cacerts. Para ello se emplea keytool de la siguiente forma:

JAVA_HOME\bin\keytool –import –trustcacerts –alias nombreCA –file cacert.pem -keystore JAVA_HOME\jre\lib\security\cacerts

1.5. Generación de un certificado para Tomcat

Para generar un certificado firmado por la CA para Tomcat es preciso dar los siguientes pasos:
Generar el par de claves para Tomcat, usando keytool. Es importante recordar que el CN de Tomcat debe ser el nombre DNS del servidor en el que se ejecute.

JAVA_HOME\bin\keytool –genkey –alias tomcat –keyalg RSA

Generar una solicitud de certificado partiendo del par de claves generado en el paso previo:

JAVA_HOME\bin\keytool –certreq –alias tomcat –file newreq.pem

Copiar el fichero newreq.pem al directorio apps de OpenSSL. Firmar la solicitud de certificado con la CA:

Perl CA.pl -sign

El comando anterior genera el fichero newcert.pem en el directorio apps de OpenSSL. Es necesario editar dicho fichero y eliminar las líneas iniciales hasta la línea que contiene BEGIN CERTIFICATE.

El bloque desde BEGIN hasta END se puede guardar en un nuevo fichero llamado tomcat.pem.

El último paso de este bloque de operaciones es instalar el certificado firmado por la CA en el keystore:

JAVA_HOME\bin\keytool –import –alias tomcat –file tomcat.pem

Con los pasos dados hasta ahora podemos probar el funcionamiento de Tomcat en modo seguro. El resultado debe ser que un navegador como Internet Explorer nos avisa de que el servidor Tomcat presenta un certificado firmado por una CA desconocida. La mayoría de los navegadores permiten, aún así, continuar la conexión y ver las páginas solicitadas. Para que el navegador reconozca la CA y, por tanto, pueda verificar la validez del certificado de

Tomcat, es necesario instalar el certificado de la CA en el propio navegador. En el caso de Internet Explorer, esto se puede conseguir fácilmente accediendo a las Opciones de Internet en el menú Herramientas, y seleccionando la solapa Contenido.
En la parte central del cuadro de diálogo aparece un botón "Certificados", con el cual podremos ver los certificados instalados en el navegador. Para instalar la CA debemos seleccionar el grupo de certificados denominado "Entidades emisoras raíz de confianza", y pulsar el botón Importar. Seleccionaremos el fichero cacert.pem.

1.6 Certificado de cliente

Para activar la solicitud de certificado al navegador, es necesario realizar un pequeño cambio en el fichero server.xml. El atributo clientAuth del elemento Factory que aparece dentro del Connector activado en el apartado "Configuración de Tomcat" debe tomar el valor "true":

```
<Connector className="org.apache.coyote.tomcat4.CoyoteConnector"
    port="8443" scheme="https" secure="true" >
  <Factory className =
"org.apache.coyote.tomcat4.CoyoteServerSocketFactory"
    keystoreFile="/home/neal/.keystore"
    keystorePass="changeit"
    clientAuth="true"
    protocol="TLS"/>
</Connector>
```

Este cambio obliga a detener y reiniciar Tomcat.
Por otra parte, es necesario generar un certificado para el cliente, que esté firmado por la misma CA que firmó el certificado del Tomcat. Para ello, se deben seguir los siguientes pasos:
Generar una solicitud de certificado (la configuración por defecto está en el fichero openssl.cnf):

Perl CA.pl -newreq

Firmar la solicitud de certificado:

Perl CA.pl -sign

Exportar el fichero con el certificado y la clave privada a un fichero con formato PKCS12 que puede ser importado por un navegador, como Internet Explorer. El último argumento permite especificar el nombre con el que se visualizará el certificado en la lista de certificados instalados en el del navegador.

Perl CA.pl –pkcs12 "Nombre para el Certificado"

Al generar el fichero pkcs12 se solicita un password para proteger la clave privada que contiene.
Ese password será solicitado por el navegador al importar el certificado, siguiendo el procedimiento de importación descrito para el certificado de la CA, con la salvedad de que, en este caso, se selecciona el grupo de certificados llamado "Personal".
Los pasos anteriores nos permiten que el navegador pueda conectarse con Tomcat a través de la conexión segura. Sin un certificado de cliente instalado en el navegador, sería imposible establecer la conexión.

2. OpenSSL

El Proyecto OpenSSL es un esfuerzo colaborativo para el desarrollo de un Kit de herramientas robusto, de tipo comercial, perfectamente terminado y de código abierto, para la implementación de los protocolos *SSL* (Secure Socket Layer v2/v3) y *TLS* (Transport Layer Security v1), así como, un gran librería criptográfica de propósito general.

El Proyecto es gestionado a nivel mundial por una comunidad de voluntarios que utilizan Internet para comunicarse, planear y desarrollar el kit de herramientas OpenSSL, así como, la documentación relacionada.

OpenSSL está basada en la excelente librería SSLeay desarrollada por Eric A. Young y Tim J. Hudson. El Kit de herramientas de OpenSSL esta licenciado bajo un estilo de licencia Apache, lo cual significa básicamente que se puede descargar y usar para propósitos comerciales y no comerciales sujeto a unas simples condiciones de la licencia. http://www.openssl.org/docs/apps/openssl.html

El programa OpenSSL es una herramienta de comandos utilizada desde el shell, para el uso de varias funciones de criptografía de la librería Cripto de OpenSSL.

Puede ser usado para:

- Creación de llaves RSA, DH y DSA
- Creación de Certificados Digitales X.509, CSRs y CRLs.
- Encriptación y des-encriptación.
- Calculo de mesagge digest
- Clientes SSL/TLS y prueba de servidores.
- Gestión de mensajes MIME encriptados y correos cifrados.

El programa OpenSSL provee una rica variedad de comandos, cada uno de los cuales a menudo cuenta con varias opciones y argumentos
Básicamente existen tres tipos de listas de pseudo-comandos:

- Comandos Estándar
- Comandos de Mensajes asimilados (digest)
- Comandos de Cifrado

2.1. Instalación de OpenSSL

La instalación de OpenSSL es muy sencilla, basta descargar el paquete y compilarlo. La última versión es la 0.9.7, publicada el 11 de abril del 2005.
La secuencia de comandos para compilarlo e instalarlo siguiendo la configuración que se utilizó en el desarrollo del prototipo es:

1. Descargar el paquete en su última actualización de http://www.openssl.org/source/
2. Descomprimirlo en la carpeta por defecto c://openssl-0.9.7
3. ejecutar install
4. Definir la variable de entorno OPENSSL_CONF con los valores C:\openssl-0.9.7\apps\openssl.cnf

El directorio de instalación por default es c:// En este directorio encontraremos los binarios base del programa en apps/ y varios scripts para ayudarnos a manejar una autoridad certificadora. El archivo de configuración de OpenSSL está en c://openssl-0.9.7/apps/openssl.cnf; vale la pena revisarlo para asegurarnos que funcione como lo deseamos.

2.2. Creación y auto-firma de nuestro certificado

El primer paso que se debe dar es crear un certificado firmado por nosotros mismos. Será con este certificado que en el futuro nosotros podremos firmar los certificados que nos sean solicitados. Utilizaremos el script CA.pl localizado en c://openssl-0.9.7/apps/ para poder concentrarnos más en el procedimiento que en los detalles sintácticos.

El primer paso es crear la estructura necesaria para levantar sobre esta nuestra autoridad certificadora:

Los siguientes comandos se deben ejecutar desde el Prom. Ubicado ene. Directorio raíz de OpenSSL c://openssl-0.9.7/apps//

CA.pl -newca

CA certificate filename (or enter to create)

Dejamos en blanco

Making CA certificate ...

Using configuration from c://openssl-0.9.7/apps/openssl.cnf

Generating a 1024 bit RSA private key

.....++++++

.....++++++

writing new private key to './demoCA/private/cakey.pem'

Enter PEM pass phrase: *esta es mi frase de paso y nadie la debe conocer*

Verifying password - Enter PEM pass phrase: *esta es mi frase de paso y nadie la debe conocer*

You are about to be asked to enter information that will be incorporated into your certificate request. What you are about to enter is what is called a Distinguished Name or a DN. There are quite a few fields but you can leave some blank

For some fields there will be a default value,

If you enter '.', the field will be left blank.

Country Name (2 letter code) [AU]:*CO*

State or Province Name (full name) [Some-State]:*Cauca*

Locality Name (eg, city) []:*Popayán*

Organization Name (eg, company) [Internet Widgits Pty Ltd]:*UNICAUCA*

Organizational Unit Name (eg, section) []:*FIET*

Common Name (eg, YOUR name) []:*Departamento de Telemática*

Email Address []:anonimo*@unicauca.edu.co*

La contraseña que dimos aquí arriba es de vital importancia - la utilizaremos siempre que queramos firmar un certificado. Sin esta, OpenSSL se quejará y abortará la operación.

Quedamos ahora con un directorio demoCA en el cual trabajaremos con nuestra autoridad certificadora. Ahora generamos una petición de certificado:

CA.pl -newreq

Using configuration from / c://openssl-0.9.7/apps/openssl.cnf

Generating a 1024 bit RSA private key

.++++++

.....++++++

writing new private key to 'newreq.pem'

Enter PEM pass phrase: *esta es otra frase de paso que solo yo deboconocer*

Verifying password - Enter PEM pass phrase: *esta es otra contraseña que solo yo debo conocer*

You are about to be asked to enter information that will be incorporated into your certificate request.

What you are about to enter is what is called a Distinguished Name or a DN.

There are quite a few fields but you can leave some blank

For some fields there will be a default value,

If you enter '.', the field will be left blank.

Country Name (2 letter code) [AU]:*CO*

State or Province Name (full name) [Some-State]:*Cauca*

Locality Name (eg, city) []:*Popayán*

Organization Name (eg, company) [Internet Widgits Pty Ltd]:*UNICAUCA*

Organizational Unit Name (eg, section) []:*FIET*

Common Name (eg, YOUR name) []:*Departamento de Telemática*

Email Address []:anonimo*@unicauca.edu.co*

Please enter the following 'extra' attributes

to be sent with your certificate request

A challenge password []:*secreta*

An optional company name []:*UNICAUCA*

Request (and private key) is in newreq.pem

Queda entonces hecha la petición de certificado, así como nuestra llave privada, en el archivo newreq.pem - El siguiente paso es firmarlo:

OpenSSL c://openssl-0.9.7/apps/CA.pl -sign

Using configuration from OpenSSL c://openssl-0.9.7/apps/openssl.cnf

Enter PEM pass phrase: *esata es la frase de paso que solo yo debo conocer*

Check that the request matches the signature

Signature ok

The Subjects Distinguished Name is as follows

countryName :PRINTABLE:'CO'

stateOrProvinceName :PRINTABLE:'Cauca'

localityName :PRINTABLE:'Popayan'

organizationName :PRINTABLE:'UNICAUCA'

organizationalUnitName:PRINTABLE:'FIET'

commonName :PRINTABLE:'Departamento de Telemática'

emailAddress :IA5STRING:'anonimo@unicauca.edu.co'

Certificate is to be certified until May 21 19:42:25 2005 GMT (365 days)

Sign the certificate? [y/n]:*y*

1 out of 1 certificate requests certified, commit? [y/n]*y*

Write out database with 1 new entries

Data Base Updated

Signed certificate is in newcert.pem

Por último, toca poner nuestro certificado firmado por nosotros mismos en donde OpenSSL espera encontrarlo - en el directorio especificado en c://openssl-0.9.7/apps /openssl.cnf.

$ *su root*

Password:

cp newcert.pem c://openssl-0.9.7/apps *demoCA//private/*

cp demoCA/private/cakey.pem c://openssl-0.9.7/apps*/demoCA/private/*

3. Muy Buena Privacidad (PGP)

El PGP (Pretty Good Privacy ó Muy Buena Privacidad) es un sistema de encriptación por llave pública escrito por Philip Zimmermann, y sirve para que nadie salvo uno mismo y el destinatario o destinatarios a los que vaya dirigido el mensaje puedan leerlo al ir los mensajes codificados, también puede usarse para comprobar la autenticidad del mensaje asegurándonos que lo ha escrito el remitente en realidad, realmente es muy bueno y es prácticamente indescifrable, esto mismo le ha llevado al autor del mismo Philip Zimmermann a tener bastantes quebraderos de cabeza con la ley en Estados Unidos, afortunadamente su caso ya se ha cerrado.

El funcionamiento es muy sencillo, cada usuario tiene dos llaves una pública y otra privada, la pública es la que distribuye a la gente y sirve para que ellos puedan enviarle un mensaje codificado que solo él mediante su llave privada podrá descifrar, también puede servir para firmar un mensaje poniendo una parte de su llave privada (irreconocible claro) en una firma, esto es como un certificado de autenticidad, ya que al recibir el mensaje el PGP comprueba la firma y texto y lo compara con la llave pública que tenemos del remitente dando un error si se ha cambiado algo en el texto o la firma no corresponde a la persona que nos envía el mensaje.

Sirve también para enviar ficheros a través de correo electrónico codificados en formato ascii y mucho mejor que otros sistemas como el unicode ya que el PGP usa antes de codificar una compresión zip al documento o programa que va a codificar.

Antes de llegar un mensaje a su destino, este pasa por muchos ordenadores, recuerde que cientos de personas pueden leer o escanear en busca de palabras clave su correo en Internet, es más, esto se hace, a nadie le gusta que su correo no sea privado, de hecho cuando usted escribe una carta postal a un amigo por que cierra el sobre para protegerlo de su lectura en vez de dejarlo abierto o enviar una postal, La intimidad del correo personal tanto postal como electrónico esta amparada por la ley y la constitución de la mayoría de los países.

3.1. El GNU Privacy Guard

GnuPG [GPG] es un reemplazo completo y libre para PGP. Debido a que no utiliza el algoritmo patentado IDEA , puede ser utilizado sin restricciones.

La versión 1.0.0 fue publicada el 7 de septiembre de 1999. La versión estable actual es 1.4.1 .

GnuPG es Software Libre . Puede ser utilizado, modificado y distribuido libremente bajo los términos de la GNU General Public License (Licencia Pública General de GNU).

PGP, en el cual se basa OpenPGP, es el que fue desarrollado originalmente por Philip Zimmermann a comienzos de los 90 y que se explicó en el anterior párrafo.

3.2. Características de GPG

GnuPG en sí mismo es una utilidad de línea de comandos sin ninguna característica gráfica. Se trata del motor de cifrado, en sí mismo, que puede ser utilizado directamente desde la línea de comandos, desde programas de shell o por otros programas. Por lo tanto GnuPG puede ser considerado como un motor para otras aplicaciones.
De todos modos, incluso si se utiliza desde la línea de comandos, proporciona toda la funcionalidad necesaria, lo que incluye un sistema interactivo de menús. El conjunto de órdenes de esta herramienta siempre será un superconjunto del que proporcione cualquier interfaz de usuario.
Dentro de las características más importantes podemos encontrar:

- Reemplazo completo de PGP.
- No utiliza algoritmos patentados.
- Con licencia GPL, escrito desde cero.
- Puede utilizarse como filtro.
- Implementación completa de OpenPGP (vea RFC 2440 en RFC Editor).
- Funcionalidad mejorada con respecto a PGP y mejoras de seguridad con respecto a PGP 2
- Descifra y verifica mensajes de PGP 5, 6 y 7.
- Soporta ElGamal, DSA, RSA, AES, 3DES, Blowfish, Twofish, CAST5, MD5, SHA-1, RIPE-MD-160 y TIGER.
- Facilidad de implementación de nuevos algoritmos utilizando módulos.
- Fuerza que el identificador de usuario (User ID) esté en un formato estándar.
- Soporta fechas de caducidad de claves y firmas.
- Sistema de ayuda en línea.

3.3. GnuPG para Windows y su Instalación

- GnuPG puede ser obtenido de http://www.gnupg.org/(es)/download/index.html para Windows, se debe bajar el binario de MS-Windows. En este momento (12/16/2003) la versión disponible es la gnupg-w32cli-1.4.1.zip (1370 KB).

- Posteriormente se debe crear un directorio en C:\ que se llame gnupg y descomprimirse allí el contenido del archivo zip.
- Se debe abrir un terminal de MS-DOS y tratar de correr pgp ejecutando el comando gpg. Si no funciona, entonces deberás decirle al sistema que incluya C:\gnupg a su ruta de ejecutables. Esto se logra editando las variables de entorno del sistema. Windows XP/NT/2000:
  Inicio>Panel de Control>Systema>Opciones Avanzadas>Variables de Entorno
  En el cuadro que dice Variables de Usuario deberás modificar PATH. Selecciona PATH y da click en Modificar, luego en el cuadro de diálogo edita la linea que dice Valor de Variable para agregar C:\gnupg. Si ya tiene otros argumentos esta variable, deberás agregarlo al final de la línea todo seguido y sin espacios usando un punto y coma entes de C:\gnupg.
  Si no existe la variable PATH puedes crearla dando click en Nueva y en Nombre de Variable ingresa PATH y en Valor de Variable C:\gnupg.

ANEXO B. DISPOSITIVO DE LECTO/ESCRITURA DE RADIO FRECUENCIA

1. Kit de Identificación por Radio Frecuencia (RFID)

El kit de RFID [Kit RFID] básicamente está compuesto por dos elementos; uno es el lector de radio frecuencia y el otro son los tag de RFID, que son las tarjetas de almacenamiento de la información.

1.2. Lector de Radio Frecuencia

El lector de radio frecuencia empleado en el desarrollo del prototipo de validación fue el lector de RFID de marca Texas Instruments y de referencia Mullion S6420 [Manual]. Éste dispositivo cumple con el estándar **ISO 15693** [ISO 15693]**,** el cual es un estándar ISO para "Tarjetas de Vecindad" (*Vicinity Cards*), que opera en la frecuencia 13.56MHz y ofrece una distancia máxima de lectura de entre 1 y 1,5 metros.
En la figura 1 se puede observar el lector de RFID empleado en el desarrollo de la aplicación.

Figura 1. Lector de RFID Mullion S6420 de Texas Instruments

1.3. Tag's de RFID

De igual forma, hacen parte de éste Kit de RFID los Tag's, los cuales están compuestos por una memoria física orientada a Bytes y organizada en bloques de tamaño fijo.
El tamaño de la memoria es de 64 bloques de 32 bits cada uno, obteniendo una capacidad total de uso de 2 KBits de memoria para almacenamiento de datos.
Posee cinco bloques adicionales de 32 bits en donde se almacena información exclusiva del tag, como por ejemplo, el identificador único del tag, el UID, entre otra, la información almacenada en éstos bloques adicionales únicamente puede ser leída y nunca puede ser modificada.

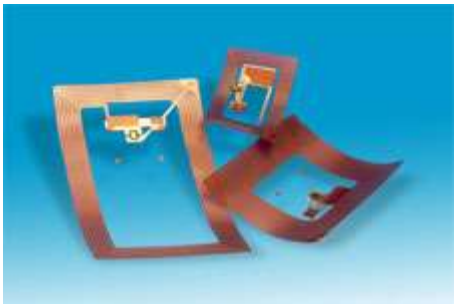

Figura 2. Tag de RFID empleado el en prototipo

1.4. Instalación en Windows

Para la instalación del kit de RFID en los sistemas operativos Windows 2000 o XP es necesario seguir los siguientes pasos y recomendaciones:

1. Conectar el lector de RFID al puesto serial del P.C.
2. Descargar la API de SUN para el manejo de los puertos en aplicaciones desarrolladas con java, commapi.jar, disponible en:
   http://java.sun.com/products/javacomm/downloads/index.html
3. Descomprimir el archivo en un directorio cualquiera, se recomienda en el directorio raíz c://.
4. Copiar el archivo win32com.dll en el directotio bin del JDK <JDK>\bin.
5. Copiar el archivo comm,jar en el directorio lib del JDK <JDK>\bin
6. Copiar javax.comm.properties en el directorio lib del JDK <JDK>\lib
   El archivo javax.comm.properties debe ser instalado en el directorio mencionado, de lo contrario, el sistema no reconocerá ningún puerto.

REFERENCIAS

[ActivePerl], Archivo de instalación que se puede obtener en el sitio: http://www.activestate.com/Products/ActivePerl.

[GPG]. Información del proyecto y paquetes disponibles en la dirección http://www.gnupg.org/(es)/

[ISO 15693], Información de la Recomendación para la tecnología de RFID ISO 15693, http://rfidusa.com/superstore/product_info.php?products_id=133

[Kit RFID], Información acerca del lector de RFID y de los Tag`s en la página de la empresa Texas Instruments. http://www.ti.com/tiris/docs/products/transponders/RI-I11-112A.shtml

[Manual], Manual de referencia del lector de RFID Mullion S6420, disponible en archivo pdf, ISO_15693_RFID.pdf

[OpenSSL]. Toda la información sobre el proyecto OpenSSL, http://www.openssl.org

[Tomcat]. Paquete de instalación Apache Tomcat disponible en la dirección http://jakarta.apache.org/tomcat/index.html