CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

ANEXO B
LA SEGURIDAD EN EL ESCENARIO MOVIL

1. CIFRADO SIMÉTRICO

Los mecanismos de seguridad en el escenario móvil han tenido una evolución continua y especialmente importante en los últimos años. Desde que entró al mercado la telefonía celular digital GSM (Global System for Mobile Communications) a inicios de los 90s se ha puesto especial atención en los mecanismos para garantizar la seguridad de la comunicación de voz como la de datos (la que usan las aplicaciones) y con el ánimo de garantizar dicha seguridad se han creado varios estándares, el estándar GSM 02.48[1] es el mas conocido (ver anexo A), y también se han tratado de migrar los mecanismos de seguridad de las redes fijas a las redes móviles. En la actualidad los mecanismos de seguridad basados en cifrado son los mas efectivos y comienzan a ser utilizados ampliamente en las redes móviles gracias a tecnologías como Java Card y SATSA (Security And Trust Services API).

El cifrado simétrico [2] es aquel que emplea la misma clave tanto para cifrar como para descifrar los datos. Presentan el inconveniente de que para ser empleados en comunicaciones la clave debe estar tanto en el emisor como en el receptor, lo cual lleva a preguntarse cómo transmitir la clave de forma segura. Los Algoritmos de cifrado simétrico mas utilizados son DES (Data Encryption Standard), Triple DES y AES (Advanced Encryption Standard). El cifrado simétrico tiene la ventaja de ser más rápido y eficiente que el cifrado asimétrico, pero el asimétrico es mas seguro.

Usualmente la calidad del cifrado se mide por la cantidad de esfuerzo que se necesita para averiguar las claves. En la tabla 1-1 se mencionan los Algoritmos de cifrado simétrico mas utilizados con la longitud correspondiente de sus claves.

Cifrado	**Longitud de Clave**
RC2	128 bits
RC4	128 bits
DES	64 bits
Triple DES (2 Keys)	128 bits
Triple DES (3 Keys)	192 bits
AES	128, 192 ó 256 bits

Tabla 1-1. Tipos de cifrado simétrico

1.1. Cifrado DES (Data Encryption Standard)

El cifrado DES fue hasta hace unos años el estándar mas utilizado para cifrado simétrico. Se utiliza una clave de 64 bits, lo que significa que utilizando el tipo de ataque más simple, fuerza bruta, que simplemente trata de ir probando una a una todas las claves se tendría 2^{64} posibilidades. ¿Cuánto tiempo llevaría probarlas todas si, por ejemplo, se dispusiera de un computador capaz de hacer mil millones de operaciones por segundo? Se llevaría más de 584 años. El problema de DES es que existen otros mecanismos distintos al de fuerza bruta los cuales pueden averiguar la clave en mucho menos tiempo.

1.2. Cifrado 3DES

El cifrado 3DES o Triple DES aparece como una evolución del DES. El cifrado 3DES consiste en cifrar una información con una clave DES, luego esa información se descifra con otra clave DES y finalmente la anterior información se vuelve a cifrar con otra clave. Por eso una clave Triple DES es generalmente la unión de tres claves DES. Utilizando el ataque de fuerza bruta y una clave de 192 bits, se tendrían 2^{192} posibilidades. ¿Cuánto tiempo llevaría probarlas todas si, por ejemplo, se dispusiera de un computador capaz de hacer mil millones de operaciones por segundo? Se llevaría más de 199^{39} años.

2. CIFRADO ASIMÉTRICO

El cifrado Asimétrico [3] se basa en la infraestructura de clave pública PKI, cuyos orígenes se remontan al artículo seminal de Diffie y Hellman en 1976, donde se explica la idea revolucionaria de servirse para las operaciones criptográficas de una pareja de claves, una pública, conocida por todos, y otra privada, sólo conocida por el usuario a quien le es asignada.

Un mensaje puede ser cifrado por cualquier persona usando la clave Pública, ya que es públicamente conocida, aunque sólo el poseedor de la clave privada podrá descifrarlo. Recíprocamente, un mensaje cifrado con la clave privada sólo puede ser cifrado por su poseedor, mientras que puede ser descifrado por cualquiera que conozca la clave pública.

Estas propiedades de que goza la criptografía asimétrica o también llamada de clave pública, la convierten en candidata ideal para prestar servicios como la **autenticación de usuarios** (para asegurarse de la identidad de un usuario, bien como emisor de documentos o para garantizar el acceso a servicios distribuidos en red, ya que sólo él puede conocer su clave privada, evitando así la suplantación), el **no repudio** (para impedir que una vez emitido un documento el emisor se retracte o niegue haberlo enviado), la **integridad de la información** (para prevenir la modificación deliberada o accidental de los datos, durante su transporte, almacenamiento o manipulación), y el acuerdo de claves secretas para garantizar la **confidencialidad de la información intercambiada**.

2.1. Pares: clave pública y clave privada

Los algoritmos de llave pública o algoritmos asimétricos, han demostrado su interés para ser empleados en Internet. Su novedad fundamental con respecto a la criptografía simétrica es que las claves no son únicas, sino que forman pares. Se basan en general en plantear al atacante problemas matemáticos difíciles de resolver. El más popular por su sencillez es RSA, otros algoritmos son los creados por ElGamal y Rabin. Los algoritmos asimétricos emplean generalmente longitudes de clave mucho mayores que los simétricos. Por ejemplo, mientras que para algoritmos simétricos se considera segura una clave de 128 bits, para algoritmos asimétricos se recomiendan claves de al menos 1024 bits. Además, la complejidad de cálculo que implican estos últimos los hace considerablemente más lentos. Los algoritmos asimétricos poseen dos claves diferentes, denominadas "Clave Pública" y "Clave Privada". Una de ellas se emplea para codificar, mientras que la otra se usa para decodificar, dependiendo de la aplicación que le demos al algoritmo, la clave pública será la de cifrado o viceversa.

Es importante mencionar que los datos cifrados con una clave no pueden ser descifrados con la misma, sino con su pareja.

Cifrado	**Longitud de Claves**
RSA	512, 736, 768, 896, 1024, 1280, 1536, 1984 ó 2048 bits
DSA	512, 768 ó 1024 bits
ECDSA (Elliptic Curve Digital Signature Algorithm)	128 ó 192 bits

Tabla 2-1. Tipos de cifrado Asimétrico

En el escenario móvil desde hace varios años las tarjetas inteligentes han desempeñado un papel importante debido a que su hardware es capaz de generar las claves Asimétricas, quedando almacenada en su interior la clave privada. Esta característica ofrece la posibilidad de trasladar las claves de forma segura lo que permite interactuar con otros sistemas PKI.

2.2. Cifrado RSA

El cifrado RSA llamado así en honor a sus inventores Rivest-Shamir-Adleman, es actualmente el algoritmo mas utilizado a nivel mundial debido a su relativamente sencilla implementación y a su gran seguridad. La forma más fácil de generar un par de claves RSA es por medio de tres enteros no negativos llamados: módulo, exponente público y exponente privado. Cabe mencionar que el cifrado RSA se utiliza para cifrar datos pequeños debido a que computacionalmente es complejo lo cual redunda en mayor cantidad de tiempo para realizar los procesos.

Por la razón mencionada anteriormente, se acostumbra a utilizar una combinación de cifrado simétrico con cifrado RSA. A continuación se describe un ejemplo híbrido.

Para cifrar un mensaje, primero se cifra con una clave simétrica del tipo 3DES (192 bits), que es un proceso muy rápido, y después se cifra dicha clave (simétrica) con una clave asimétrica pública (larga) del tipo RSA (1024 bits). Con este procedimiento se consigue que sólo el poseedor de la clave privada (asimétrica) pueda descifrar la clave simétrica (rápida) que le permitirá descifrar el mensaje. La figura 2-1 muestra paso a paso los procesos anteriormente descritos:

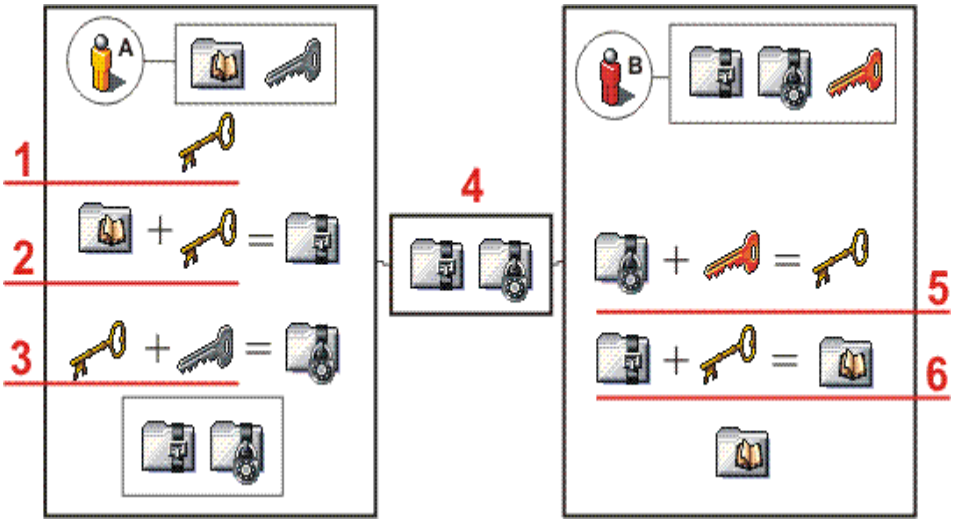

Figura 2-1. Pasos de un cifrado híbrido

A/ El usuario A tiene la clave pública del destinatario B, y el mensaje sin cifrar.

1/ Genera una clave simétrica de 192 bits (3DES por ejemplo).

2/ Cifra el mensaje con la clave simétrica.

3/ Cifra la clave simétrica con la clave pública del destinatario (Asimétrica RSA de 1024 bits, por ejemplo).

4/ Envía el mensaje cifrado y la clave cifrada.

B/ El destinatario tiene su clave privada y recibe el mensaje cifrado y la clave cifrada.

5/ Descifra la clave simétrica de 192 bits (3DES) con su clave privada de 1024 bits.

6/ Finalmente descifra el mensaje con la clave simétrica que lo cifró.

El anterior ejemplo describe el sistema de cifrado usado tanto en los sistemas de clave simétrica como el de clave asimétrica. Funciona mediante el cifrado de clave pública para compartir una clave para el cifrado simétrico. En cada mensaje, la clave simétrica utilizada es diferente por lo que si un atacante pudiera descubrir la clave simétrica, solo le valdría para ese mensaje y no para los restantes. PKI usa sistemas de cifrado híbridos. La clave simétrica es cifrada con la clave pública, y el mensaje saliente es cifrado con la clave simétrica, todo combinado automáticamente en un sólo paquete. El destinatario usa su clave privada para descifrar la clave simétrica y acto seguido usa la clave simétrica para descifrar el mensaje.

3. FIRMA DIGITAL

La finalidad de la firma digital [4] es la de poder garantizar la autoría de la misma y la integridad de los datos firmados. Un mensaje firmado es completamente legible, no está cifrado, es como una postal firmada.

3.1. Generación

Lo primero que se hace es generar un hash del mensaje a firmar. Una función Hash permite obtener un resumen o un hash a partir de un mensaje. Dicho resumen es mucho más pequeño que el mensaje original, y es muy difícil encontrar otro mensaje que tenga el mismo resumen. La función hash presenta las siguientes características:

- Es irreversible, es decir que a partir del resultado de la función hash nunca se podrá obtener el documento original.

- Un ligero cambio en el documento original, de una simple letra por ejemplo, genera un resultado de la función hash completamente distinto del documento original.

Las funciones hash mas utilizadas son MD2, MD4, MD5 y SHA, que dan como resultado resúmenes de 128 y 160 bits. La siguiente figura muestra un ejemplo de la generación de un hash o también conocido como messsage digest.

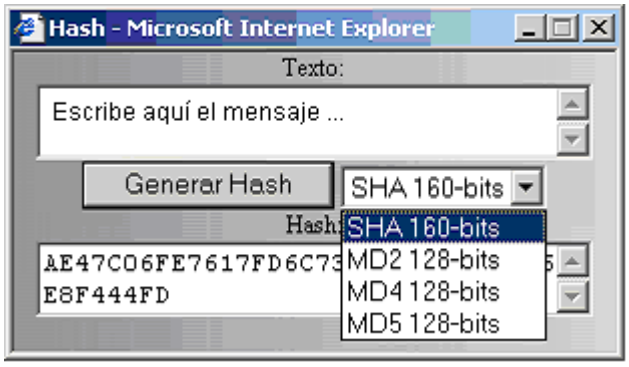

Figura 3-1. Generación de un Hash

El proceso que sigue, es cifrar dicho hash con una clave asimétrica, la clave privada. Finalmente se envía al destinatario el mensaje, la firma y un certificado digital del emisor.

3.2. Verificación

Para comprobar una firma ésta se descifra con la clave pública del certificado digital y se obtiene un hash. Luego se le calcula el hash del mensaje a comprobar y finalmente se compara con el hash anterior. Si son iguales se garantiza que el que lo firmó fue el poseedor de la clave privada (ya que se ha podido descifrar con la pública) y que el documento no ha sido modificado (ya que los hash coinciden). Si cualquiera de estas dos cosas no se cumplen se tiene lo que se suele llamar una "Rotura de firma".

La figura 3-2 describe paso a paso el proceso generación y verificación de la firma digital.

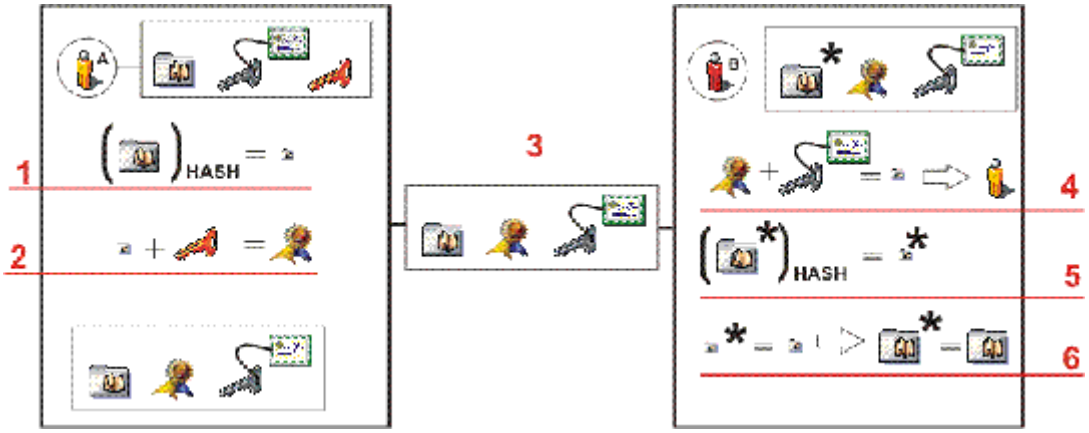

Figura 3-2. Pasos en la firma digital

A/ El usuario "A" tiene el mensaje, su certificado (con su clave pública), y su clave privada.

1/ Genera el hash del mensaje (con la función SHA por ejemplo).

2/ Cifra el hash con su clave privada y obtiene la firma.

3/ Envía el mensaje, la firma, y una copia de su certificado.

B/ El destinatario lo recibe, pero no sabe si han sido modificados por el camino (Hacker).

4/ Comprueba el certificado y descifra la firma con la clave pública del certificado, lo que implica que se cifró con la pareja privada del mismo, garantizando la autoría, y obtiene un hash.

5/ Genera el hash del mensaje recibido.

6/ Comprueba que los dos hash son iguales. Si los hash son iguales quiere decir que los originales que los generaron también, con lo que se garantiza la integridad de los datos.

4. CERTIFICADOS X.509

Uno de los problemas que surgen en Internet es el de la identificación de las personas o entidades, por ejemplo, cómo asegurarse de que una clave pública encontrada en Internet pertenece realmente a quién dice pertenecer. Un certificado es esencialmente una clave pública y un identificador, firmados digitalmente por una AC (Autoridad Certificadora), y su utilidad es demostrar que una clave pública pertenece a un usuario concreto.

El formato de certificado X.509 [5] es el más común y extendido en la actualidad. Estos certificados se estructuran de forma jerárquica, de tal forma que se puede verificar la autenticidad de un certificado comprobando la firma de la autoridad que lo emitió, que a su vez tendrá otro certificado expedido por otra autoridad de rango superior. De esta forma se va subiendo en la jerarquía hasta llegar al nivel más alto, que deberá estar ocupado por un certificado que goce de la confianza de toda la comunidad.
El mecanismo que debe emplearse para conseguir un certificado es generar una pareja de claves asimétricas, pública y privada, y un identificador con los datos del solicitante. Enviar la clave pública --¡Nunca la privada!-- junto con un identificador a la autoridad certificadora, después de que el solicitante se haya identificado positivamente frente a ella, se le enviará su certificado que no es otra cosa que su clave pública y su identificador, firmados con la clave privada de la Autoridad certificadora. (Ver figura 4-1).

issuer: C=ES ST=L=Barcelona O=SECURITY ZUTANEZ OU=Division de certificados CN=Fulano Menganez Email=Fulano@fulanez.es subject: C=ES ST=O=OU=CN=Jaimito Email=Jaimito@jaimito serial:15

Certificate: Data: Version: 1 (0x0) Serial Number: 21 (0x15) Signature Algorithm: md5WithRSAEncryption Issuer: C=ES ST=L=Barcelona O=SECURITY ZUTANEZ OU=Division de certificados CN=Fulano Menganez Email=Fulano@fulanez.es Validity Not Before: Nov 18 15:15:31 1998 GMT Not After : Nov 13 15:15:31 1999 GMT Subject: C=ES, ST=, O=, OU=, CN=Jaimito Email=Jaimito@jaimito Subject Public Key Info: Public Key Algorithm: rsaEncryption RSA Public Key: (1024 bit) Modulus (1024 bit): 00:9e:74:de:c9:1a:6b:f4:fe:d1:04:30:58:7e:8b: 51:7a:98:23:e9:45:a9:c2:a7:7c:f8:f8:b5:9a:a2: ea:c1:99:68:ba:f7:c3:d8:06:05:1b:6a:47:a1:44: 5c:2c:a6:e0:4b:6f:ce:02:c4:06:32:20:34:be:13: 97:39:a3:aa:6f:2f:41:a7:bc:14:c8:f3:0c:ad:9d: 09:63:8a:f5:eb:60:5b:06:a6:01:fb:1a:07:b2:c6: 39:48:bb:b7:00:56:4e:20:6d:87:3f:67:0b:2f:f4: b0:5f:74:7f:90:6b:b4:47:6f:56:1a:b5:c5:42:54: 9b:e5:e3:00:e2:4f:e3:14:47 Exponent: 65537 (0x10001) Signature Algorithm: md5WithRSAEncryption 3b:2b:e9:ff:48:48:35:ab:30:5c:e2:d1:88:c9:29:8b:bc:09: b2:58:80:17:9c:e7:08:0a:7d:8a:5e:46:a8:83:3b:ee:84:de: 62:e3:ea:51:cb:92:bc:fa:db:90:bd:cd:9f:25:d4:4a:48:63: ac:b8:93:f9:dc:9c:cf:ef:fd:45

- -----BEGIN CERTIFICATE-----
MIICOzCCAeUCARUwDQYJKoZIhvcNAQEEBQAwgaYxCzAJBgNVBAYTAkVTMRIwEAYD
VQQIEwlDYXRhbHVueWExDDAKBgNVBAcTA0JjbjEVMBMGA1UEChMMU0VDVVJJVFkg
QkNOMRowGAYDVQQLExFzZWNjaW8gZCdlbXByZXNlczEdMBsGA1UEAxMURGF2aWQg
R3VlcnJlcm8gVmlkYWwxIzAhBgkqhkiG9w0BCQEWFGd1ZXJyZXJvQGdyZWMudXBj

LmVzMB4XDTk4MTExODE1MTUzMVoXDTk5MTExMzE1MTUzMVowZjELMAkGA1UEBhMC
RVMxCTAHBgNVBAgTADEJMAcGA1UEChMAMQkwBwYDVQQLEwAxGDAWBgNVBAMUD0Nh
bHZpbiAmIEhvYmJlczEcMBoGCSqGSIb3DQEJARYNY2FsdmluQGhvYmJlczCBnzAN
BgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAnnTeyRpr9P7RBDBYfotRepgj6UWpwqd8
+Pi1mqLqwZlouvfD2AYFG2pHoURcLKbgS2/OAsQGMiA0vhOXOaOqby9Bp7wUyPMM
rZ0JY4r162BbBqYB+xoHssY5SLu3AFZOIG2HP2cLL/SwX3R/kGu0R29WGrXFQlSb
5eMA4k/jFEcCAwEAATANBgkqhkiG9w0BAQQFAANBADsr6f9ISDWrMFzi0YjJKYu8
CbJYgBec5wgKfYpeRqiDO+6E3mLj6lHLkrz625C9zZ8l1EpIY6y4k/ncnM/v/UU= - -----END
CERTIFICATE-----

Figura 4-1. Ejemplo de un certificado

Este es un certificado, válido durante un año, emitido por la autoridad certificadora SECURITY ZUTANEZ para el usuario Jaimito cuya clave pública RSA se observa en la figura 4-2.

Exponente: 65537

modulo: 111271955529718274977020001053287254971153574325639486465242654
426491145396140308823101054430403215854018849918550447888175506
616458932058891843404404841771733136829794829081324994736239836
651771075446109365198267065678811090107152632592388889101510156
761040462390674445104852526457688536483681077362150397411847 1

Figura 4-2. Ejemplo de una clave pública

Todos los datos están firmados por la AC usando la función hash MD5 y su clave privada RSA.

Para verificar que el certificado es correcto se debería obtener el certificado digital emitido para dicha AC por una segunda AC. Para verificar la veracidad de este segundo certificado se debería obtener el certificado digital emitido por una segunda AC por una tercera AC. Como este proceso podría eternizarse, existen las llamadas autoridades raíz que firman sus propios certificados, un ejemplo de autoridad raíz es Verisign.

Aparte de los datos del emisor y del propietario del certificado éste puede contener información referente a las limitaciones que se hayan establecido para su uso: e-mail, www, etc...

Como se mencionó antes, las claves asimétricas para los certificados se generan en el equipo del usuario ya que toda la infraestructura PKI se basa en que sólo el propietario de la pareja de claves posee la clave privada. El encargado de generar dichas claves es el "CSP" (Proveedor de servicios criptográficos) de cada usuario. Cuando se accede a las páginas de emisión de certificados (online) de una autoridad certificadora, se "llama" al registro de su sistema en busca de los "CPS's" disponibles, para que el usuario seleccione uno. Es en este momento donde se puede escoger si el certificado se almacenará en un disco duro o en un soporte criptográfico externo, llaves USB (iKey),

tarjetas inteligentes, etc. En el momento de la instalación de estos dispositivos se añaden al sistema los CPS que los manejan.

- **Certificados Revocados**

  Aunque el usuario es el responsable de custodiar su clave privada, en su tarjeta SIM (Modulo de Identificación del Suscriptor) por ejemplo, en ocasiones puede ser necesario que un certificado sea revocado. Cuando alguien se autentica o firma algo lo que se hace es garantizar la autoría, y se hace bajo la premisa de que sólo esa persona tiene su clave privada. Si se sospecha que la integridad o confidencialidad de su clave a estado en peligro, se puede pedir la revocación de su certificado a la autoridad certificadora para evitar la suplantación

  En el momento de admitir un certificado para la autenticación de un usuario, o al comprobar una firma, hay que cerciorarse de que el certificado no esté en la lista de revocados de la autoridad certificadora que lo emitió para asegurarse de la autoría y el no repudio. Actualizar y mantener estas listas en red es uno de los mayores problemas de las autoridades certificadoras. No hay que confundir un certificado revocado, con un certificado caducado, prácticamente ningún programa o plataforma PKI admitirá un certificado caducado para autenticarse o firmar, pero es habitual que un certificado siga siendo utilizado después de haber sido revocado, porque por lo general hay muchos problemas con las listas de revocados, tanto de actualización, disponibilidad, y ralentización de las transacciones por el tiempo empleado en cotejar listas enormes en servidores remotos.

5. SEGURIDAD EN JAVA CARD

La plataforma Java Card [6] fue diseñada y desarrollada desde un principio específicamente para proporcionar seguridad a las tarjetas inteligentes. Como una plataforma neutral, la tecnología Java Card está implementada sobre una amplia variedad de soluciones para tarjetas inteligentes con niveles de seguridad que varían desde unos modestos a otros extremadamente seguros. Las características modernas del lenguaje de programación Java proporcionan un rico arreglo de herramientas de desarrollo confiable y seguro de aplicaciones. La JCVM (*Java Card Virtual Machine*) separa las aplicaciones de los niveles inferiores de Hardware y sistema operativo. El Java Card API estándar provee una interfaz uniforme para tarjetas inteligentes disparejas.

La seguridad no es “buena o mala para todos” los asuntos. Más de 100 millones de productos Java Card han sido desplegados. Esta base instalada ha demostrado que la tecnología Java Card proporciona una plataforma segura para el rápido desarrollo y despliegue de aplicaciones de tarjetas inteligentes que conozcan los requerimientos de seguridad del mundo real para operadores de sistemas seguros. Este rango de

operadores son en su mayoría operadores de telefonía móvil, agencias de gobierno y proveedores de servicios financieros.

5.1. Seguridad de las tarjetas Java Card

La seguridad de tarjetas inteligentes es un problema complejo multi-dimensional. Hay diferentes costos asociados con diferentes niveles de seguridad. Diferentes algoritmos criptográficos requieren chips de costos variados, justo como diferentes evaluaciones y certificaciones de seguridad pueden variar el costo en tiempo y dinero. Sin embargo, productos con algún nivel de seguridad pueden beneficiarse de la tecnología Java Card.

La plataforma Java Card proporciona un seguro ambiente de ejecución con un firewall entre diferentes aplicaciones sobre la misma tarjeta. Esto permite a diferentes aplicaciones sobre la misma tarjeta funcionar separadamente e independientemente de cada una de las otras como si ella estuviera en una tarjeta aparte. El lenguaje de programación Java y el Java Card API permiten desarrollar, usando la programación orientada a objetos, aplicaciones seguras de forma rápida y fácil. En contraste, la tradicional programación de tarjetas inteligentes ha usado lenguajes ensambladores o el lenguaje de programación C, los cuales fuerzan la evaluación de seguridad a mirar la aplicación entera como una unidad de conducta verificativa. Las aplicaciones Java Card pueden encapsular datos sensitivos y algoritmos dentro de objetos, los cuales tienen un comportamiento verificable e incrementan la seguridad. El resultado es código que es simple y fácil de desarrollar y mantener. Así, los beneficios incluyen bajo costo, rápido tiempo para poner al mercado y alta seguridad.

Uno de los más importantes beneficios de seguridad de la plataforma Java Card es la comunicación que proporciona entre las tarjetas inteligentes y el gran mundo de tecnología Java. La plataforma Java ha tomado el liderazgo en la comunidad académica y desarrolladora. Creando una fuerte base de conocimiento técnico y entendimiento compartido. Muchas de las publicaciones que tratan la seguridad y “exactitud” de las operaciones usando el lenguaje de programación Java y la tecnología Java Card han sido ampliamente investigadas e implementadas en múltiples contextos. Expertos familiarizados con estas publicaciones pueden ser encontrados alrededor del mundo. Los desarrolladores de aplicaciones de tarjetas inteligentes y emisores se benefician de todas estas investigaciones sobre la plataforma Java Card y los principios generales por debajo de toda la tecnología Java.

5.2. Seguridad del lenguaje Java Card

Una de las características fundamentales de la máquina virtual de Java es la fuerte seguridad proporcionada en parte por el verificador de archivos class. Muchos dispositivos que implementan la plataforma Java Card pueden ser muy pequeños para soportar verificación de archivos CAP sobre el Dispositivo (tarjeta) mismo. Esta consideración dirige hacia un diseño que habilite la verificación sobre un dispositivo pero no basado sobre este. Los datos en un archivo CAP que se necesitan solo para verificación son empaquetados separadamente de los datos necesitados para la ejecución actual de su applet. Esto se permite para flexibilidad en el cómo sea manejada la seguridad en una implementación. Las firmas criptográficas también son parte de los procedimientos de instalación para cargar un applet Java Card en la tarjeta inteligente.

Hay varias opciones para proporcionar seguridad a nivel de lenguaje sobre un dispositivo habilitado para la tecnología Java Card. Lo conceptualmente simple es verificar el contenido de un archivo CAP sobre el dispositivo cuando este sea bajado o después de que sea bajado. Esta opción podría solo ser factible en los dispositivos de “grandes” capacidades. Sin embargo, algún subconjunto de verificación podría ser posible eventualmente en dispositivos con “pequeñas” capacidades. Otras opciones se basan en alguna combinación de uno o mas de: seguridad física del terminal de instalación, una criptográficamente forzada cadena de confianza desde la fuente del archivo CAP, y verificación previa a la descarga del contenido de un archivo CAP.

Los estándares de la plataforma Java Card dicen tan poco como es posible acerca de la instalación de archivos CAP y políticas de seguridad. Desde tarjetas inteligentes que deben servir como procesadores seguros en muchos diferentes sistemas con diferentes requerimientos de seguridad, esto es necesario para permitir tener un gran líder en flexibilidad y para encontrar las necesidades de emisores y usuarios de tarjetas inteligentes.

5.3. Seguridad del Java Card RE

Mientras la JCVM se responsabiliza por garantizar el nivel de seguridad del lenguaje Java, el Java Card RE impone adicionales requerimientos de seguridad de ejecución sobre el dispositivo que implementa el Java Card RE, lo cual resulta ser una necesidad para características adicionales sobre la JCVM.

La característica de seguridad de ejecución básica impuesta por el Java Card RE fuerza al aislamiento de applets usando lo que es llamado un applet firewall. El applet firewall previene los objetos que fueron creados por un applet desde que esté siendo usado por otro applet. Esta prevención desautoriza el acceso a ambos los campos y métodos de instancias de clases, así como también la longitud y contenido de arreglos.

El aislamiento de applets es una importante característica de seguridad, pero esta requiere un mecanismo que permita a los applets compartir objetos en situaciones donde hay una necesidad de interoperar. El Java Card RE permite compartir usando el concepto de objetos de interfaz compartible. Estos objetos proveen la única forma en que un applet puede hacer objetos habilitados para el uso por otros applets.

El applet firewall además protege desde el inautorizado uso de objetos pertenecientes al Java Card RE mismo. El Java Card RE puede usar mecanismos no reflejados en el API Java Card para hacer estos objetos disponibles para usarse por applets.

5.4. ¿Qué tan seguras son las tarjetas Java Card?

En los últimos cinco años, los productos que incorporan la plataforma Java Card han pasado las evaluaciones de seguridad del mundo real para las mayorías de industrias alrededor del mundo. La plataforma Java Card es la plataforma líder para tarjetas multi-aplicación en la telefonía móvil. Esta es además la única plataforma que ha pasado las evaluaciones de seguridad de emisión por la mayoría de asociaciones de pagos

financieras. En adición, esta ha pasado las valoraciones de seguridad lideradas por autoridades gubernamentales, incluyendo el departamento de defensa y la Agencia de seguridad nacional de los Estados Unidos.

6. SEGURIDAD EN J2ME

J2ME (Java 2 Micro Edition) es una tecnología que ha tenido una constante evolución en casi todos sus aspectos, durante el transcurso de los últimos años. Por el lado de la seguridad J2ME ha tratado tres subtemas en especial:

- La seguridad del lenguaje J2ME junto con la de su máquina virtual.
- La seguridad en los MIDlets (aplicaciones) que son descargados a un dispositivo J2ME.
- La segura comunicación de los MIDlets con otros sistemas, por ejemplo con servidores.

Los principales avances con respecto a la seguridad se evidencian principalmente en tres especificaciones:

- MIDP (*Mobile Information Device Profile)* 2.0 o JSR 118 (Java Specification Request 118).
- JTWI (*Java Technology for the Wireless Industry*) 1.0 o JSR 185
- SATSA (*Security And Trust Services API*) o JSR 177 el cual se describe en el Anexo C.

6.1. MIDP 2.0

MIDP 2.0 define el framework para autenticar la fuente (el emisor) de un MIDlet suite y que autoriza que el MIDlet suite ejecute funciones protegidas por otorgamiento de permisos que pueden haberse solicitado basados en las políticas de seguridad del dispositivo. Además se identifican las funciones que son estimadas de seguridad vulnerable y se definen permisos para esas funciones protegidas. Finalmente busca realizar conexiones seguras, en donde se resaltan las conexiones HTTPS (Protocolo de Transferencia de Hiper-Texto Seguro).

Por el lado de autenticar la fuente, MIDP 2.0 define los llamados dominios que pueden ser *untrusted* (no confiable)*, trusted, minimum* y *maximum*. A continuación se describen algunas de sus características:

- Al dominio *untrusted* pertenecen todos los MIDlet suite que no están firmados digitalmente.

- Si el MIDlet suite está firmado es colocado en cualquiera de los dominios protegidos (*trusted o maximum*) que están asociados con el certificado raíz de la cadena de certificados de llave firmada.

- Si un desarrollador de aplicaciones quiere que su MIDlet pertenezca a algún dominio protegido, con lo cual podrá utilizar con total libertad todas la características del dispositivo J2ME, debe obtener (comprar) un certificado digital de una Certificate Authority reconocida (Ej: VeriSign o Western) y firmar su MIDlet suite.

Además MIDP 2.0 define los permisos utilizados por el grupo de funciones. Estos permisos de listan en la tabla 6-1.

Function Group	Protocol	Permission
Net Access	http	javax.microedition.io.Connector.http
Net Access	https	javax.microedition.io.Connector.https
Net Access	datagram	javax.microedition.io.Connector.datagram
Net Access	datagram server (without host)	javax.microedition.io.Connector.datagramreceiver
Net Access	socket	javax.microedition.io.Connector.socket
Net Access	server socket	javax.microedition.io.Connector.serversocket
Net Access	ssl	javax.microedition.io.Connector.ssl
Local Connectivity	comm	javax.microedition.io.Connector.comm
Application Auto Invocation	All	javax.microedition.io.PushRegistry

Tabla 6-1. Permisos de grupos de función

Por el lado de las conexiones seguras MIDP 2.0 define la conexión HTTPS [7], que no es más que una conexión HTTP soportada en los estándares SSL (*Secure Sockets Layer*) y WTLS (*Wireless Transport Layer Security*) para permitir la transmisión de datos cifrados. Además MIDP 2.0 deja abierta la opción de utilizar esquemas de seguridad basados en XML [8] (*eXtensible Markup Language*) los cuales han tenido últimamente un gran auge debido al uso de los *Web Services* [9].

6.1.1. Requerimientos de acceso a red

Las aplicaciones *untrusted* deben usar los APIs normales HttpConnection y HttpsConnection para acceder a la web y a seguros Web Services. No hay restricciones sobre los números de puertos de servidores web a través de estas interfaces. La implementación permite que los servidores web puedan identificar aplicaciones untrusted. En MIDP 2.0 :

- La implementación de HttpConnection y HttpsConnection debe incluir un encabezado separado “User-Agent” con el Product-Token “UNTRUSTED/1.0”. Los encabezados User-Agent proporcionados por la aplicación **no tienen que** ser borrados.

- La implementación de SocketConnection usando sockets TCP **tiene que** lanzar la java.lang.SecurityException cuando un MIDlet suite *untrusted* intente conectarse sobre los puertos 80 y 8080 (http) y 443 (https).

- La implementación de SecureConnection usando sockets TCP **tiene que** lanzar la java.lang.SecurityException cuando unos MIDlet suites *untrusted* intenten conectarse sobre el puerto 443 (https).

- La implementación del método DatagramConnection.send **tiene que** lanzar una java.lang.SecurityException cuando un MIDlet suite *untrusted* intente enviar datagramas a algún de los puertos 9200-9203 (Gateway WAP).

- Los requerimientos de arriba **deben** ser aplicados independiente de que el API usado acceda a la red. Por ejemplo, los métodos javax.microedition.io.Connector.open y Manager.createPlayer **deben** lanzar un java.lang.SecurityException si se intenta acceder a estos números de puertos a través o por medio de otras formas que los normales HttpConnection y HttpsConnection.

6.1.2. SSL (Secure Socket Layer)

El protocolo SSL permite establecer conexiones seguras a través de Internet, de forma sencilla y transparente. La idea consiste en interponer una fase de codificación de los mensajes antes de enviarlos por la red. Una vez que se ha establecido la comunicación, cuando una aplicación quiere enviar información a otro computador, la capa SSL la recoge y la codifica, para luego enviarla a su destino a través de la red. Análogamente, el módulo SSL del otro ordenador se encarga de decodificar los mensajes y se los pasa como texto plano a la aplicación destino. Todo se hace de forma transparente para el usuario.

Una comunicación SSL efectuada por el puerto 443, consta fundamentalmente de dos fases:

1. Fase de saludo (*handshaking*). Consiste básicamente en una identificación mutua de los interlocutores, para la cual se emplean habitualmente los certificados X.509. Tras el intercambio de claves públicas, los dos sistemas escogen una clave de sesión, de tipo simétrico.

2. Fase de comunicación. En esta fase se produce el auténtico intercambio de información, que se codifica mediante la clave de sesión acordada en la fase de saludo.

Cada sesión SSL lleva asociado un identificador único que evita la posibilidad de que un

atacante escuche la red y repita exactamente lo mismo que ha oído, aún sin saber lo que significa, para engañar a uno de los interlocutores.

Las ventajas de este protocolo son evidentes, ya que liberan a las aplicaciones de llevar a cabo las operaciones criptográficas antes de enviar la información, y su transparencia permite usarlo de manera inmediata sin modificar apenas los programas ya existentes, aunque requiere un mayor ancho de banda para realizar sus comunicaciones. Desde hace tiempo los principales navegadores de Internet incorporan un módulo SSL, que se activa de forma automática cuando es necesario. En la figura 6-1 se muestran los pasos para establecer una conexión HTTPS.

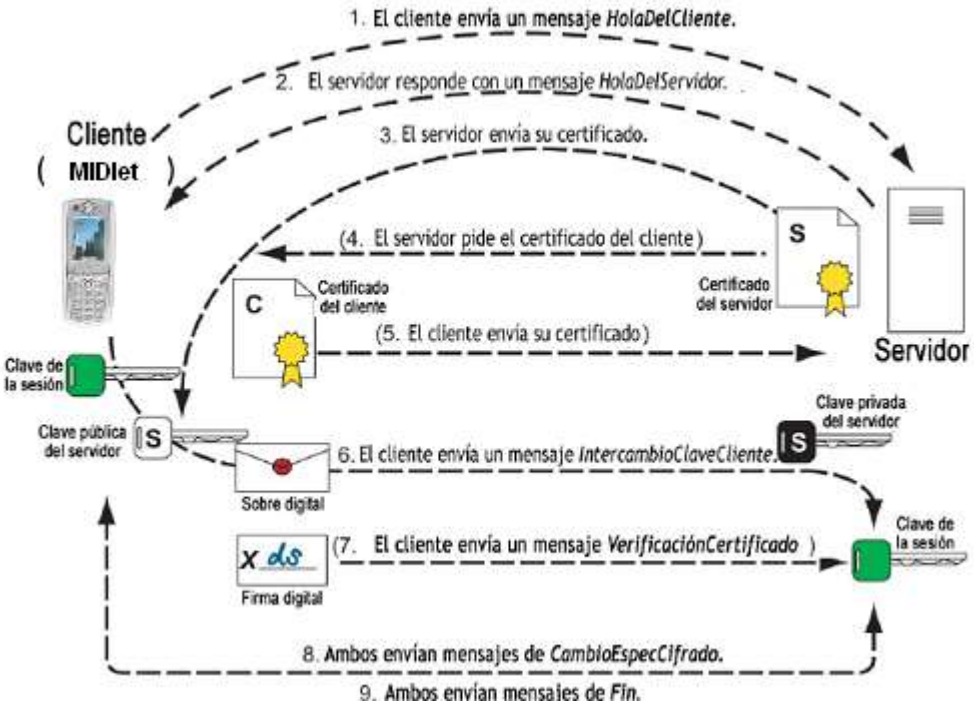

Figura 6-1. Pasos para el establecimiento de una comunicación segura

6.1.3. WTLS o Certificados WAP

Hoy en día existen pocos dispositivos J2ME que puedan utilizar certificados personales. Esto se debe a que la memoria y la capacidad de proceso de los terminales son muy limitadas y no pueden generar claves ni almacenar certificados, lo cual se piensa solucionar con el uso de SATSA y terminales mas avanzados. Actualmente algunos clientes WAP manejan WTLS y es curioso comprobar que, aunque para la mayoría de terminales todavía no es posible manejar dichos certificados, Verisign ya ha conseguido incluir sus certificados raíces *.wtls en la mayoría de terminales móviles "desde la fabrica". La figura 6-2 describe el uso de WTLS.

Figura 6-2. Uso de WTLS en las redes móviles

6.2. JTWI 1.0 (*Java Technology for the Wireless Industry*)

Dentro del conjunto total de la especificación JTWI (JSR 185) 1.0, se definen las llamadas "Políticas de seguridad para dispositivos GSM / UMTS". A continuación se muestra un resumen de estas políticas.

- Esta parte de la especificación se deriva de partes no confiables de las "*Recommended Security Policy for GSM/UMTS Compliant Devices*" (RP) un suplemento de MIDP 2.0.

- Para aplicaciones *untrusted*, ésta especificación substituye el documento RP. En todas las otras consideraciones, el RP es la guía para dispositivos cumplan con JTWI.

- El propósito de esta especificación es extender la base del MIDlet suite security framework definido en MIDP 2.0 y definir las siguientes áreas:

  - El modelo de confianza requerido para dispositivos GSM/UMTS.
  - Capacidades de MIDlets basadas sobre permisos definidos por MIDP 2.0 y otras especificaciones.
  - La definición de los modos de interacción de permisos de usuario.
  - La línea guía sobre indicadores de usuarios y notificaciones.

Los dispositivos GSM / UMTS que implementen estas políticas de seguridad deben seguir al framework de seguridad especificado en el MIDP 2.0. Los dispositivos que no sean GSM /UMTS deberían adoptar estos modelos.

JTWI define las siguientes configuraciones de seguridad para los MIDlet Suites (en dominios *untrusted*) basada en las funciones realizadas.

Grupos de función	**Configuración por defecto**	**Configuraciones permitidas**
Acceso a red	*Session*	*Session, Oneshot, No*
Mensajería	*Oneshot*	*Oneshot, No*
Auto-invocación de aplicación	*Session*	*Session, Oneshot, No*

Conectividad local	*Session*	*Blanket, Session, Oneshot, No*
Grabación de multimedia	*Oneshot*	*Session, Oneshot, No*

Tabla 6-2. Grupos de función y configuraciones de usuario

Los MIDlet suites *untrusted* pueden solicitar permisos usando los atributos en el descriptor de aplicación o en el JAR manifest. Los permisos de usuario solicitados son presentados usando el grupo de función. La presentación es específica de la implementación, pero las siguientes reglas deben ser seguidas:

- El usuario debe ser capaz de cambiar cada grupo de función configurando alguna configuración permitida para los grupos de función del listado 6-2.

- El dispositivo **tiene que** tener un conjunto de grupo de función que se configure para cada MIDlet suite *untrusted*. La implementación **no tiene que** ofrecer una forma para configuraciones de conjuntos de grupo de función para varios MIDlet suites de dominios *untrusted* al mismo tiempo.

- Una implementación **tiene que** ser capaz de presentar el nombre del MIDlet suite, número de la versión y configuraciones de grupo de función al usuario para cada MIDlet suite instalado. Una implementación **tiene que** presentar alguna información adicional relacionada a seguridad al usuario.

La especificación MIDP 2.0 define un número de restricciones a los APIs para los cuales los permisos son definidos y el acceso puede restringirse por políticas. El mapeo de los permisos de los grupos de función es listado en la tabla 6-1.

Además JTWI define la características de seguridad que deben cumplir algunos de los API mas importantes, por ejemplo define las Políticas *untrusted* del *Wíreless Messaging API*.

REFERENCIAS BIBLIOGRÁFICAS

[1] GSM 02.48 Disponible en http://www.3gpp.org/ftp/Specs/html-info/0248.htm

[2] Artículo en internet, Clava simétrica, disponible en http://www.eumed.net/cursecon/ecoinet/seguridad/clavesimetrica.ppt

[3] Artículo en internet, Public Key Infraestructure, Enero de 2005, Disponible en http://foro.elhacker.net/index.php/topic,53455.0.htm

[4] Artículo en internet, Firmas digitales, Disponible en http://www2.sharesafe.net/sharesafe/TuturialPKI2.asp#certf

[5] Enrique Ortiz Enríquez de Salamanca, CERTIFICADOS DIGITALES, Noviembre de 2003, disponible en http://www2.sharesafe.net/sharesafe/TuturialPKI2.asp?men2=none

[6] Especificación Java Card, Disponible en http://java.sun.com/products/javacard

[7] Especificación de la conexión HTTPS, disponible en http://www.ietf.org/rfc/rfc2818.txt

[8] XML Digital Signature APIs, disponible en http://jcp.org/en/jsr/detail?id=105

[9] http://java.sun.com/webservices/

GLOSARIO

API	Application Programing Interface
AES	Advanced Encryption Standard
DES	Data Encryption Standard
GSM	Global System for Mobile Communications
J2ME	Java 2 Micro Edition
JCRE	Java Card Runtime Environment
JCRMI	Java Card Remote Method Invocation
JCVM	Java Card Virtual Machine
JTWI	*Java Technology for the Wireless Industry*
ME	Mobile Equipment
MIDP	Mobile Information Device Profile
PKCS#11	Cryptographic Token Interface Standard
PKI	Public Key Infrastructure
RSA	algoritmo Rivest-Shamir-Adleman
SATSA	Security and Trust Service API
SIM	Subscriber Identity Module