CONTENIDO

ANEXO C *SECURITY AND TRUST SERVICES API* (SATSA)

1. INTRODUCCIÓN……1

1.1. Paquete opcional SATSA-APDU……2

1.2. Paquete opcional SATSA-JCRMI……..2

1.3. Paquete opcional SATA-PKI……..3

1.4. Paquete opcional SATSA-CRYPTO……..3

2. java.security……..4

3. javacard.framework……5

4. javacard.security……...6

5. javax.crypto……...6

6. javax.microedition.apdu……...8

7. javax.microedition.jcrmi……..9

8. javax.microedition.pki……10

9. javax.microedition.securityservice……10

REFERENCIAS BIBLIOGRÁFICAS……11

GLOSARIO……12

ANEXO C
SECURITY AND TRUST SERVICES API (SATSA)

1. INTRODUCCIÓN

El *Security And Trust Services API (SATSA)* [1] define cuatro paquetes opcionales para la plataforma *Java 2 Micro Edition (J2ME)*. La especificación ha sido producida en respuesta a la *Java Specification Request 177 (JSR-177)*. El propósito de la especificación es describir una colección de APIs que proporcionen servicios de seguridad y confianza mediante el uso de un *Security Element (SE).* Un SE, es un componente de un dispositivo J2ME, que proporciona los siguientes beneficios:

- Almacenamiento seguro protegiendo datos sensitivos, tales como claves privadas de usuario, certificados de llave pública, información personal, etc.

- Operaciones de cifrado que soportan protocolos de pago, integridad y confidencialidad de los datos.

- Un ambiente de ejecución para desplegar características de seguridad personalizadas. Las aplicaciones J2ME pueden dejar que el SE maneje características de valor agregado, tales como identificación y autenticación de usuario, pagos bancarios, aplicaciones de lealtad, etc.

Un SE puede tener una variedad de formas. Las tarjetas inteligentes son comúnmente usadas para implementar un SE. Ellas están ampliamente desplegadas en los teléfonos móviles, tales como tarjetas SIM en teléfonos GSM, tarjetas UICC en teléfonos de 3G y tarjetas RUIM en teléfonos CDMA. Alternativamente, un SE puede ser totalmente implementado in software. SATSA no excluye ninguna posible implementación de un SE, aunque algunos paquetes son optimizados para implementaciones sobre tarjetas inteligentes.

Los cuatro paquetes opcionales del API pueden ser implementados independientemente. Esto paquetes se describen a continuación:

- SATSA-APDU: paquete que soporta la comunicación con aplicaciones sobre tarjetas inteligentes usando el protocolo APDU.

- SATSA-JCRMI: paquete que define un cliente *Java Card RMI* que le permite a las aplicaciones J2ME invocar un método sobre un objeto Java Card remoto.

- SATSA-PKI: paquete que le permite a las aplicaciones calcular una firma digital (pero no verificarla) y gestionar credenciales de usuario básicas.

- SATSA-CRYPTO: paquete que define un subconjunto del API de criptografía de J2SE. Este proporciona operaciones de cifrado básicas soportando *message digest,* verificación de firma digital, cifrado y descifrado de datos.

La comunicación con las tarjetas inteligentes está basada en el Generic Connetion Framework del paquete javax.microedition.io. Dos interfaces de conexión con las tarjetas inteligentes son definidas para establecer una conexión con una tarjeta inteligente. La interface APDUConnection es usada para comunicarse con tarjetas inteligentes que cumplen con el estándar ISO 7816-4. La interfaz JavaCardRMIConnection es usada para iniciar una sesión JCRMI. Estas conexiones son definidas en los paquetes javax.microedition.apdu y javax.microedition.jcrmi respectivamente.

1.1. Paquete opcional SATSA-APDU

El paquete opcional SATSA-APDU incluye dos componentes para soportar la comunicación con tarjetas inteligentes que cumplen con el ISO 7816-4 usando el protocolo APDU.

- Un subconjunto del paquete *java.lang*. Este soporta la clase de excepción *UnsupportedOperationException*, la cual no está incluida en el API CLDC o el API MIDP.

- El paquete *javax.microedition.apdu*. Este incluye a la interface APDUConnection para soportar intercambio de APDUs.

1.2. Paquete opcional SATSA-JCRMI

El paquete opcional SATSA-JCRMI incluye cuatro componentes para proporcionar un API cliente Java Card RMI:

- Un subconjunto del paquete java.lang. Este soporta la clase de excepción UnsupportedOperationException, la cual no está incluida en el API CLDC o el API MIDP.

- Un subconjunto del paquete java.rmi de la plataforma J2SE para proporcionar las interfaces RMI básicas. El subconjunto incluye la interfaz Remote y la clase RemoteException.

- El paquete javax.microedition.jcrmi. Este incluye la interfaz JavaCardRMIConnection usada para iniciar una sesión JCRMI e interfaces usadas por los stubs generados por el compilador jcrmi stub.

- Un subconjunto del API Java Card. El subconjunto incluye las clases de excepción definidas en el paquete javacard.framework, javacard.framework.service y

javacard.security. Estas excepciones pueden ser lanzadas durante una invocación a un método de un objeto Java Card debido a errores criptográficos (en el paquete javacard.security), errores de acceso al framework de la tarjeta (en el paquete javacard.framework) o errores detectados en el acceso a los servicios de la tarjeta (en el paquete javacard.framework.service). Por lo tanto, las clases de excepciones Java Card son incluidas para proporcionar el mismo comportamiento cuando los métodos sean invocados remotamente.

1.3. Paquete opcional SATSA-PKI

El paquete opcional SATSA-PKI incluye dos componentes para proveer generación de firma digital y la gestión de credencial de usuario básica (por ejemplo, un certificado X.509 es una credencial de usuario que incluye una clave pública).

- El paquete javax.microedition.pki. Este soporta generación de solicitudes de certificados y registro local de credenciales de usuario. Las credenciales de usuario son usadas en conjunción con otros parámetros para calcular la firma digital. Cuando se despliega sobre una plataforma MIDP 2.0, este paquete además incluye la clase Certificate y CertificateException, las cuales son definidas en el API MIDP 2.0.

- El paquete javax.microedition.securityservice. Este soporta generación de firmas digitales a nivel de aplicación que conforman el formato Cryptographic Message Syntax (CMS) [2].

1.4. Paquete opcional SATSA-CRYPTO

El paquete opcional SATSA-CRYPTO es un subconjunto del API de criptografía de la plataforma J2SE. Cuando este paquete opcional es implementado sobre una plataforma basada en CDC/FP que ya soporta algunas clases de este paquete opcional, el API completo es la unión del API en el CDC/FP y el API en el paquete opcional SATSA-CRYPTO.

- Los paquetes java.security y java.security.spec. Ellos proporcionan el soporte básico para acceder a claves públicas, computar digests y verificar firmas digitales.

- Los paquetes javax.crypto y javax.crypto.spec. Ellos proporcionan el soporte básico para cifrar y descifrar datos.

- Un subconjunto del paquete java.lang. Este incluye las clases de excepción IllegalStateException, las cuales no están incluidas en el API CLDC o el API MIDP.

2. java.security

El paquete java.security de J2SE proporciona las clases e interfaces para el framework de seguridad. Este paquete soporta la generación y el almacenamiento de claves pares para cifrado, así como también un número de operaciones de cifrado exportables incluyendo aquellas para message digest y generación de firma digital.

SATSA soporta un subconjunto del paquete java.security de la plataforma J2SE, el cual contiene:

Interfaces

- Key: es una interfaz de alto nivel para todas las claves.
- PublicKey: interfaz que define una clave pública.

Clases

- KeyFactory: esta clase es usada para convertir de key specifications (representaciones transparentes del material fundamental de la clave) a Keys (clave criptográfica opaca del tipo Key).
- MessageDigest: esta clase proporciona a las aplicaciones la funcionalidad de un algoritmo de message digest, tales como MD5 o SHA.
- Signature: esta clase es usada para proporcionar a las aplicaciones la funcionalidad de un algoritmo de firma digital.

Excepciones

- DigestException: esta es la excepción genérica del Message Digest.
- GeneralSecurityException: esta clase es una excepción genérica de seguridad que proporciona el tipo de seguridad para todas las clases de excepciones relacionadas a seguridad que hereden de ella.
- InvalidAlgorithmParameterException: esta es la excepción para inválidos o inapropiados parámetros del algoritmo.
- InvalidKeyException: esta es la excepción para Keys inválidas (codificación inválida, longitud errónea, no inicializada, etc).
- KeyException: esta es la excepción básica para claves.
- NoSuchAlgorithmException: esta excepción es lanzada cuando un algoritmo de cifrado en particular es solicitado pero no está disponible en el entorno.

- SignatureException: esta es la excepción genérica para firmas digitales.

SATSA también proporciona un subconjunto del paquete java.security.spec de la plataforma J2SE versión 1.4.2. El subconjunto lo componen:

Interfaces

- AlgorithmParameterSpec: una especificación (transparente) de los parámetros criptográficos.

- KeySpec: una especificación (transparente) del material que constituye una clave criptográfica.

Clases

- EncodedKeySpec: esta clase representa una clave pública en formato codificado.

- X509EncodedKeySpec: esta clase representa una codificación ASN.1 de una clave pública, codificada de acuerdo al tipo ASN.1 SubjectPublicKeyInfo.

Excepciones

- InvalidKeySpecException: esta es la excepción para especificaciones de clave inválidas.

3. javacard.framework

Este paquete proporciona las excepciones del API Java Card que pueden ser lanzadas por un método remoto. Las clases de excepción Java Card son incluidas para asegurar el mismo comportamiento cuando un método es invocado remotamente. El paquete contiene:

Excepciones

- APDUException: representa una excepción relacionada a una APDU.

- CardException: esta clase define un campo reason y dos métodos para accederlo getReason() y setReason().

- CardRuntimeException: esta clase define un campo reason y dos métodos para accederlo getReason() y setReason().

- ISOException: esta clase encapsula una respuesta ISO7816-4 status word como su código de razón (reason).

- PINException: esta clase representa una excepción relacionada con el acceso a la clase OwnerPIN.

- SystemException: esta clase representa una excepción relacionada con la clase JCSystem.

- TransactionException: representa una excepción en el subsistema de transacciones.

- UserException: representa una excepción de usuario.

Además se cuenta con el paquete javacard.framework.service el cual contiene una Excepción:

- ServiceException: representa una excepción relacionada al framework de servicio Java Card.

4. javacard.security

Proporciona la excepción del API JavaCard que puede ser lanzada por un método remoto. La excepción es la siguiente:

- CryptoException: representa una excepción relacionada a criptografía.

5. javax.crypto

Este paquete es un subconjunto del paquete javax.crypto de la plataforma J2SE. Soporta cifrado simétrico y asimétrico. Sólo un Proveedor de Servicio Criptográfico por defecto es soportado. El paquete contiene:

Clases

- Cipher: esta clase proporciona la funcionalidad de cifrado criptográfico para cifrar y descifrar datos.

Excepciones

- BadPaddingException: esta excepción es lanzada cuando un mecanismo de relleno (padding) en particular es esperado para los datos de entrada pero los datos no están rellenados (padded) apropiadamente.

- IllegalBlockSizeException: esta excepción es lanzada cuando la longitud de los datos proporcionada para un bloque de cifrado es incorrecta, por ejemplo, que el tamaño del bloque no sea igual al del cifrado.

- NoSuchPaddingException: esta excepción es lanzada cuando un mecanismo de relleno en particular es solicitado pero no esta disponible en el entorno.

- ShortBufferException: esta excepción es lanzada cuando un buffer de salida proporcionado por el usuario es demasiado pequeño como para mantener el resultado de la operación.

Además se cuenta con el paquete javax.crypto.spec el cual es un subconjunto del paquete disponible en la plataforma J2SE. Este paquete proporciona las clases que especifican las claves y parámetros de los algoritmos. La especificación de una clave es una representación transparente del material que constituye una clave. Una especificación de un parámetro de algoritmo es una representación transparente del conjunto de parámetros usados con un algoritmo. El paquete contiene:

Clases

- IvParameterSpec: esta clase especifica un vector de inicialización (IV).

- SecretKeySpec: esta clase especifica una clave secreta en una forma independiente del proveedor.

La especificación recomienda que el paquete opcional SATSA-CRYPTO soporte los siguientes algoritmos, modos de algoritmos y esquemas de relleno:

Message Digest

- Algoritmo: SHA -1

Firma digital

- Algoritmo: SHA1withRSA

Cifrado Asimétrico

- Algoritmo: RSA

Cifrado Simétrico

- Algoritmos: DES, DESede, AES

- Modos de algoritmo: ECB, CBC

- Esquemas de relleno: NoPadding, PKCS5Padding

6. javax.microedition.apdu

Este paquete define el manejador del protocolo APDU para comunicaciones ISO7816-4 con una tarjeta inteligente. El paquete contiene una sola interfaz:

- APDUConnection: esta interfaz define la conexión mediante APDUs (Application Protocol Data Units).

Las aplicaciones J2ME pueden usar esta conexión para comunicarse con aplicaciones sobre una tarjeta inteligente. El estándar ISO7816-4 define el protocolo APDU como un protocolo de nivel de aplicación entre una tarjeta inteligente y una aplicación sobre el dispositivo. Hay dos tipos de mensajes APDU: command APDUs y response APDUs. Los command APDUs son enviados a la tarjeta inteligente por aplicaciones J2ME. Los response APDUs son los mensajes recibidos desde las tarjetas inteligentes.

Además una aplicación J2ME puede usar el método getATR de esta interfaz para obtener información relativa a un Answer To Reset (ATR) retornado por la tarjeta inteligente cuando se le da un reset. También se puede utilizar el método enterPIN para preguntarle al usuario por el PIN el cual será enviado para que la tarjeta lo verifique.

Una conexión APDU es creada al pasarle al método Connector.open una cadena URI de conexión genérica mas el Application IDentifier (AID) y opcionalmente el slot en el cual la tarjeta está insertada. Por ejemplo [3], la cadena de conexión:

apdu:0;target=A0.00.00.00.62.03.01.0C.02.01

Esta cadena indica que la conexión es dirigida a la aplicación con AID *A0.00.00.00.62.03.01.0C.02.01*, la cual reside en la tarjeta inteligente insertada en el slot número 0. Si se quiere crear una conexión con SAT (SIM Application Toolkit) entonces se debe cambiar el parámetro AID por la cadena “SAT”. Por ejemplo:

apdu: 0;target=SAT

Cada conexión APDU tiene reservado un canal lógico exclusivamente para esto. Eso es, el canal está dedicado a las aplicaciones J2ME y a la aplicación seleccionada de la tarjeta inteligente hasta que la conexión sea cerrada.

Una vez una conexión APDU es creada, la aplicación J2ME puede usar el método exchangeAPDU para enviar command APDUs y recibir response APDUs a y desde la tarjeta.

Finalmente la aplicación J2ME puede llamar al método Connection.close sobre una conexión APDU para cerrarla.

7. javax.microedition.jcrmi

Este paquete proporciona las clases en interfaces para una conexión Java Card RMI. La interfaz JavaCardRMIConnection define la conexión Java Card RMI que puede ser usada por aplicaciones J2ME para comunicarse con aplicaciones sobre una tarjeta inteligente usando el protocolo Java Card RMI.

Una aplicación J2ME usa stubs para comunicarse con objetos remotos sobre una tarjeta inteligente. Un stub es un proxy para un objeto remoto. Cuando se pasa un objeto remoto como valor de retorno a una llamada de un método remoto, el stub de ese objeto remoto es pasado en su lugar. Las aplicaciones J2ME invocan un método sobre el stub local el cual es responsable de la ejecución de la llamada al método sobre el objeto remoto. La clase RemoteStub es la superclase común para los stubs de objetos remotos.

La interfaz RemoteRef representa al manejador para un objeto remoto. Cada stub contiene una instancia de RemoteRef. La interfaz RemoteRef contiene la representación concreta de una referencia. Esta referencia es usada para ejecutar llamadas remotas sobre el objeto remoto que está referenciado.

La clase RemoteStub y la interfaz RemoteRef son transparentes para las aplicaciones J2ME. Ellas son usadas por los stubs generados por el compilador de stub Java Card RMI.

Este paquete contiene:

Interfaces

- JavaCardRMIConnection: esta interfaz define la conexión Java Card RMI la cual puede ser usada por aplicaciones J2ME para comunicarse con aplicaciones sobre una tarjeta inteligente usando el protocolo Java Card RMI.

- RemotoRef: esta interfaz representa al manejador para un objeto remoto.

Clases

- RemoteStub: esta clase es la superclase común para los stubs de objetos remotos.

8. javax.microedition.pki

Este paquete define las clases para el soporte básico a la gestión de certificados de usuario. El paquete contiene:

Clases

- UserCredentialManager: esta clase proporciona la funcionalidad para gestionar credenciales de usuario lo cual incluye crear solicitudes de firmado de certificados, adicionar credenciales de usuario, y remover credenciales que pueden ser usadas para generar firmas digitales como se especifica en la clase CMSMessageSignatureService.

Excepciones

- UserCredentialManagerException: esta clase es usada para identificar condiciones de error en la gestión del almacenamiento de los certificados de usuario.

9. javax.microedition.securityservice

Este paquete define las clases para generar firmas digitales a nivel de aplicación que conforman el formato Cryptographic Message Syntax (CMS). El paquete contiene:

Clases

- CMSMessageSignatureService: esta clase proporciona servicios para mensajes criptográficos.

Excepciones

- CMSMessageSignatureServiceException: esta excepción es usada para identificar condiciones de error detectadas mientras se firman mensajes.

REFERENCIAS BIBLIOGRÁFICAS

[1] Especificación de SATSA, disponible en http://jcp.org/en/jsr/detail?id=177

[2] Definición de la Cryptographic Message Syntax (CMS), disponible en http://www.ietf.org/rfc/rfc2630.txt

[3] Implementación de referencia de SATSA, disponible en http://java.sun.com/products/satsa/

GLOSARIO

A

AES: Advanced Encryption Estandard

AID: Application Identifier

APDU: Application Protocol Data Unit

API: Application Program Interface

ASN: Abstract Syntax Notation

ATR: Answer To Reset

C

CBC: Cipher Block Chaining

CDC/FP: Connected Device Configuration/Foundation Profile

CDMA: Code Division Multiple Access

CLDC: Connected Limited Device configuration

CMS: Cryptographic Message Syntax

D

DES: Data Encryption Standard

DESede: DES encryption decryption encryption

E

ECB: Electronic Code Book

G

GSM: Global System for Mobile Communications

I

ISO: International Standards Organization

IV: Initialization Vector

J

J2ME: Java 2 Micro Edition

J2SE: Java 2 Standard Edition

JCRMI: Java Card RMI

JSR: Java Specification Request

M

MIDP: Mobile Information Device Profile

P

PKI: Public Key Infrastructure

R

RMI: Remote Method Invocation

RUIM: Removable User Identity Module

RSA: algoritmo Rivest-Shamir-Adleman

S

SAT: SIM Application Toolkit

SATSA: Security And Trust Services API

SE: Security Element

SIM: Subscriber Identity Module

U

UICC: Universal Integrated Circuit Card