

Smart Card for Global Enterprise Authentication

Ruchir Choudhry, Alok Prasad,

ruchirchoudhry@cint.co.in alokprasad@cint.co.in;

Abstract.

Global Smart cards aren't new. They were introduced in Europe two decades ago in the form of (not so smart) memory cards, used to store critical phone information with the purpose of reducing thefts from pay phones.

Smart-card technology is an industry standard defined and controlled by the Joint Technical Committee 1 (JTC1) of the International Standards Organization (ISO) and the International Electronic Committee (IEC). The series of international standards ISO/IEC 7816, introduced in 1987 with its latest update in 2003, defines various aspects of a smart card, including physical characteristics, physical contacts, electronic signals and transmission protocols, commands, security architecture, application identifiers, and common data elements.

Global smart card is a plastic card that contains an embedded integrated circuit (IC). A smart card resembles a credit card. When used as a SIM card, the plastic card is small - just big enough to fit inside a cell phone. The Global Smart cards are highly secure by design, and tampering with one result in the destruction of the information it contains.

In some areas of use Global smart cards are just memory cards that merely provide protected non-volatile storage. This global smart card is very advanced smart cards have both microprocessors and memory, for secure processing and storage, and can be used for security applications that use public-key or shared-key algorithms.

The non-volatile memory in a global smart card is its most precious resource and can be used to store secret keys and digital certificates. Unlikely some smart cards have separate cryptographic coprocessors that support such algorithms as RSA, AEC, and (3)DES. We have tried to have a integrate it.

Global Smart cards don't contain a battery, and comes to active state only when connected with a card reader. When connected, after performing a reset sequence the card remains passive, waiting to receive a command request from a client (host) application.

Global Smart cards are contact less. This contact less smart cards communicates by means of a radio frequency signal, with a typical range of less than 2 feet. The radio communication of contact less smart cards is based on technology similar to Radio Frequency ID (RFID) tags used in stores to counter theft and track inventory. Figure 1 depicts contact and contact less smart cards:

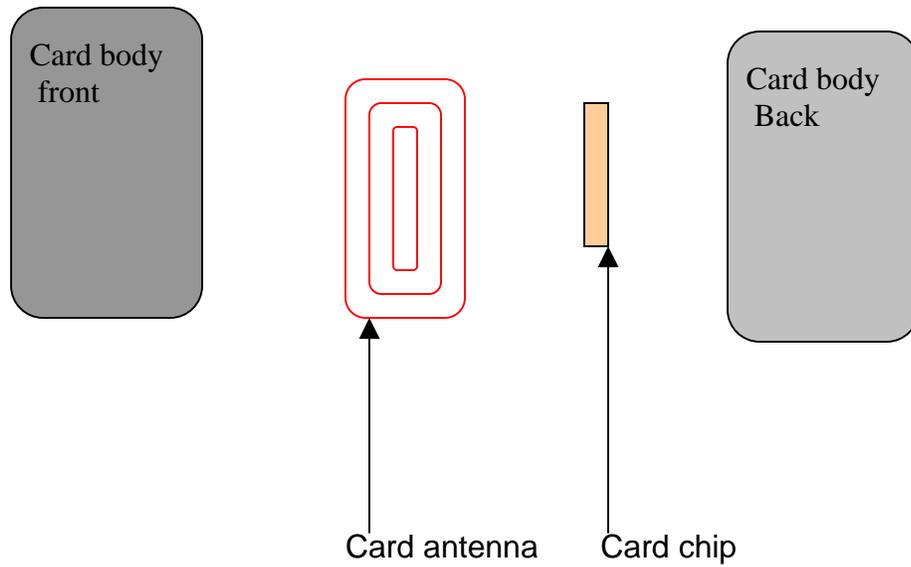


Figure 01: Smart Card/tag

Layer 01 Tag object Domain Figure 02:

The Tagged Object Domain contains the tagged products in a supply chain or any other assets or locations that are intended to be tracked or monitored, including the use of sensors on tags. Because object and tag are attached physically, they are considered as components of the same domain.

In contrast to other domains, most of the artifacts in the Tagged Object Domain are mobile, that is they can move across different RFID Infrastructures/segments. The data is constantly traversed between the different layers and at different locations across the globe; this restriction imposes strict interoperability requirements on those artifacts that would be addressed ideally through open standards.

Layer 02 Antennas and Reader Device Domain Figure 02

The Antenna and Reader Domain are located across the globe and it serves as an interface between the physical world of objects, tags, radio frequencies, and so forth and the world of software systems. As the card/tag passes through this system it detects and activates the card/tag and the data is traversed between the systems

The Global Card platform is a multiple-application environment. As Figure 03 illustrates, one or more Java Card applets may reside on the card, along with supporting software - the card's operating system and the Java Card Runtime Environment (JCRE). The JCRE consists of the Java Card VM, the Java Card Framework and APIs, and some extension APIs.

All Java Card applets extend the Applet base class and must implement the `install()` and `process()` methods; the JCRE calls `install()` when installing the applet, and `process()` every time there is an incoming APDU for the applet.

Java Card applets are instantiated when loaded and stay alive when the power is switched off. A card applet behaves as a server and is passive. After a card is powered up, each applet remains inactive until it's selected, at which time initialization may be done. The applet is active only when an APDU has been dispatched to it. How an applet becomes active (selected) is described in the section

We are working on the software/hardware with, more sophisticated functionality and is expected to move from the Edge Domain down to readers (for example, event filtering), making readers *smarter* and able to handle more tasks programmatically. These readers can be any device, like PDA, Set top box, ATM, Intelligent terminals, POS, Kiosk Telephone, etc.

The platforms dose the exchange of information using XML standards and thus can be integrated with any platform supporting the XML standards for data interchange.

Layer 03 Edge Domain Figures 02

The Edge Domain includes the functionality of filtering and aggregating volumes of data that is provided by the readers, supporting the analysis of data and applying local decision-making and intelligence. The architecture of this platform is compatible with readers from multiple vendors and must hide individual reader (and also tag interface) idiosyncrasies effectively from the remainder of the infrastructure.

The Edge Domain functions are implemented at various locations across the globe and are typically in a low-cost appliance just upstream of the readers and uses embedded software technologies to establish a software stack on the outer edge of the RFID infrastructure. To enable manageability of a production RFID system, deployment aspects are another key aspect that must be addressed. Software updates in the Edge Domain can be deployed automatically and remotely all across the globe.

Layer 04 Premises Domain Figure 02

The Premises Domain is the intermediary between enterprise applications and the Edge Domain. The Premises Domain filters and aggregates, monitors and escalates RFID events to detect critical business operations, enabling programmatic decision-making. It also tracks and logs all-important information about products and locations and manages *downstream* components in other domains, such as readers or RFID controllers.

The Premises Domain deals with events *on a higher level* than the Edge Domain, that is events that are important in context of a business operation or process. This domain can store data, and it interacts with enterprise back-end systems through business process integration.

Business logic can be specific for premises' operational requirements. As an example, a distribution center for food can have different business logic from a distribution center for hardware, although they belong to the same retail chain and are part of the same hierarchy.

Layer 05 Enterprise and Business Applications Domain Figure 02

The Enterprise and Business Application Domain include the existing applications that requires information about product movement that is captured by the RFID infrastructure.

These applications correspond to an organization's unique mix of business requirements. This domain includes systems that help ordering, managing, or supplying goods and that can be enhanced greatly by being able to monitor product movement automatically. Examples of these types of systems are Process Automation, Inventory Management, Enterprise Resource Planning (ERP), Manufacturing Execution Systems, Supply Chain Execution Systems,

Warehouse Management Systems, Data Warehouse, Merchandise Management, Store Systems, Work In Process Manufacturing, and so on.

Layer 06 Object Directory Domain Figures 02

The Object Directory Domain provides information about the physical object that is using its unique ID as the lookup key. It enables the rapid retrieval of product information and also provides a framework for allowing companies to securely share product information with trading partners.

The EPCglobal Network consists of the following:

Object Naming Service (ONS)

Electronic Product Code Discovery Service (EPCDS)

Electronic Product Code Information Service (EPCIS)

The EPCglobal Network is one example of an Object Directory Domain.

This domain typically delivers three kinds of information about a product:

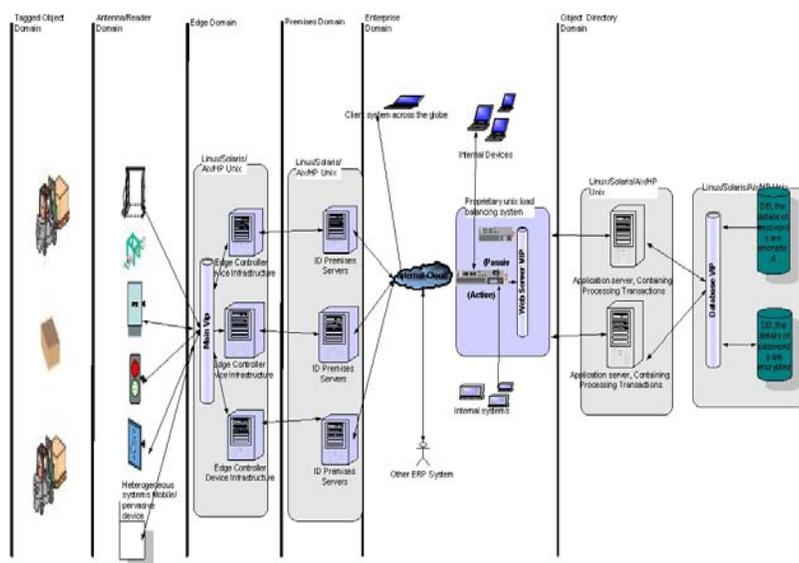
Core product information

Manufacturing time information

Life cycle history information

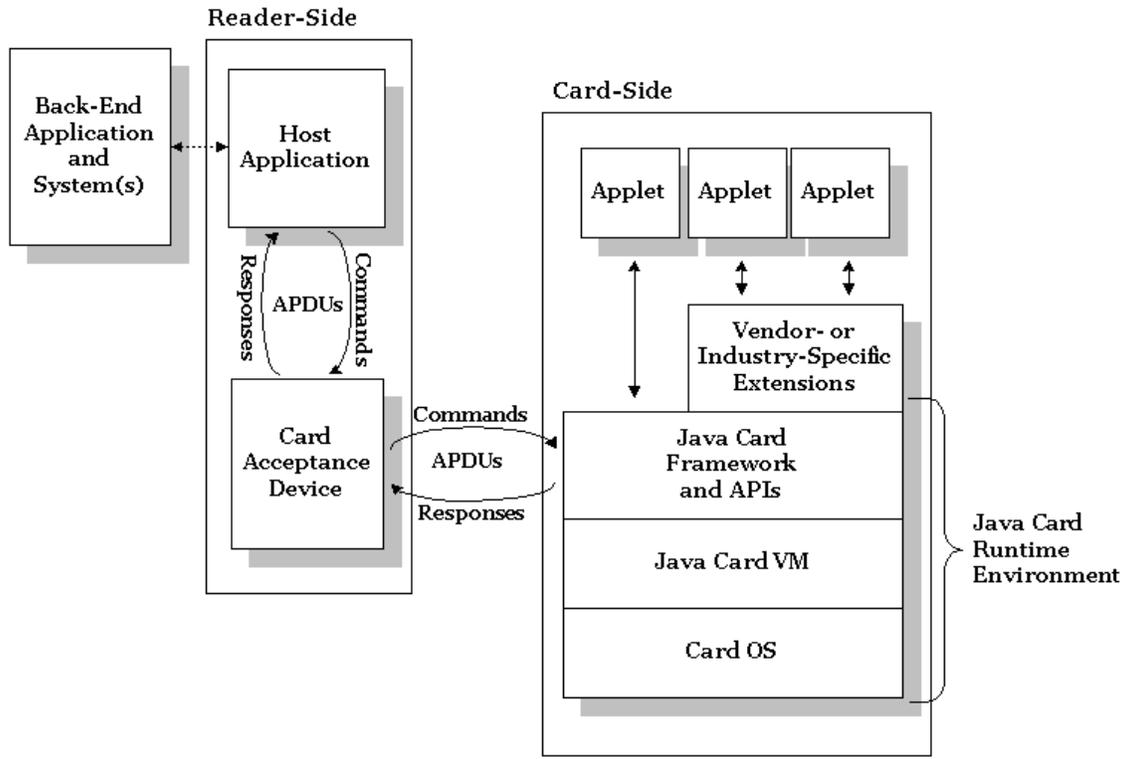
Core product information is usually obtained from product information managers, product catalogs, or data pools.

Figure 02 Global smart card Architecture



Project	Global Smart Cards
Group	Smart Card SECURITY
ROLE	Enterprise Architect JAVA CARD_DEF_OME
Developed by	Ruchr Choudhry Atok Prasad
Version	1.0.0.1.0.2

Figure 03 Global smart card Architecture internal



Inside the Global Card Virtual Machine

The Java Card Virtual Machine (JCVM) specification defines a subset of the Java programming language and a Java-compatible VM for smart cards, including binary data representations and file formats, and the JCVM instruction set.

The VM for the Java Card platform is implemented in two parts, with one part external to the card and the other running on the card itself. The on-card Java Card VM interprets bytecode, manages classes and objects, and so on. The external Java VM part is a development tool, typically referred to as the *Java Card Converter tool*, that loads, verifies, and further prepares the Java classes in a card applet for on-card execution. The output of the converter tool is a **Converted Applet (CAP)** file, a file that contains all the classes in a Java package in a loadable, executable binary representation. The converter verifies that the classes conform to the Java Card specification.

The JCVM supports only a restricted subset of the Java programming language, yet it preserves many of the familiar features including objects, inheritance, packages, dynamic object creation, virtual methods, interfaces, and exceptions. The JCVM specification drops the support for a number of language elements that would use too much of a smart card's limited memory:

Table 3. Summary of Java Card Language Limitations

Language Features	Dynamic class loading, security manager (java.lang.SecurityManager), threads, object cloning, and certain aspects of package access control are not supported.
Keywords	native, synchronized, transient, volatile, strictfp are not supported.
Types	There is no support for char, double, float, and long, or for multidimensional arrays. Support for int is optional.
Classes and Interfaces	The Java core API classes and interfaces (java.io, java.lang, java.util) are unsupported except for Object and Throwable, and most methods of Object and Throwable are not available.
Exceptions	Some Exception and Error subclasses are omitted because the exceptions and errors they encapsulate cannot arise in the Java Card platform.

There are also programming-model limitations. For instance a loaded library class can no longer be extended in the card; it is implicitly made final.

In line with the memory constraint the JCVM specification additionally defines constraints on many program attributes. Table 4 summarizes the JCVM resource constraints. Note that many of these constraints are typically transparent to Java Card developers.

Table 4. Summary of Java Card VM Constraints

Packages	A package can refer to up to 128 other packages
	A fully qualified package name is limited to 255 bytes. Note that the character size depends on the character encoding.
	A package can have up to 255 classes.
Classes	A class can directly or indirectly implement up to 15 interfaces.
	An interface can inherit from up to 14 interfaces.
	A package can have up to 256 static methods if it contains applets (an <i>applet package</i>), or 255 if it doesn't (a <i>library package</i>).
	A class can implement up to 128 public or protected instance methods, and up to 128 with package visibility.

In the Java Card VM, as in the J2SE VM, *class files* are central, but the JCVM specification defines two other file formats to further platform independence, the *Converted Applet (CAP)* and *Export* formats, which will be described in the second installment of this article, in the section "Developing a Java Card Application."

The Global Java Card API

The Java Card API specification defines a small subset of the traditional Java programming language API - even smaller than that of J2ME's CLDC. There is no support for Strings, or for multiple threads. There are no wrapper classes like Boolean and Integer, and no Class or System classes.

In addition to its small subset of the familiar Java core classes the Java Card Framework defines its own set of core classes specifically to support Java Card applications. These are contained in the following packages:

`java.io` defines one exception class, the base `IOException` class, to complete the RMI exception hierarchy. None of the other traditional `java.io` classes are included.

`java.lang` defines `Object` and `Throwable` classes that lack many of the methods of their J2SE counterparts. It also defines a number of exception classes: the `Exception` base class, various runtime exceptions, and `CardException`. None of the other traditional `java.lang` classes are included.

`java.rmi` defines the `Remote` interface and the `RemoteException` class. None of the traditional `java.rmi` classes are included. Support for Remote Method Invocation (RMI) is included to simplify migration to, and integration with, devices that use Java Card technology.

`javacard.framework` defines the interfaces, classes, and exceptions that compose the core Java Card Framework. It defines important concepts such as the Personal Identification Number (PIN), the Application Protocol Data Unit (APDU), the Java Card applet (`Applet`), the Java Card System (`JCSystem`), and a utility class. It also defines various ISO7816 constants and various Java Card-specific exceptions. Table 5 summarizes this package's contents:

Table 5. Java Card v2.2 javacard.framework

Interfaces	<p>ISO7816 defines constants related to ISO 7816-3 and ISO 7816-4.</p> <p>MultiSelectable identifies applets that can support concurrent selections.</p> <p>PIN represents a personal identification number used for security (authentication) purposes.</p> <p>Shareable identifies a shared object. Objects that must be available through the applet firewall must implement this interface.</p>
Classes	<p>AID defines an ISO7816-5-conforming Application Identifier associated with an application provider; a mandatory attribute of an applet.</p> <p>APDU defines an ISO7816-4-conforming Application Protocol Data Unit, which is the communication format used between the applet (on-card) and the host application (off-card).</p> <p>Applet defines a Java Card application. All applets must extend this abstract class.</p> <p>JCSystem provides methods to control the applet life-cycle, resource and transaction management, and inter-applet object sharing and object deletion.</p> <p>OwnerPIN is an implementation of the PIN interface.</p> <p>Util provides utility methods for manipulation of arrays and shorts, including arrayCompare(), arrayCopy(), arrayCopyNonAtomic(), arrayFillNonAtomic(), getShort(), makeShort(), setShort().</p>
Exceptions	<p>Various Java Card VM exception classes are defined: APDUException, CardException, CardRuntimeException, ISOException, PINException, SystemException, TransactionException, UserException.</p>

avacard.framework.service defines the interfaces, classes, and exceptions for *services*. A service processes incoming commands in the form of an APDU. Table 6 summarizes the framework service API:

Table 6. javacard.framework.service

Interfaces	<p>Service, the base service interface, defines the methods processCommand(), processDataIn(), and processDataOut().</p> <p>RemoteService is a generic Service that gives remote processes access to services on the card.</p> <p>SecurityService extends the Service base interface, and provides methods to query the current security status, including isAuthenticated(), isChannelSecure(), and isCommandSecure().</p>
Classes	<p>BasicService is a default implementation of a Service; it provides helper methods to handle APDUs and service collaboration.</p> <p>Dispatcher maintains a registry of services. Use a dispatcher if you want to delegate the processing of an APDU to several services. A dispatcher can process an APDU completely with the process() method, or dispatch it for processing by several services with the dispatch() method.</p>
Exceptions	<p>ServiceException a service-related exception.</p>

javacard.security defines the classes and interfaces for the Java Card security framework. The Java Card specification defines a robust security API that includes various types of private and public keys and algorithms, methods to compute cyclic redundancy checks (CRCs), message digests, and signatures:

Table 7. javacard.security

Interfaces	<p>Generic base interfaces Key, PrivateKey, PublicKey, and SecretKey, and subinterfaces that represent various types of security keys and algorithms: AESKey, DESKey, DSAKey, DSAPrivateKey, DSAPublicKey, ECKey, ECPrivateKey, ECPublicKey, RSAPrivateCrtKey, RSAPrivateKey, RSAPublicKey</p>
Classes	<p>Checksum: abstract base class for CRC algorithms</p> <p>KeyAgreement: base class for key-agreement algorithms</p>

KeyBuilder: key-object factory

KeyPair: a container to hold a pair of keys, one private, one public

MessageDigest: base class for hashing algorithms

RandomData: base class for random-number generators

Signature: base abstract class for signature algorithms

Exceptions CryptoException: encryption-related exceptions such as unsupported algorithm or uninitialized key.

javacardx.crypto is an extension package that defines the interface KeyEncryption and the class Cypher, each in its own package for easier export control. Use KeyEncryption to decrypt an input key used by encryption algorithms. Cypher is the base abstract class that all ciphers must implement.

javacardx.rmi is an extension package that defines the Java Card RMI classes. It defines two classes, CardRemoteObject and RMIService. CardRemoteObject defines two methods, export() and unexport(), to enable and disable remote access to an object from outside the card. RMIService extends BasicService and implements RemoteService to process RMI requests.

Conclusion

In order to achieve the vision of using global smart card technology to streamline various processes, security constrains, ease of use a role of this paper is to provide assistance to various open source community and smart card community in the implementation of smart card technologies for a wide range of purposes including personal identification, physical and logical access, digital signatures, travel, and small purchases. It s our prime objective to use the existing open source technology and use the Global Smart card Access Common ID contract, in reengineering their business processes to achieve streamlined operations and cost savings through enhanced operational efficiency.

We will continuously strive for this goal and keep updating this document every 6-month.

Source:

Name	Web Address	Description
Access Certificates for Electronic Services (ACES)	http://www.gsa.gov/aces	The ACES website provides information on the Government-wide public key infrastructure.

Identity, Credential and Access Management (ICAM)	http://www.idmanagement.gov/smartcard/information/smartcardhandbook.pdf	This site provides the core body of knowledge on Identity Credential and Access Management
Avanti	http://www.biometric.freereserve.co.uk/avanti.htm	This site provides background information about biometrics, their use in everyday business situations and how they are deployed.
Biometrics Consortium	http://www.biometrics.org	The Biometric Consortium serves as the US government's focal point for research, development, test, evaluation, and application of biometric-based personal identification/verification technology.
Card Europe	http://www.cardeurope.demon.co.uk/index.htm	Although primarily focused on Europe, Card Europe has expanded to encompass the whole world. The site provides access to a database of information as a starting point for information concerning smart card related products, services and activities.
CardTech/ SecurTech	http://www.ctst.com	CardTech/ SecurTech promotes the advancement of card, biometric and transaction security technologies through educational resources for professionals at

every level of expertise.

CommerceNet of Massachusetts, Information Technology Division Legal Department, The PKI Page E-Authentication	http://www.magnet.state.ma.us/itd/legal/backers.htm http://www.cio.gov/eauthentication/	This site provides background papers on PKI, cryptography, digital signatures and electronic commerce. The E-Authentication site promotes public trust in meeting the authentication business needs in E-Gov transactions.
Electronic Frontiers Georgia (EFGA)	http://www.efga.org	The EFGA web page provides information about emerging technology, with links to information about digital signatures and cryptography.
Federal Bridge Certificate Authority	http://www.cio.gov/fbca/	The FBCA site provides information relevant to an entity accepting certificates issued by another entity for a transaction.
Federal Identity and Credentialing Committee	http://www.cio.gov/ficc	The FICC site describes policy recommendations for the use of identity credentials in the federal sector.
Federal PKI Policy Authority	http://www.cio.gov/fpkipa/	The FPKIPA site describes the group's supervision of FBCA and its promotion of agency-to-agency PKI interoperability.

