Open Platform
Smart Card: Chip Operating System Platform

- Proprietary (native)
  - Non-portable across different vendors
  - Not open to outsiders
  - Single source

- Open
  - MultOS card
  - Java card
  - Windows for Smart Card
Non-portable Application on Proprietary Platform

Card A
- Electronic Purse
- Credit Application

Card B
- Access Control
- Credit Application

Proprietary Platform

Infineon Chip

STM Chip
Portable Application on Open Platform

Card A
- Electronic Purse
- Credit Application
- Loyalty Application

Card B
- Access Control
- Credit Application
- Loyalty Application

Open Platform
- Operating System

Infineon Chip

STM Chip
Mondex was conceived in 1990 and launched in Dec. 1993.
MultOS was invented in 1993 with chips from Philips and Hitachi. The platform was from Keycorp and Dai Nippon (enabled card manufacturers to just become embedders).
Visa Cash Disposable pilots in many events around the same period.
Embedders started to emerge, attracted by the plastic card printers that had high added value.
In 1996, MasterCard purchased 51% of MultOS, while Mondex became an applet.
Also in 1996, Java card was conceived, and naturally supported by Visa, Sun Microsystem and leading card manufacturers.
Open Platform: Card History

- Visa defined the Visa Open Platform (VOP) to complement the Java card.
- In 1998, GSM SIM ToolKit was made available. Java as an applet development became a logical choice.
- Visa renamed VOP to Global Platform in Oct. 1999, and opened it to become a vendor independent specification compatible with Window Smart Card and Java SIM ToolKit.
- Microsoft give-up and in May 2001, license the source code
- In 2008, Gemalto bought the MultOS business from Keycorp.
MultOS Card

- MULTI-application OS defined by MAOSCO
- Silicon using Philips and Hitachi, and later, Infineon
- Platform OS developed by Keycorp & DNP
- ITSEC E6-certified firewalls for card application segregation
MultOS Card

Diagram showing the components of MultOS Card:
- **Mondex purse**
- **EMV credit & debit**
- **Loyalty**
- **Access**

Below the components:
- **MAOS - Operating system**
- **Silicon**

On the right side:
- **Applications**
- **MULTOS**
- **Firewalls**
- **Silicon (hardware)**
- **Interpreter and API**
- **Application load certificate**
MultOS Card

- Uses a dedicated programming language, MultOS Executable Language (MEL)
- Can develop using C programming language, and then convert to MEL (eg. using Swiftcard compiler)
- Can also develop using Java compiled to MEL
- MEL is a virtual machine assembly language
Development Process

- Development on the PC using MEL assembly or C programming language
- Simulation and debugging on the PC
- Downloading onto the development card with the development card certificates from MOASCO
  - Weak application development card
  - Strong application development card
- Application for certificates from MULTOS CA to load into life cards
Code-let Loading: Full Flexibility

- Unprotected
- Integrity Protection
- Integrity & Secrecy Protection
- Loading can be unique to each card or a classification of cards
MultOS Card: Development Process

- MAOSCO
- MULTOS Application Toolkit Providers
- Application Developer
- Application Load Unit Generator
- Application Load Facility
- Certification Authority
1. License Agreement. Application Developer registers & purchases a license from MAOSCO, which allows them to develop MULTOS™ applications & tools.

2. Application Developer Toolkits. MAOSCO endorses an open market policy for application development tools.

3. Application Developer. MULTOS™ application development should follow a typical software development life-cycle: Requirements, Design, Code & Test. Applications may be coded directly in MEL or a high level language (e.g. ‘C’) and compiled into MEL.

4. Application Registration. As an Issuer, the bank wants the maximum flexibility & return on each card they issue. The card Issuer can sell space on its MULTOS™ card to companies, for instance the supermarket, recovering card costs. The MULTOS™ CA provides the cryptographic services for the MULTOS™ scheme and allows Issuers to register multiple applications independently by requesting Application Load Certificates (ALCs). To register an application, the Issuer requires application specific information from the application provider. Note: It is not necessary for any Application Code or Data to be sent to the MULTOS™ CA or Card Issuer to register.
5. Application Source Code. The supermarket will have its Loyalty application developed according to its business requirements. Once complete, the MEL code, data, ISO directory (DIR) and ISO File Control Information (FCI) should be packaged in an Application Load Unit (ALU). Either the Application Provider, Issuer, Bureau or even a third party can perform this.

6. Application Load Unit (ALU). ALU’s provide 2 security features:

- Protection for confidential Application Source Code & Data. The data is encrypted and is only decrypted once safely loaded onto the correct MULTOS Card.
- An ALU can be digitally signed, ensuring data integrity and protecting the ALU from corruption. This signature is checked by the MULTOS™ card during the application load process.
7. Application Load Certificate (ALC). An ALC allows the Issuer to control which applications are loaded onto its card base. It uniquely identifies the Card Issuer, Application and Issuer card(s) onto which the application is to be loaded. The ALC also contains information so that MULTOS™ can verify that the application loaded is genuine. The Issuer requests the ALC from the MULTOS™ CA and forwards them to the Application Load Facility. In the example, the supermarket requests permission from the bank to load their Loyalty application onto the bank’s cards. The bank requests the ALCs from the MULTOS CA and passes these to the supermarket.

8. Load MULTOS™ Application. The IFD commands used to load the ALU & ALC onto a MULTOS™ card. Application load certificates are used to enable the card to ensure that the application which is about to be loaded has been approved by the card issuer to be loaded onto the card. The certificate is generated by a Certification Authority (CA) based on information supplied by the application provider. The information passed to the CA is encrypted using the private key for the issuer or application provider. In order to check the validity of the certificate, the card needs the public key of whoever generated the ALC. This public key is included in the ALC. The card receives the ALC and decrypts it.
Java Card

Specification

JC Applet .java

Compilation

JC Applet .class

JC Conversion

JC Applet .cap

Load / Install

PC CARD

Java Card

Client / Host Application

Run !
Java Card Architecture

- Applets
  - Instance A1
  - Package A
  - Instance B1
  - Package B
  - Instance B2
  - Package C
  - Instance C1

- JCRE
  - Dispatcher
  - JavaCard Virtual Machine
  - JavaCard API
  - GlobalPlatform API
  - Proprietary API
  - Native methods

- BIOS
  - Hardware functions

- EEPROM

- ROM
Java Card Overview

- Java Card allows:
  - Having several independent applications on a single card
  - Adding new applications after the card has been issued
  - Source and binary compatibility of the applications

- Java Card defines:
  - A language
  - A standard binary format for the applets (bytecode)
  - A virtual machine, used to execute the applet bytecodes
  - An API, available for the applets
  - The applet selection mechanism
  - No file system
Java Card Applets

- Packages
  - Can group several applet classes
  - Corresponds to the data unit transmitted to the card when an applet is loaded

- Applet Classes
  - Contains the code of the applet

- Applet Instances
  - Contains the data of the applet
  - Allows several instances of the same applet class
AID: Application IDentifier

- Uniquely identifies applet classes, instances and packages

- RID: unique identifier of the application provider (provided by ISO; eg. A0 00 00 00 77)

- PIX: unique for a given RID, identifies the application (allocated by the application provider)
Java Card Runtime Environment

- **Java Card Virtual Machine**
  - Interprets (executes) the applet code (bytecode)
  - The bytecode is an abstract machine language, independent from any chip hardware

- **Java Card API**
  - Provides various services to the applets: APDU command management, cryptography, etc.

- **Dispatcher**
  - Responsible for the applet selection mechanism, and transfers the commands to the appropriate applet
Applet Selection

- The dispatcher receives all the APDU commands coming from the terminal.
- It processes the SELECT APPLET commands, and keeps track of the currently selected applet.
- The AID of the applet instance to select is given in the data field of the SELECT APPLET command.
- The Dispatcher transfers all the following commands to the currently selected applet.
- At power on, the default applet is selected.
Java Card Language Subset

- Supported Java features
  - Small primitive data types: boolean, byte and short
  - One-dimensional arrays
  - Java packages, classes, interfaces
  - Inheritance, virtual methods, overloading
  - Dynamic object creation
  - Class members access scope
  - Exception handling
  - Garbage collection (JC 2.2)

- Unsupported Java features
  - Large primitive data types: int (optional), long, double and float
  - Characters and strings
  - Multidimensional arrays
  - Threads
  - Dynamic class loading
  - Sandbox model and security manager
  - Introspection
  - Object serialization
  - Object cloning
Java Card API

- The standard Java API is inappropriate for smart cards.
- Java Card includes minimum standard Java classes:
  - The *Object class*
  - The *Throwable* class and some basic exception classes
  - No other standard Java package is available
- Java Card provides three packages dedicated for smart cards
  - javacard.framework
  - javacard.security
  - javacardx.crypto
GlobalPlatform Overview

- Previously called Visa Open Platform

- GlobalPlatform allows
  - Communicating with the card in a secure way
  - Applet loading, installation and deletion
  - Defining specific security environments for each application provider

- GlobalPlatform defines
  - The Card Manager application
  - The Security Domain applications
  - A common behavior for all other applications
  - An API
Card Manager

- The card manager is an applet.
- It is also called the Issuer Security Domain.
- It is the first applet to be installed on the card.
- Usually, default is selected when the card is powered up.
- It is the entry point for managing the content of the card.
- Its APDU commands are specified by GP.
GP APDU Commands

- Delete
- Get Data
- Get Status
- Install
- Load
- Manage Channel
- Put Key
- Select
- Store Data
- Initialize Update
- External Authenticate
- Begin R-MAC Session
- End R-MAC Session
- Get Challenge
- Internal Authenticate
- Manage Security Environment
- Perform Security Operation
Applications Life Cycle

INSTALLED

SELECTABLE

Application
Specific States

LOCKED

LOCKED

LOCKED
Secure Messaging

- Secure communication channel between card and host
- Required for all sensitive operations with the Card Manager
- A mechanism that can be taken advantage of by any application
- Several Secure Channel Protocols (defined at pre-perso time)
  - SCP01 mode 5: compatibility with OP2.0.1’
  - SCP02 mode 55: improved security
- Three security levels (chosen when the SC is opened)
  - Mutual authentication
  - Mutual authentication and integrity checking
  - Mutual authentication, integrity checking and confidentiality
Key Sets

- The Card Manager and Security Domains can hold several key sets.
- Each key set is identified by a version number.
- Knowledge of a key set is required to open a secure channel.
- A key set is composed of three keys:
  - Encryption/Authentication Key
  - MAC Key
  - Key Encryption Key
Secure Channel Opening

Host challenge generation

- INITIALIZE UPDATE (host challenge)

Response (card challenge, card cryptogram, status)

- Card challenge generation
- Session keys computation
- Card cryptogram computation
- Card cryptogram verification
- Host cryptogram computation

EXTERNAL AUTHENTICATE (host cryptogram)

Response (status)

- Host cryptogram computation
- Host cryptogram verification

Card

Host
Package Loading

- INSTALL for LOAD command
  - AID of the package to load
  - AID of the Security Domain to associate with the package (or the Card Manager AID if not applicable)

- LOAD command
  - Sends a portion of the load file to the card
  - The load file contains the CAP file, encapsulated in a TLV
  - The load file may also contain DAP blocks if applicable
Application Installation

- INSTALL for INSTALL command
  - AID of the package containing the applet
  - AID of the applet class within the package
  - AID that must be assigned to the instance
  - Application privileges
  - Install parameters

- Usually combined with INSTALL for the MAKE SELECTABLE command
Security Domains

- A Security Domain is an applet.
- It represents an application provider.
- It maintains its own key sets.
- Security Domains may have different privileges.
  - Simple Security Domain
  - DAP (Data Authentication Pattern) and mandated DAP
  - Security Domain with delegated management
Simple Security Domain

- It provides new key sets to the application that takes advantage of the secure messaging services.
- These key sets do not give the right to modify the card content, as the Card Manager key sets do.
- When a package is loaded, it is associated to an SD and all applets from this package will be linked to it.
- An applet can be extradited, in order to change the link and associate it with another SD.
DAP Verification

- **Goal**
  - Checking the integrity of the package code
  - Checking that the package comes from an authorized entity (application provider)
- Such Security Domains require one more key: the DAP key (RSA public key).
- The DAP is a signature of the CAP file, using the DAP key.
- When loading the package, the Card Manager queries the SD to check the DAP value.
- DAP is optional (unless the package must be associated with the SD), but with mandated DAP, it becomes mandatory.
Card-related Application Design: using Open Platform COS

- Knowledge on smart card application security requirements is a pre-requisite.
- Design the application APDU command set, application data and keys.
- Design the SAM APDU command set, application data and keys.
- Design the SAM-Application-Card APDU transaction flow for all subsystems.
- Provide test cards and test SAMs to each subsystem vendor, so that each subsystem vendor knows how to use the card and the SAM.
Questions?