Java Card Platform Evolution

Florian Tournier,
Director, Product Management,
Internet Of Things Cloud Service

Saqib Ahmad
Consulting Member of Technical Staff,
Java Card Engineering,
Internet Of Things Cloud Service

October 26, 2015

Safe Harbor Statement

Program Agenda

1. Introduction to Java Card
2. Drivers for 3.0.5 Release
3. 3.0.5 Framework Update
4. 3.0.5 Security Update
5. 3.0.5 Enhanced Cryptography
6. Looking Forward

Java Card

Java Technology for Smart Cards and Secure Elements

ORACLE
JAVA CARD

Key Business Features/Benefits

- Strong security for the development and deployment of trusted, efficient, easy-to-use identity services
- Proven, mature global standard with a strong growing market demand
- Vibrant ecosystem of 3B+ Java Card devices deployed each year

Java Card Devices

- SIM Cards, Payment Cards
- ID Cards, ePassport, Transportation Cards
- Embedded Secure Elements for NFC

Use Cases and Vertical Segments

- **Telecom** : Network authentication, mobile security, mobile wallet
- **Government** : National/Military/Administration/Healthcare eID, ePassport, driving license…
- **Financial Services** : Contact / contactless payment
- **NFC** : Secure elements for mobile payment
- **M2M** : Smart Meter, Telehealth, Automotive, …

Java Card Benefits

- Object Oriented Programming
- Secure Programming Platform
- Hardware Independent
- Operating System Independent
- Multi-Application Support
- Secure Applet Loading
- Open Standard
- 50+ licensees, 10+ billion deployed

From the News

EAL5+ Certification for Oracle
After several years of efforts, Oracle got awarded an EAL5+ certification for the Java Card platform developed for Infineon.

Java Card Classic 3.0.5
A new release of the Java Card Classic specification in June, 2015. Work on that release to continue for an update of the Java Card Protection Profile

New Applications for Java Card
In Mobile payment schemes (Android/iOS), in wearables, eID, in NFC SIM cards, and many more.

IoT as a new target
Java Card vendors positioning themselves as « digital security » specialists – acquiring / developing M2M & IOT Platforms.

US Banks moving to smart card
EMV migration & emergence of contactless payment creating opportunity for Java Card

JavaOne ORACLE

Oracle Java Card Offering

Platform, Implementations, Programs

Platform	Implementations & Tools	Programs
• Two Oracle Java Card Editions • Java Card Connected • Java Card Classic • Specifications, Test Suites (TCK), Common Criteria Protection Profiles	• Reference Implementation (commercial source) • Developer Tools (free SDK)	• Support Programs for Source Licensees • Java Card S

* Not Generally available

Program Agenda

1 Introduction to Java Card

2 Drivers for 3.0.5 Release

3 3.0.5 Framework Update

4 3.0.5 Security Update

5 3.0.5 Enhanced Cryptography

6 Looking Forward

Java Card 3.0.5 Core Focus : Security Interoperability

- Security certification of Java Card products remains a challenge
  - Highly time- and resource-consuming
  - Difficult to make secure applications portable between platforms
- Security interoperability aims at simplifying this
  - Defining specific APIs, under the responsibility of the platforms
  - The applications only have to use these APIs properly

Security interoperability features introduced in 3.0.5

- Integrity of arrays
  - Verified at system level
  - Application can trigger verification
- Enhanced exchanges with APIs
  - Checking the output of system calls
  - Essential on some operations like signature verifications

Java Card 3.0.5 : additional features

- Spec Updates
  - Crypto updates : adhere to the latest crypto standards
    - Better support of key agreement, including Diffie-Hellman
    - Support for new crypto algorithms, key formats
    - Improvements in the crypto framework
  - One-to-many biometrics
  - Minor standards-related changes
    - Alignment with contactless schemes
    - Alignment with GlobalPlatform, ETSI

- Tools / SDK improvements
  - More security in the Reference Implementation
    - Making the Virtual Machine more defensive
    - To encourage licensees to raise the security level of their own implementations
  - Moving the SDK to Eclipse
    - Providing better support for Java Card Classic
  - Later: Revising the Java Card Protection Profile
    - Taking in consideration the new security APIs

Program Agenda

1. Introduction to Java Card
2. Drivers for 3.0.5 Release
3. 3.0.5 Framework Update
4. 3.0.5 Security Update
5. 3.0.5 Enhanced Cryptography
6. Looking Forward

Updates to javacard.framework.APDU

- HCI support
  - Added new constant PROTOCOL_MEDIA_HCI_APDU_GATE to support APDU over HCI defined for the APDU gate in ETSI TS 102 622
- Media type F support for FeliCa
  - Added new constant PROTOCOL_MEDIA_CONTACTLESS_TYPE_F JIS X 6319-4:2010 transport protocol Type F
- These new constants help in identifying the source of the incoming APDU to take appropriate action.

PIN API Enhancement

- New interface: OwnerPINx
  - OwnerPINx allows for updating the PIN, try-limit and try-counter
- New interface: OwnerPINxWithPredecrament
  - OwnerPINxWithPredecrament extends the OwnerPINx functionality by supporting decrementing of the try-counter before any PIN validation attempts
- New class: OwnerPINBuilder
  - Factory class to build OwnerPIN objects which can be of type
    - OwnerPIN
    - OwnerPINx
    - OwnerPINxWithPredecrament

New Utility Class: javacardx.apdu.util.APDUUtil

- APDUUtil class provides utility functions to extract information from CLA byte of the incoming command APDU.
  - static byte getCLAChannel(byte CLAByte)
  - static boolean isCommandChainingCLA (byte CLAByte)
  - static boolean isISOInterindustryCLA(byte CLAByte)
  - static boolean isSecureMessagingCLA(byte CLAByte)
  - static boolean isValidCLA(byte CLAByte)

Extended Biometry Support: Biometry 1:N

- Motive: Allow applications to store multiple templates to be matched against.
  - If N=1, then the package provides the same functionality as the existing javacardx.biometry package.
- Added a new extension package javacardx.biometry1toN with the following classes/ Interfaces:
  - Interface BioMatcher
  - Interface BioTemplateData
  - Interface OwnerBioMatcher
  - Interface OwnerBioTemplateData
  - Interface SharedBioMatcher
  - Interface SharedBioTemplateData
  - Class Bio1toNBuilder
  - Exception Bio1toNException

Program Agenda

1 Introduction to Java Card

2 Drivers for 3.0.5 Release

3 3.0.5 Framework Update

4 3.0.5 Security Update

5 3.0.5 Enhanced Cryptography

6 Looking Forward

Diffie-Hellman Modular Exponentiation

- New API added to support DH Modular Exponentiation
- New interface DHKey
- New interface DHPrivateKey
- New interface DHPublicKey
- Support for DH keys in javacard.security.KeyBuilder class

Domain Data Conservation

- Motive: Domain/Curve data is common and is therefore duplicated in Elliptic Curve/Diffie-Hellman/DSA Keys. Allow to share domain data to conserve NVM footprint.
- API
  - New constants in javacard.security.KeyBuilder
    - ALG_TYPE_DH_PARAMETERS, ALG_TYPE_DSA_PARAMETERS, ALG_TYPE_EC_F2M_PARAMETERS, ALG_TYPE_EC_FP_PARAMETERS
  - New method in javacard.security.KeyBuilder
    - public static Key buildKeyWithSharedDomain(byte algorithmicKeyType, byte keyMemoryType, Key domainParameters, boolean keyEncryption) throws CryptoException

Support for new algorithms and key lengths

- Support for SHA-3
- Support for Optimal Asymmetric Encryption Padding (OAEP)
- Support for RSA 3K
- New Constants in class RandomData (old ones are deprecated)
  - ALG_FAST
  - ALG_KEYGENERATION
  - ALG_PRESEEDED_DRBG
- Support for AES CMAC algorithm
- Support for plain ECDSA
- New constant ALG_AES_CTR to support using AES in counter mode

Support for Application-Level Countermeasure Implementation

- Motive: Ease of implementation of countermeasures for application developers while also helping with security certification for applications
- Introduction of Class SensitiveArrays
- Introduction of Class SensitiveResult

Sensitive Arrays

- Motive: Applications are required to perform countermeasures on some arrays that contain sensitive information.
- SensitiveArrays API allows the application to create an array the integrity of which is managed by the platform.
- API for creating sensitive arrays
  - static Object makeIntegritySensitiveArray(byte type, byte memory, short length)
    - Return value must be cast to an array
  - static boolean isIntegritySensitiveArraysSupported()
  - static boolean isIntegritySensitive(Object obj)
  - static void assertIntegrity(Object obj)

Class SensitiveResult

- Motive: Provide the applet developer an API to double-check the result of an operation performed by the API.
- Used by all of the API functions considered sensitive and vulnerable to attack
- Applet developer's responsibility to invoke the API to confirm absence of an attack

SensitiveResult Class

- void assertEquals(Object obj)
- void assertEquals(short val)
- void assertFalse()
- void assertGreaterThan(short val)
- void assertLessThan(short val)
- void assertNegative()
- void assertPositive()
- void assertTrue()
- void assertZero()
- void reset()

Program Agenda

1 Introduction to Java Card

2 Drivers for 3.0.5 Release

3 3.0.5 Framework Update

4 3.0.5 Security Update

5 3.0.5 Enhanced Cryptography

6 Looking Forward

One-shot Classes

- Motive: Allow applications to use one-time quick cryptography avoiding NVM writes
- One-Shot objects are temporary JCRE entry point objects.
- All Java Card platforms are required to support at least one one-shot instance.
- New Classes:
  - Cipher.OneShot
  - Signature.OneShot
  - InitializedMessageDigest.OneShot
  - MessageDigest.OneShot
  - RandomData.OneShot

Signature.OneShot Example

```
Signature.OneShot sig = null;
 try {
    sig = Signature.OneShot.open(MessageDigest.ALG_SHA, Signature.SIG_CIPHER_RSA, Cipher.PAD_PKCS1);
    sig.init(someRSAKey, Signature.MODE_SIGN);
    sig.sign(someInData, (short) 0, (short) someInData.length, sigData, (short) 0);
 } catch (CryptoException ce) {
    // Handle exception
 } finally {
    if (sig != null) {
       sig.close();
       sig = null;
    }
 }
```

AEAD Cipher Support

- New Class added: javacardx.crypto.AEADCipher class
  - Abstract base class for Authenticated Encryption with Associated Data (AEAD) ciphers.
- Support for only AES
- Support for only CCM (Counter with CBC-MAC) and GCM (Galois/Counter Mode) modes

Program Agenda

1 Introduction to Java Card

2 Drivers for 3.0.5 Release

3 3.0.5 API Update: Framework

4 3.0.5 API Update: Security

5 3.0.5 Enhanced Cryptography

6 Looking Forward

Market Evolution for Java Card

Latest trends

SIM integration
Mobile Security

IoT Security

SIM Integration and Mobile Security

eUICC and beyond

- Increasing interest for eUICC
- Including solutions without distinct SE
  - From new security startups & established vendors

Mixed security for payment

- Started with Apple Pay
  - Some credentials on the eSE
  - Some other credentials on the TEE
- Similar models adopted elsewhere

TEE

eUICC

eSE

Boundaries between AP, TEE, SE become blurry

IOT Security

Endpoint devices are the main issue

- Deployed everywhere, accessible
- Price-sensitive
- Security still not a clear priority

Three major issues to address

- Device management
- Device integrity
- Device asset protection

IoT

Moving Market – Standardization is less important than ability to be agile & prototype

Java Card Positioning Rationale

Shifting ecosystems

- From Smart Cards…
  - Fully integrated
  - Serving one customer
- … to Digital Security
  - Embedded in other Hardware
  - Credential management
  - Multiple stakeholders
  - Different implementations

Shifting value

- From enablement …
  - Interoperability
  - Multiple applications
- … to Business Efficiency
  - Easing certification
  - On multiple platforms

Java Card – beyond Card

Potential Features

Feature	Details
RAM-based memory model	• More RAM is available to store “transient” objects • Garbage collection is mandatory, efficient on small memory
Alternative persistence and life cycle	• Persistent objects stored in secondary memory, need to be read • Atomicity is different, a synchronization is required
Mainstream Java APIs (Streams, NIO, …)	• Made possible by additional RAM • Radically changes the programming model at limited cost
Enhanced bytecode verification	• Split VM model does not need to be universal • Bytecode should be verified at least during/after loading
Multiple stakeholders	• A Java Card platform must support several top stakeholders • Mandatory to support several generic SIM profiles

Java Card Evolution

Next steps

- Validate Technical Requirements with Industry
  - Java Card Forum
  - End-users / technology adopters
- Seed the market with existing Java Card technology
  - Can be applied to many new form factors, even with some limitations
- Drive Architecture and Concepts for strong IoT security

Additional Resources

- Java Card on OTN
  http://www.oracle.com/technetwork/java/embedded/javacard/overview/index.html
  - Specifications and SDK download

- Oracle Java Card page
  http://www.oracle.com/us/technologies/java/embedded/card/overview/index.html
  - Commercial information

Integrated Cloud

Applications & Platform Services

JavaOne™

ORACLE®

ORACLE®