Programming Smart Cards

Sameh Elnikety 2002
Rajnish Kumar 2001
Comp527

Smart Cards
- Cyberflex Access Developer 32k
- Built-in micro-processor and memory
- Applications
  - Pre-paid calling card
  - Security needs
  - e-cash

Developing an Applet

1. Compile a class file from the java file and fill it with bytecodes.
2. Create an .jar file from the .class file (Program File Generator).
3. Download the .jar file onto the card as a load file. (Smart Card Toolkit or Install and Load commands)
4. Install an applet instance from the load file. (Smart Card Toolkit or Install command)

Outline
1. Java Card Applet Development
2. Java Card Applet Installation
3. Cryptography Support
4. Writing Terminal Application
javacard.framework

- Each applet extends javacard.framework.Applet
- It has useful stuff (e.g. Pin, Signature)

Applet

```java
package com.slb.javacard.SimplePurse;

import javacard.framework.*;
import javacard.security.*;

public class Wallet extends javacard.framework.Applet {
}
```

Methods

- `install()` : instantiate applet object
- `select()` : prepare applet for execution
- `process()` : switch statement
Beware !!

- Card resource limitations
  - No double, float
  - Limited heap and stack size

- Operation time out

- Transactions for handshaking

Command format (APDU)

<table>
<thead>
<tr>
<th>From Terminal</th>
<th>CLA</th>
<th>INS</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>From Card, To Terminal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SW1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SW2</td>
</tr>
</tbody>
</table>

- CLA : Command Class ( 1byte )
  - 00h for loader class ( Card specific ).
  - 03h for Wallet ( User defined )

Command format

<table>
<thead>
<tr>
<th>From Terminal</th>
<th>CLA</th>
<th>INS</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>From Card, To Terminal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SW1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SW2</td>
</tr>
</tbody>
</table>

- INS : Command Instruction Identifier ( 1b )
  - CLA + INS uniquely identifies the command

Command format

<table>
<thead>
<tr>
<th>From Terminal</th>
<th>CLA</th>
<th>INS</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>From Card, To Terminal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SW1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SW2</td>
</tr>
</tbody>
</table>

- P1,P2 : Command parameters
- P3 : number of bytes of data to follow, or expected by the terminal
- SW1,SW2 : status words.
Code Walk

- Sorry, small print
- Just to get the idea
```java
public void process(ASN1Input) {
    // Process the input data...
    byte[] buffer = getBuffer();
    // Process the buffer...
}
```

```java
private void getBalance(ASN1Input) {
    byte[] buffer = getBuffer();
    // Get the balance...
    balance = buffer;  // Assume getBuffer returns the balance
    // Process the balance...
}
```
2- Java Card Applet Installation

```java
private void validateAPDU(APDU apdu) {
    byte[] buffer = apdu.getBuffer();
    // process the APDU data which requires to be validated
    // for security purposes, some steps might be done as part of the APDU
    byte[] bufferA = (byte[]) apdu.getReceivedBuffer();
    // validate the interface and set the validation flag in the user interface
    // if validation is true, the interface will be enabled
    // if validation is false, a warning message could be displayed
    if (bufferA[0] == buffer[0]) {
        // continue
        // If bufferA[1] == buffer[1] (or similar checks)
        // exception: exception occurs: incorrect buffer[1] (or similar)
        // exception occurs: exception occurs (or similar)
        // exception occurs: exception occurs (or similar)
    }
    private void Exchange(APDU apdu) {
        byte[] buffer1 = apdu.getBuffer();
        byte[] buffer2 = apdu.getBuffer();
        byte[] buffer3 = apdu.getBuffer();
        byte[] buffer4 = apdu.getBuffer();
    }
```
Homework
- Tutorial from Cyberflex Access Cards Programmer’s Guide
  - Page 139 till 180

3- Cryptography Support

Crypto Libraries
- Javacard.security
- Javacardx.crypto
- See exception in page 181 and 182

Cryptography Support
- Symmetric/Asymmetric Authentication
- Internal/External Authentication
- Key Files
- Supported Encryption Algorithms
  - DES
  - 3-DES
  - RSA
Javacardx.crypto

- Classes
  - DES_Key
  - DES3_Key
  - RSA_PrivateKey
  - RSA_PublicKey
  - MessageDigest

En/Decryption

- DES_Key class methods
  - encryptECB, encryptCBC
  - decryptECB, decryptCBC
  - generateMAC, verifyMAC
  - setKey
  - getBlockSize

4- Writing Terminal Application

Terminal Application

- Use _slb.iop.* classes to write terminal application in Java
- Example method:
  - SmartCard.sendCardAPDU( CLA, INS, P1, P2, dataArray, Mode )

- Use transactions
Reference
- SDK Guide
- Cyberflex Access Programmer’s Guide
- Search google for Java cryptography
- www.cyberflex.com/Support/support.html

Get Done
- Install Reader and Software
- Do the tutorial
- Play around with wallet.java
- Write coke machine code
- Add Crypto

Cooperate!
- Start early, small step at a time
- You WILL have problems
- Post to the newsgroup specific questions
- Answer them!