

Programming Smart Cards

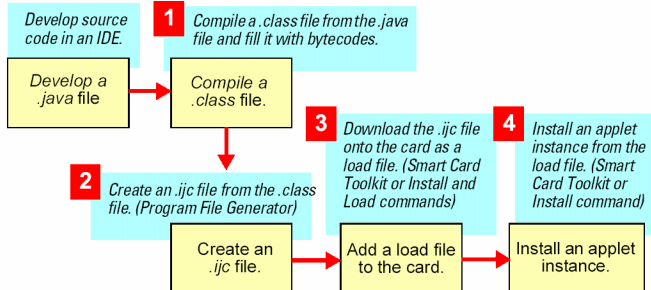
Sameh Elnikety 2002
Rajnish Kumar 2001
Comp527

Smart Cards

- Cyberflex Access Developer 32k
- Built-in micro-processor and memory

- Applications
 - Pre-paid calling card
 - Security needs
 - e-cash

Developing an Applet



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

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

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

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



Applet

```
{
    private Wallet()
    public static void install()
    public boolean select()
    public void process(APDU apdu)
}
```



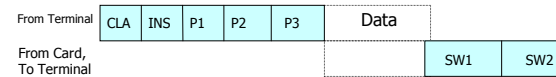
Methods

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

Beware !!

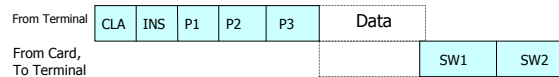
- Card resource limitations
 - No double, float
 - Limited heap and stack size
- Operation time out
- Transactions for handshaking

Command format (APDU)



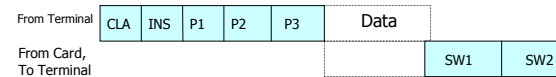
- CLA : Command Class (1byte)
 - 00h for loader class (Card specific).
 - 03h for Wallet (User defined)

Command format



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

Command format



- 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

```
package com.sib.javacard.SimplePurse;
import javacard.framework.*;
import javacard.security.*;

public class Wallet extends javacard.framework.Applet {

    /* constants declaration */
    // code of CLA byte in the command APDU header
    final static byte Wallet_CLA = (byte)0x80;

    // codes of INS byte in the command APDU header
    final static byte Deposit = (byte) 0x10;
    final static byte Debit = (byte) 0x20;
    final static byte Balance = (byte) 0x30;
    final static byte Validate = (byte) 0x40;
    final static byte Encrypt = (byte) 0x50;
    final static byte PinChange = (byte) 0x60;

    // maximum number of incorrect tries before the PIN is blocked
    final static byte PinTryLimit = (byte)0x03;

    // maximum size PIN
    final static byte MaxPinSize = (byte)0x04;

    // status word (SW1-SW2) to signal that the balance becomes negative;
    final static short SW_NEGATIVE_BALANCE = (short) 0x6910;
    final static short SW_WRONG_PIN = (short) 0x6300;
}
```

```
/* instance variables declaration */
OwnerPIN pin;
static byte base_balance = (byte) 0x20;
byte balance;
DESKey key;
Signature signature;

private Wallet(byte buffer[], short offset, byte length) {
    // It is good programming practice to allocate
    // all the memory that an applet needs during its
    // lifetime inside the constructor
    pin = new OwnerPIN(PinTryLimit, MaxPinSize);
    balance = (byte) 0;
    key = (DESKey)KeyBuilder.buildKey(KeyBuilder.TYPE_DES_TRANSIENT_RESET, KeyBuilder.LENGTH_DES, false);
    signature = Signature.getInstance(Signature.ALG_DES_MACG_ISO9797_M1, false);
    signature.init(key, Signature.MODE_SIGN);
    if (buffer[offset] == (byte)0) {
        register();
    }
    else {
        register(buffer, (short)(offset+1), (byte)(buffer[offset]));
    }
} // end of the constructor
```

```
public static void install(byte buffer[], short offset, byte length){
    // create a Wallet applet instance (card) *!*
    new Wallet(buffer, offset, length);

    // create a Wallet applet instance (Simulator) *!*
    userApplet = new Wallet();
} // end of install method

public boolean select() {
    // reset validation flag in the PIN object to false
    pin.reset();
    // returns true to JCRC to indicate that the applet
    // is ready to accept incoming APDUs.
    return true;
} // end of select method
```

```

public void process(APDU apdu) {
    // APDU buffer
    // APDU object carries a byte array (buffer) to
    // transfer incoming and outgoing APDU header
    // and data bytes between card and CAD
    byte buffer[] = apdu.getBuffer();

    // Implement a select handler
    if (selectingApplet()) {
        ISOException.throwIt(ISO7816.SW_NO_ERROR);
    }

    // verify that if the applet can accept this
    // APDU message
    if (buffer[ISO7816.OFFSET_CLA] != Vallet.CLA)
        ISOException.throwIt(ISO7816.SW_CLA_NOT_SUPPORTED);

    byte ins = buffer[ISO7816.OFFSET_INS];
    if (ins == Balance) getBalance(apdu);
    else if (ins == Debit) debit(apdu);
    else if (ins == Deposit) deposit(apdu);
    else if (ins == Validate) validate(apdu);
    else if (ins == Encrypt) encrypt(apdu);
    else if (ins == PinChange) PinChange(apdu);
    else ISOException.throwIt(ISO7816.SW_INS_NOT_SUPPORTED);
} // end of process method

```

```

private void deposit(APDU apdu) {
    byte buffer[] = apdu.getBuffer();

    // access authentication
    if (! pin.isValidated())
        ISOException.throwIt(ISO7816.SW_SECURITY_STATUS_NOT_SATISFIED);

    // indicate that this APDU has incoming data and
    // receive data starting from the offset
    // ISO7816.OFFSET_CDATA
    byte bytesRead = (byte)(apdu.setIncomingAndReceive());

    // it is an error if the number of data bytes read does not
    // match the number in Lc byte
    if (bytesRead != (byte)1)
        ISOException.throwIt(ISO7816.SW_WRONG_LENGTH);

    // increase the balance by the amount specified in the
    // data field of the command APDU
    balance = (byte)(balance + buffer[ISO7816.OFFSET_CDATA]);

    // return successfully
    return;
} // end of deposit method

```

```

private void debit(APDU apdu) {
    byte buffer[] = apdu.getBuffer();

    // access authentication
    if (! pin.isValidated())
        ISOException.throwIt(ISO7816.SW_SECURITY_STATUS_NOT_SATISFIED);

    byte bytesRead = (byte)(apdu.setIncomingAndReceive());

    if (bytesRead != (byte)1)
        ISOException.throwIt(ISO7816.SW_WRONG_LENGTH);

    // balance can not be negative
    if ((byte)( (byte) balance - (byte) buffer[ISO7816.OFFSET_CDATA]) < (byte) 0 )
        ISOException.throwIt(SW_NEGATIVE_BALANCE);

    balance = (byte) (balance - buffer[ISO7816.OFFSET_CDATA]);
} // end of debit method

```

```

private void getBalance(APDU apdu) {
    byte buffer[] = apdu.getBuffer();

    // access authentication
    if (! pin.isValidated())
        ISOException.throwIt(ISO7816.SW_SECURITY_STATUS_NOT_SATISFIED);

    // inform system that the applet has finished processing
    // the command and the system should now prepare to
    // construct a response APDU which contains data field
    apdu.setOutgoing();

    // indicate the number of bytes in the data field
    apdu.setOutgoingLength((byte)1);

    // move the data into the APDU buffer starting at offset 0
    buffer[0] = (byte) balance;

    // send 1 byte of data at offset 0 in the APDU buffer
    apdu.sendBytes((short)0, (short)1);
} // end of getBalance method

```

```

private void validate(APDU apdu) {
    byte buffer[] = apdu.getBuffer();
    // retrieve the PIN data which requires to be valid ated
    // the user interface data is stored in the data field of the APDU
    byte byteRead = (byte)apdu.setIncomingAndReceive();
    // validate user interface and set the validation flag in the user interface
    // object to be true if the validation succeeds
    // if user interface validation fails, PinException would be
    // thrown from pin check() method
    if (!pin.check(buffer, ISO7816.OFFSET_CDATA, byteRead))
        ISOException.throwIt(SW_WRONG_PIN);
    key.setKey(buffer, (short)0);
} // end of validate method

```

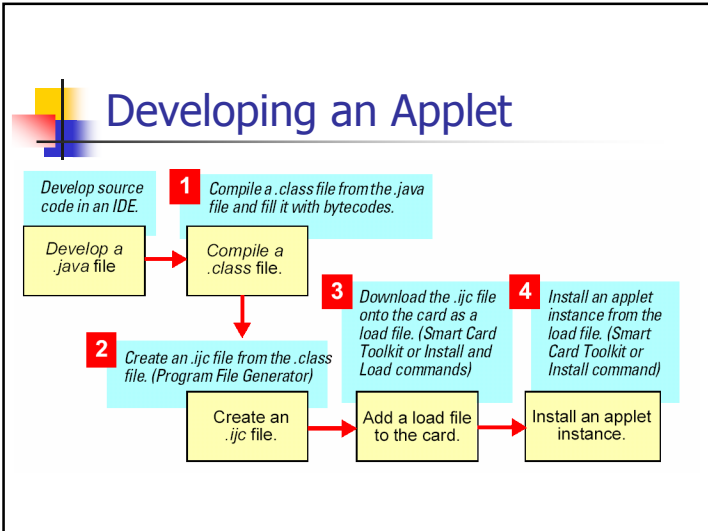
```

private void encrypt(APDU apdu) {
    byte buffer[] = apdu.getBuffer();
    // get 4 bytes of data and return the encrypted result.
    // access authentication
    if (!pin.isValidated())
        ISOException.throwIt(ISO7816.SW_SECURITY_STATUS_NOT_SATISFIED);
    short byteRead = apdu.setIncomingAndReceive();
    signature.sign(buffer, (short)5, byteRead, buffer, (short)0);
    apdu.setOutgoingAndSend((short)0, (short)8);
}

private void PinChange(APDU apdu) {
    byte buffer[] = apdu.getBuffer();
    byte newPIN[] = {(byte)buffer[ISO7816.OFFSET_P1], (byte)buffer[ISO7816.OFFSET_P1], (byte)buffer[ISO7816.OFFSET_P1], (byte)buffer[ISO7816.OFFSET_P1]};
    pin.update(newPIN, (short)0, (byte)4);
}

```

2- Java Card Applet Installation





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!