Colluding apps

Java cards:

Secure interactions in Java cards

Java cards

- Smart cards: embedded systems that allow to store and process information

- Typical Aplications: Credit cards, Electronic cash, Loalty systems, Helthcare, Government identification ....

- Java cards:
  Java Virtual machine / applications (applets) are portable

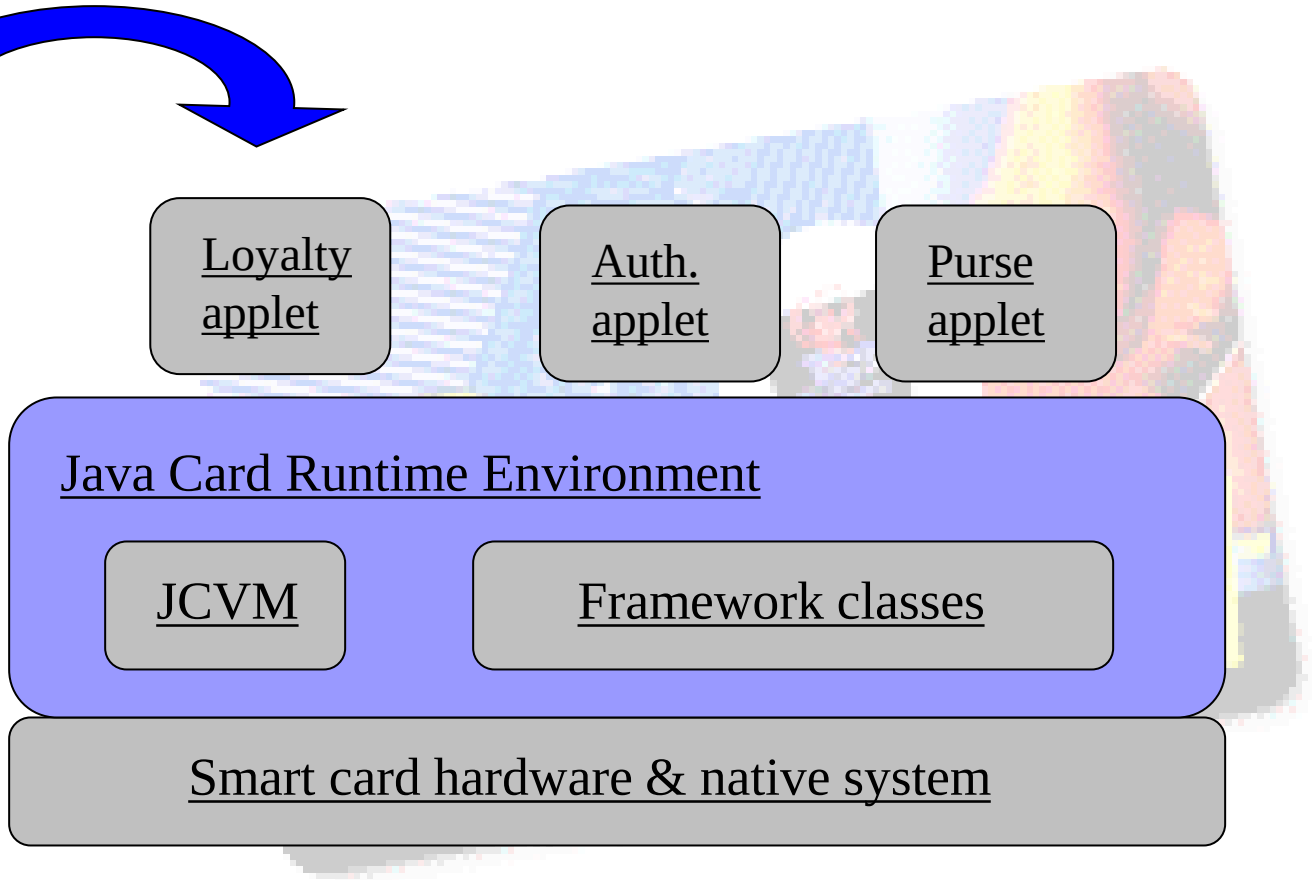
  - Multiapplicative Java cards: applets can be downloaded and installed on card after the card issuance

  - Applet's sensitive data must be protected against anouthorised accesses

Java cards

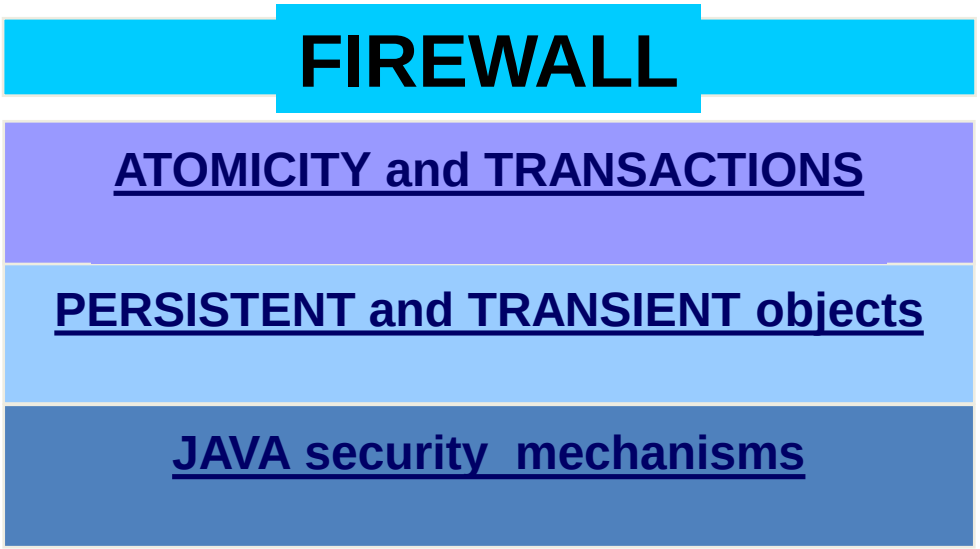
Card reader

Multiapplicative Java cards



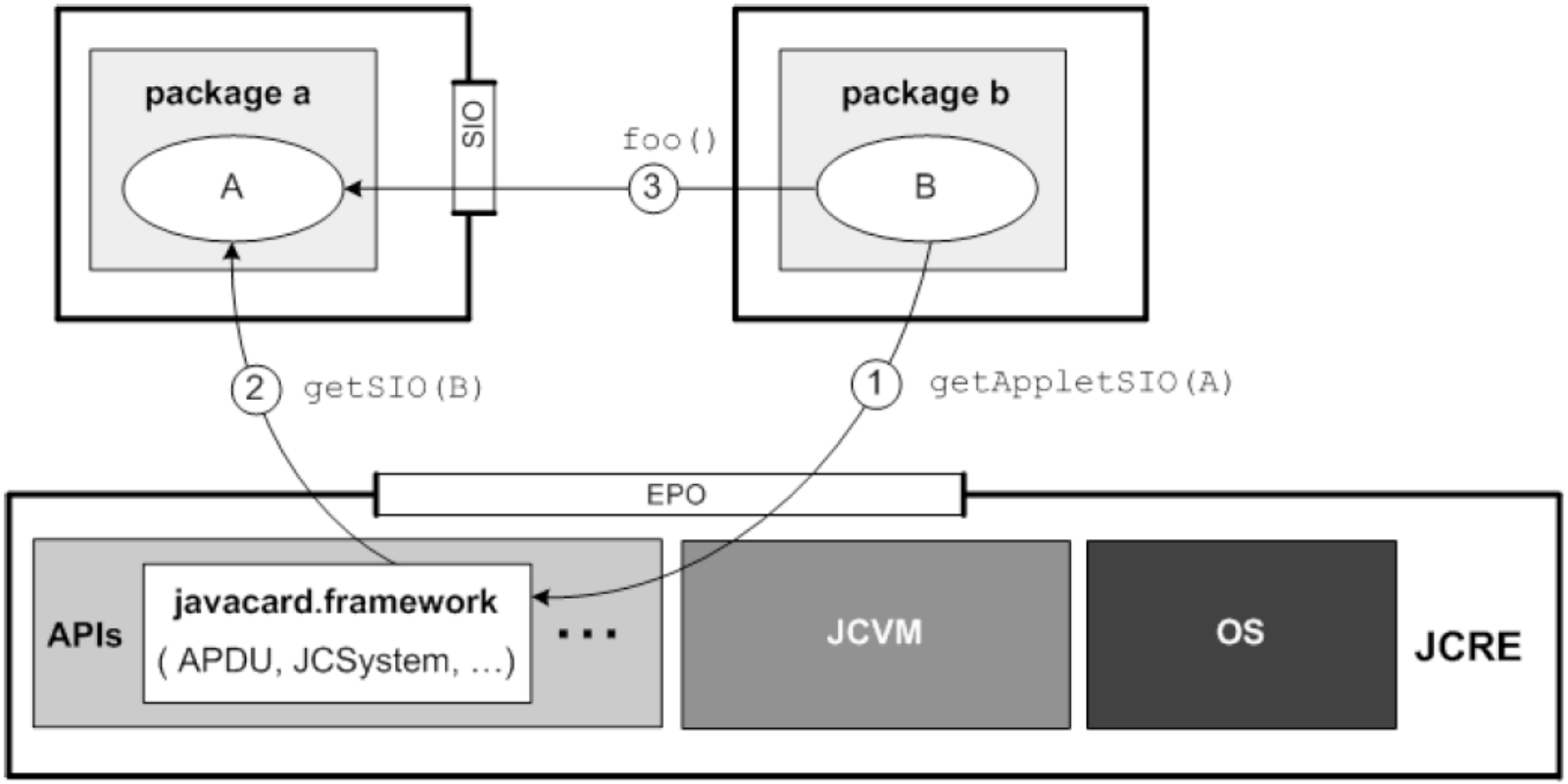
Java cards security

- Security in Java cards is a combination of the security mechanisms in Java and additional security procedures imposed by the card platform



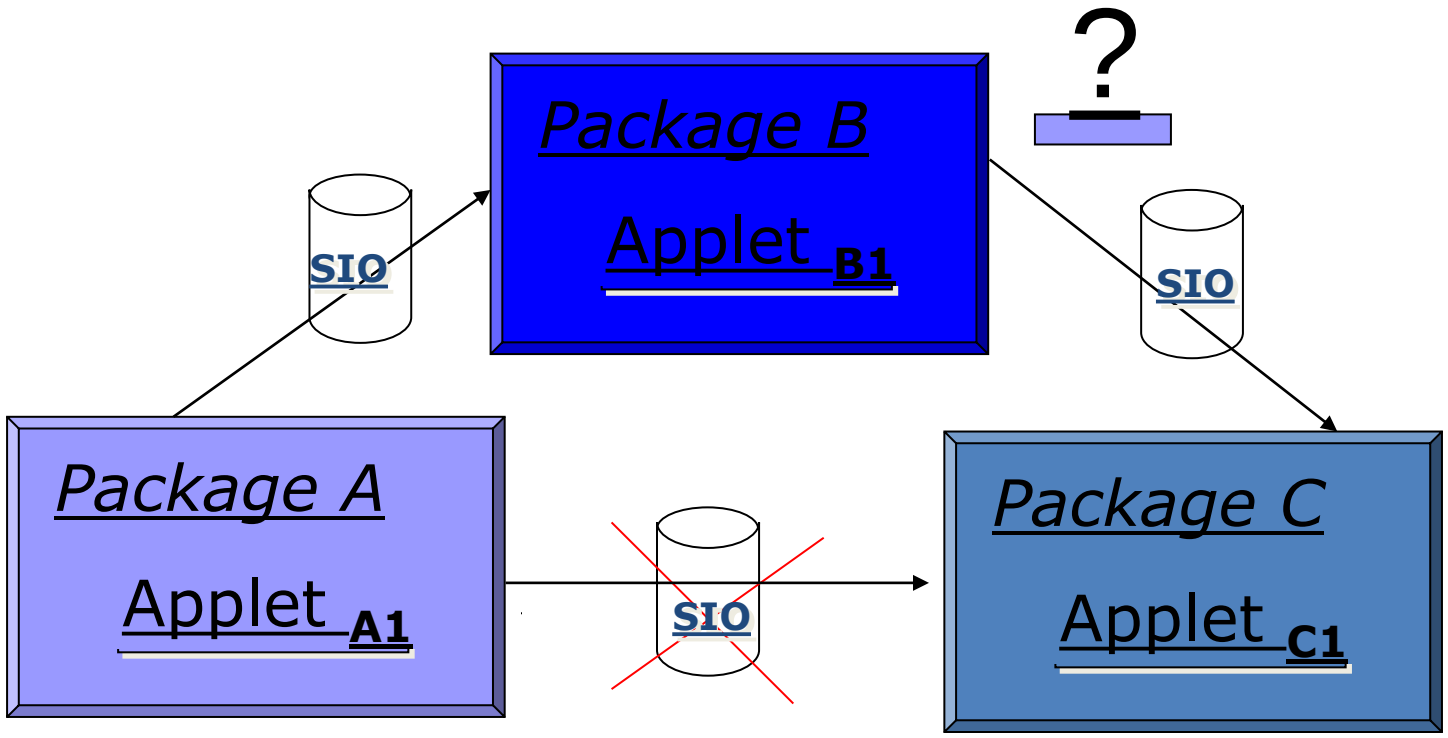
- The Firewall forces the isolation between objects of applets belonging to different packages

Communication between packages



Limits of the firewall

- Based on access control checks
- Place restrictions on the applets that can access to methods of applets belonging to other packages
- Does not control the propagation of the information from an applet of a package towards applets of other packages



file AInt.java

```
import javacard.framework.Shareable;
public interface AInt extends Shareable{
   public short foo();  }
```

file A.java

```
import javacard.framework;
import a.AInt;
public class A extends Applet implements AInt{
  private short balance;
  public Shareable getSIO(AID client, byte num){
    if(client.equals(B))  return this;
    return null;
  }
  public short foo(){
    AID client = getpreviouscontextAID();
    if(client.equals(B)) return balance;
    return 0;
  }
}
```

Fig. 3. Package a.

file BInt.java

```
import javacard.framework.Shareable;
public interface BInt extends Shareable{
   public a.AInt bar();  }
```

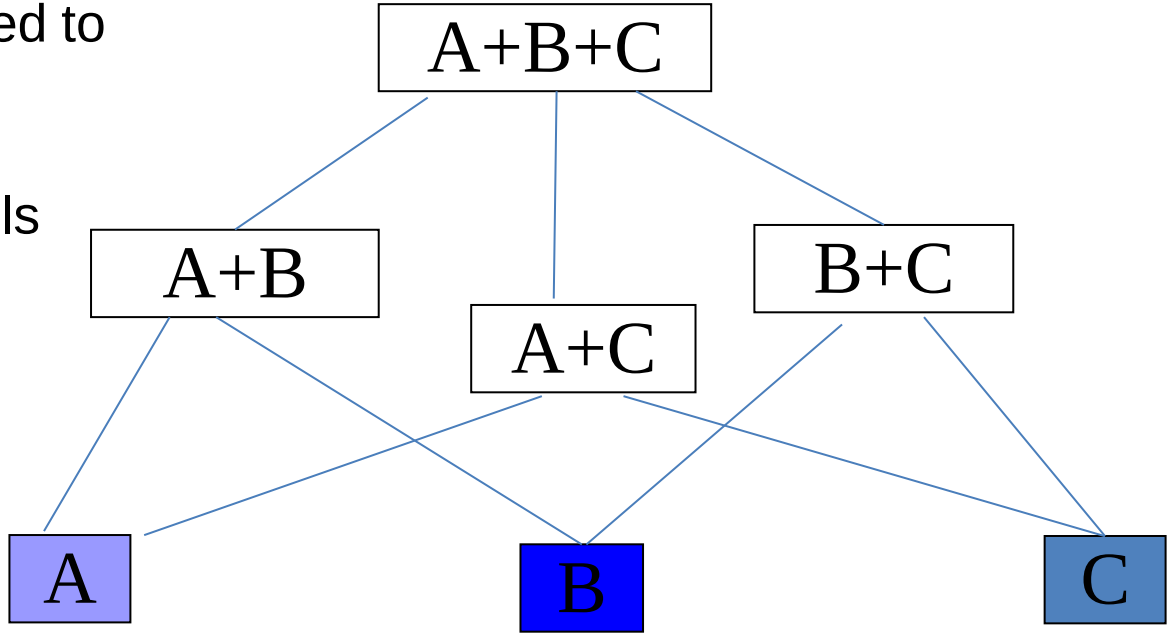
file B.java

```
import javacard.framework;
import a.AInt;
public class B extends Applet implements BInt{
  private static AInt AObj;
  private short ABalance;
  private void work (){
    AObj = (AInt) (JCSystem.getAppletSIO(A, 0));
    ABalance = AObj.foo();
  }
  public Shareable getSIO(AID client, byte num){
   return this;
  }
  public AInt bar(){return AObj;}
}
```

Fig. 4. Package b.

Secure Information Flow

- Security levels assigned to packages
- Lattice of security levels



- Abstract Interpretation framework: abstract execution of the applets using security levels instead of real data

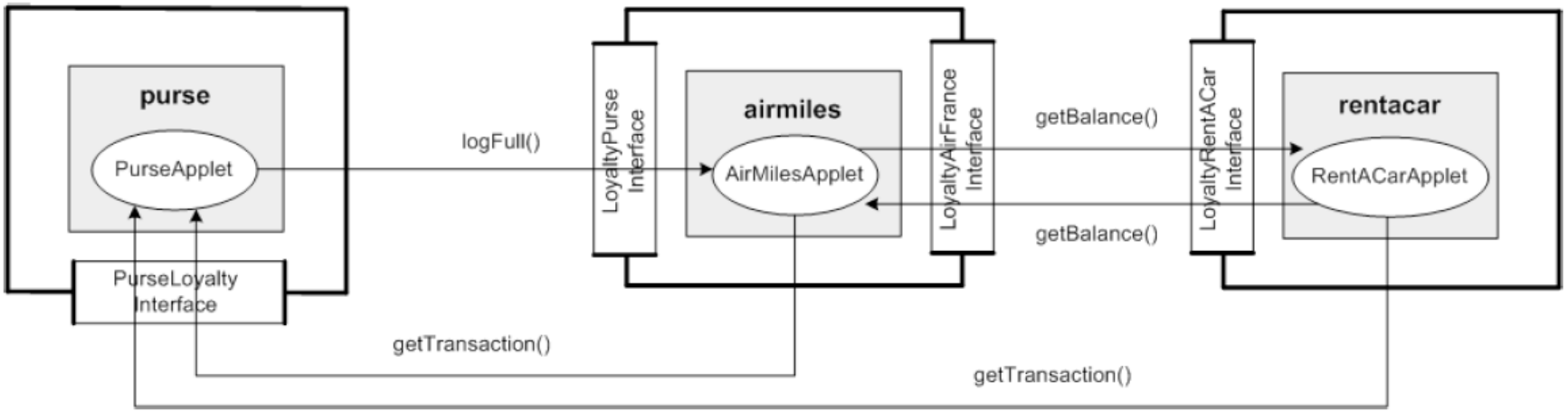
- Secure Information Flow: Check that information exchanged between A and B has a security equal to or level lower than A+B

Java Card Information Flow Verifier

JCIFV performs the analysis according to the following main steps

1. Unique security levels are automatically assigned to packages and shareable interface objects. An initial security level is assigned to the other methods and object fields
2. CAP file (native code of an applet) is decoded and saved as a bytecode
3. Abstract interpretation of the bytecode is performed
4. The analysis stops when the state of the abstract interpreter does not longer change and all methods have been analyzed
5. Secure information flow is checked

Electronic Purse



illicit information flow from Purse to RentACar caused by a method invocation (no parameters) from AirMiles and RentAcar

Purse: log-full service (logFull()), which notifies registered applets that the transaction log is going to be over-written.
Airmail: registered for the log-full service

RentACar: not registered for the log-full service

Electronic Purse

Assume that AirFrance requests RentACar the amount of miles (getBalance()) every time Purse notifies AirFrance that the transaction log is full.

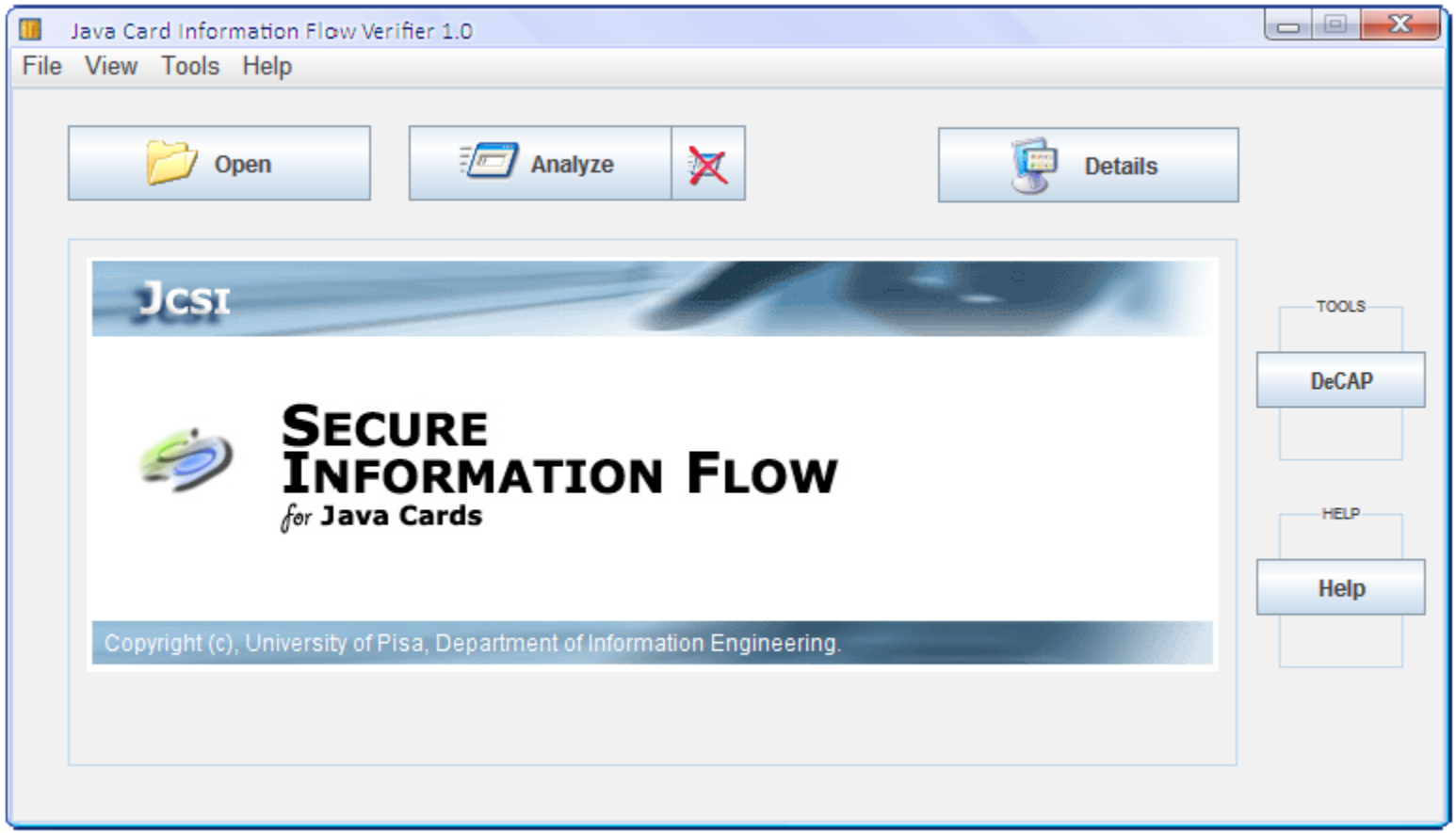
logFull() method implemented by AirFrance contains an invocation of method getTransaction() of Purse followed by an invocation of method getBalance() of RentACar.

Applet RentaACar, whenever observes an invocation of getBalance(), can infer that Purse is going to over-write the transaction log.

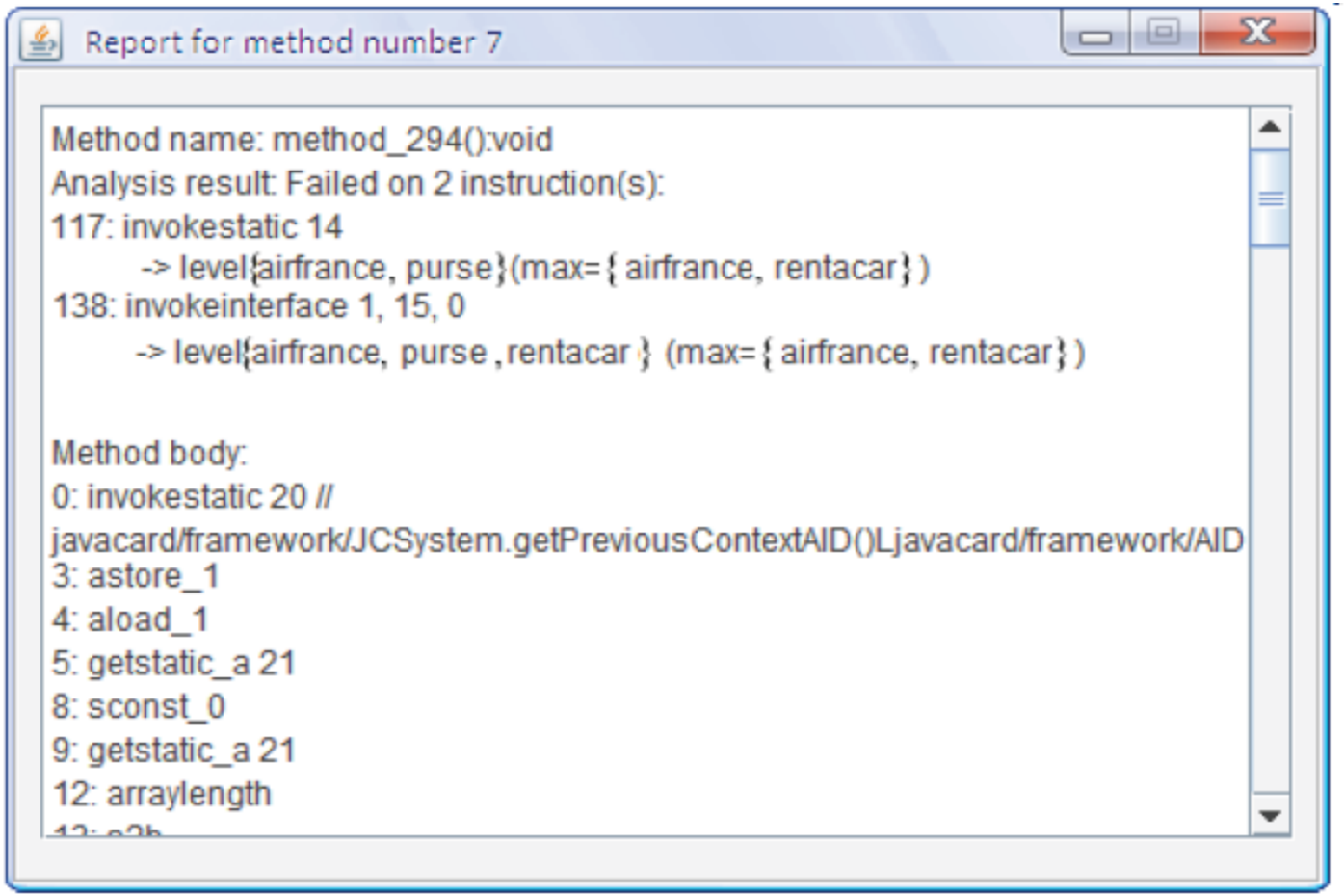
Thus, even without subscribing to the log-full service, RentACar is able to benefit from such a service.

Purse is not able to detect such information flow.

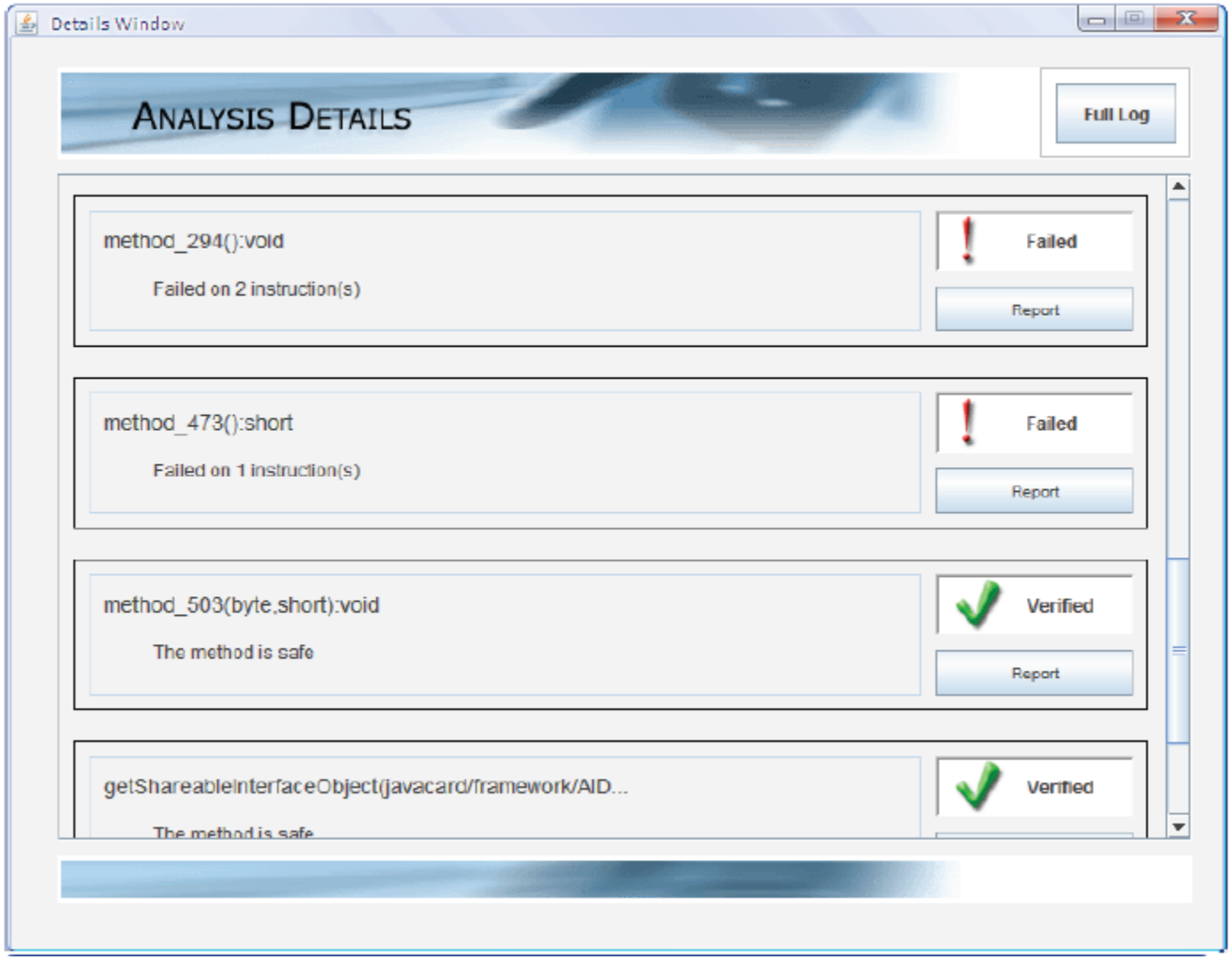
The tool


Java Card Information Flow Verifier 1.0
File View Tools Help
Open
Analyze
Details
JCSI
SECURE INFORMATION FLOW
for Java Cards
Copyright (c), University of Pisa, Department of Information Engineering.
TOOLS
DeCAP
HELP
Help

Report


Report for method number 7
Method name: method_294():void
Analysis result: Failed on 2 instruction(s):
117: invokestatic 14
-> level{airfrance, purse}(max={ airfrance, rentacar})
138: invokeinterface 1, 15, 0
-> level{airfrance, purse ,rentacar } (max={ airfrance, rentacar})
Method body:
0: invokestatic 20 //
javacard/framework/JCSystem.getPreviousContextAID()Ljavacard/framework/AID
3: astore_1
4: aload_1
5: getstatic_a 21
8: sconst_0
9: getstatic_a 21
12: arraylength

Analysis


Details Window
ANALYSIS DETAILS
Full Log
method_294():void
Failed on 2 instruction(s)
Failed
Report
method_473():short
Failed on 1 instruction(s)
Failed
Report
method_503(byte,short):void
The method is safe
Verified
Report
getShareableInterfaceObject(javacard/framework/AID...
The method is safe
Verified

Discussion

- JCIFV tool is able to certify applets against secure information flow of sensitive data
  saved present on the card

- Verification (Abstract Interpretation)
  - JVIFV certifies only secure applets
  - some correct applets could also be rejected
    - due to the characteristics of the applet code the percentage of erroneously rejected applets is very low