UNIVERSITY
OF TRENTO - Italy

secure
CHANGE

Load-Time Security Certification for Real Smart-Cards

Olga Gadyatskaya
joint work with F.Massacci, E.Lostal
(University of Trento, Italy)

Evaluation by B. Chetali, Q-H. Nguyen
TrustedLabs/Gemalto (FR)

This talk

UNIVERSITY
OF TRENTO - Italy

- How to design lightweight yet flexible and effective access control framework in a very restricted environment (Java Card)
- How to integrate the framework on a real card
- Bonus: demo of the prototype

Agenda

UNIVERSITY
OF TRENTO - Italy

- **Motivations and the Security-by-Contract idea**
- **The Java Card Background**
- **Contracts**
- **A (thin) hint of theory**
- **A (larger) taster of engineering**
- **Demo**
- **Conclusions**

Agenda

UNIVERSITY
OF TRENTO - Italy

- **Motivations and the Security-by-Contract idea**
- **The Java Card Background**
- **Contracts**
- **A (thin) hint of theory**
- **A (larger) taster of engineering**
- **Demo**
- **Conclusions**

Mobile payments

UNIVERSITY OF TRENTO - Italy

- **NFC** technology as enabler
- **Secure element** for storing secrets

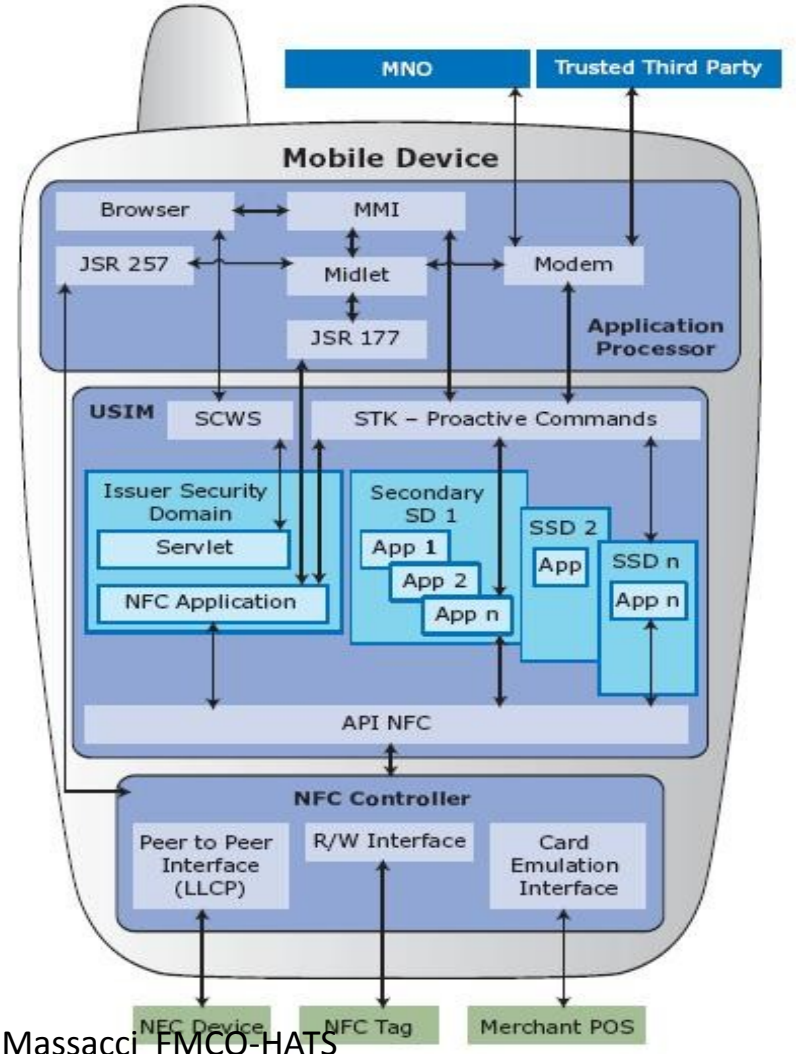


Image courtesy of the NFC forum

Pros of each secure element technology

UNIVERSITY
OF TRENTO - Italy

- **Cheaper**

Dedicated chip/phone memory

- **SIM card is already managed by the telco**
- **Standardized development and deployment**
- **It is there in ALL ~~smart~~phones**

SIM card

SIM as secure element

UNIVERSITY OF TRENTO - Italy

- Not only credentials/PINs.
  These are apps!
  - and some of them may even interact
  - this is Java
- New apps may be added/ old ones removed over time
- **Sensitive apps require strict control (on the secure element) over who talks to whom**

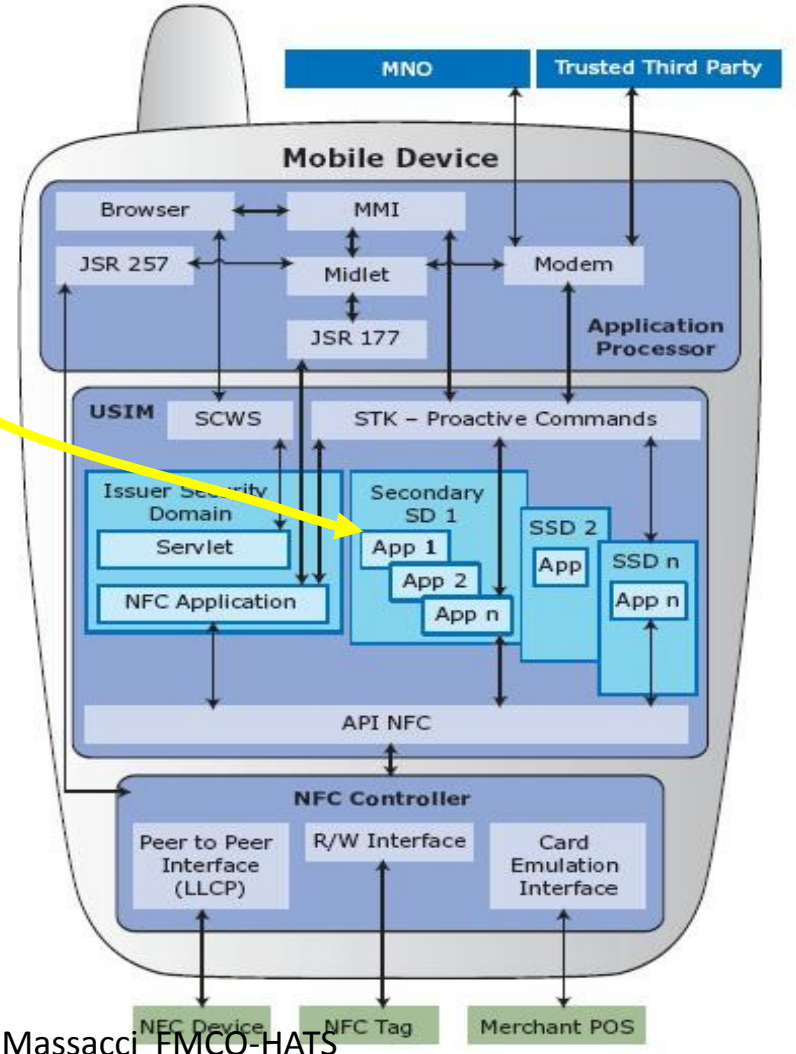


Image courtesy of the NFC forum

Design goals

UNIVERSITY
OF TRENTO - Italy

- **We need an on-card system that:**
  - Allows to add or remove applets
  - Enables applets to declaratively control access to their shared resources (**services**)
  - The access control policy can mention arbitrary applet identifiers (AIDs)
  - The applet bytecode is validated by the card itself to respect the policies of other applets on card

Design constraints

UNIVERSITY
OF TRENTO - Italy

- **No modifications to the standard loading protocol, run-time environment or the virtual machine**
  - Too expensive
- **Most part of the trusted computing base is in ROM**
  - Cannot be modified after the card is in the field
- **Applet providers can set up their policies independently**
  - Telco does not want to be bothered

It was not achieved before

UNIVERSITY
OF TRENTO - Italy

Existing solutions for Java Card:

- Can verify full information flow, but for predefined set of applets and off-card
- Can verify transitive control flow on card, but only for predefined and limited set of domains (applet owners)
- [Java Card protection] The policies are embedded into the applet code.

The threat model

UNIVERSITY
OF TRENTO - Italy

- We assume an attacker that can:
  - Load or remove her applets on the card
  - Update access control policy of her own applets
- The attacker cannot:
  - Force loading or removal of someone else's applets or change their policies
  - Spoof someone else's applets pretending to be their legitimate owner
- **The attacker's goal**
  - **Enable her applets to access illegally sensitive services of other applets**

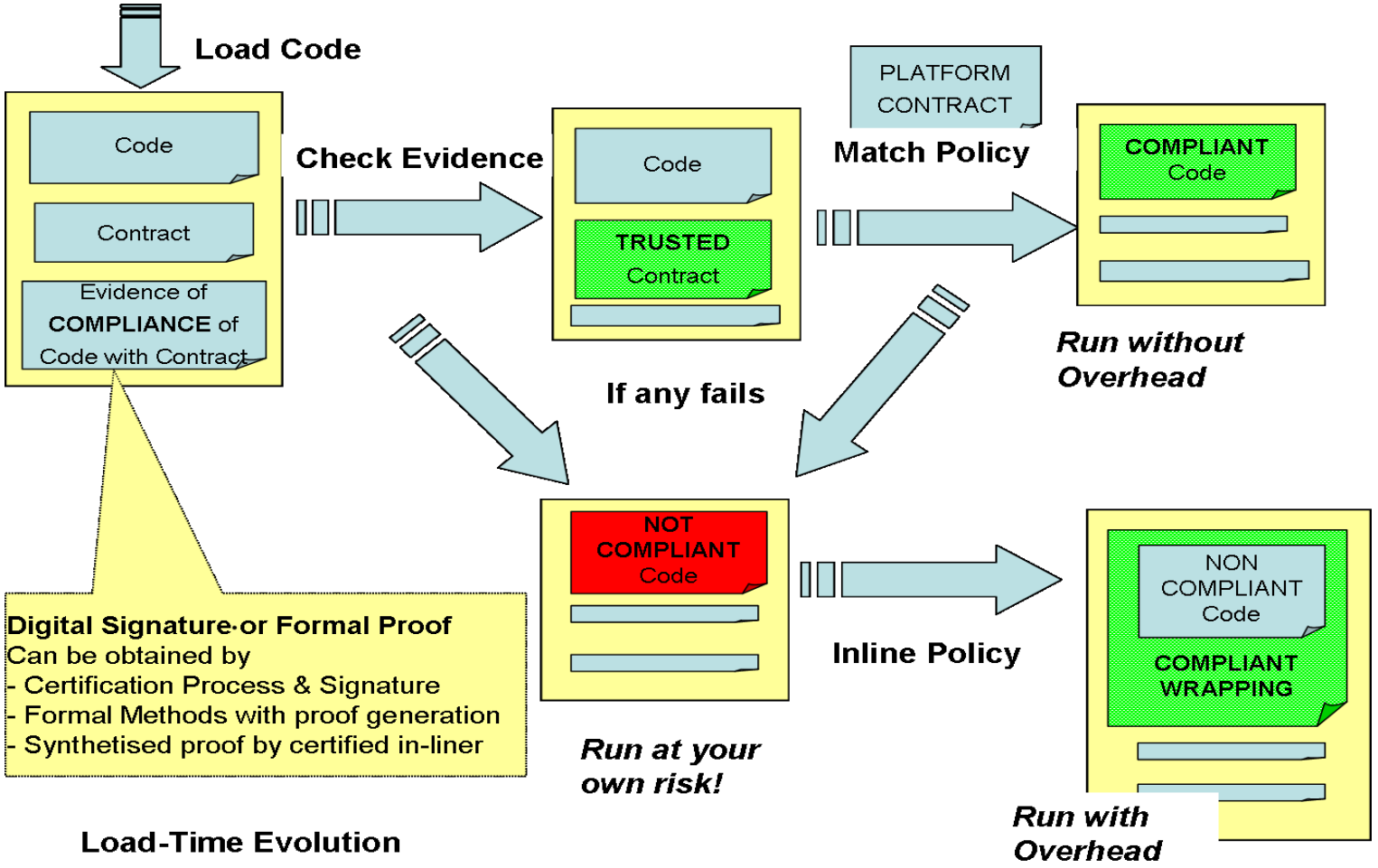
The Security-by-Contract idea

UNIVERSITY
OF TRENTO - Italy

- **SxC – particular instance of Load Time Verification**
  - Derived from Proof carrying code and Model carrying code ideas
- **Well-tested for mobile platforms**
  - Java & .NET implementation (2008)
  - Android (Manifest) implementation (Enck et al, 2010)
- **But a smartphone isn't a card…**

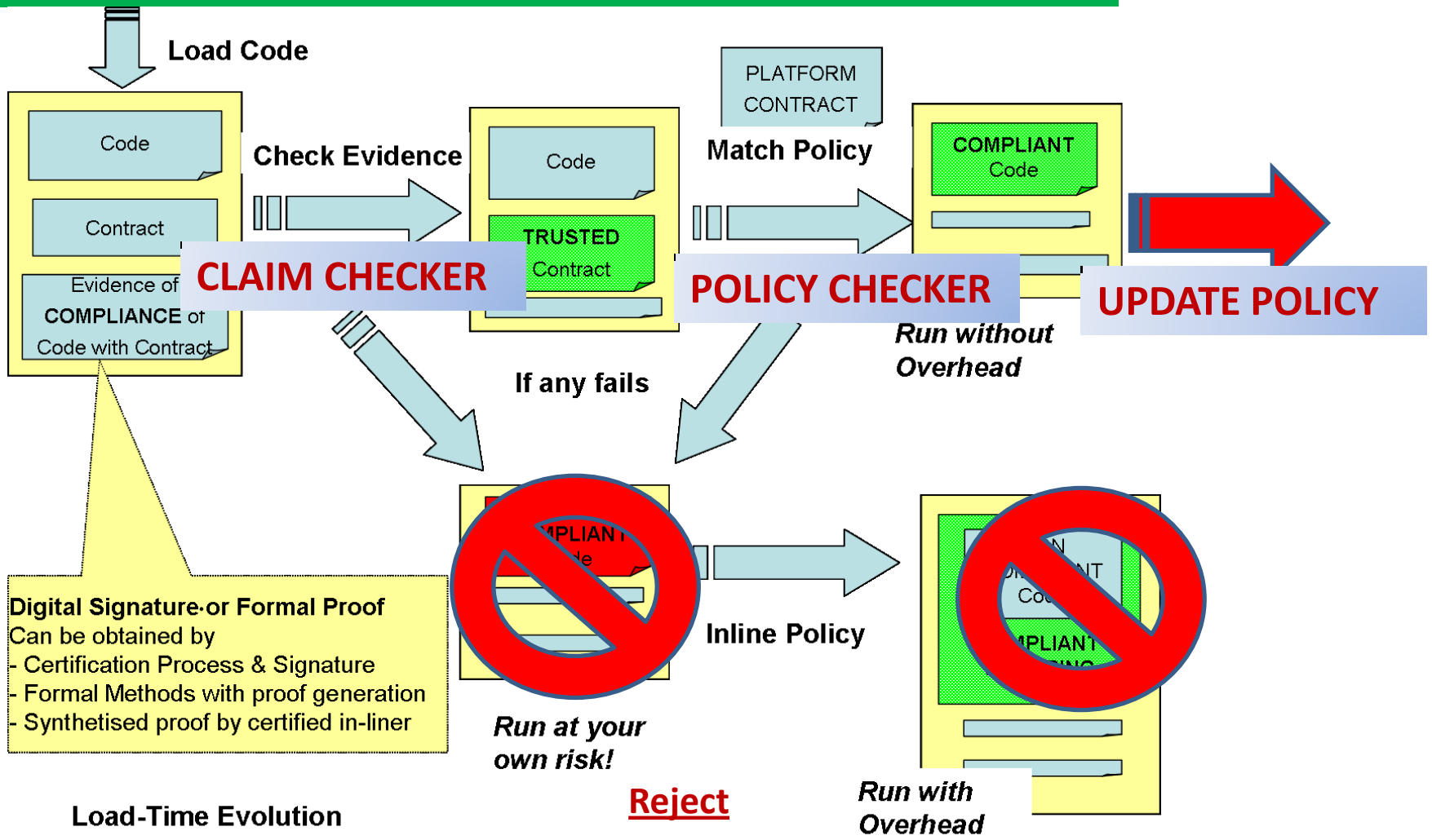
SxC workflow on mobile

UNIVERSITY OF TRENTO - Italy



SxC workflow on smart cards

UNIVERSITY OF TRENTO - Italy

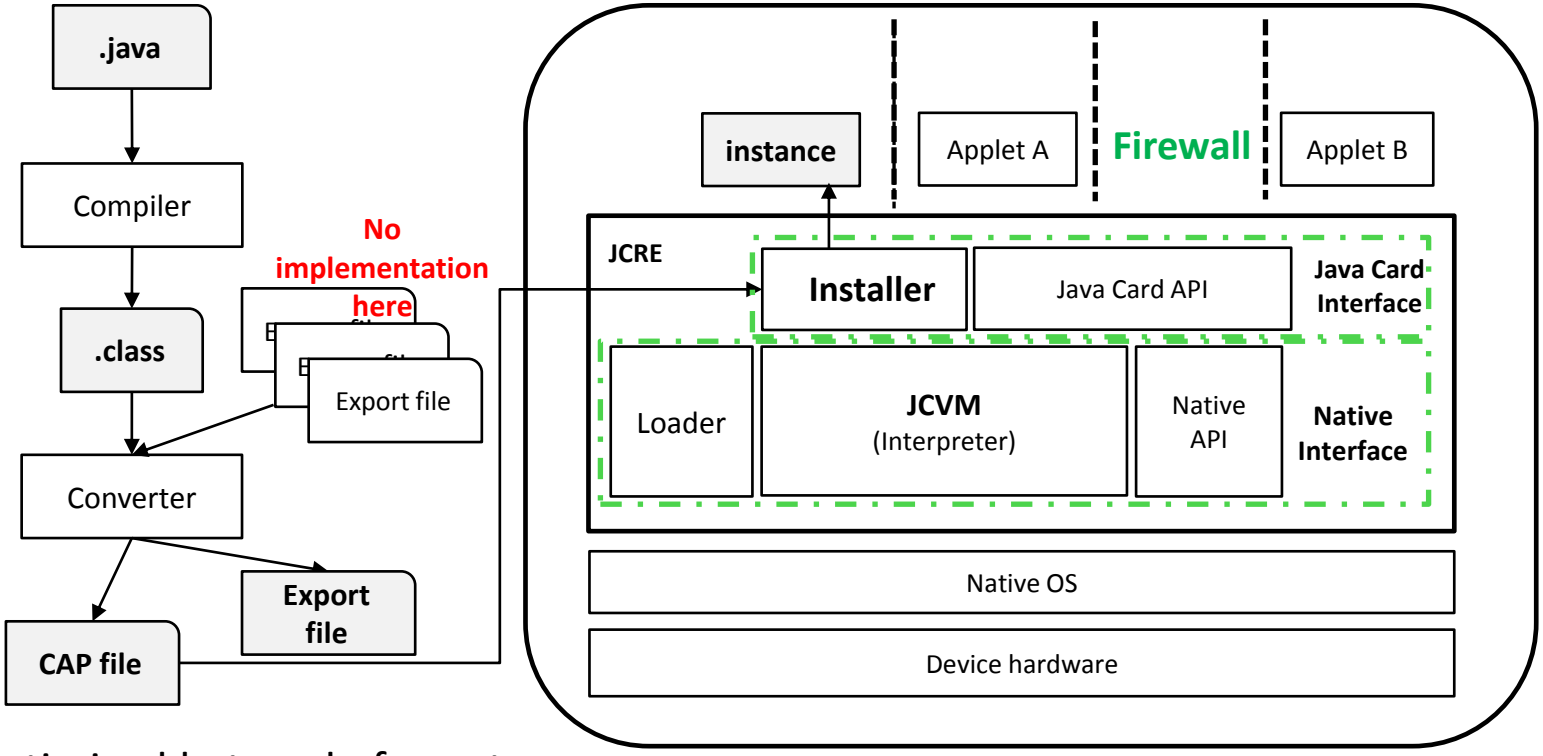


Agenda

UNIVERSITY
OF TRENTO - Italy

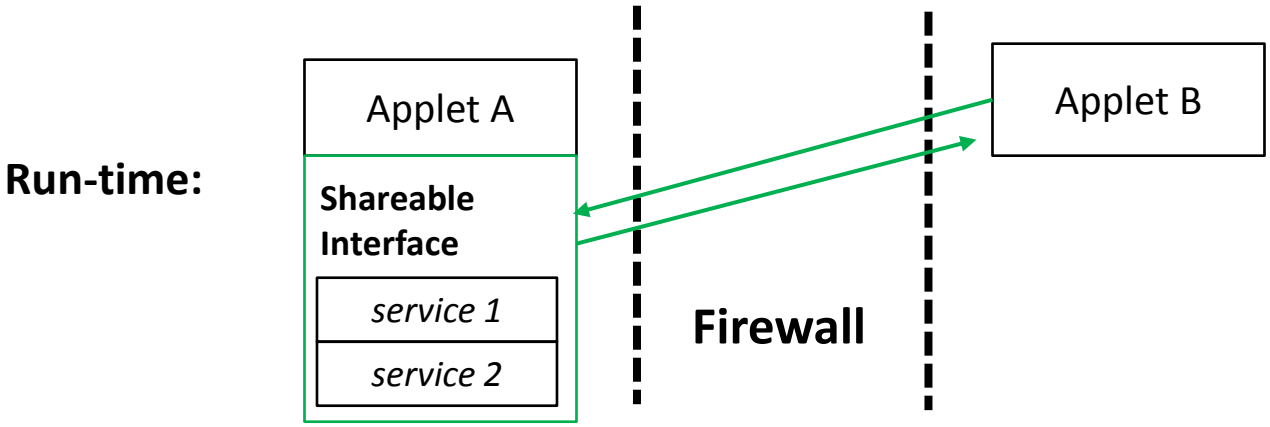
- **Motivations and the Security-by-Contract idea**
- **The Java Card Background**
- **Contracts**
- **A (thin) hint of theory**
- **A (larger) taster of engineering**
- **Demo**
- **Conclusions**

The Java Card platform



Optimized bytecode format

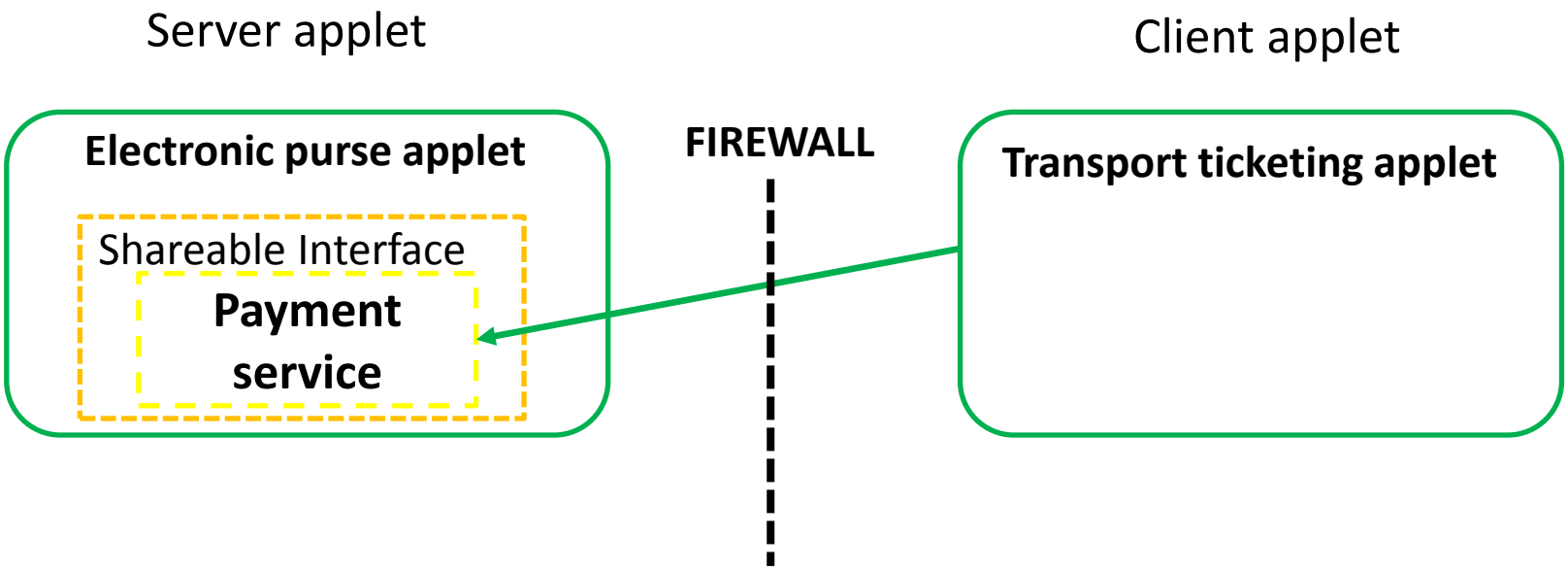
How does JC really work?



Access control is embedded into functional code

- Technical Consequence 1 → If A checks who calls it, the access control policy cannot be updated unless the code is updated
  - sometimes code updates are not even possible
- Technical Consequence 2 → If A does not check, then everybody can use it

Example


Server applet
Client applet
Electronic purse applet
Shareable Interface
Payment service
FIREWALL
Transport ticketing applet

ePurse applet: the ACL in the code

UNIVERSITY
OF TRENTO - Italy

```
01 byte ClientsNumber = 1;
02 byte [] TransportAIDset =
{0x01,0x02,0x03,0x04,0x05,0x0C,0x0A};
03 final AID TransportAID = JCSystem.lookupAID
(TransportAIDset,(short)0,(byte)TransportAIDset.length);
04
05 //the access control list
06 AID [] clientAIDs = {TransportAID};
07 //ACL check implementation
08 public short authorizedClient(AID clientAID){
09    for (short i=0; i<ClientsNumber; i++)
10       if (clientAIDs[i].equals(clientAID))
11          return i; //clientAIDs is in the ACL
12    return -1;
13 }
```

ACL checks

ePurse applet: Shareable interface

UNIVERSITY OF TRENTO - Italy

```
14 //SI definition
15 public interface PaymentInterface extends Shareable {
16    //definition of the payment service
17   byte payment(short account_number);
18 }
19 public class PaymentClass implements PaymentInterface {
20   byte payment_code = 0x08;
21   public byte payment(short account_number){
22      //implementation of  the service
23      AID clientAID = JCSystem.getPreviousContextAID();
24      if (authorizedClient(clientAID) == -1) //ACL check
25          return (byte) 0x00; //no service is provisioned
26      else return payment_code; //provision of the service
27   }
28 }
29 public PaymentClass PaymentObject;
```

service

Agenda

UNIVERSITY
OF TRENTO - Italy

- **Motivations and the Security-by-Contract idea**
- **The Java Card Background**
- **Contracts**
- **A (thin) hint of theory**
- **A (larger) taster of engineering**
- **Demo**
- **Conclusions**

Contract I

UNIVERSITY
OF TRENTO - Italy

- Applets come equipped with a contract
  - **Claims**
    - I may **provide** these shareable interfaces with these services
    - I may **call** those methods from those interfaces
  - **Security Rules**
    - This service can only be called by this application
  - **Functional Rules**
    - I need these services from those applications
- When new applet arrives platform will check
  - **contract complies with bytecode**
  - **contract is acceptable to other applets**

Contract II

UNIVERSITY
OF TRENTO - Italy

Contract of an applet

AppClaim

Provided services

<Interface token, method token>

Called services

<Provider application AID, Interface token, method token>

AppPolicy

Security rules

<Interface token, method token, Authorized application AID>

Functional rules

<Provider application AID, Interface token, method token>

How do we get the tokens? - from Export files

UNIVERSITY OF TRENTO - Italy

Export file (snippet) of the Purse applet:

```
export_classes {
  class_info {  // packagePurse/PaymentInterface
    token   0 // Shareable interface token
    …
    export_methods_count  1
    methods {
      method_info {
        token 0  // shared method token
        …
        name_index  0  // payment
```

Service `PaymentInterface.payment` → gets token <0,0>

Invoked service tokens

UNIVERSITY OF TRENTO - Italy

Source code of *Transport*

```
01 private void connectServer(){
02    final AID appletAID = JCSystem.lookupAID
(serverAppletAID,(short)0,(byte)serverAppletAID
.length);
03   if (appletAID == null)
04
ISOException.throwIt(ISO7816.SW_CONDITIONS_NOT_
SATISFIED);
05   PaymentObject = (PaymentInterface)
(JCSystem.getAppletShareableInterfaceObject(app
letAID, InterfaceDetails));
06 }
07 private void newBalance() {
     // Actual service invocation
08    payment_code =
PaymentObject.payment(account_number);
09    return;
10 }
```

CAP file of *Transport*

Import component

```
package_info[1]{…
AID_length 6
AID {1.2.3.4.5.b} }
```

Constant Pool component

```
constantPool[16]{…
external package_token 1
class_token 0
```

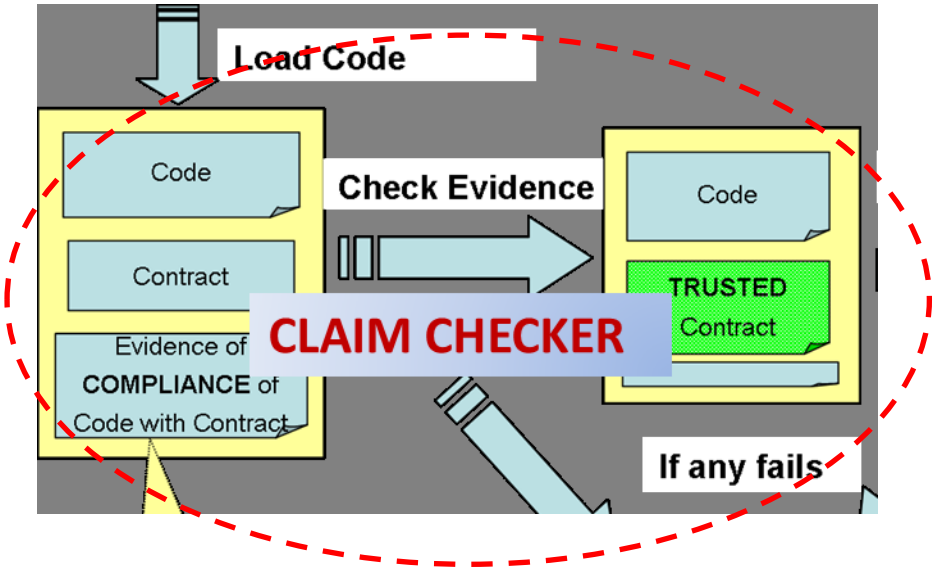
Method component

```
…   // bytecode of newBalance()
getstatic_a 17;
getfield_b_this 2;
invokeinterface 2 16 0;
putfield_b 3;
return;
```

Called service <0 , 0> from AID 0x01020304050B

The Claim Checker

UNIVERSITY OF TRENTO - Italy



Matches the Contract with the bytecode

For provided services:

- Checks the Shareable interfaces in CAP Export component

For called services:

- Finds all `invokeinterface` instructions (Method component and friends) and checks the invocation was declared

Agenda

UNIVERSITY
OF TRENTO - Italy

- Motivations and the Security-by-Contract idea
- The Java Card Background
- Contracts
- A (thin) hint of theory
- A (larger) taster of engineering
- Demo
- Conclusions

Formally

UNIVERSITY
OF TRENTO - Italy

- A deployed applet is a tuple <AID, Bytecode, ConstPool>
- A platform Θ is a set of currently deployed applets
- Security policy of the platform is a set of contracts $\{\{Contract_1\}, \ldots, \{Contract_N\}\}$ of currently deployed applets

Taxonomy of the JCVM instructions

UNIVERSITY
OF TRENTO - Italy

Type	Instructions
I	Arithmetic instructions; instructions that do not affect control flow. Cannot produce exceptions, execution proceeds to the next instruction: iadd
II	Can throw run-time exceptions, but not security exceptions: irem
III	Modify execution flow: goto, ifnull
IV	Return instructions: return
V	Can throw security exceptions: checkcast, iastore. The JCRE checks the object access rights here
VI	Invoke methods: invokeinterface, invokespecial, invokestatic, invokevirtual

The security theorem

UNIVERSITY
OF TRENTO - Italy

- **IF the JCRE is correct wrt specs:**
  - [Firewall] applets only interact through Shareable interfaces
  - The Converter was correct and the CAP file was not tampered with
  - `invokeinterface` is the only invocation instruction that can be used for invoking services
- **AND the SxC framework is correct wrt the specs**

- **THEN all methods invoked by any deployed applet B are authorized in the platform policy**

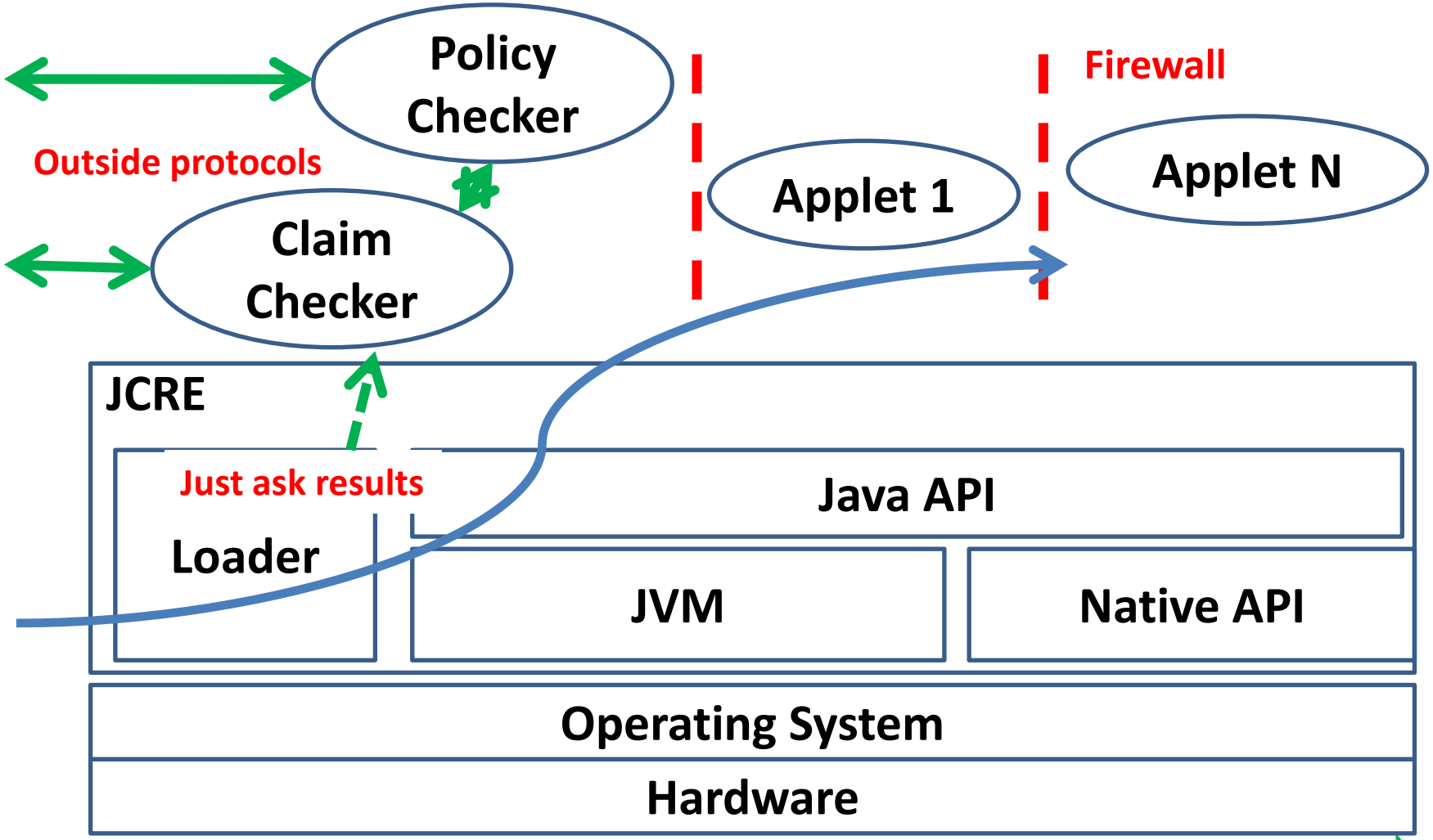
Proof goes by cases of method invocation on the platform and inductively over the length of platform execution.

Agenda

UNIVERSITY
OF TRENTO - Italy

- **Motivations and the Security-by-Contract idea**
- **The Java Card Background**
- **Contracts**
- **A (thin) hint of theory**
- **A (larger) taster of engineering**
- **Demo**
- **Conclusions**

Our first architecture: "as-on-mobile"

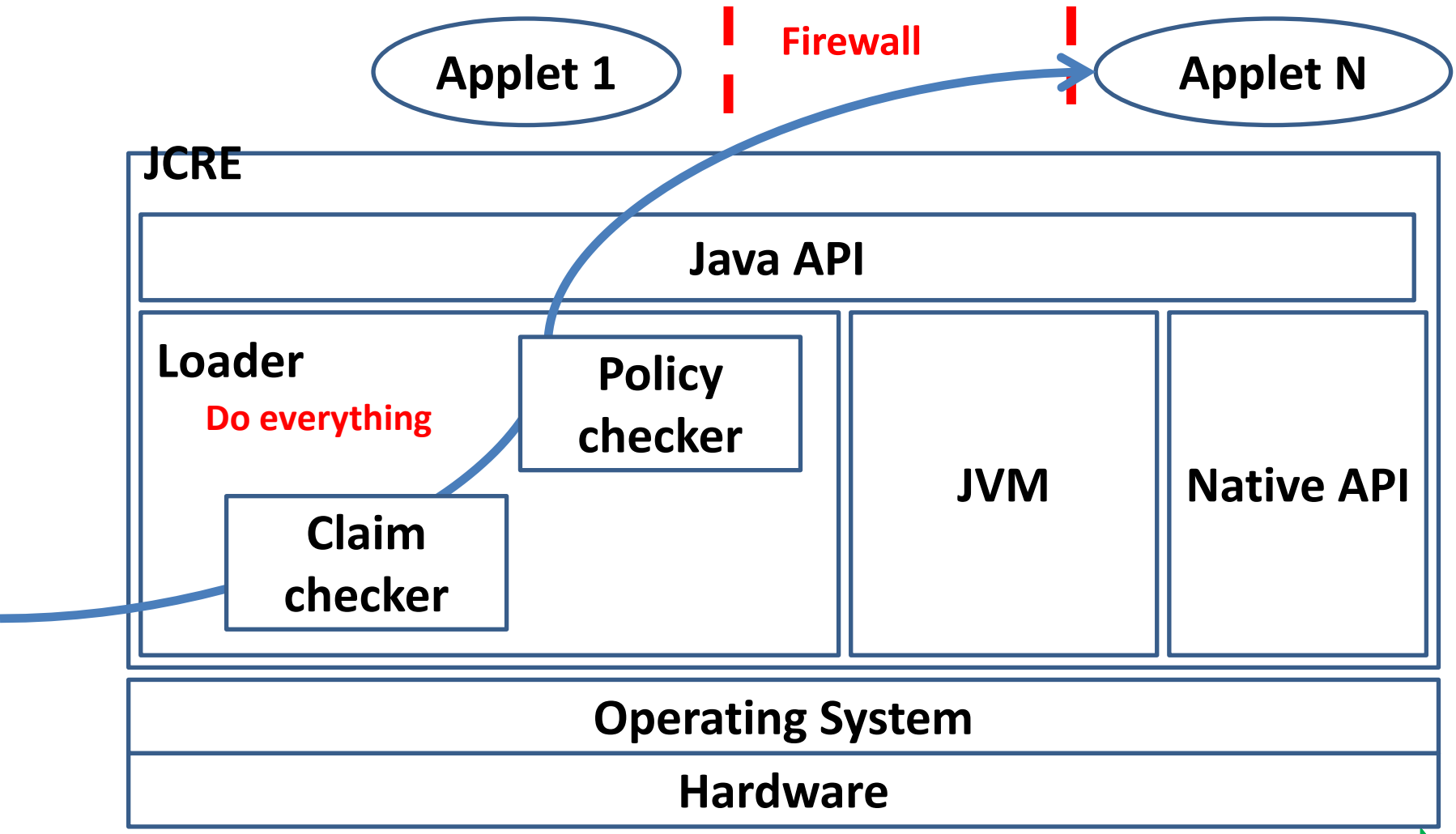


First engineering problem

UNIVERSITY
OF TRENTO - Italy

- **We implemented Policy Checker as an applet**
  - Footprint of checker 11KB and contracts 2KB
- **BUT requires changing existing protocols!**
  - Loading protocol standard plus check results of 1+2
  - New protocol with policy checker
  - New protocol with claim checker
- **Loader can trust Policy Checker, but Claim Checker?**
  - Needs signatures and certification
  - Too small improvement to justify new protocols

Our second architecture

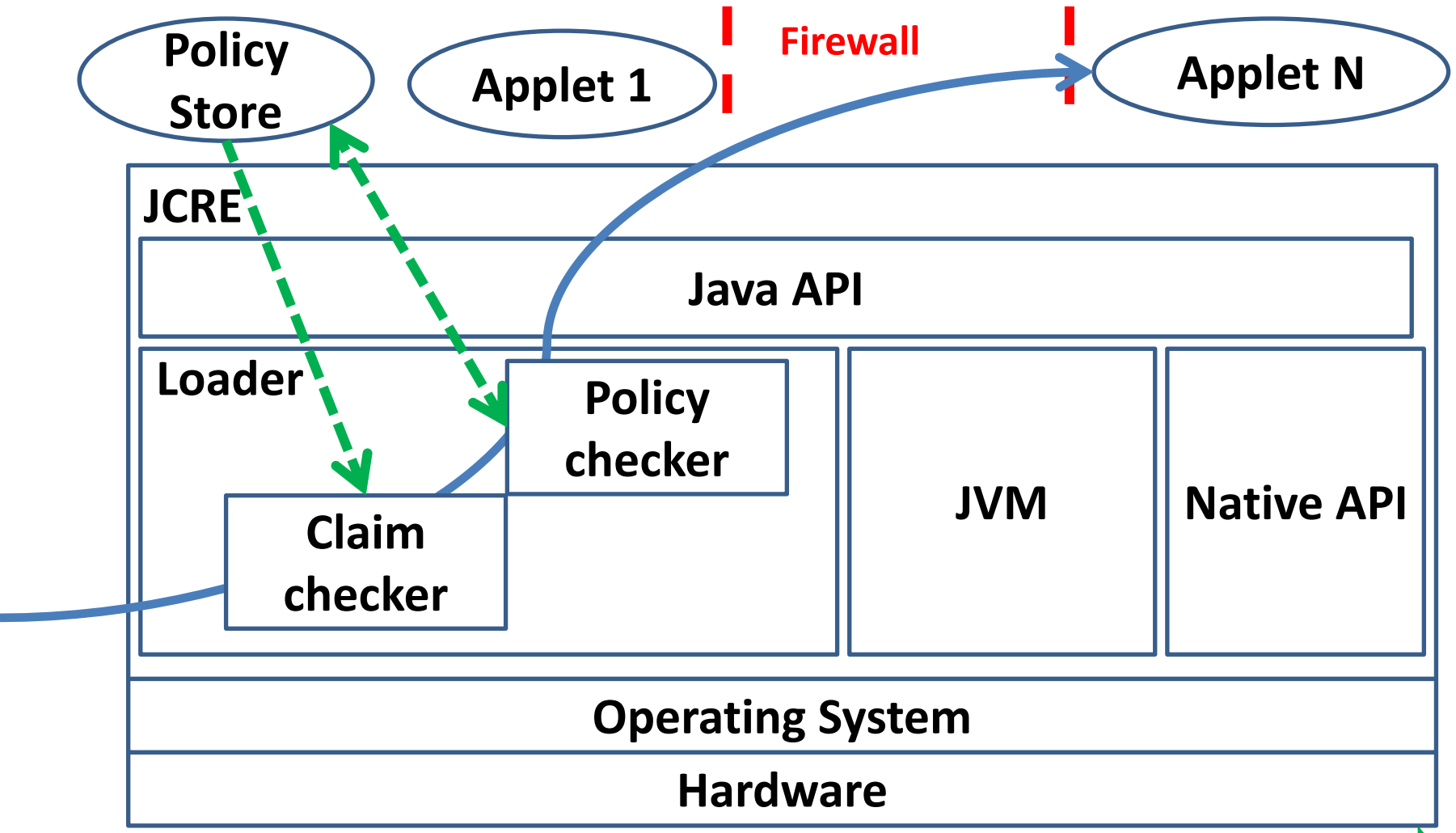


Second Engineering Problem

UNIVERSITY
OF TRENTO - Italy

- More Effective and Efficient
  - Checkers no longer trust external checks of code
  - Eliminate check of signature!
  - Both checkers can be implemented in C
- But where do we put the policy?
  - We need to retrieve it and store it somewhere…
  - But the Loader is "printed"
    - We could have a "`static int policy[]`" but that's not going to work in the ROM

Our third architecture



Third Engineering Problem

UNIVERSITY
OF TRENTO - Italy

- **How to deliver the Contract to the Checkers?**
  - Can't change the loading protocol
- **Both Checkers need applet AID…**
  - AIDs are "big" → don't want to use them in the algorithms
  - AIDs only known at loading time → can't "print" them in Loader
- **A bit of help from the platform**
  - AID are mapped into Package ID (much shorter)
  - But still you have rules for AIDs not yet on board

Third Engineering Idea

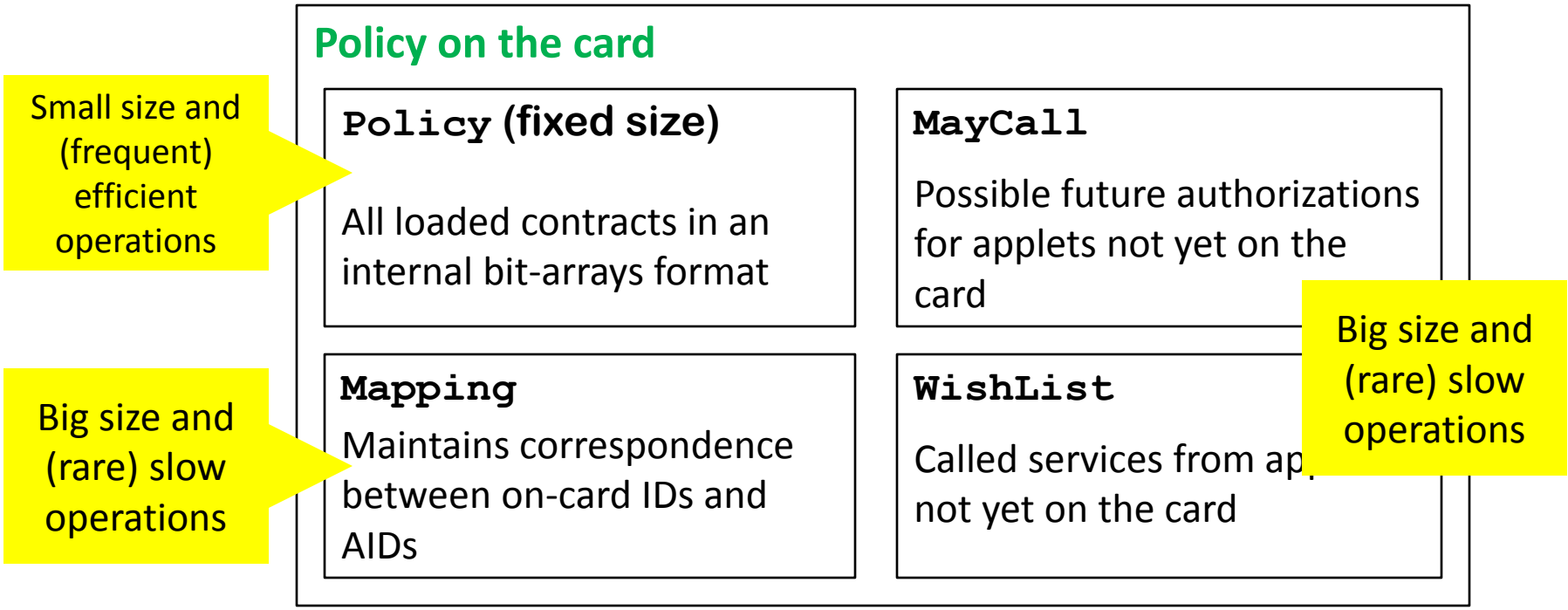
UNIVERSITY
OF TRENTO - Italy

- **Each applet includes contract in CAP file Custom component**
  - No need to send it separately
  - Arrives and leaves with applet
  - Updates identical to old code updates
  - Enables backward compatibility for cards and applets
- **Checkers do not need trust anyone**
  - Contract update would anyhow require code check
- **PolicyStore references applet contract with PID**
  - Mapping table from PID to AID
  - Checkers only get short matrix with loaded PIDs

Security policy on the card

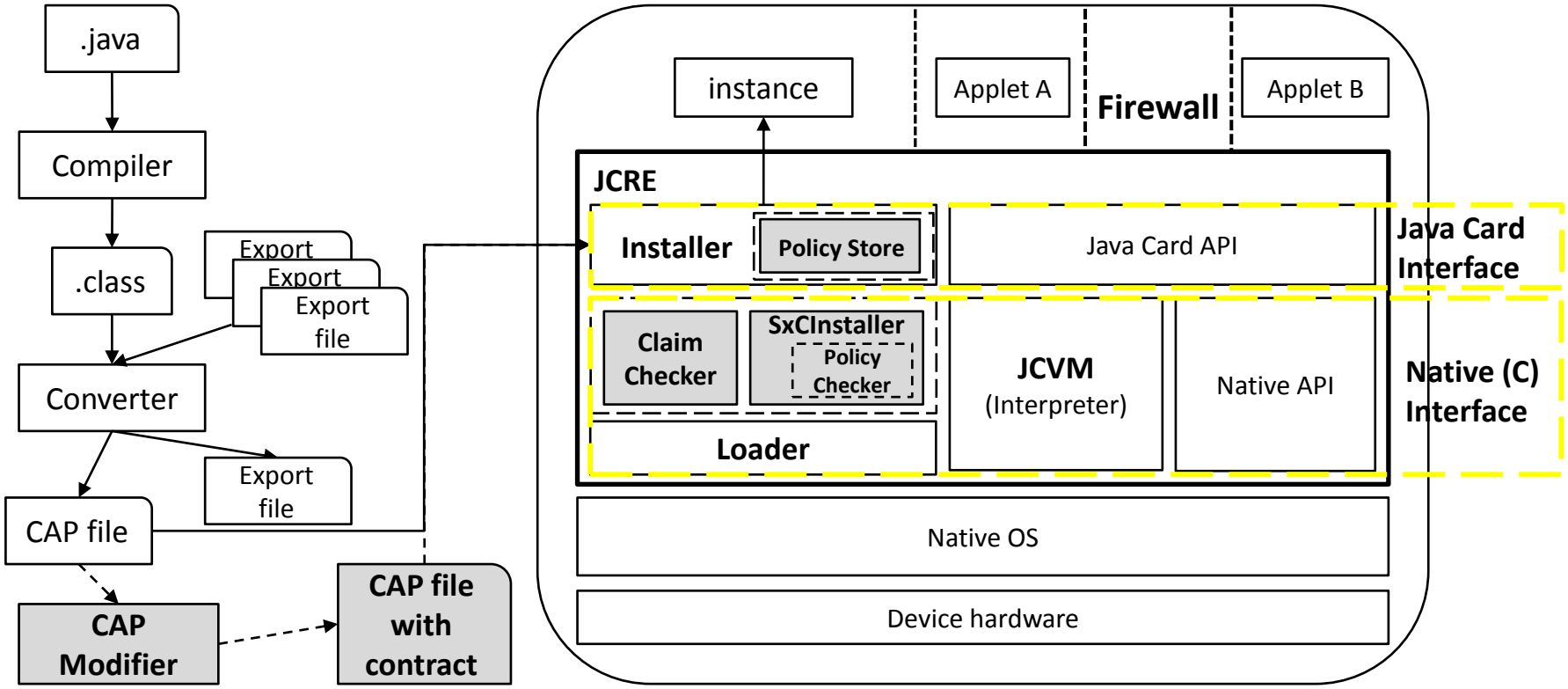
UNIVERSITY OF TRENTO - Italy

Arbitrary AIDs in the Mapping



The final architecture

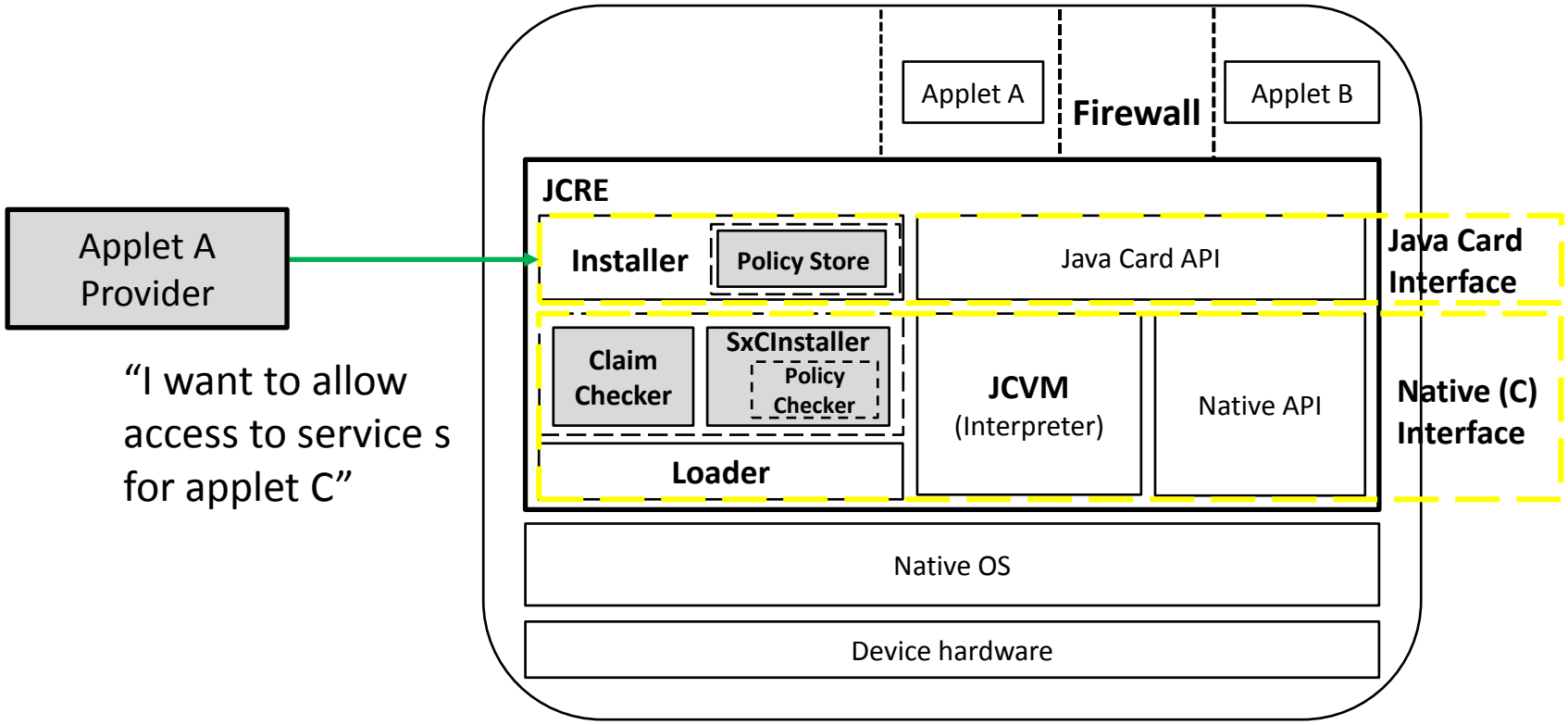
UNIVERSITY OF TRENTO - Italy



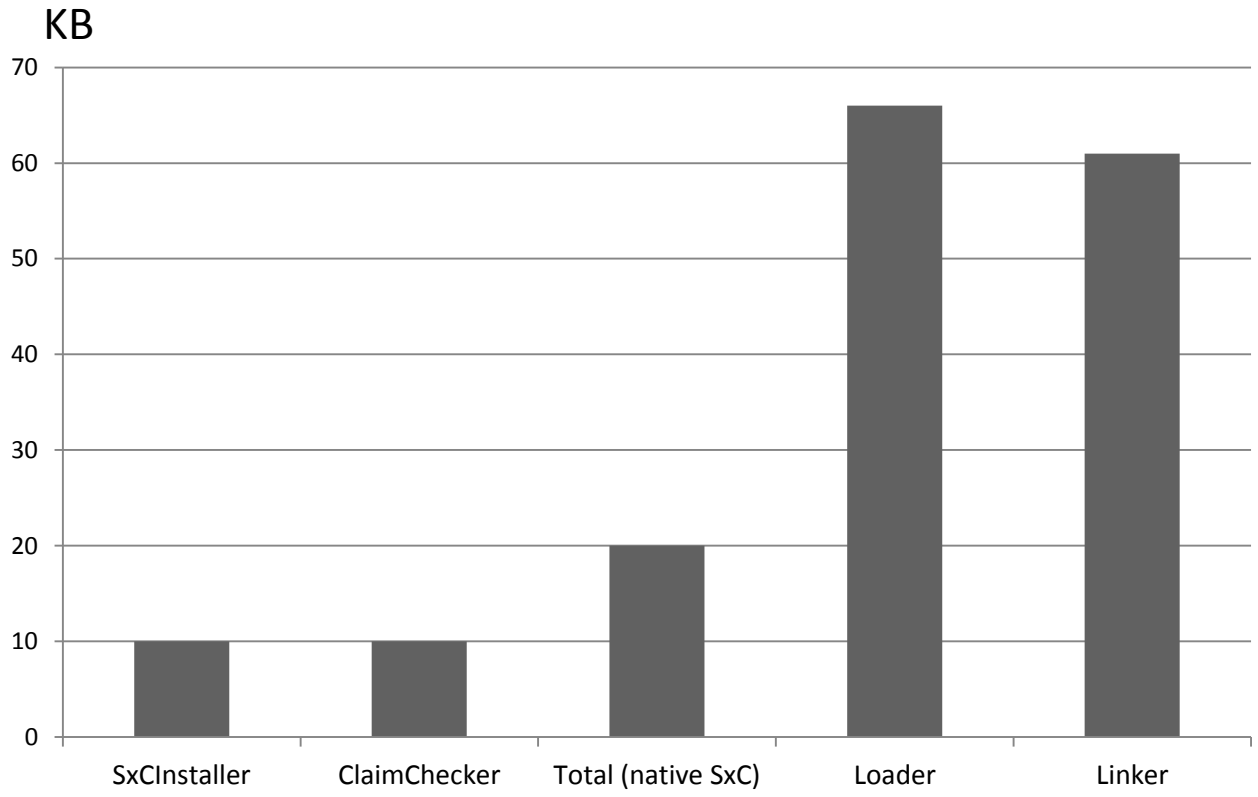
The SxC deployment process does not modify the standard Java Card tools

New applet policy update protocol

UNIVERSITY OF TRENTO - Italy

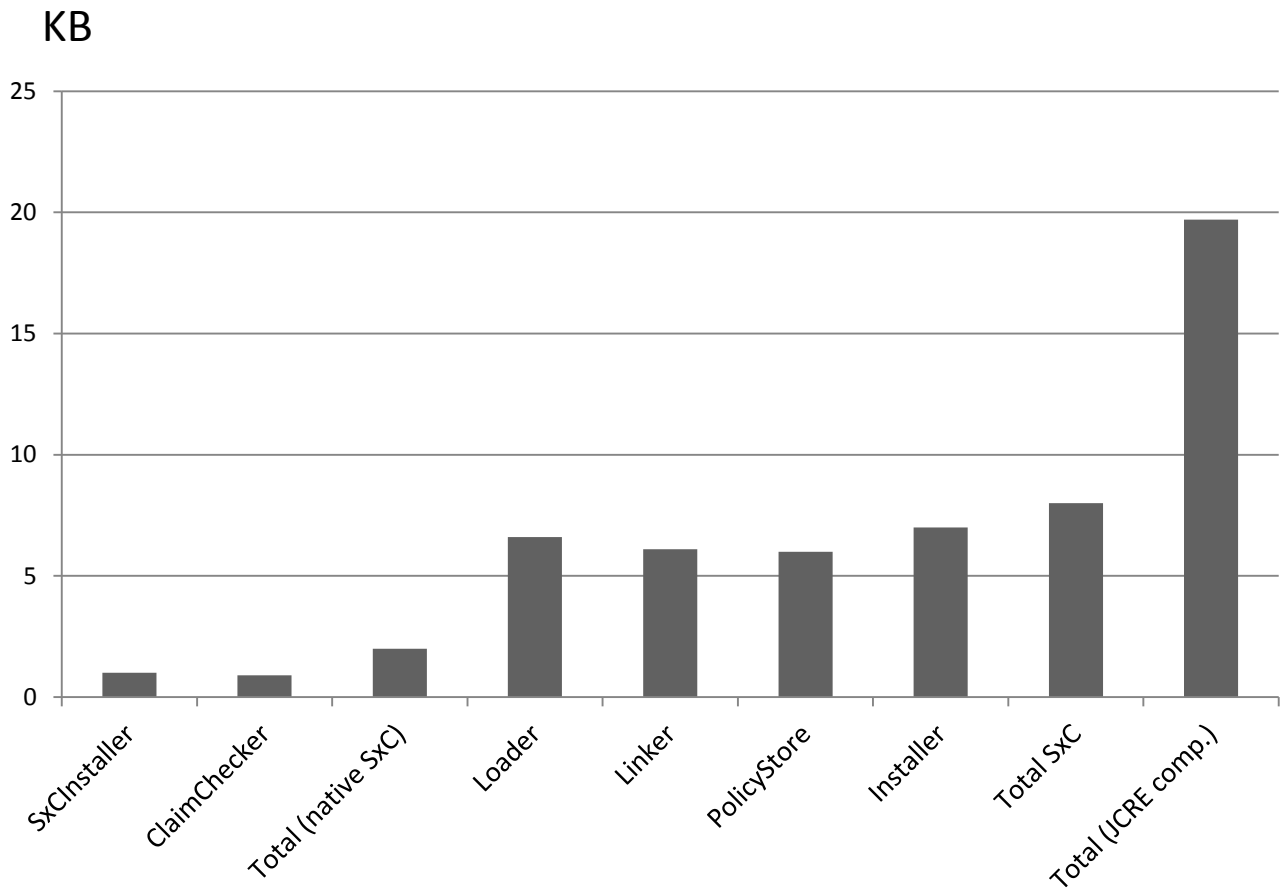


It is small enough



Native components compiled on PC

It really works on the card



On-card components

Works on real applets

UNIVERSITY
OF TRENTO - Italy

Applet	CAP file size	# of methods in CAP file	# of services	LOCs (.java)
Purse	2.5KB	6	1	66
Transport	2.5KB	5	0	92
EID	11.2KB	81	1	1419
ePurse	4.7KB	16	1	431

coop.jca - Notepad

File Edit Format View Help

```
                        L6:     aload_2;
                                sconst_0;
                                getstatic_b 4;          // byte coop/CoopLoyalty.Points
                                bastore;
                                aload_1;
                                sconst_0;
                                sconst_1;
                                invokevirtual 15;               // setOutgoingAndSend(SS)V
                                goto L8;
                        L7:     sspush 27904;
                                invokestatic 11;                // javacard/framework/ISOException.throwIt(S)V
                        L8:     return;
                }

rcharge()V {


                                getfield_a_this 0;              // reference coop/CoopLoyalty.PurseAID
                                sconst_0;
                                getfield_a_this 0;              // reference coop/CoopLoyalty.PurseAID
                                arraylength;
                                s2b;
                                invokestatic 16;
                                astore_1;
                                aload_1;
                                ifnonnull L2;
                                sspush 27013;
                                invokestatic 11;
                                aload_0;
                                aload_1;
                                sconst_2;
                                invokestatic 17;
framework/Shareable;
                                checkcast 0 18;
                                putfield_a 2;
                                getfield_a_this 2;
                                getstatic_b 4;
                                invokeinterface 2 18
                                putstatic_b 4;
                                return;
```

Edit Contract

Edit Provides | Edit Calls/FuncRules | Edit SecRules

```
Provides
---------
Count: 1
Interface token 0x0 and service token 0x0

Calls/FuncRules
---------------
Count: 1
Interface token 0x0 and service token 0x0 from AID 0x1 0x2 0x3 0x4 0x5 0x0

SecRules
---------
Count: 1
Authorise AID 0x1 0x2 0x3 0x4 0x5 0x6 0x7 0x0 to call:
     Interface token 0x0 and service token 0x0
```

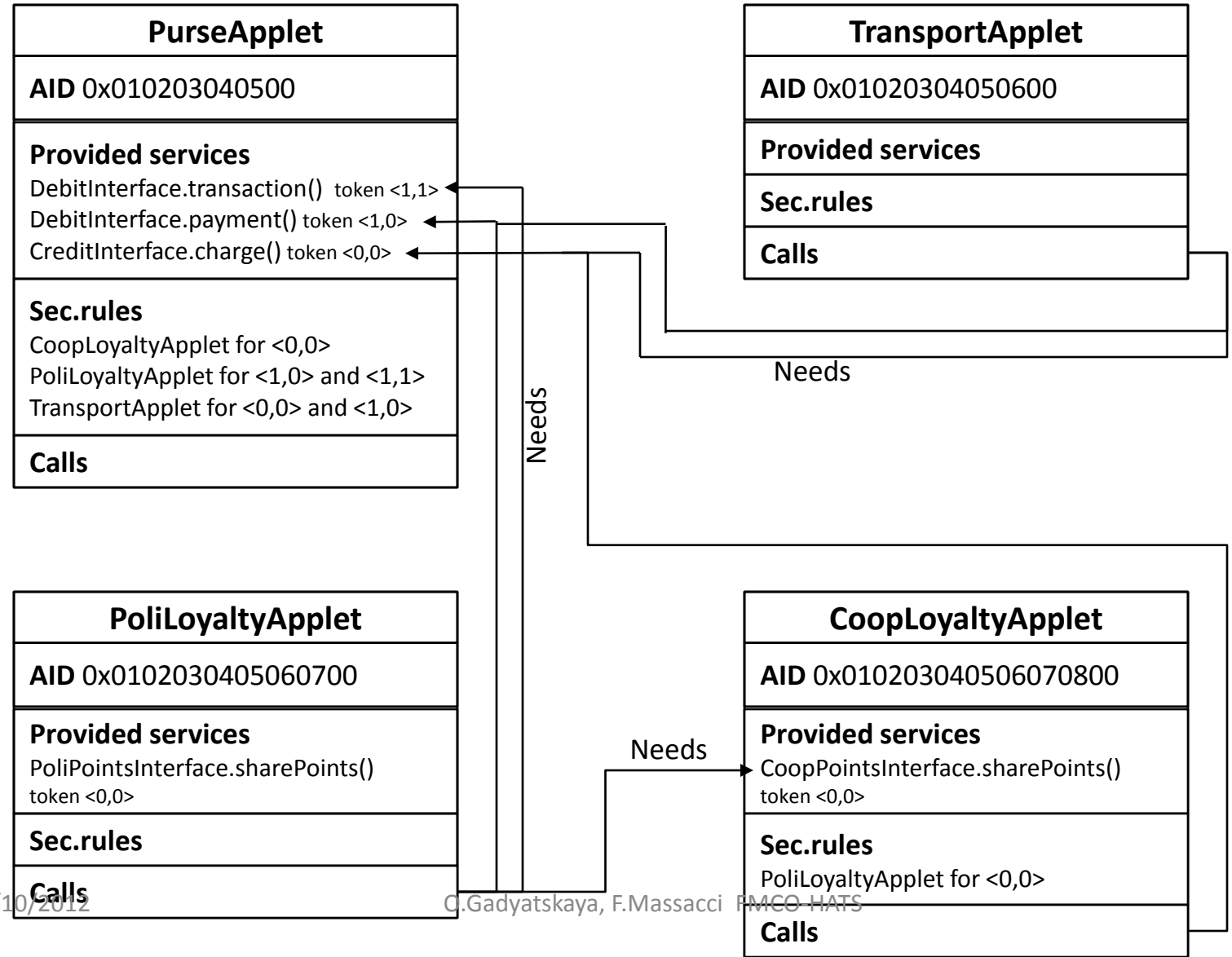
Export | Update CAP | Update Scripts | Main Menu

Command Prompt

```
Waiting for 0 seconds, press a key to continue ...
The update was carried out properly.


C:\Users\Olga\Desktop\DEMO\WorkFolder\2-ClaimChecker Demo>AdditionDemoFullC.bat
coopNonCompliant.cap
Evolution is addition!

ClaimChecker Error: Some callscc1 not present in callscc or tries to call an app
let not yet on the card

The evolution wasn't compliant and was rejected.


C:\Users\Olga\Desktop\DEMO\WorkFolder\2-ClaimChecker Demo>AdditionDemoFullC.bat
coopCompliant.cap
Evolution is addition!

Applets currently on the card (C)
pID 1, AID length 9 , AID 123456780

The evolution was compliant and policy is ready to be updated.


C:\Users\Olga\Desktop\DEMO\WorkFolder\2-ClaimChecker Demo>
```

DEMO

Demo scenario



Agenda

UNIVERSITY
OF TRENTO - Italy

- Motivations and the Security-by-Contract idea
- The Java Card Background
- Contracts
- A (thin) hint of theory
- A (larger) taster of engineering
- Demo
- Conclusions

Industrial conclusions

UNIVERSITY
OF TRENTO - Italy

- VISA is sceptical
- But
  - **less sensitive applets require cheaper validation techniques**

You can find more details in

UNIVERSITY
OF TRENTO - Italy

- [POLICY'2011] N. Dragoni, E. Lostal, O. Gadyatskaya, F. Massacci, F. Paci: *A Load Time Policy Checker for Open Multi-application Smart Cards*
- [ICISS'2011] O. Gadyatskaya, E. Lostal, F. Massacci: *Load Time Security Verification*
- [BYTECODE'2012] O. Gadyatskaya, E. Lostal, F. Massacci: *Extended Abstract: Embeddable Security-by-Contract Verifier for Java Card*
- Some technical reports on my web page **www.unitn.it/~gadyatskaya**

Conclusions

UNIVERSITY
OF TRENTO - Italy

- **SxC framework performs loading time application certification**
  - an applet is accepted only if it respects policies of other deployed applets
- **Security code separated from the functional code**
- **It really works on a smart card**
  - non-invasive addition to the standard Java Card deployment process

UNIVERSITY
OF TRENTO - Italy

secure
CHANGE

Send us your applets …

Olga.Gadyatskaya@unitn.it

Fabio.Massacci@unitn.it