Load-Time Security Certification for Real Smart-Cards

Fabio Massacci
Joint work with O. Gadyaskaya, E. Lostal
University of Trento (IT)

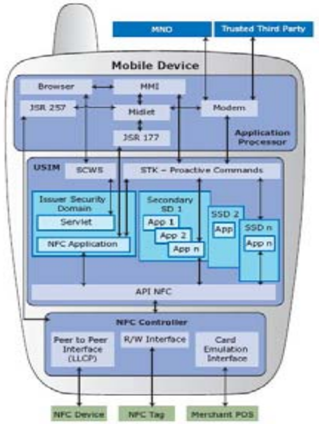
Evaluation by B. Chetali, Q-H. Nguyen
TrustedLabs/Gemalto (FR)

The talk plan

- Motivation & design targets
- The Security-by-Contract Idea
- A (thin) hint of theory
- A (larger) taster of engineering
- Evaluation and challenges

A Marriage for Interest

- M-payments, M-ticketing, M-facebook etc. etc
- Wanna do lots sensitive things on the phone?
  - how ePurse talks to eTicket securely?
- Use the smart card as the secure element!
  - Apps talks within the card securely



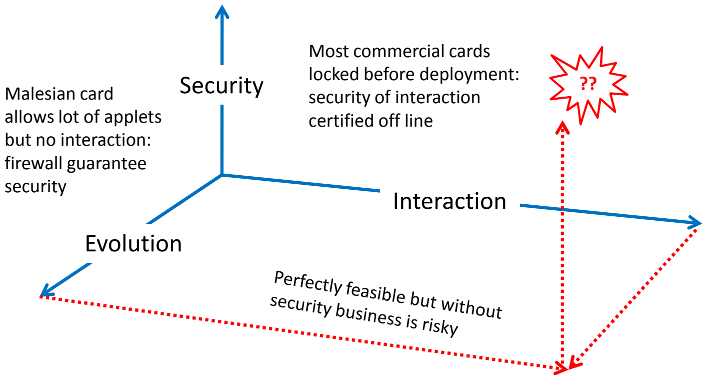
Example : Java Card + GlobalPlatform

- GlobalPlatform = Middleware for secure management of applets (with open specification)
  - Lots of smart cards deployed with GP
- In theory
  - GP supports OTA loading, update and un-loading
  - JC Firewall confines unwanted interactions
  - allow interactions among apps (through Shareable interfaces)
- In practice
  - Few multi-application cards allows independent OTA updates

M-Business wants more…

- **“EIS” – Evolution, Interaction and Security**
  - **E: Own updates are independent (over the air OTA)**
  - **I: Own applet can interact with business partner**
  - **S: Own security policy is respected**
- **Any pair widely popular in the wild, it’s the combination that’s missing**

… but it was not here (yet)



Malesian card allows lot of applets but no interaction: firewall guarantee security

Most commercial cards locked before deployment: security of interaction certified off line

The usual evocative picture

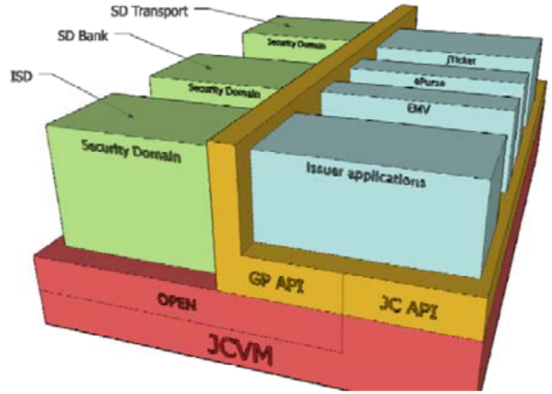
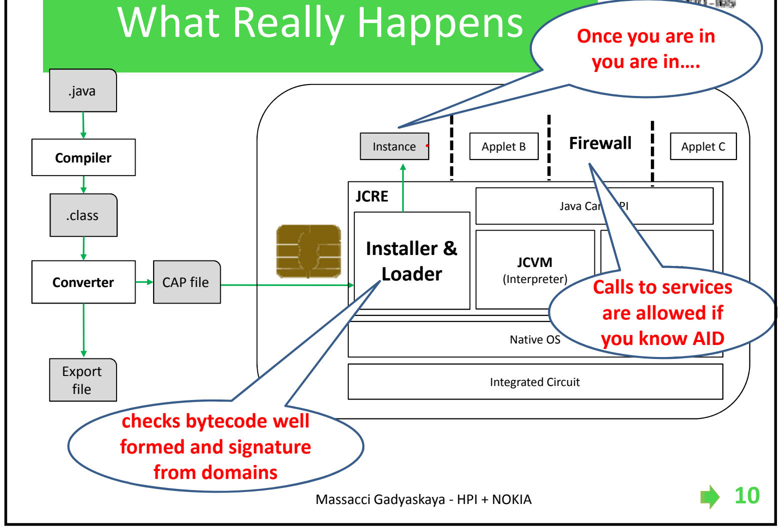


Image from D1.1 of SecureChange project

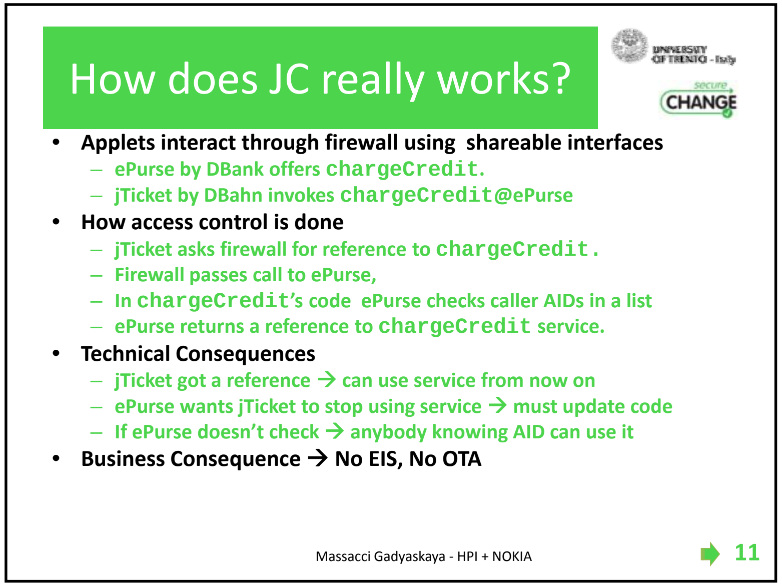
What Really Happens



Once you are in you are in….

Calls to services are allowed if you know AID

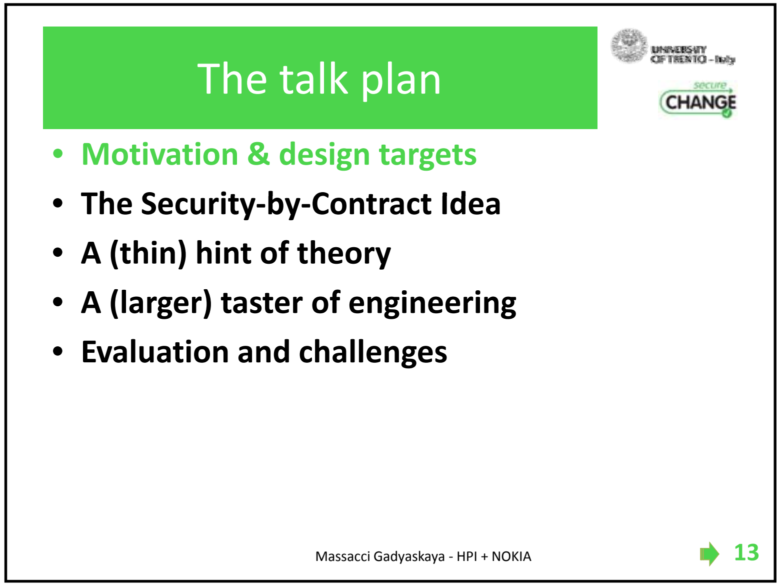
checks bytecode well formed and signature from domains



How does JC really works?

- Applets interact through firewall using shareable interfaces
  - ePurse by DBank offers `chargeCredit`.
  - jTicket by DBahn invokes `chargeCredit@ePurse`
- How access control is done
  - jTicket asks firewall for reference to `chargeCredit`.
  - Firewall passes call to ePurse,
  - In `chargeCredit`'s code ePurse checks caller AIDs in a list
  - ePurse returns a reference to `chargeCredit` service.
- Technical Consequences
  - jTicket got a reference → can use service from now on
  - ePurse wants jTicket to stop using service → must update code
  - If ePurse doesn't check → anybody knowing AID can use it
- Business Consequence → No EIS, No OTA

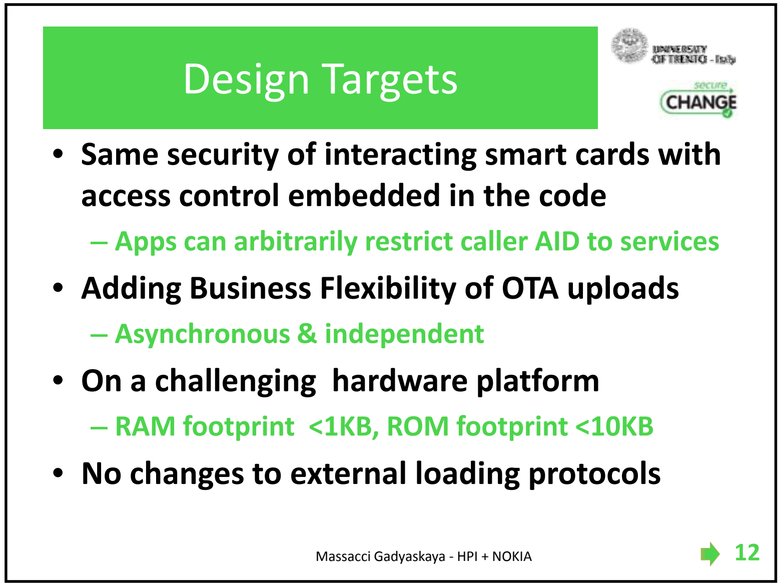
Massacci Gadyaskaya - HPI + NOKIA 11



The talk plan

- Motivation & design targets
- The Security-by-Contract Idea
- A (thin) hint of theory
- A (larger) taster of engineering
- Evaluation and challenges

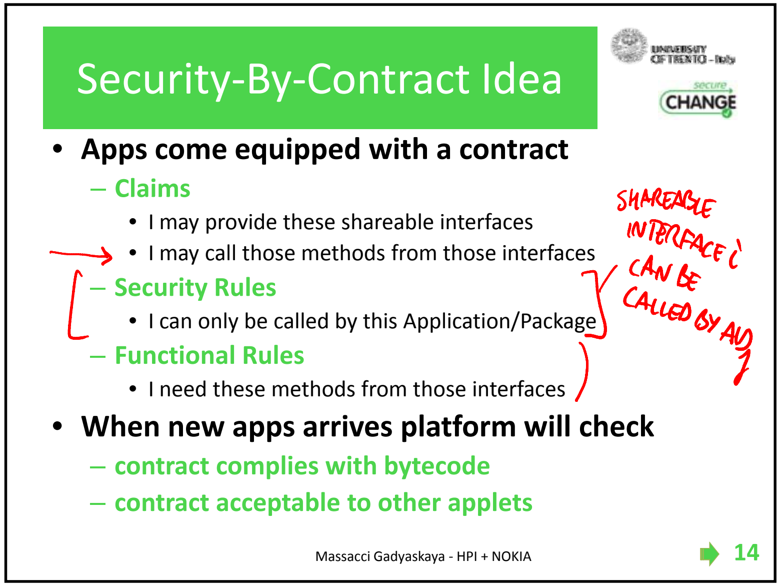
Massacci Gadyaskaya - HPI + NOKIA 13



Design Targets

- Same security of interacting smart cards with access control embedded in the code
  - Apps can arbitrarily restrict caller AID to services
- Adding Business Flexibility of OTA uploads
  - Asynchronous & independent
- On a challenging hardware platform
  - RAM footprint <1KB, ROM footprint <10KB
- No changes to external loading protocols

Massacci Gadyaskaya - HPI + NOKIA 12



Security-By-Contract Idea

- Apps come equipped with a contract
  - Claims
    - I may provide these shareable interfaces
    - I may call those methods from those interfaces
  - Security Rules
    - I can only be called by this Application/Package
  - Functional Rules
    - I need these methods from those interfaces
- When new apps arrives platform will check
  - contract complies with bytecode
  - contract acceptable to other applets

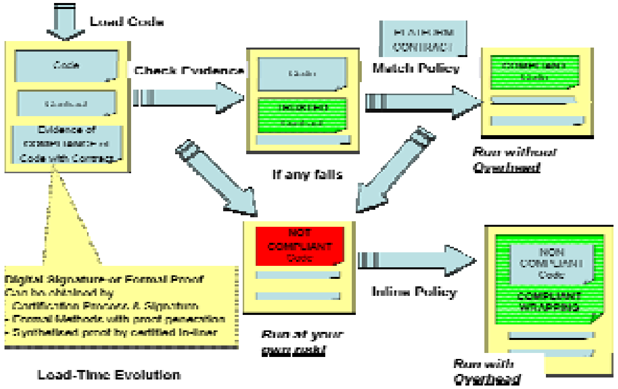
SHAREABLE INTERFACE i CAN BE CALLED BY AID j

Massacci Gadyaskaya - HPI + NOKIA 14

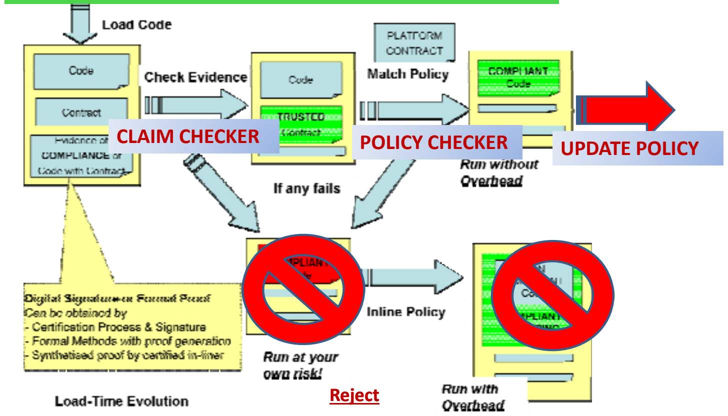
SxC as Load-time verification

- **SxC particular instance of Load Time Verification**
  - **Derived from Proof carrying code CC and Model Carrying Code ideas**
- **Well-tested for mobile platforms**
  - **Java & .NET implementation (KUL, KTH, UNITN)**
  - **Android (Manifest) implementation (Enck et al)**
  - **Published on JCS, JLAP, Comp. & Security, SCP, Elsevier ISTR**
  - **Policy checker could even run a small model checker**
    - “allowed file.size > 1024Kb “ vs “filesize < 512kb”
- **But a smartphone ain’t a card…**

SxC Workflow on Mobile

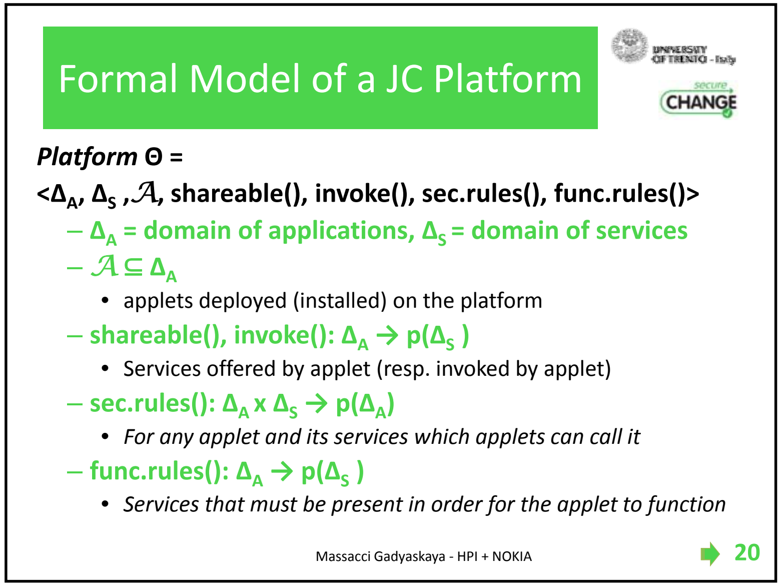


S$_x$C Workflow on Smart Cards



The talk plan

- **Motivation & design targets**
- **The Security-by-Contract Idea**
- **A (thin) hint of theory**
- **A (larger) taster of engineering**
- **Evaluation and challenges**

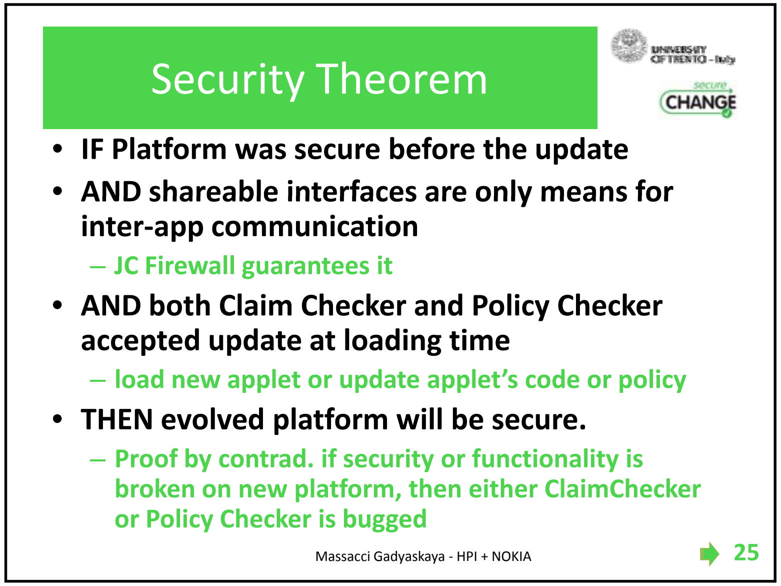


Formal Model of a JC Platform

Platform Θ =

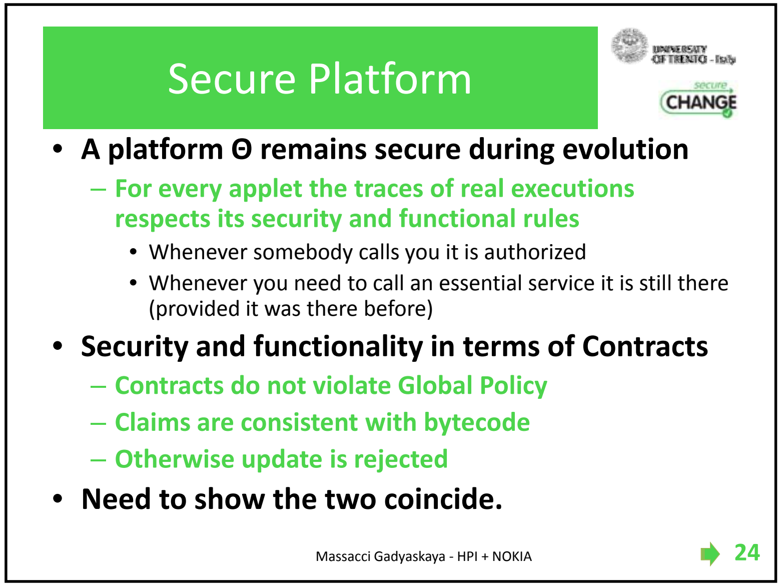
$<\Delta_A, \Delta_S, \mathcal{A}$, shareable(), invoke(), sec.rules(), func.rules()>

- Δ_A = domain of applications, Δ_S = domain of services
- $\mathcal{A} \subseteq \Delta_A$
  - applets deployed (installed) on the platform
- shareable(), invoke(): $\Delta_A \rightarrow p(\Delta_S)$
  - Services offered by applet (resp. invoked by applet)
- sec.rules(): $\Delta_A \times \Delta_S \rightarrow p(\Delta_A)$
  - *For any applet and its services which applets can call it*
- func.rules(): $\Delta_A \rightarrow p(\Delta_S)$
  - *Services that must be present in order for the applet to function*



Security Theorem

- IF Platform was secure before the update
- AND shareable interfaces are only means for inter-app communication
  - JC Firewall guarantees it
- AND both Claim Checker and Policy Checker accepted update at loading time
  - load new applet or update applet's code or policy
- THEN evolved platform will be secure.
  - Proof by contrad. if security or functionality is broken on new platform, then either ClaimChecker or Policy Checker is bugged

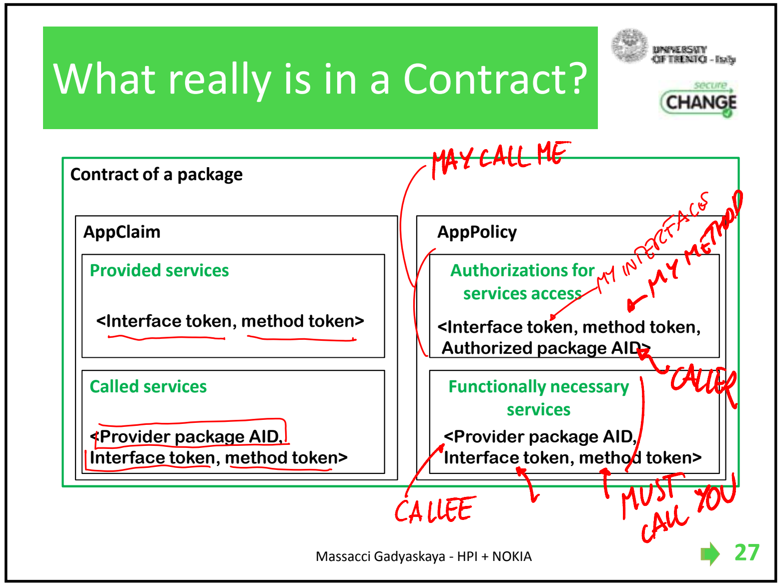


Secure Platform

- A platform Θ remains secure during evolution
  - For every applet the traces of real executions respects its security and functional rules
    - Whenever somebody calls you it is authorized
    - Whenever you need to call an essential service it is still there (provided it was there before)
- Security and functionality in terms of Contracts
  - Contracts do not violate Global Policy
  - Claims are consistent with bytecode
  - Otherwise update is rejected
- Need to show the two coincide.

The talk plan

- Motivation & design targets
- The Security-by-Contract Idea
- A (thin) hint of theory
- A (larger) taster of engineering
- Evaluation and challenges



What really is in a Contract?

Contract of a package

AppClaim

Provided services

<Interface token, method token>

Called services

<Provider package AID, Interface token, method token>

AppPolicy

Authorizations for services access

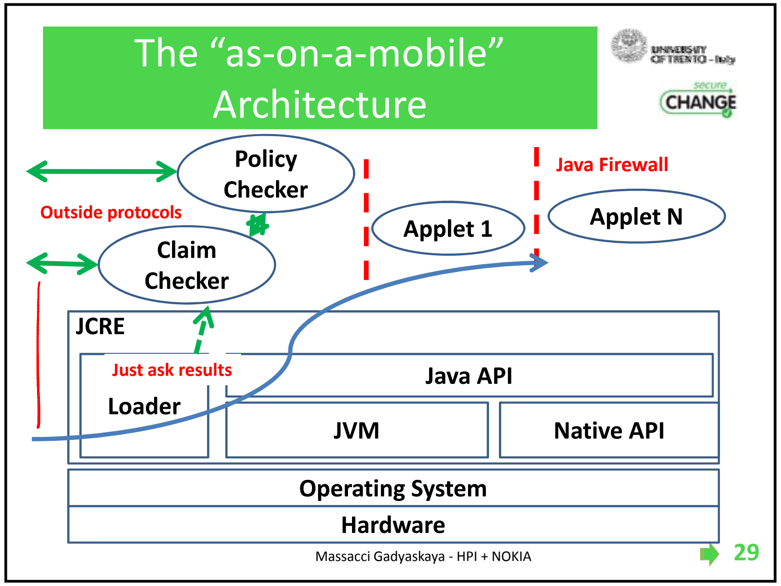
<Interface token, method token, Authorized package AID>

Functionally necessary services

<Provider package AID, Interface token, method token>

Massacci Gadyaskaya - HPI + NOKIA

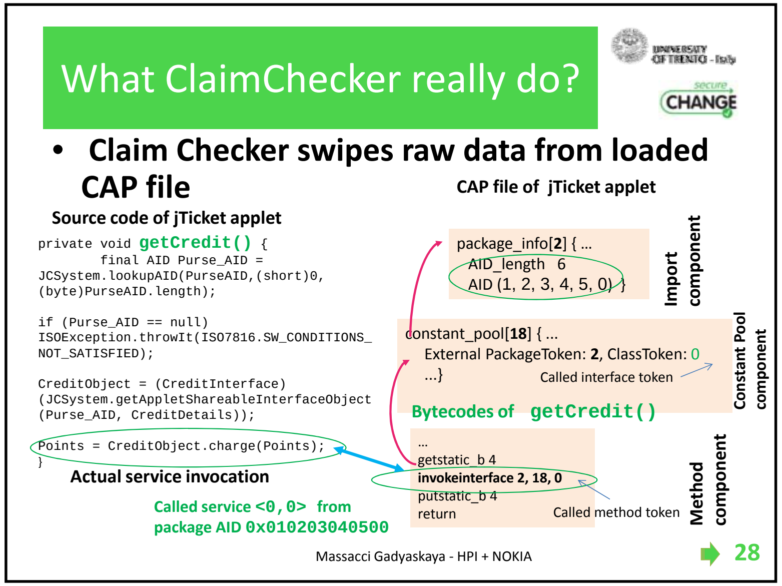
27



The "as-on-a-mobile" Architecture

Massacci Gadyaskaya - HPI + NOKIA

29



What ClaimChecker really do?

- Claim Checker swipes raw data from loaded CAP file

Source code of jTicket applet

```
private void getCredit() {
        final AID Purse_AID =
JCSystem.lookupAID(PurseAID,(short)0,
(byte)PurseAID.length);

if (Purse_AID == null)
ISOException.throwIt(ISO7816.SW_CONDITIONS_
NOT_SATISFIED);

CreditObject = (CreditInterface)
(JCSystem.getAppletShareableInterfaceObject
(Purse_AID, CreditDetails));

Points = CreditObject.charge(Points);
}
```

Actual service invocation

Called service <0,0> from package AID 0x010203040500

CAP file of jTicket applet

```
package_info[2] { ...
  AID_length  6
  AID (1, 2, 3, 4, 5, 0) }
```

Import component

```
constant_pool[18] { ...
  External PackageToken: 2, ClassToken: 0
  ...}
```

Called interface token

Constant Pool component

Bytecodes of getCredit()

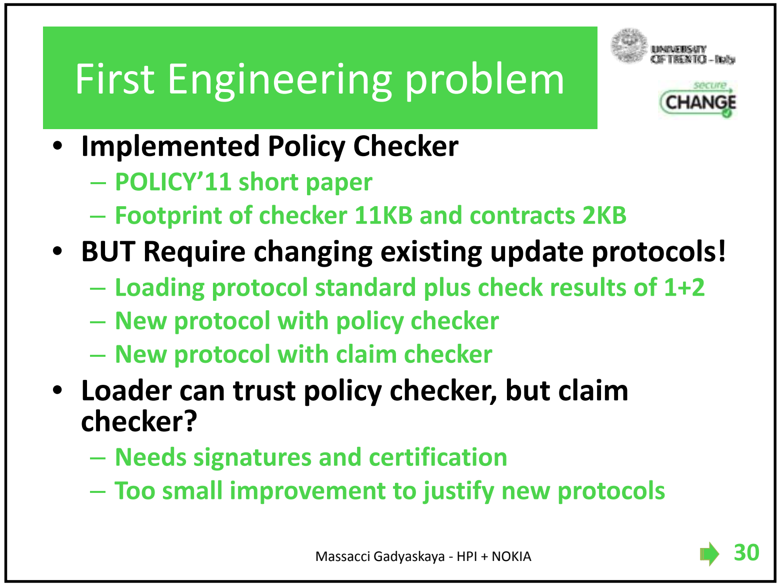
```
...
getstatic_b 4
invokeinterface 2, 18, 0
putstatic_b 4
return
```

Called method token

Method component

Massacci Gadyaskaya - HPI + NOKIA

28



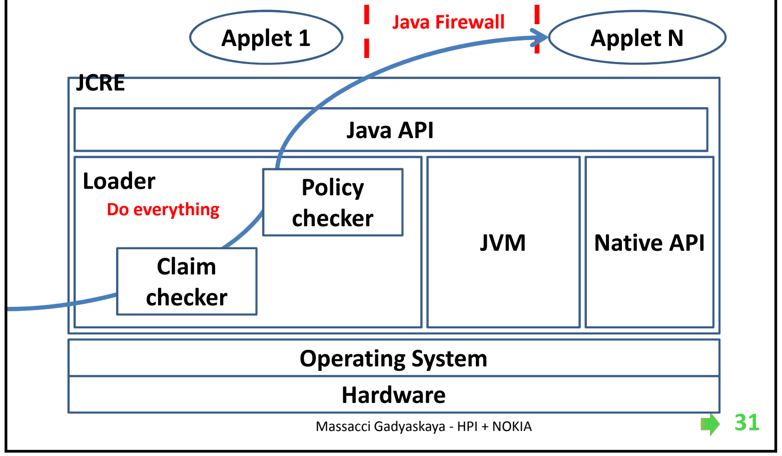
First Engineering problem

- Implemented Policy Checker
  - POLICY'11 short paper
  - Footprint of checker 11KB and contracts 2KB
- BUT Require changing existing update 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

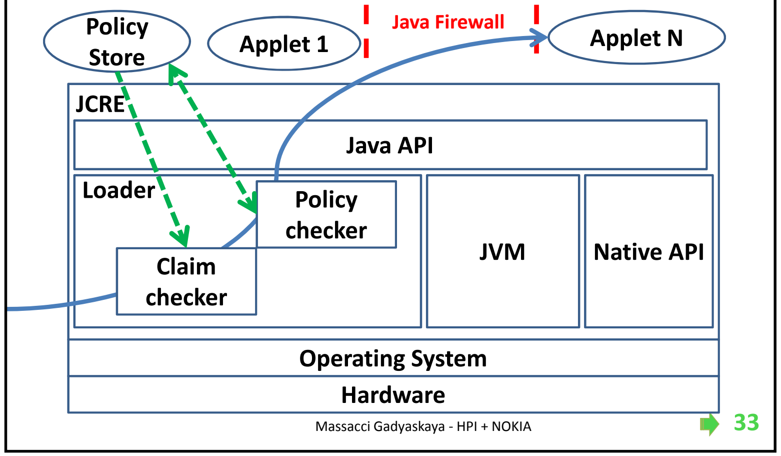
Massacci Gadyaskaya - HPI + NOKIA

30

Our Second Architecture



Our Third Architecture

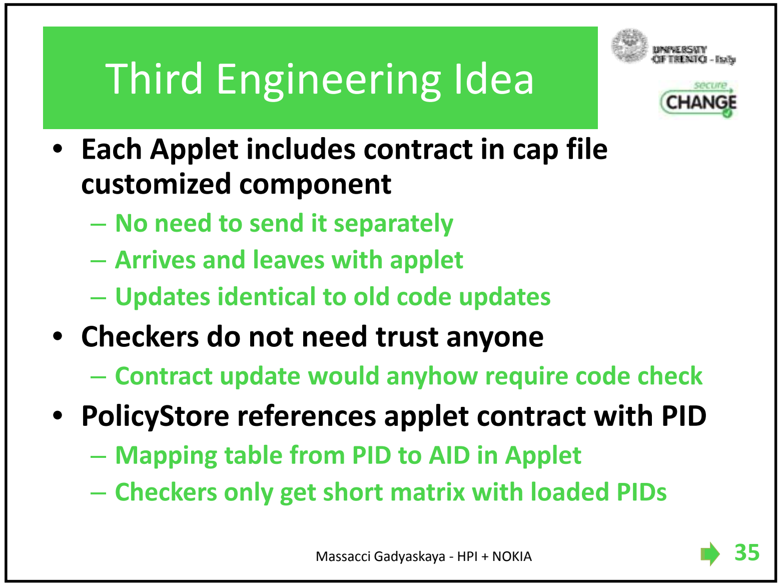


Second Engineering Problem

- 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 loader is "printed"
    - We could have a "static int policy[]'' but that's not going to work in the ROM

Third Engineering Problem

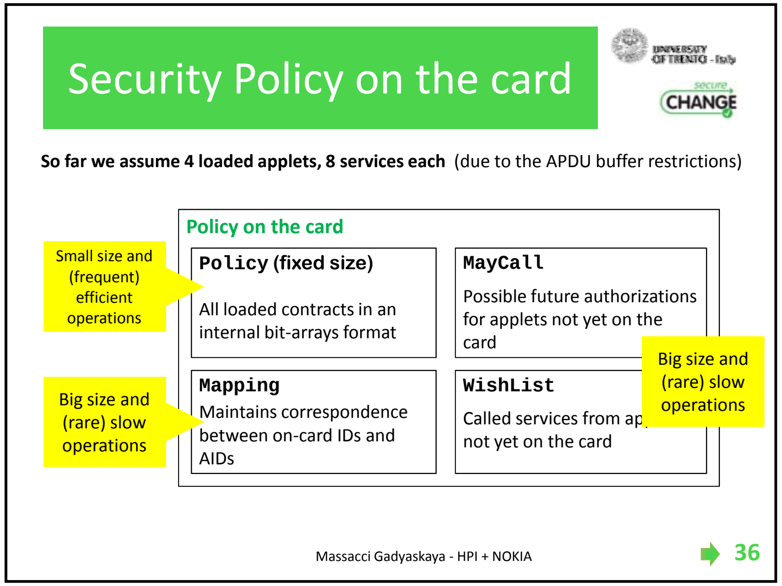
- Who's giving the contract to the checker?
  - Can't change the protocol of updates…
- Both Checkers needs Applets AIDs…
  - AIDs are "big" → access matrix won't fit in RAM
  - AIDs only known at OTA's time → can't "print" them in Loader
- A bit of help from the platform
  - AID are mapped into Package ID (much shorter)
  - But still you have rules for AIDs not yet on board



Third Engineering Idea

- Each Applet includes contract in cap file customized component
  - No need to send it separately
  - Arrives and leaves with applet
  - Updates identical to old code updates
- Checkers do not trust anyone
  - Contract update would anyhow require code check
- PolicyStore references applet contract with PID
  - Mapping table from PID to AID in Applet
  - Checkers only get short matrix with loaded PIDs

Massacci Gadyaskaya - HPI + NOKIA 35



Security Policy on the card

So far we assume 4 loaded applets, 8 services each (due to the APDU buffer restrictions)

Policy on the card

Small size and (frequent) efficient operations

`Policy` (fixed size)

All loaded contracts in an internal bit-arrays format

`MayCall`

Possible future authorizations for applets not yet on the card

Big size and (rare) slow operations

`Mapping`

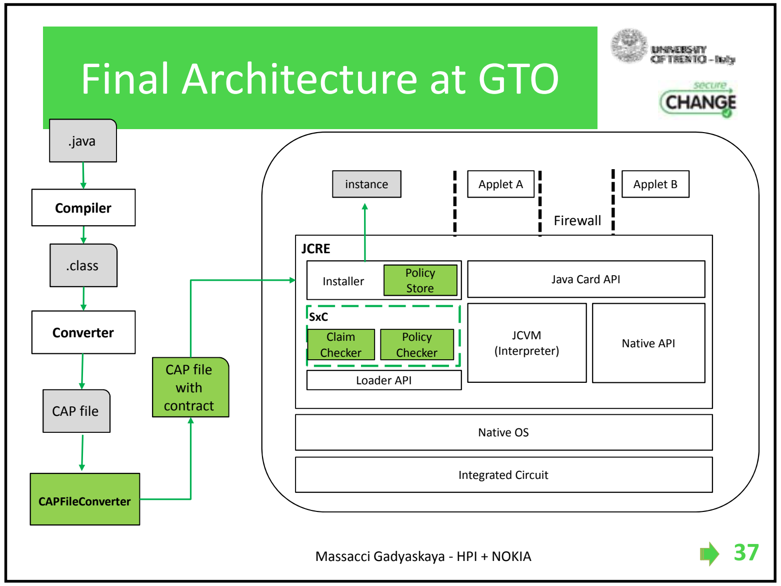
Maintains correspondence between on-card IDs and AIDs

`WishList`

Called services from ap not yet on the card

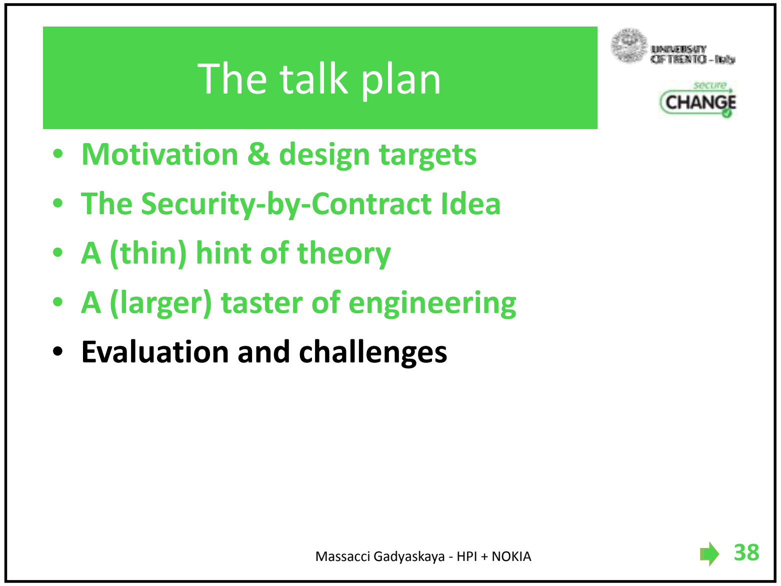
Big size and (rare) slow operations

Massacci Gadyaskaya - HPI + NOKIA 36



Final Architecture at GTO

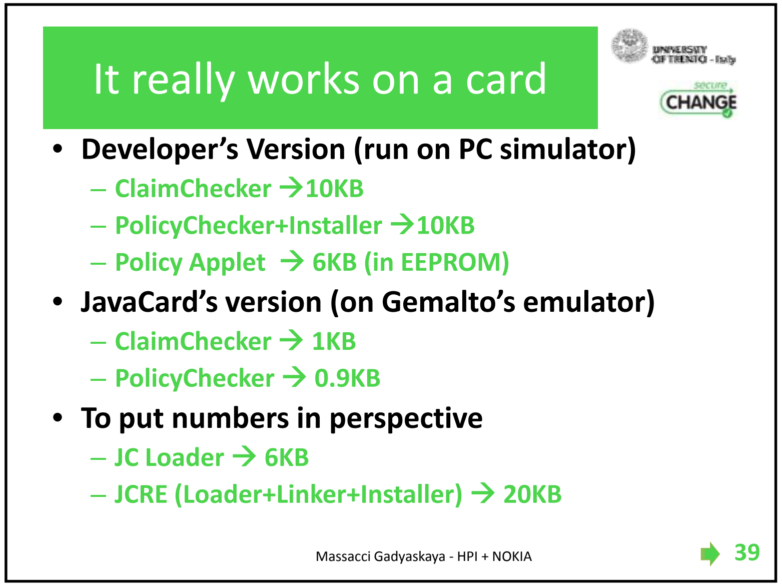
Massacci Gadyaskaya - HPI + NOKIA 37



The talk plan

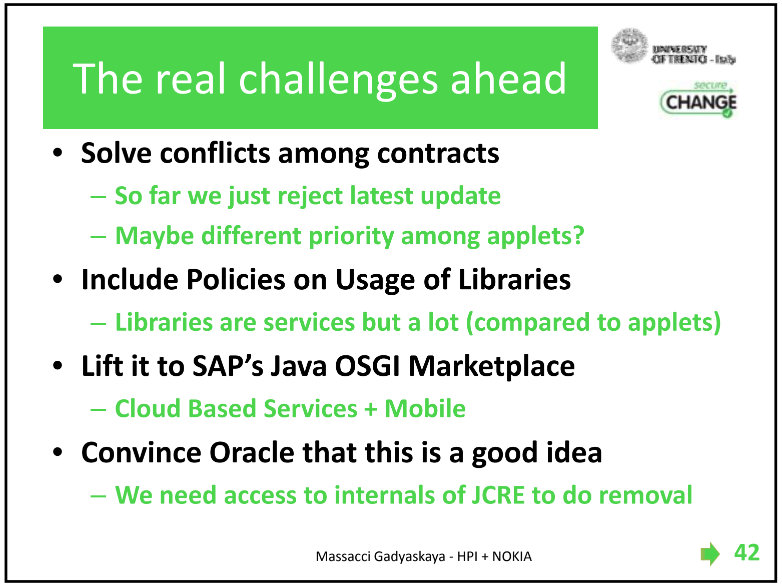
- Motivation & design targets
- The Security-by-Contract Idea
- A (thin) hint of theory
- A (larger) taster of engineering
- **Evaluation and challenges**

Massacci Gadyaskaya - HPI + NOKIA 38



It really works on a card

- Developer's Version (run on PC simulator)
  - ClaimChecker →10KB
  - PolicyChecker+Installer →10KB
  - Policy Applet → 6KB (in EEPROM)
- JavaCard's version (on Gemalto's emulator)
  - ClaimChecker → 1KB
  - PolicyChecker → 0.9KB
- To put numbers in perspective
  - JC Loader → 6KB
  - JCRE (Loader+Linker+Installer) → 20KB



The real challenges ahead

- Solve conflicts among contracts
  - So far we just reject latest update
  - Maybe different priority among applets?
- Include Policies on Usage of Libraries
  - Libraries are services but a lot (compared to applets)
- Lift it to SAP's Java OSGI Marketplace
  - Cloud Based Services + Mobile
- Convince Oracle that this is a good idea
  - We need access to internals of JCRE to do removal

UNIVERSITY OF TRENTO - Italy

secure CHANGE

Send us your applets …

Fabio.Massacci@unitn.it

www.massacci.org