Security-by-Contract for Applications’ Evolution in Multi-Application Smart Cards

Nicola Dragoni
ndra@imm.dtu.dk
http://www2.imm.dtu.dk/~ndra
Embedded Systems Engineering (ESE) Section
DTU Informatics
Technical University of Denmark (DTU)
Talk Outline

- **Domain**: multi-application smart cards
- **Problem**: supporting applications’ evolution
- **Approach**: Security-by-Contract (key idea)
Multi-Application Smart Cards

- Multi-application smart cards: several applications run on the same card
- Applications (Web clients and servers) are owned and asynchronously controlled by different stakeholders
- Applications can dynamically be loaded, changed and removed during the active life of the card
JAVA CARD = JRE + GLOBALPLATFORM

- GlobalPlatform = Middleware (with Open Specifications)
  - Lots of smart card deployed with those specifications

- Java Card = GlobalPlatform + Java Runtime Environment
  - Support loading and unloading of many applications on the fly and asynchronously
  - Allow interactions among applications on the card (through Shareable Interface inherited services)
JAVA CARD = JRE + GLOBALPLATFORM

- GlobalPlatform = Middleware (with Open Specifications)
  - Lots of smart card deployed with those specifications

- Java Card = GlobalPlatform + Java Runtime Environment
  - Support loading and unloading of many applications on the fly and asynchronously
  - Allow interactions among applications on the card (through Shareable Interface inherited services)

- Still/Yet/But…
  - There is NO fielded open multi-application smart card
Applications run in dedicated security domains. The name is evocative of a separate space (such as in a virtual machine) but in reality a domain just supports some security services...
Security Domains??!!

From GP specification:

**Security Domains** act as the on-card representatives of off-card authorities.

**Security Domains** support security services such as key handling, encryption, decryption, digital signature generation and verification for their providers' (Card Issuer, Application Provider or Controlling Authority) applications.

Each **Security Domain** is established on behalf of a Card Issuer, an Application Provider or a Controlling Authority when these off-card entities require the use of keys that are completely isolated from each other.
Problem: Control of Interactions Among Applications

- Policy for security domains:
  - Only Bank can be called by Transport
  - Transport will only call Bank

- Policy for applications:
  - EMV@Bank will only be called by ePurse@Bank
  - ePurse@Bank can only be called by jTicket@Transport and will only call EMV@Bank
  - jTicket@Transport will only call ePurse@Bank

- New application Points@Loyalty arrives on the platform. Desired behavior:
  - Points@Loyalty will only call ePurse@Bank
  - And here we have a problem!
Ok... But... Wait a Minute... GP??!!

- GP does not solve the problems of illegal information exchange even for the applications from different security domains
- All inter-application interactions are pushed to lower levels ==> runtime environment or even hardware
- Example: in Java Card, the control of the communications between the applications and the applications and the platform rests on the JRE!!
Yes... Ok... But... Wait a Minute... JRE Firewall??!!

- JRE has a firewall security mechanism that **isolates** each applet from the other applets within its own context!!

  ▶ The internal operations of an applet have **no effect** on other applets embedded on the card!!
Yes... Ok... But... Wait a Minute... JRE Firewall??!!

• JRE has a firewall security mechanism that isolates each applet from the other applets within of its own context!!

  ‣ The internal operations of an applet have no effect on other applets embedded on the card!!

  ‣ Still, applications can interact in this environment by explicitly implementing shareable methods callable via an API!! (Application service in Java Card 3.0 specification or Global Services in the GP specification)

• If application A knows shareable interface of application B, then it may use it for its own purposes, and there is no means for B or B security domain owner to prevent it, unless special controls are hacked into the Java firewall

  ‣ However this completely prevents the asynchronous download or update of different applications!!
What About the Available Business Solutions?

- There are business solutions for multi-application smart cards on top of GP and Java Card from Venyon Oy, Gemalto and companies alike developed for banking, transport and mobile operators.

- **But** typical solution from such companies is only responsible for:
  - handling loading of card customer applications
  - security domain key handling
  - management and removal of applications

- Such a solution is only an improvement of GP, but it is not dealing with:
  - certification of new applications on the card
  - checks of compliance with new applications to the initial card security policy
  - checks if the removal of some application is even possible and will not break the work of others remaining on the card
Back to “The” Problem

• What remains out of reach is a secure way to deploy new applications on the multi-application smart card once it is in the field

• A costly manual review is necessary!

• What owners of different trust domains wants: to make sure their applications cannot be accessed by new applications added after theirs!

• What smart card developers have to prove: all the changes that are possible to apply to the card are security-free!!

  ‣ In this way their formal proof of compliance with Common Criteria is still valid after that changes and they do not need to obtain a new certificate...
“The” (Security) Requirement of Smart Cards

• Java Card applications must be Common Criteria certified to respect a certain policy of each stakeholder
  ‣ Pre-issuance certification when the card is prepared
  ‣ All later changes must show they meet the same policy
“The” (Security) Requirement of Smart Cards

• Java Card applications must be Common Criteria certified to respect a certain policy of each stakeholder
  ‣ Pre-issuance certification when the card is prepared
  ‣ All later changes must show they meet the same policy

• Solution 1 – Theory
  ‣ Certify the application for all possible changes of itself AND its fellow applications
“The” (Security) Requirement of Smart Cards

- Java Card applications must be Common Criteria certified to respect a certain policy of each stakeholder
  - Pre-issuance certification when the card is prepared
  - All later changes must show they meet the same policy

- Solution 1 – Theory
  - Certify the application for all possible changes of itself AND its fellow applications

- Solution 2 – Another Theory
  - Run-time monitor new applications to prevent their interactions with old applications if it’s forbidden
“The” (Security) Requirement of Smart Cards

• Java Card applications must be Common Criteria certified to respect a certain policy of each stakeholder
  ‣ Pre-issuance certification when the card is prepared
  ‣ All later changes must show they meet the same policy

• Solution 1 – Theory
  ‣ Certify the application for all possible changes of itself AND its fellow applications

• Solution 2 – Another Theory
  ‣ Run-time monitor new applications to prevent their interactions with old applications if it’s forbidden

• Solution 3 – Practice
  ‣ Don’t allow post-issuance evolution…
What We Want: Supporting Applications’ Evolution

• Download new applications, delete old applications, update applications, update/change policy of the smart card

• Applications are signed and come with their own “policy”, describing their security behavior

• New applications should

  ✓ not interact with some “old” applications (if that is not wanted)

  ✓ interact with other “old” applications (if that is wanted)
What We Want: Supporting Applications’ Evolution

• Download new applications, delete old applications, update applications, update/change policy of the smart card

• Applications are signed and come with their own “policy”, describing their security behavior

• New applications should

  ✓ not interact with some “old” applications (if that is not wanted)

  ✓ interact with other “old” applications (if that is wanted)

We want to support applications’ evolution in multi-application smart cards
Possible Interactions

- **Policy for security domains:**
  - SD1 cannot call SD2
  - SD3 only can be called after a call to SD2

- **Applications installed on the platform:**
  - Application A with services ID1, ID2 belongs to SD2
  - Application B with service ID3 belongs to SD3
Possible Interactions

- **Policy for security domains:**
  - SD1 cannot call SD2
  - SD3 only can be called after a call to SD2
- **Applications installed on the platform:**
  - Application A with services ID1, ID2 belongs to SD2
  - Application B with service ID3 belongs to SD3
- **New application C arrives on the platform. Desired behavior:**
  - C will only call shareable interfaces ID1, ID2, ID3
  - C will only call shareable interface ID
  - C will only call ID2 after calling ID3
- **Advanced Desired Behavior:**
  - Information flow only TO and FROM service ID1 at any point
  - Call Flow TO service ID2 only after service call FROM ID3
Approach: Security-By-Contract

• Stolen from Meyer’s Programming-by-Contract and Model-Carrying-Code
  ‣ Works for Mobile Systems! (S3MS project)

• Applications come with a contract describing its security relevant behavior
  ‣ Security API + Shareable Interfaces
    ‣ Needed calls to other security domains and/or applications
    ‣ Allowed calls by other security domains and/or applications
    ‣ Forbidden calls by other security domains and/or applications

• Security policy is represented in the same model as contract, so it is possible to check contract and policy for compliance
SxC Workflow: Code Carries with Contract

Load Code

Check Evidence

Y/N
SxC Workflow: Code Carries with Contract

Load Code

Check Evidence

Y/N

Y

Match?

SC Policy

Load Code

Code

Contract

Evidence of Compliance

SC Policy

02234 - Autumn 2010
SxC Workflow: Code Carries with Contract

Load Code

Check Evidence

Y/N

Y

MATCH?

N

SC POLICY

COMPLIANT CODE

CONTRACT

EVIDENCE

RUN WITHOUT OVERHEAD

NOT COMPLIANT CODE

CONTRACT

EVIDENCE

RUN AT YOUR RISK!

Load Code

CONTRACT

EVIDENCE OF COMPLIANCE

Y/N

N

Y

Y

N

N

TRUSTED CONTRACT

EVIDENCE

CODE

CODE
SxC Workflow: Code Carries with Contract

Load Code

CODE

CONTRACT

EVIDENCE OF COMPLIANCE

Check Evidence

Y/N

N

MATCH?

Y

N

SC POLICY

NOT COMPLIANT CODE

CONTRACT

EVIDENCE

RUN AT YOUR RISK!

Inline Policy

NOT COMPLIANT CODE

COMPLIANT WRAPPING

CONTRACT

EVIDENCE

RUN WITH OVERHEAD

COMPLIANT CODE

CONTRACT

EVIDENCE

RUN WITHOUT OVERHEAD

19
SxC Workflow: Contract Extracted From Code

Load Code

Contract Extraction

Code

Trusted Contract

Match?

Y

N

Compliant Code

Contract

Run Without Overhead

Inline Policy

Not Compliant Code

Contract

Run At Your Risk!

Compliant Wrapping

Contract

Run With Overhead

SC Policy
On-Card Contract-Policy Matching

Verification process for a new application

02234 - Autumn 2010
Off-Card Contract-Policy Matching

Verification process for a new application

02234 - Autumn 2010
Catalogue of Possible Research Projects
1 - Formalization of Trust

X accepts Y only if Y’s behavior is trusted
1 - Formalization of Trust

X accepts Y only if Y’s behavior is trusted

- Ask Nicola for a paper
- Think about a possible formalization of Trust
- Write down your idea
- Work on the details
2 - Security Domains

- Our prototype does not consider Security Domains yet...
  
- Extend the design of the system (new protocols) in order to take Security Domains into consideration
2 - Security Domains

• Our prototype does not consider Security Domains yet...

› Extend the design of the system (new protocols) in order to take Security Domains into consideration

☑ Ask Nicola for a MSc Thesis + documentation
☑ Think about a possible extension of the system
☑ Write down your idea
☑ Work on the details
Thanks! Questions?

“I don’t have any solution, but I certainly admire the problem”
(Ashleigh Ellwood Brilliant)

Nicola Dragoni
ndra@imm.dtu.dk
http://www2.imm.dtu.dk/~ndra
Embedded Systems Engineering (ESE) Section
DTU Informatics
Technical University of Denmark (DTU)