



# Vers la certification des applications Java Card<sup>TM</sup>



Damien SAUVERON

sauveron@labri.u-bordeaux.fr

<http://dept-info.labri.u-bordeaux.fr/~sauveron>

# Plan

---

## La certification

- Le schéma français
- L'échelle d'assurance
- Périmètre d'évaluation et les problèmes de la multi-application

## Le projet Sécurité Java Card

- Besoin de SERMA Technologies
- JCAT Tools
- JCAT Emulator
- De la théorie à la pratique
- Formalisation

## Perspectives

# La certification

---

## Objectifs :

- ☞ Garantir le niveau de sécurité d'un produit TI (confidentialité, intégrité, disponibilité).
- ☞ Bénéficier des accords de reconnaissance mutuelle.

## Avantages pour :

- les utilisateurs : comparer sur une base objectif ;
- les industriels : prouver leur compétence et d'étendre leurs marchés ;
- les autorités d'homologation : s'assurer que les objectifs de sécurité sont satisfaits.

## Le schéma français

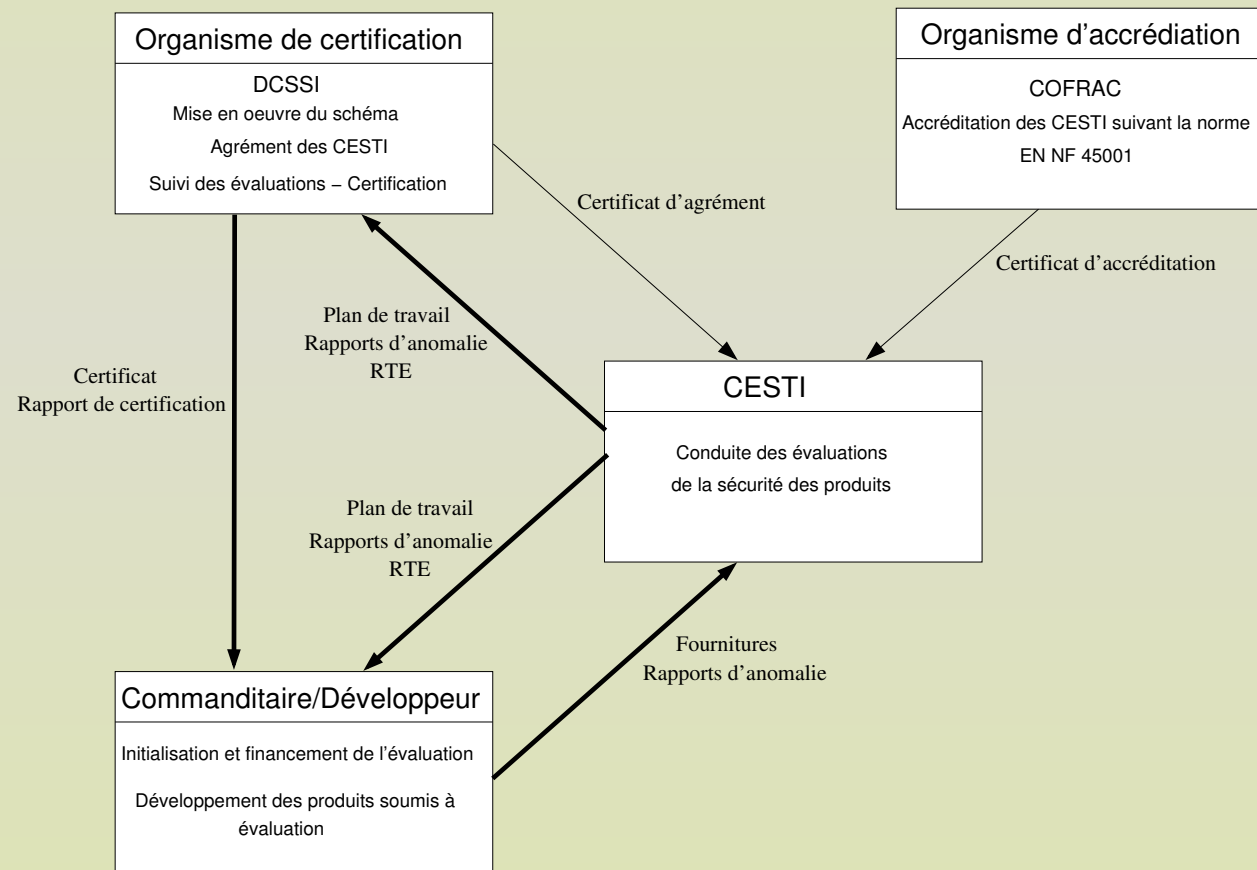


FIG. 1 – Rôles des différents intervenants et échanges d'information

## L'échelle d'assurance

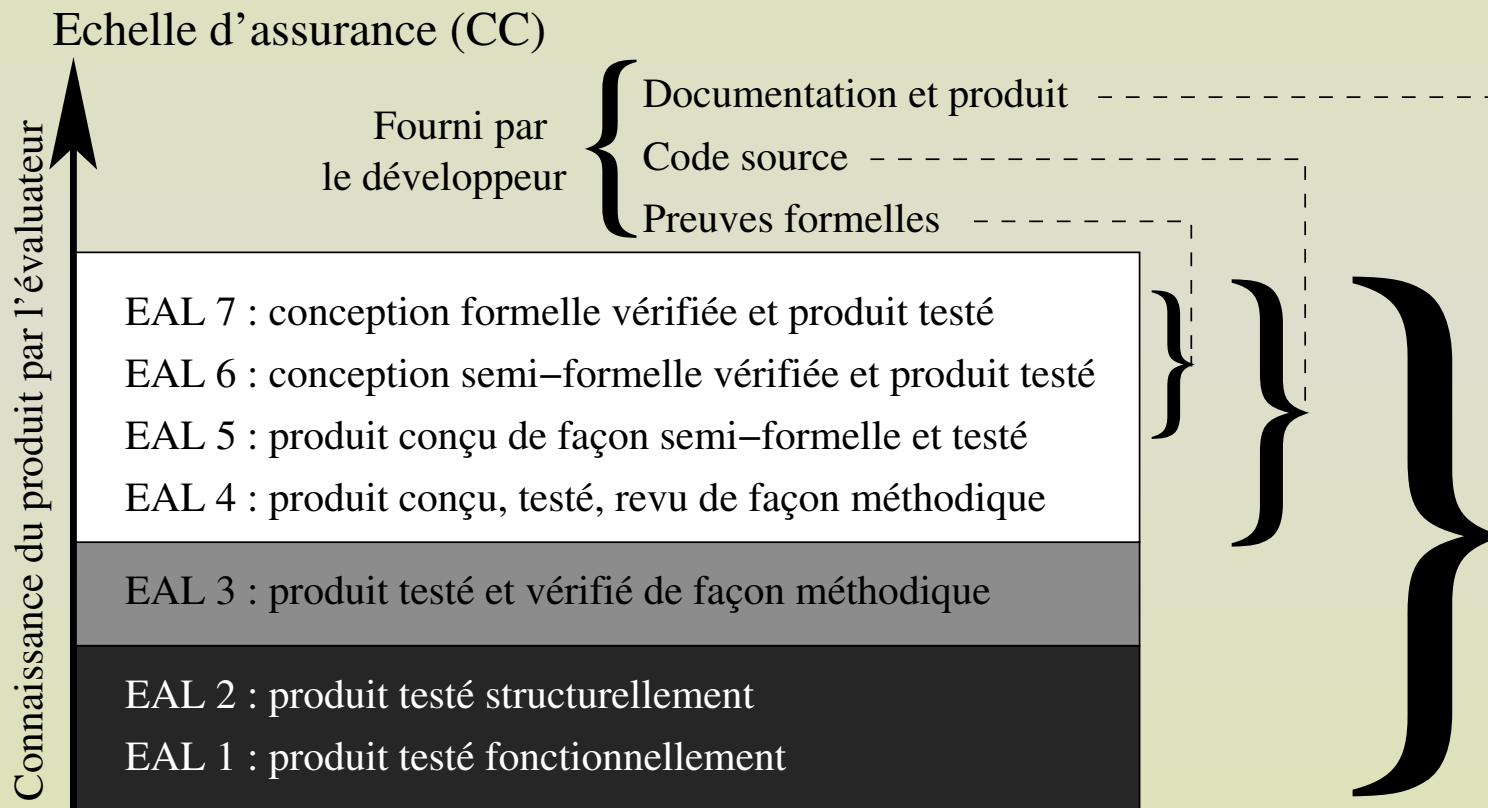
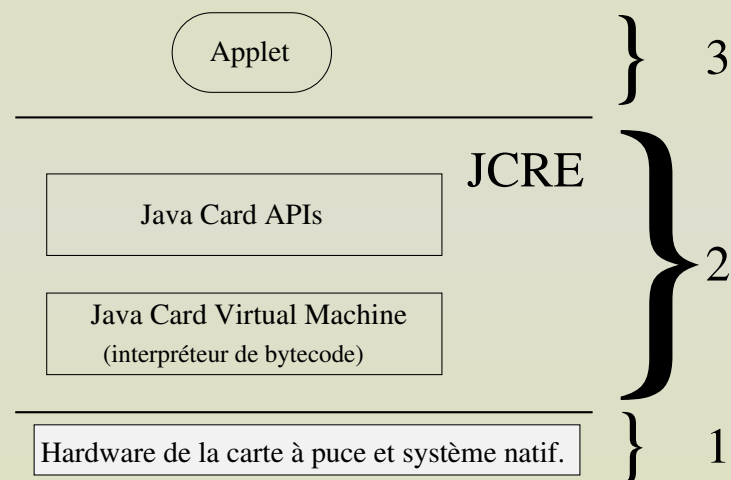


FIG. 2 – L'échelle d'assurance des Critères Communs

## Périmètre d'évaluation ...

Défini dans la cible de sécurité dans la description de la cible d'évaluation.



Cible à évaluer	Périmètre		
	1	2	3
Puce	X		
JCRC	X	X	
VM & APIs	Certifié	X	
Applet	X	X	X
		Certifié	X
			X

Nouveaux problèmes apportés par les applets Java Card :

- ☞ pas de définition précise du périmètre d'évaluation ;
- ☞ pas de méthodologie d'évaluation d'une applet ;
- ☞ la multi-application.

## ... et les problèmes de la multi-application

### Avantages :

- possibilité de réduire les coûts de développement et d'évaluation.

### Problèmes :

- la peur des industriels et des utilisateurs (verrou psychologique).

### Solution :

- ☞ utilisation du standard Open Platform pour maintenir un niveau de sécurité supplémentaire afin de rassurer les industriels et le schéma ;
- ☞ utilisation de Java Card comme plate-forme mono-applicative afin de ne pas inquiéter les consommateurs.

# Le projet Sécurité Java Card

## Besoins de SERMA Technologies :

- ➡ posséder de bonnes connaissances des technologies Java et Java Card ;
- ➡ développer une méthodologie d'évaluation :
  - pour les applets,
  - pour les plate-formes ;
- ➡ développer des batteries de tests d'attaques ;
- ➡ rester proche de la réalité (attaques hardware/software) ;
- ➡ améliorer la qualité de ses évaluations par l'utilisation de preuves formelles.

## Objectifs :

- soumettre la méthodologie à DCSSI et renforcer le schéma ;
- confirmer l'agrément du CESTI pour évaluer la technologie Java Card.

## JCAT Tools (1/2)

### Réponses aux besoins de SERMA Technologies :

Buts	Java Card Attack & Test			
	Emulator	View	Converter	Autres
Connaissance détaillée de la technologie Java Card	X	X		
Définition d'une méthodologie d'évaluation de plate-forme	X			
Création d'une batterie de tests d'attaques	X	X	X	X
Expérimentation de tests d'attaques software & hardware	X			

Développements réalisés en *Java*.

## JCAT Tools (2/2)

---

### *JCAT Emulator* :

Destiné à tester et attaquer des implémentations d'applets.

### *JCAT View* :

Outil de visualisation des formats de fichiers CAP de différents constructeurs.

### *JCAT Converter* :

Outil de conversion d'un format de fichier CAP d'un constructeur vers celui d'un autre.

*Autres JCAT Tools* : par exemple, l'outil d'aide à la modification de bytecode  $\implies$  faciliter la création de batterie de tests et d'attaques.

## JCAT Emulator (1/2)

---

Implémentation complète des spécifications JC 2.1.1 :

- ➡ VM et APIs
- ➡ JCRE (firewall et transaction)

### Objectifs :

- tester et déboguer des applets ;
- détecter des problèmes dans leur comportement.

### Fonctionnalités :

- exécution pas à pas ;
- visualisation des mémoires, des objets, buffer de transaction, pile de frame (en cours), etc.

## JCAT Emulator (2/2)

### **Vers une méthodologie d'évaluation d'une plate-forme :**

Liste des points obscurs dans les spécifications qui sont importants pour la sécurité.

Exemple : transaction annulée dans un `install()` après l'appel à `register()`.

### **Futures améliorations :**

- instrumentations de la machine  $\implies$  statistiques ;
- améliorer la simulation d'attaques physiques ;
- améliorer l'implémentation des APIs crypto ;
- compatibilité JC 2.2 (RMI, multi-channels) et Open Platform.

jcvm

File Options Help

ret

**Loaded Package**

0xa0	0x00	0x00	0x00	0x62	0x00	0x01
0x00	0x00	0x00	0x00	0x00	0x00	
0x00	0x00	0x00	0x00	0x00	0x00	0x01
0xa0	0x00	0x00	0x00	0x62	0x01	0x01
0x00	0x11	0x22	0x33	0x55		
0x00	0x11	0x22	0x33	0x66		

**Transaction**

In progress : ●

index : 0

0	0	14	0	6	0	0	2
7	fd	0	27	0	0	0	0
2	7	fd	0	29	0	0	0
0	2	7	fd	0	2a	0	0
0	0	0	0	0	0	6	fe
0	1	0	1e	0	1	2	7
fd	0	2b	0	0	0	0	2
7	fd	0	2b	0	2	0	0
2	7	fd	0	2b	0	4	0
0	2	7	fd	0	2b	0	0
0	0	2	7	fd	0	2b	0
2	0	0	2	7	fd	0	2b
0	4	0	0	2	7	fd	0
16	0	2	0	0	2	7	fd
0	14	0	6	0	1	2	7
fd	0	14	0	6	0	2	2
7	fd	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

**eeeprom**

- 0x06 0x00 0x11 0x22 0x33 0x55 0x01 0x00
- fields = ...
- 0x00 0x11 0x22 0x33 0x55 0x01
- fields = ...
- 0x00 0x11 0x22 0x33 0x55 0x01
- fields = 35 ...
- 0x06 0x00 0x11 0x22 0x33 0x66 0x01 0x00

**ram**

- ram
  - 0x00 0xe6 0x05 0x00 0x11 0x22 0x33 0x66 0x

```

/* 0x815 */ return , Operand Stack : []
/* 0xa2b */ putfield_a , fields = 21 0 0 0 , Operand Stack : []
/* 0xa2d */ return , Operand Stack : []
/* 0x7da */ dup , Operand Stack : [20, 20]
/* 0x7db */ putstatic_a , static fields = 0 11 0 12 0 13 0 10 0 0 0 16 0 15 0 14 0 17 0 18 0 0
/* 0x7de */ putstatic_a , static fields = 0 11 0 12 0 13 0 10 0 20 0 16 0 15 0 14 0 17 0 18 0 0
/* 0x7e1 */ getstatic_a , Operand Stack : [20]
/* 0x7e4 */ invokestatic , 0( null . 0 ), Operand Stack : []
/* 0x02d */ invokestatic , 0( null . 0 ), Operand Stack : []
/* 0x0b5 */ impdep1 , [23]
/* 0x030 */ return , Operand Stack : []
/* 0x7e7 */ getstatic_a , Operand Stack : [18]
/* 0x7ea */ invokevirtual , 0( null . 0 ), Operand Stack : []
/* 0x1bf */ getfield_a_this, Operand Stack : [19]
/* 0x1c1 */ areturn , Operand Stack : [19]
/* 0x7ed */ putstatic_a , static fields = 0 11 0 12 0 13 0 10 0 20 0 16 0 15 0 14 0 17 0 18 0 0
/* 0x7f0 */ getstatic_a , Operand Stack : [19]
/* 0x7f3 */ invokestatic , 0( null . 0 ), Operand Stack : []
/* 0x033 */ invokestatic , 0( null . 0 ), Operand Stack : []
/* 0x0bc */ impdep1 , [24]
/* 0x036 */ return , Operand Stack : []
/* 0x7f6 */ sconst_1 , Operand Stack : [1]
/* 0x7f7 */ putstatic_b , static fields = 0 11 0 12 0 13 0 10 0 20 0 16 0 15 0 14 0 17 0 18 0 1
/* 0x7fa */ return , Operand Stack : []

```

Cap viewer

0xa0 0x00 ...

```

Component header : {
  magic      : decaffeinated
  minor version : 1
  major version : 2
  flags      : 2
  this package : {
    pkg minor version : 0
    pkg major version : 1
    aid length        : 7
    AID               : 0xa0.0x0.0x0.0x0.0x62.0x1.0x1
  }
}

Component directory : {
  component sizes : {
    Header      : 17
    Directory   : 31
    Applet      : 0
    Import      : 21
    ConstantPool : 718
    Class       : 318
    Method      : 3122
    StaticField : 10
    Reference Location : 509
    Export      : 165
    Descriptor  : 2638
  }
  static field size : {
    image size    : 32
    array init count : 0
    array init size : 0
  }
  import count : 2
  applet count : 0
  custom count : 0
}

Component Import : {

```

Properties Values

Reference 0x21

Owner

PackageAID : 0x00 0x11 0x22 0x33 0x55

Applet Owner : 0x21

Attributes

Special attributes : NONE

Transient type : NOT\_A\_TRANSIENT\_OBJECT

class

```

flags                : 4
interface count      : 3
super classRef       : external class 0x3 of package 0x1
declared instance size : 0
first reference token : 255
reference count       : 0
public method table base : 5
public method table count : 7
package method table base : 0
package method table count : 0
public virtual method table : { 0x81 0xffff 0x15 0x78 0x7b 0x7e 0
package virtual method table : { }
implemented interfaces : {
  interface[0] {

```

## De la théorie à la pratique

---

Les travaux LaBRI/SERMA Technologies ont facilité :

**Evaluation EAL4+ d'une plate-forme Java Card** (première fois en France) :

- ☞ utilisation des résultats et des recommandations issus du développement de JCAT Emulator ;
  - ☞ mise en place d'une première batterie de tests d'attaques logicielles et logicielles/matérielles.
- ⇒ Recommandations pour les développeurs d'applets sur cette plate-forme.

**2 évaluations EAL4+ de produits comportant des applets Java Card**

- ☞ batterie spécifique de tests d'attaques logicielles et logicielles/matérielles.

## Formalisation (1/2)

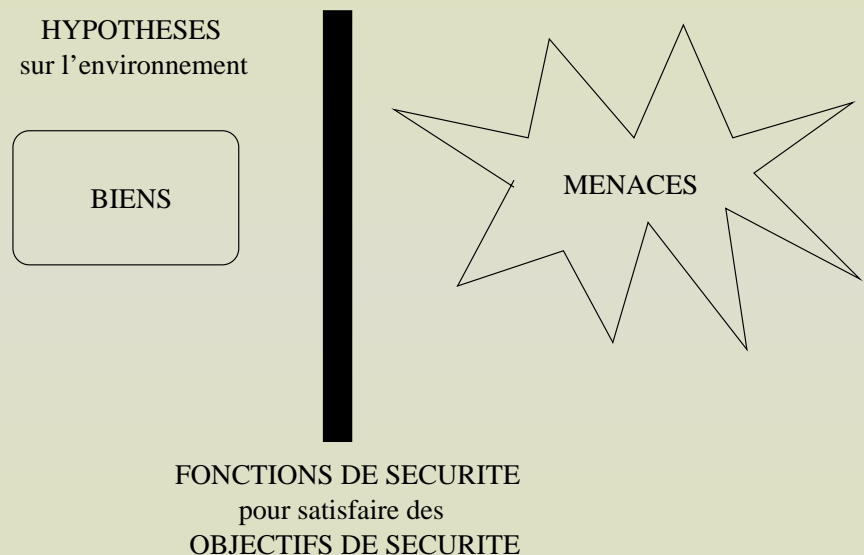


FIG. 3 – La sécurité dans les CC

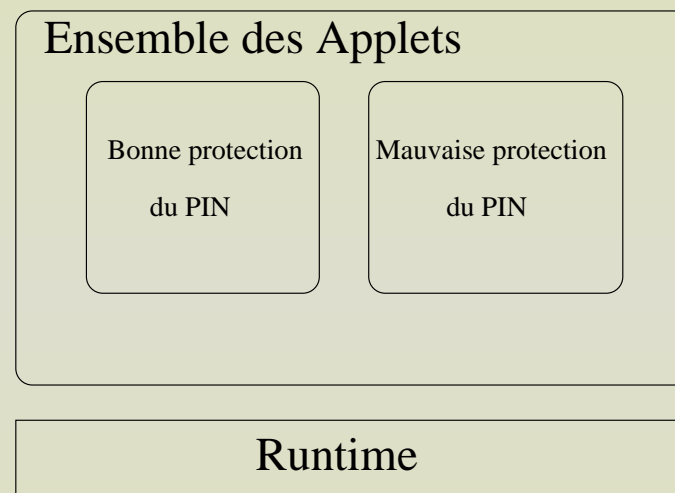
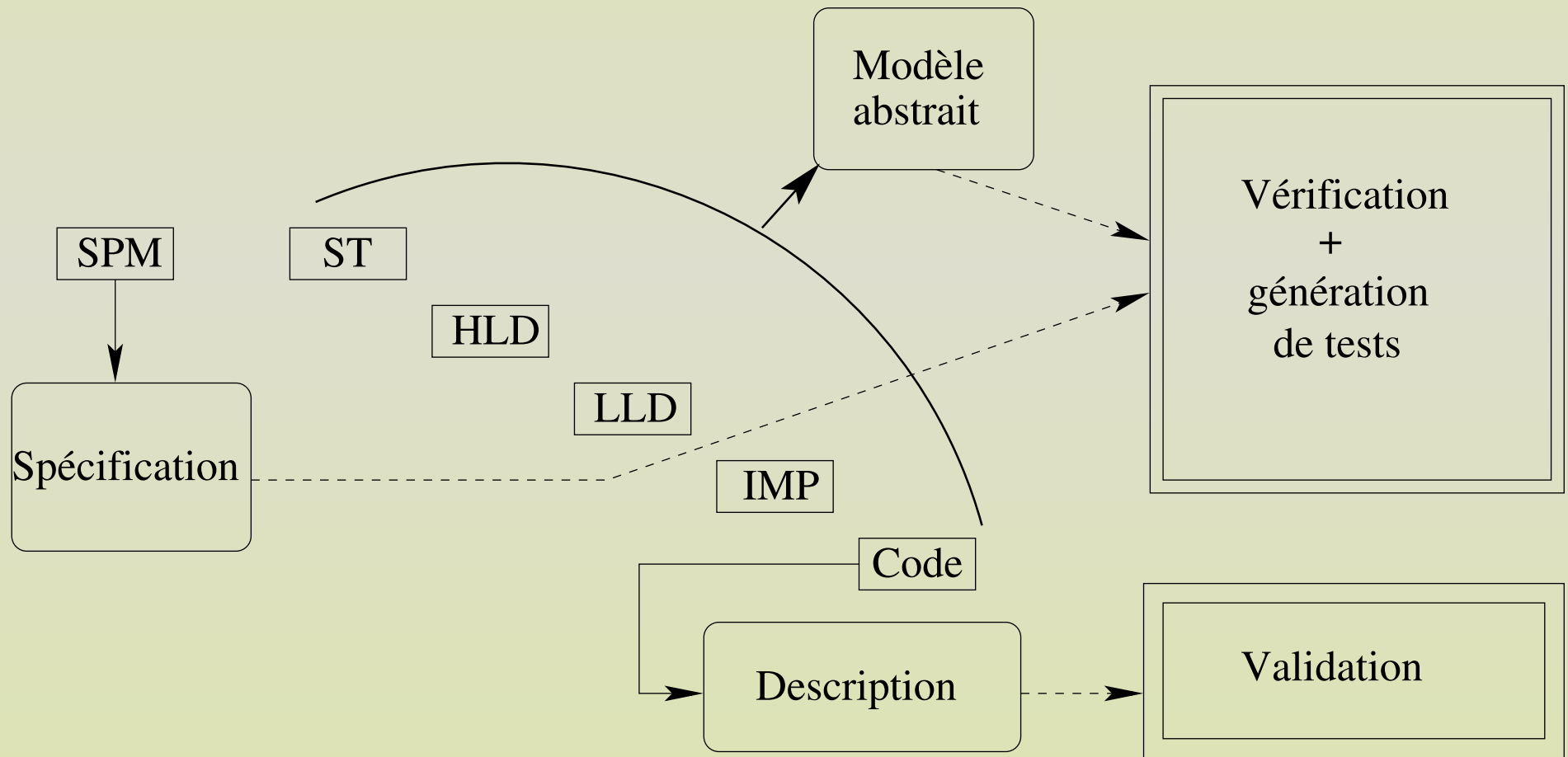


FIG. 4 – Exemple de propriété de sécurité

- ⇒ outils de validation à postériori.
- ⇒ méthodologie de développement.

## Formalisation (2/2)



# Perspectives

---

Compléter JCAT Emulator en partenariat avec différents industriels implémentant la technologie Java Card par leur représentation des structures de données (mémoires, objets, etc.).

Rendre tous nos outils compatibles avec la spécification Java Card 2.2.

Faire certifier notre implémentation afin d'accéder au Java Card Forum.

Aider SERMA Technologies à améliorer la qualité de ses évaluations via le développement d'une méthodologie visant à fournir des preuves formelles.