

La carte à puce multi-applicative et sa sécurité



Damien SAUVERON
sauveron@labri.fr

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

Plan

Présentation de la carte à puce

Quelques vulnérabilités liées au concept de
multi-application

Présentation des différentes technologies des
cartes à puce multi-applicatives

Présentation de la carte à puce

Qu'est ce qu'une carte à puce ?

- ➡ *un morceau de plastique* de la taille d'une carte de crédit
- ➡ *un circuit électronique* capable de manipuler (stocker, calculer, etc) des informations

Historique

- En 1968, deux Allemands Jürgen DETHLOFF et Helmut GRÖTRUPP introduisent un circuit intégré dans une carte plastique
- Entre 1974 et 1978, le français Roland MORENO, *le père de la carte à puce* dépose 47 brevets dans 11 pays
- En 1983, apparition des premières cartes téléphoniques à mémoire
- En 1984, adoption par le G.I.E carte bancaire de la “ carte bleue ”
- Entre 1984 et 1987, normes ISO 7816 (carte à puce à contact)
- En 1997, apparition des premières Java Cards

Les différentes cartes à puce

Il existe plusieurs sortes de cartes à puce.

Plusieurs classements possibles :

les cartes à mémoire



versus

les cartes à microprocesseur

les cartes à contact



versus

les cartes sans contact

La carte à mémoire

- ➡ Premier modèle de cartes à puce
- ➡ Majorité des cartes vendues dans le monde en 1999

Elle possède :

- ➡ *une puce mémoire* de 1 à 4 Ko
- ➡ *une logique câblée non programmable*

Avantages :

- sa technologie simple
- son faible coût (1\$)

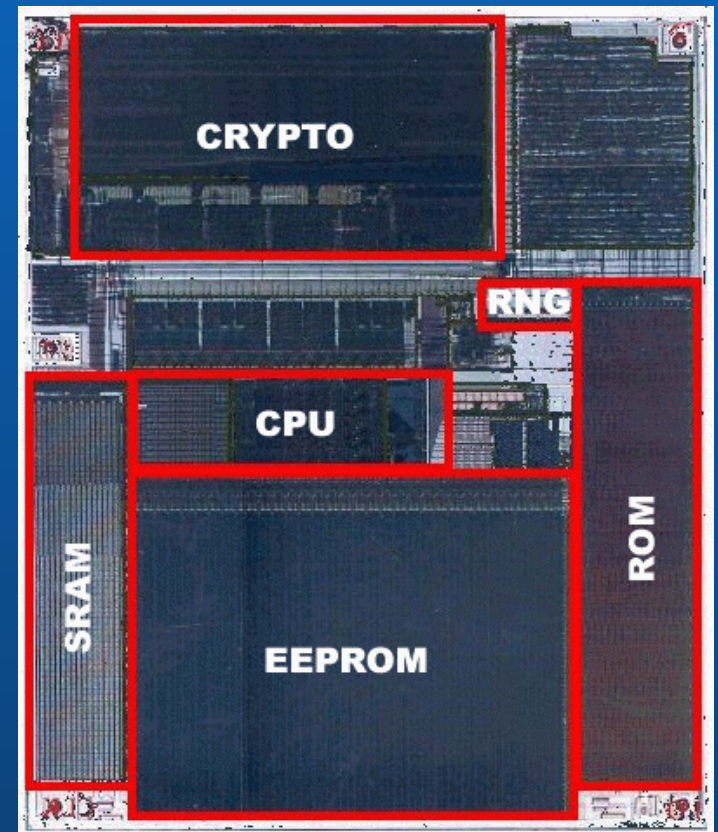
Inconvénients :

- sa dépendance vis-à-vis du lecteur de carte
- “ assez ” facile à dupliquer

La carte à microprocesseur

Taille de la puce : $25mm^2$

- ➡ **Microprocesseur** (CPU) : 8, 16 ou 32 bits (à architecture RISC ou pas)
- ➡ **ROM** : 16 à 24 Ko
- ➡ **EEPROM** : 8 à 64 Ko
- ➡ **RAM** : 1 Ko
- ➡ Coprocesseur cryptographique
- ➡ Générateur de nombres aléatoires (RNG)



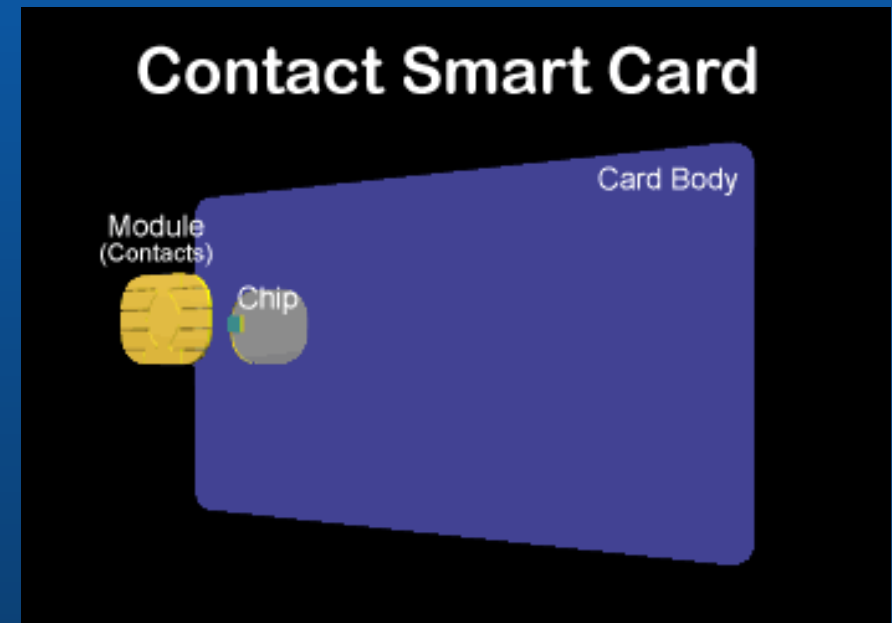
Avantage : le coût acceptable pour tant de sécurité (entre 1\$ et 20\$).

La carte à contact

- ➡ Suit le standard *ISO 7816*
- ➡ *Communication série via huit contacts* $\Rightarrow$ insertion dans un lecteur de carte

Problèmes :

- l'insertion et le retrait sont des facteurs d'usure de la carte
- orientation de la carte dans le lecteur

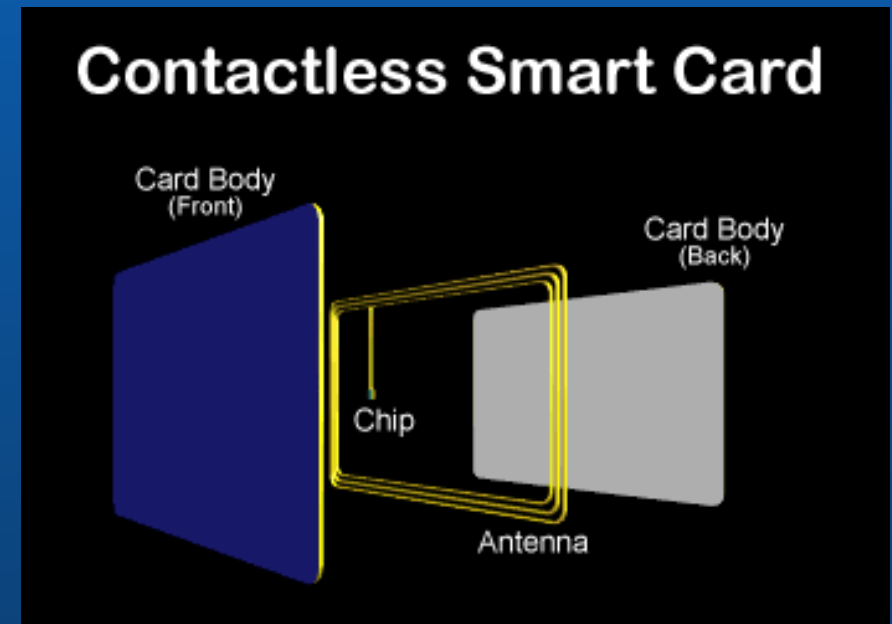


La carte sans contact

- ➡ *Communication via une antenne* dans la carte
- ➡ Récupère son énergie d'un couplage capacitif ou d'un couplage inductif

Problèmes :

- distance de communication limitée (environ 10 cm)
- temps de transaction est de l'ordre de 200 ms $\Rightarrow$ limite les données à échanger
- le coût élevé



La carte combi

C'est une combinaison entre :

- ☞ la carte à contact
- ☞ et la carte sans contact

Ces deux possibilités de communication $\implies$ la carte “idéale”

Quelques applications

- ➡ l'industrie des télécommunications
- ➡ l'industrie bancaire et monétaire
- ➡ le secteur de la santé
- ➡ l'industrie audiovisuelle avec la télévision à péage, ...
- ➡ le porte-monnaie électronique
- ➡ les transports en commun
- ➡ le contrôle d'accès physique de personnes à des locaux, ...
- ➡ l'identification : à des sites sur l'Internet, ...
- ➡ les applications de fidélité
- ➡ les “ e-services ”

Quelques vulnérabilités liées au concept de multi-application

Le concept de multi-application

Une carte multi-applicative = système d'exploitation + des applications

Système mono-tache

Application fournie :

- ➡ par l'émetteur de carte $\Rightarrow$ code en ROM
- ➡ par un fournisseur d'applications $\Rightarrow$ code en EEPROM

Spécifications *Open Platform* :

- ➡ mécanismes de gestion des cartes multi-applicatives (installation, désinstallation, cycle de vie, etc)
- ➡ mécanismes sécuritaires

Identification du fournisseur d'applications

Vulnérabilité potentielle : pirate voulant charger une application agressive

Solution : *authentication mutuelle* pour identifier les fournisseurs autorisés

Possibilité de création d'un “ *canal sécurisé* ” $\implies$ assurer la confidentialité et l'intégrité des communications

Ces mécanismes de “ mutual authentication ” et de “ secure channel ” sont décrits dans Open Platform.

A terme, la plate-forme devra être capable de se “ défendre ” sans utiliser ces mécanismes.

L'intégrité des données et du code

Hypothèse : “ casser ” le mécanisme d'authentification mutuelle

Vulnérabilité potentielle : au niveau du code de l'application à charger

Solution : signature cryptographique avant son chargement et ensuite
vérification de la signature par la carte

⇒ *mécanisme de DAP* (Data Authentication Pattern) de Open Platform

Hypothèse : “ casser ” le mécanisme d'authentification mutuelle et de DAP

Vulnérabilité potentielle : au niveau du code et des données des applications
et du système d'exploitation

Solution : *mécanismes de “ firewall ”* et de partage sécurisé de données

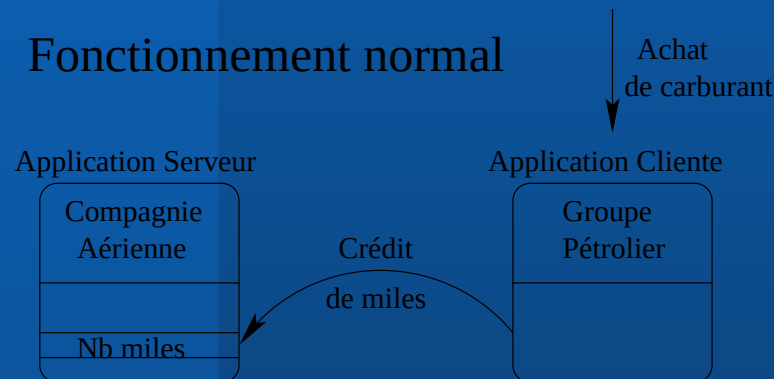
La sécurité : l'affaire de tous

Hypothèse : Plate-forme sûre.

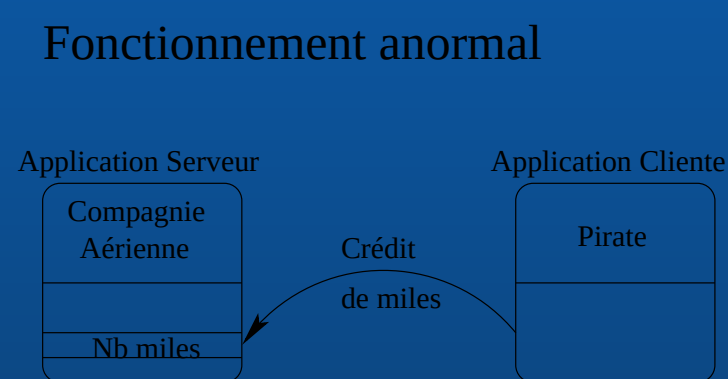
Exemple : Application de fidélité entre un groupe pétrolier et une compagnie aérienne.

1 litre de carburant acheté = 1 miles offert

Fonctionnement normal



Fonctionnement anormal



Conclusion : La sécurité doit être assurée au niveau de la plate-forme, mais aussi au niveau applicatif. La sécurité est le problème de tous.

Présentation des différentes technologies des cartes à puce multi-applicatives

MULTOS

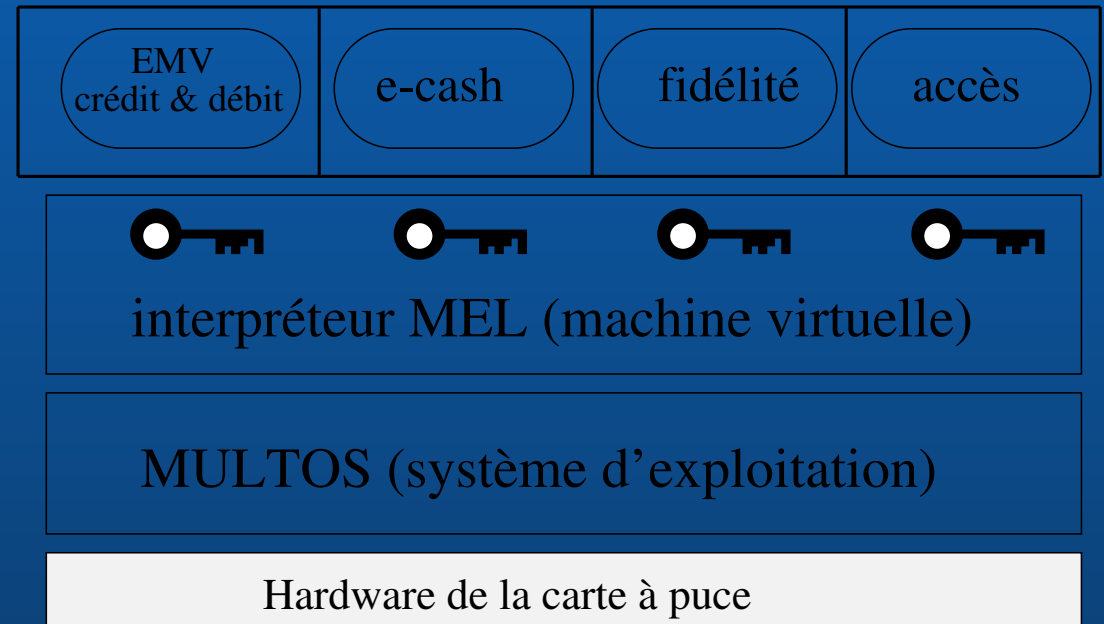
➡ Technologie proposée par
MAOSCO

➡ Standard ouvert

Spécifie :

- ➡ une machine virtuelle
- ➡ un langage : MEL (MULTOS Enabling Language)
- ➡ des APIs
- ➡ un “ firewall ”
- ➡ une compatibilité ISO 7816 et EMV

 chargement de certificat par l'application
 Firewall



Windows for Smart Card

- ➡ Technologie proposée par Microsoft
 - ➡ Carte multi-applicative orientée authentication
 - ➡ Standard fermé
- ⇒ Le projet est depuis *abandonné*.

La technologie Java CardTM

Technologie permettant de faire fonctionner des applications écrites en langage Java^a pour :

- ➡ les cartes à puce
- ➡ d'autres périphériques à mémoire limitée

La technologie Java Card définit une plate-forme pour cartes à puce sécurisée, portable et multi-applications qui incorpore beaucoup des avantages du langage Java.

^aJava and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. The author is independent of Sun Microsystems, Inc.

Historique

- En Novembre 1996, un groupe d'ingénieurs de Schlumberger cherche à simplifier la programmation des cartes à puce tout en préservant la sécurité.
⇒ la spécification Java Card 1.0
- En Février 1997, Bull et Gemplus se joignent à Schlumberger pour cofonder le “ *Java Card Forum* ”.
- En Novembre 1997, Sun présente les spécifications Java Card 2.0.

Historique

- En Mars 1999 sort la version 2.1 des spécifications Java Card. Elles consistent en trois spécifications :

- The Java Card 2.1 API Specification.
- The Java Card 2.1 Runtime Environment Specification.
- The Java Card 2.1 Virtual Machine Specification.

Contribution la plus significative :

- Définition explicite de la machine virtuelle de la Java Card.
- *Le format de chargement des applets.*

- En Mai 2000, sort une petite correction $\implies$ version 2.1.1
- En Octobre 2000, plus de 40 entreprises ont acquis la licence d'exploitation de la technologie Java Card.

Avantages de la technologie Java Card

La facilité de développement des applications grâce :

- ➡ à la **programmation orientée objet** offerte par Java
- ➡ à l'utilisation des **environnements de développement** existants pour Java
- ➡ à une **plate-forme ouverte** qui définit des APIs et un environnement d'exécution standard
- ➡ à l'**encapsulation de la complexité** fondamentale du système des cartes à puce

Avantages de la technologie Java Card

La sécurité grâce :

- ➡ à **plusieurs niveaux de contrôle d'accès** aux méthodes et aux variables (public, protected, private)
- ➡ à un **langage fortement typé**
- ➡ à l'**impossibilité de construire des pointeurs**
- ➡ à un **“ firewall ”**

Avantages de la technologie Java Card

L'indépendance au hardware réalisée grâce au langage Java

⇒ “ Write Once, Run Anywhere ”

La capacité de stockage et de gestion de multiples applications.

⇒ possibilité de mise à jour des applications de la Java Card sans avoir besoin de changer de cartes

La *compatibilité* avec les standards existants sur les cartes à puce.

Présentation de son architecture

Problème : contraintes mémoires

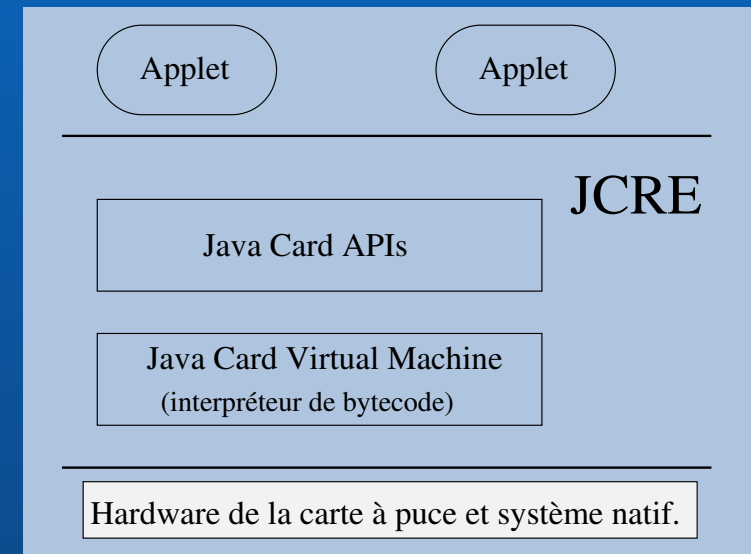
Solutions :

- un sous-ensemble des caractéristiques du langage Java
- découper la machine virtuelle Java en deux parties

Problème : pas de vérificateur embarqué

Solution :

- fournir des mécanismes sécuritaires avec l'environnement d'exécution



Le langage Java Card

Caractéristiques Java non supportées

- ✗ Type simple de donnée de grosse taille : long, double, float
- ✗ Tableau plusieurs dimensions
- ✗ Caractères et chaînes
- ✗ Chargement dynamique des classes
- ✗ Security Manager
- ✗ Ramasse-miettes et finalisation
- ✗ Threads
- ✗ Serialisation d'objet
- ✗ Clonage d'objet

Le langage Java Card

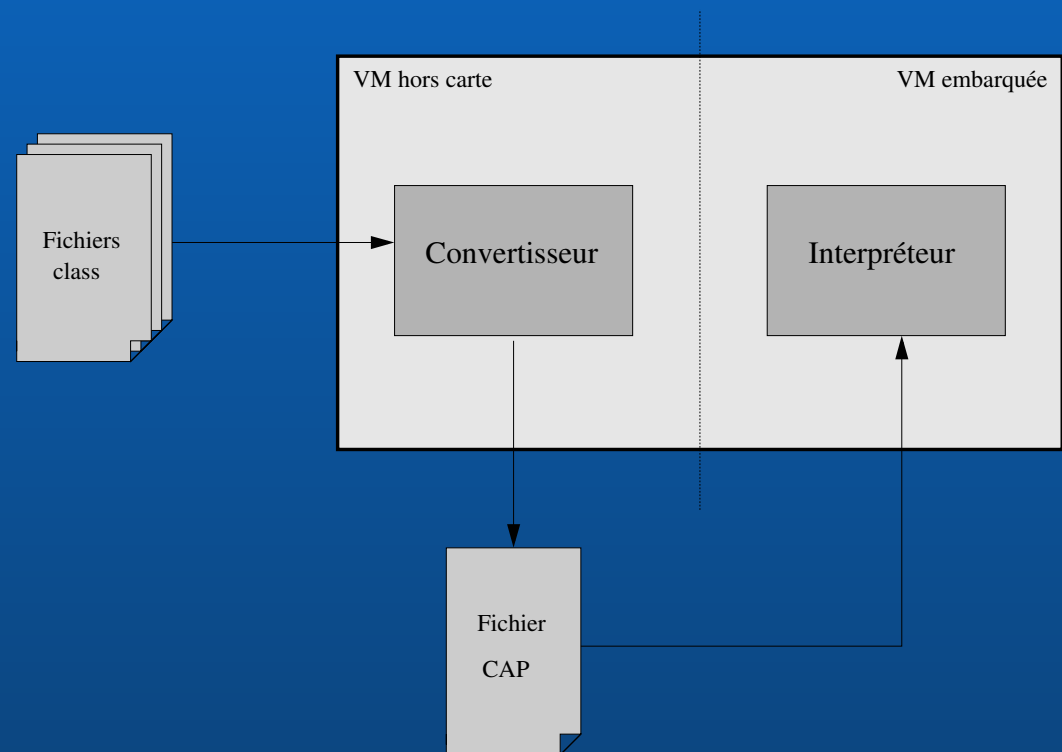
Caractéristiques Java supportées

- ✓ Type simple de donnée de petite taille : boolean, byte, short
- ✓ Tableau à une dimension
- ✓ Paquetage Java, classes, interfaces et exceptions
- ✓ Caractéristiques orientées objet : héritage, méthodes virtuelles, surcharge et création dynamique d'objet, contrôle d'accès
- ✓ Le mot clé `int` et le support des entiers sur 32 bits sont optionnels

La machine virtuelle Java Card : JCVM

Les deux parties implémentent toutes les fonctions d'une machine virtuelle.

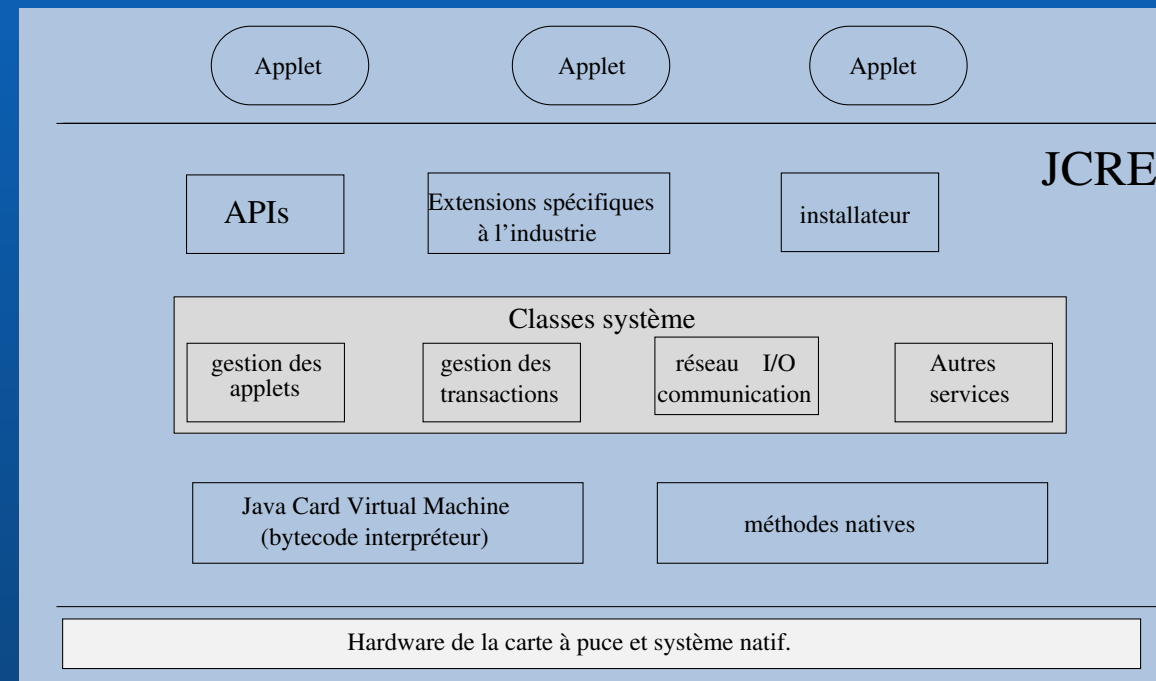
A cause du découpage de la JCVM, *la plate-forme est distribuée dans le temps et dans l'espace.*



L'environnement d'exécution Java Card

Responsable :

- ➡ de la gestion des ressources de la carte
- ➡ de la communication réseau
- ➡ de l'exécution des applets
- ➡ du système de la carte
- ➡ de la sécurité des applets



L'environnement d'exécution Java Card

Les caractéristiques du JCRE

- ➡ *Objets persistants* existant au travers des sessions avec le lecteur.
- ➡ *Objets temporaires* dont les données ne persistent pas au travers des sessions avec le lecteur.
- ➡ Chaque opération d'écriture de la JCVM est *atomique*.
- ➡ *Une transaction* est un bloc d'opérations atomiques.
- ➡ Le “ *firewall* ” isole les applets à l'intérieur de leur espace (contexte).
- ➡ Pour mettre en commun des données, il existe des *mécanismes sécurisés de partage*.

Les APIs Java Card

Ensemble de paquetages optimisés pour la programmation des cartes à puce en accord avec le modèle ISO 7816.

`java.lang` : un sous ensemble strict de son équivalent sur la plate-forme Java

`javacard.framework` : classes et interfaces pour le noyau fonctionnel des applets Java Card

`javacard.security` : modèle pour les fonctions cryptographiques supportées sur la plate-forme Java Card

`javacardx.crypto` : un paquetage d'extension

Conclusion

- ➡ En 30 ans la carte à puce s'est imposée en Europe.
- ➡ *Il reste un marché énorme* (États-Unis, etc).

La technologie Java Card saura-t-elle le conquérir ?

Pour cela il lui faudra prouver qu'elle est parfaitement *sûre*.