



Mémoire de DEA Informatique

Responsable : **Serge CHAUMETTE**

Équipe : **Systèmes et Objets Distribués**

Sécurité et vérification d'applications embarquées en environnement Java CardTM ¹

Damien SAUVERON

sauveron@labri.u-bordeaux.fr

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

Version : 1.0

Dernière révision : 3 Juin 2001

RÉSUMÉ – Avec l'explosion du commerce électronique, de la téléphonie mobile et des besoins d'identification, la confidentialité et la sécurité des échanges sont devenues des priorités. Les solutions actuellement proposées sont basées sur *la carte à puce*. Cette carte dispose en effet de nombreux atouts sécuritaires et peut s'adapter à de multiples problèmes.

Dans une première partie, nous verrons les caractéristiques principales des cartes à puces et comment la technologie Java a réussi à s'immiscer avec succès dans le monde de la carte à puce pour construire une carte multi-applicative : *la Java Card*. Ensuite, dans une seconde partie, nous présenterons les grands standards de la carte à puce qu'utilisent beaucoup des produits commerciaux que nous présenterons dans la troisième partie. Enfin nous exposerons les développements logiciels que nous avons réalisés.

MOTS-CLÉS : *carte à puce, Java Card, sécurité, embarqué*

ABSTRACT – With the explosion of electronic business, mobile telephony and needs of identification, the confidentiality and the security of the transactions are priorities. The solutions currently suggested use the *smart card*.

In a first part, we will show the main features of smart cards and how Java technology succeeded in becoming a major technology within smart card to build a multi-applicative card: *the Java Card*. Then, in a second part, we will show the major smart card standards used by products showed in the third part. Finally, we will see our software development.

KEYWORDS: *smart card, Java Card, security*

¹Java 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.

Table des matières

Table des matières	3
Introduction	5
1 Cadre, objectifs et déroulement du mémoire	7
1.1 Cadre	7
1.2 Objectifs	7
1.3 Déroulement du mémoire	8
1.4 Remerciements	8
2 La technologie Java Card	9
2.1 Présentation de la carte à puce	9
2.1.1 Qu'est ce qu'une carte à puce?	9
2.1.2 Les différentes cartes à puce	9
2.1.3 Les différentes cartes multi-applicatives	12
2.2 La Java Card	15
2.2.1 Qu'est ce que Java Card?	15
2.2.2 Les avantages de la technologie Java Card	15
2.2.3 Présentation de la Java Card	15
3 Les standards et les spécifications des cartes à puce	21
3.1 ISO 7816	21
3.2 GSM	21
3.3 EMV	22
3.4 Open Platform	23
3.5 OpenCard Framework	24
3.6 PC/SC	24
3.7 Résumé de l'étendue des divers standards présentés	25
4 Implémentations et kits Java Cards existants	27
4.1 L'anneau Java (Java Ring)	27
4.2 Les cartes à puce Java	28
4.2.1 Le kit de développement de Sun Microsystems, Inc.	28
4.2.2 Les kits de développement et les implémentations Java Card	28

5	Développement logiciel	31
5.1	Fichier CAP	31
5.2	Présentation des caractéristiques de notre outil	32
5.3	Présentation de l'interface de notre outil	32
5.4	Objectifs de cet outil	35
5.5	Améliorations à réaliser	35
6	Perspectives	37
6.1	Le projet entre le LaBRI et SERMA Technologies	37
6.2	La thèse CIFRE	37
	Conclusion	39
	Références	41
	Table des figures	43
	Liste des tableaux	43
	Glossaire	45
	Index	46

Introduction

À l'heure de l'Internet et des nouvelles technologies, alors que la sécurité devient l'une des préoccupations principales, la carte à puce est un des objets les plus sûrs qui puissent exister. Son développement se base sur l'utilisation des technologies matérielles les plus sécurisées et les plus secrètes. Depuis quelques années la carte à puce cherche aussi à s'imposer comme l'objet universel d'authentification et de sécurisation dans les systèmes informatiques et dans la vie quotidienne. Cette révolution est en cours grâce aux cartes multi-applicatives et plus particulièrement grâce à la technologie Java Card.

Chapitre 1

Cadre, objectifs et déroulement du mémoire

1.1 Cadre

Le mémoire de DEA se déroule au sein de l'équipe Systèmes et Objets Distribués du LaBRI sous la responsabilité de Serge CHAUMETTE. Il s'inscrit en parallèle d'une collaboration entre le LaBRI et la société SERMA Technologies sur un projet « Sécurité Java Card » soutenu et financé par le Ministère de l'Industrie.

La société SERMA Technologies, située à Pessac, dispose d'un CESTI (Centre d'Évaluation pour la Sécurité des Technologies de l'Information) agréé par la DCSSI (Direction Centrale de la Sécurité des Systèmes d'Information). La DCSSI, qui est l'organisme de certification français des systèmes d'information, est un service dépendant du Premier ministre. Le CESTI de SERMA Technologies est le seul CESTI « hardware » à posséder un agrément pour les logiciels développés en Java Card. L'objectif principal pour SERMA Technologies au sein de cette collaboration est de développer une bonne expertise de la structure des produits répondant aux spécifications Java Card.

1.2 Objectifs

L'objectif principal de ce mémoire était de mettre en place les bases de travail nécessaires au commencement d'une thèse CIFRE, à la rentrée 2001, intitulée « Étude et réalisation d'un environnement d'expérimentation et de modélisation pour les nouvelles technologies embarquées : application à la sécurité dans un environnement Java Card ».

La réalisation de cet objectif incluait la mise en place de documentations sur la technologie Java Card et a donné lieu à :

- *une présentation de la technologie Java Card* le 28 février 2001 *en groupe de travail de l'équipe ParaDis, composante SOD* ;
- *un rapport interne du LaBRI RR-1259-01 [3] intitulé : « La technologie Java Card ».*

Il fallait aussi décider de l'achat d'un kit de développement afin de pouvoir dès le commencement de la thèse mieux appréhender la plateforme et les problèmes concrets qui se posent.

Un second objectif de ce mémoire était une pré-étude des pistes à explorer pour mener à bien les investigations relatives à la sécurité qui seront étudiées pendant la thèse.

1.3 Déroulement du mémoire

La méthodologie que nous avons choisie de suivre lors du déroulement du mémoire de DEA est la suivante :

- dresser un inventaire des spécifications existantes ;
- dresser un inventaire des implémentations et kits de développement existants ;
- étudier ces spécifications.

En parallèle de l'étude des spécifications, nous avons entrepris le développement en Java d'un outil de visualisation et de modification d'applications ou de paquets Java Card. Le développement de cet outil a pour objectif de réaliser des attaques sur la carte.

1.4 Remerciements

Je tiens à remercier Serge CHAUMETTE pour m'avoir proposé ce mémoire, mais aussi pour m'avoir orienté dans mes recherches. Je souhaite aussi remercier les membres de l'équipe Systèmes et Objets Distribués qui m'ont tous si gentiment accueilli au sein de l'équipe.

Je voudrais aussi remercier David RENAULT (étudiant du DEA) qui a été mon « L^AT_EX Companion ». Enfin, je tiens à remercier toutes les personnes de la salle du DEA pour l'ambiance si chaleureuse qui y a régné.

Chapitre 2

La technologie Java Card

Dans la première partie de ce chapitre, nous aborderons les différentes technologies utilisées pour les cartes à puce. Puis dans la seconde partie, nous verrons comment la technologie Java Card s'intègre dans ce domaine. Si vous désirez plus d'informations sur ces points vous pouvez vous reporter au rapport interne de LaBRI RR-1259-01 [3].

2.1 Présentation de la carte à puce

2.1.1 Qu'est ce qu'une carte à puce ?

De la même taille qu'une carte de crédit, une carte à puce est une carte plastique qui contient un circuit électronique capable de manipuler (stocker, calculer, etc) des informations. Une carte à puce est un ordinateur portable et résistant aux altérations.

2.1.2 Les différentes cartes à puce

Il existe plusieurs sortes de cartes à puce que l'on peut classer de plusieurs façons. On peut soit faire un classement entre les cartes à mémoire et les cartes à microprocesseur soit entre les cartes à contact et les cartes sans contact.

La carte à mémoire

Cette carte possède une puce mémoire et elle peut aussi comporter une logique câblée non programmable (instruction programmée et non reprogrammable) mais pas de microprocesseur. La taille de la mémoire est de 1 Ko à 4 Ko.

La carte à microprocesseur

Cette carte est composée d'un microprocesseur et de différents types de mémoires (*cf.* FIG. 2.1). Le plus souvent la carte possède un processeur 8 bits mais il existe des modèles à base de processeurs 16 et 32 bits. On dispose même de processeurs à architecture RISC. Côté mémoire ces cartes en comportent trois types :

- La ROM (Read Only Memory) qui n'a pas besoin d'énergie pour sauvegarder l'information qu'elle contient. Elle sert à stocker le COS (Card Operating System) et des données

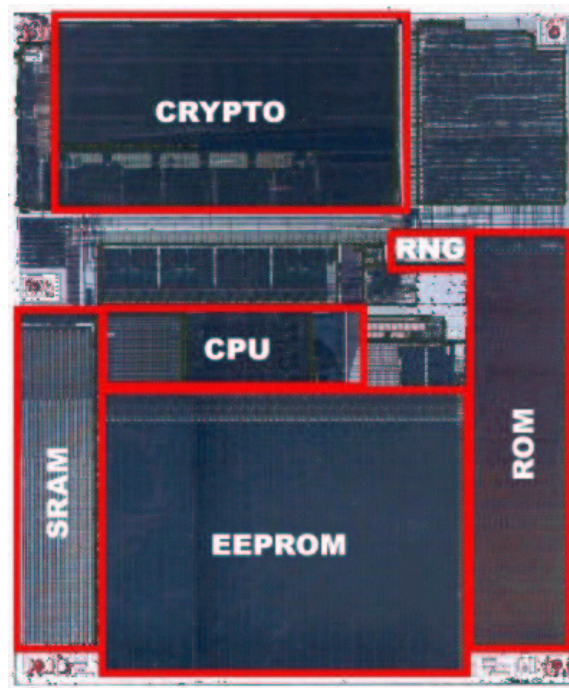


FIG. 2.1 – Un microprocesseur de carte à puce

permanentes. Elle est programmée en usine et ne peut plus être modifiée par la suite. Elle a le plus souvent une taille de 16 Ko mais il en existe disposant de 24 Ko.

- L'EEPROM (Electrical Erasable Programmable Read Only Memory) qui peut comme la ROM préserver son information même quand la carte n'est plus sous tension. La différence avec la ROM est qu'elle peut être modifiée par une application. Sa taille varie de 8 Ko à 64 Ko.
- La RAM (Random Access Memory) qui est utilisée comme espace de stockage temporaire grâce à la rapidité des temps d'accès. Elle possède un caractère non persistant, c'est à dire que dès que la carte n'est plus sous tension, elle perd son contenu. Elle peut être lue et écrite infiniment. Sa capacité est de 256 octets à 1 Ko.

La carte à microprocesseur contient aussi souvent un coprocesseur cryptographique associé à un générateur de nombres aléatoires RNG (Random Number Generator).

La carte à contact

Les cartes à contact pour pouvoir fonctionner doivent être insérées dans un lecteur de cartes. Elles utilisent une communication série via huit contacts (*cf.* FIG. 2.2).

La carte sans contact

Avec ces cartes, les problèmes rencontrés dans les cartes à contact n'existent plus puisque ces cartes ne sont pas insérées dans un lecteur. Elles communiquent via une antenne dans la carte (*cf.* FIG. 2.3). La puce tire son énergie soit d'un couplage capacitif (couplage intérieur à la carte, *e.g.* une batterie) soit d'un couplage inductif (couplage distant, collecté par l'antenne).

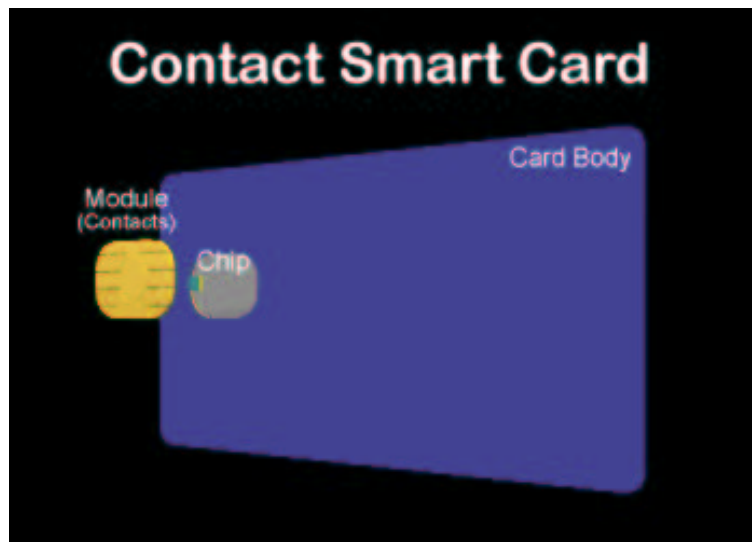


FIG. 2.2 – La carte à contact

Le couplage inductif fonctionne sur le principe du transformateur où une bobine induit un courant dans une autre bobine. La fréquence de transmission utilisée est de l'ordre de quelques MHz. La puce est capable en changeant sa résistance, de transmettre le signal qui est capté par le lecteur et interprété comme un signal de donnée.

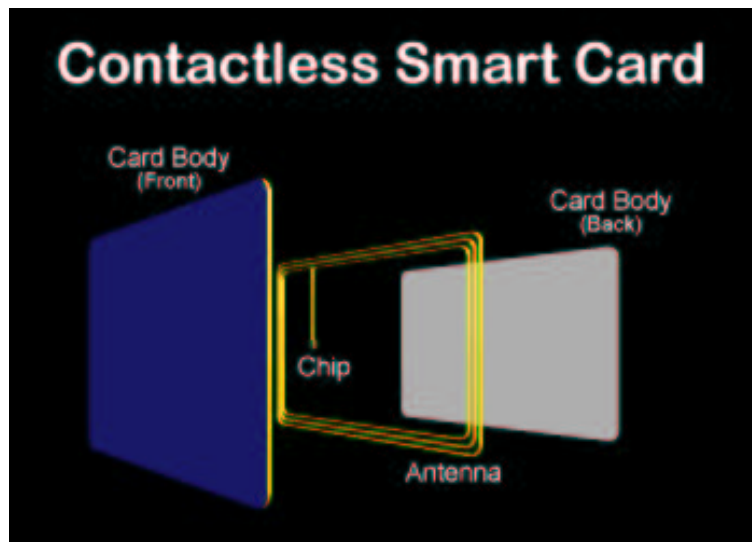


FIG. 2.3 – La carte sans contact

La carte combi

Ces cartes ne sont ni plus ni moins qu'une combinaison des cartes à contact et des cartes sans contact. Elles disposent des deux possibilités de communication, ce qui en fait des cartes

« idéales ».

2.1.3 Les différentes cartes multi-applicatives

MULTOS

MULTOS est la première carte à puce ouverte, hautement sécurisée et possédant un système d'exploitation multi-applicatif (d'où MULT-OS).

Un consortium industriel, nommé MAOSCO, est chargé de promouvoir MULTOS comme système d'exploitation multi-applicatif pour les cartes à puce, de gérer les spécifications de MULTOS, de fournir les licences et les certifications de services MULTOS. Les membres fondateurs du consortium MAOSCO sont : American Express, Dai Nippon Printing, Mondex International Ltd, Siemens, Fujitsu, Hitachi, Motorola, MasterCard International, Keycorp.

Les éléments clés de la plateforme MULTOS sont :

- une architecture hautement sécurisée ;
- la possibilité d'avoir plusieurs applications sur la même carte ;
- l'interopérabilité entre les applications ;
- l'indépendance des applications par rapport à la plateforme (au matériel) ;
- le chargement et l'effacement des applications durant la vie de la carte ;
- la compatibilité avec les standards industriels ISO 7816 et EMV.

Pour le développement des applications, Mondex International Ltd a mis en place :

- un langage optimisé pour les cartes à puce : MEL (MULTOS Enabling Language) ;
- des spécifications pour les APIs MULTOS.

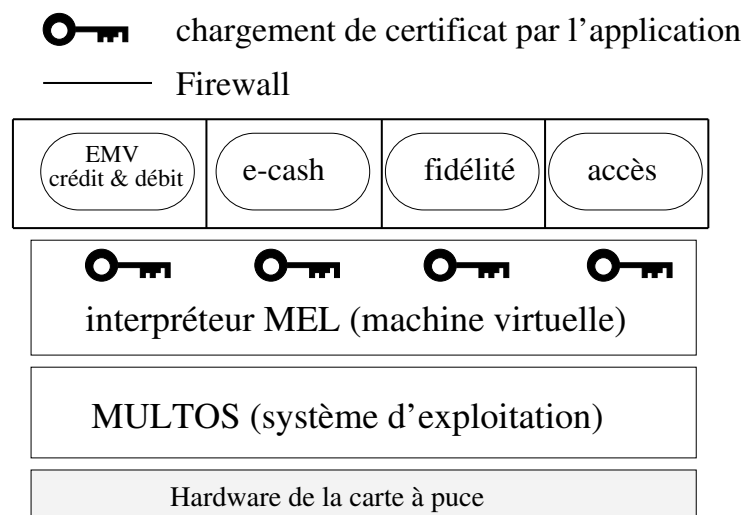


FIG. 2.4 – Architecture de MULTOS

MULTOS s'exécute sur la puce de la carte. A l'installation d'une nouvelle application, la carte à puce MULTOS vérifie la validité de l'application qui a été envoyée, alloue un espace mémoire protégé au programme via le « firewall ». Chaque nouveau service ou application est gardé rigoureusement séparé, par le « firewall », des autres programmes déjà sur la carte

afin qu'aucun d'eux, en cas de dysfonctionnements, ne puisse interférer sur une opération d'un autre programme.

MULTOS est pour le moment implémenté sur des cartes à puce :

- par DNP avec des puces fabriquées par Hitachi ;
- par Keycorp avec des puces fabriquées par Siemens ;
- par Motorola avec ses propres puces.

Il existe déjà des dizaines d'applications pour MULTOS écrites par les fabricants de cartes à puce comme Gemplus, Schlumberger, De La Rue, Orga, etc.

L'implémentation réalisée par Keycorp est l'implémentation complète de MULTOS 4.0 basée sur la puce SLE66 de Siemens. Cette implémentation est certifiée au niveau E6 des critères ITSEC (Information Technology Security Evaluation Criteria) faisant ainsi de ce produit une des cartes les plus sécurisées du monde.

Les caractéristiques de cette carte sont :

- une implémentation complète de MULTOS ;
- 16 ou 32 Ko d'EEPROM ;
- la possibilité de communication entre les applications ;
- une gestion efficace de la mémoire ;
- les standards de sécurité de MULTOS 4 ;
- un haut niveau de cryptographie.

Windows for Smart Card de Microsoft

La WinCard, proposée par Microsoft, est une carte multi-applicative orientée authentification. Comme cette carte est un standard fermé, nous ne disposons que de peu d'information, même commerciales.

Les caractéristiques de cette carte sont :

- un système d'exploitation sécurisé avec :
 - un contrôle d'accès pour l'émetteur de carte à un système de fichiers FAT,
 - des APIs Win32 embarquées « langage-neutre » (*i.e.* qui ne dépendent pas du langage utilisé),
 - un environnement d'exécution,
 - des algorithmes de cryptographie des plus sécurisés (*e.g.* RSA, DES, 3DES et SHA) ce qui la rendrait virtuellement inviolable ;
- des outils de développement d'applications rapides avec :
 - le support du système de développement Visual Studio,
 - un simulateur PC pour le débogage d'applications,
 - une carte à puce de développement avec une mémoire de type FLASH où l'on chargera le masque ;
- des applications simples fournies avec la carte et prêtes à être déployées.

Les applications pour la WinCard se programment en Visual Basic ou Visual C++ permettant au programmeur l'utilisation des environnements de développement déjà existants. Ces langages permettent d'écrire des applications qui pourront être téléchargées durant la vie de la carte.

Sur la FIG. 2.5, les composants qui ont les bords hachés sont optionnels. En effet, en utilisant le kit de développement, le développeur peut sélectionner les composants du système d'exploitation qu'il désire créant ainsi le masque qu'il chargera sur la carte. Pour le moment seul l'environnement d'exécution (*i.e.* l'interpréteur) pour Visual Basic est disponible.

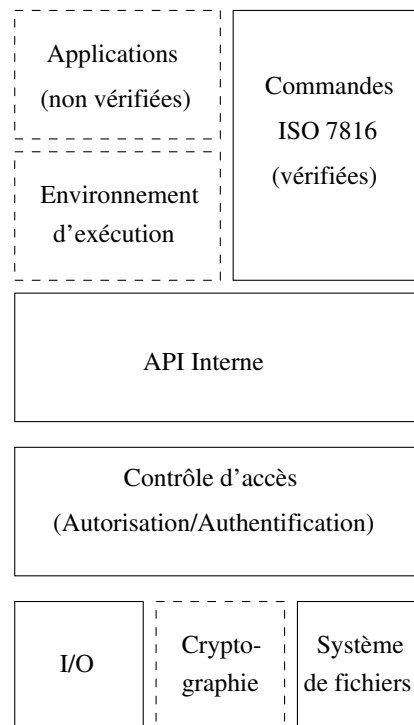


FIG. 2.5 – Architecture de Windows for Smart Card

Gemplus, avec l'annonce de GemShield, a été la première société à réaliser une carte supportant Windows for Smart Card. Depuis, quelques autres sociétés telles NexSmart technology ou DAWAR technologies, ont développé des WinCards.

La carte de NexSmart technology Windows Powered Smart Card With Crypto possède les caractéristiques suivantes :

- un système d'exploitation Windows Powered Smart Card 1.1 ;
- un processeur RISC 8 bits ;
- un coprocesseur cryptographique 1024 bits ;
- 32 Ko de mémoire FLASH ;
- 32 Ko d'EEPROM ;
- 1 Ko de RAM ;
- compatibilité avec le standard ISO 7816 1-4.

La technologie Java Card

Comme la Java Card (*i.e.* la carte multi-applicative utilisant la technologie Java Card) est amenée à connaître un gros succès, nous lui consacrons la deuxième partie de ce chapitre.

2.2 La Java Card

2.2.1 Qu'est ce que Java Card ?

Cette technologie permet aux cartes à puce et à d'autres périphériques à mémoire limitée de faire fonctionner des applications écrites en langage Java. Une Java Card est une carte à puce qui peut charger et exécuter des programmes écrits en Java. Contrairement aux cartes à puce traditionnelles, les programmes exécutés par la carte ne sont pas forcément fournis par l'émetteur de carte.

Pour résumer, la technologie Java Card définit une plateforme sécurisée pour cartes à puce, portable et multi-application qui incorpore beaucoup des avantages du langage Java.

2.2.2 Les avantages de la technologie Java Card

Les avantages apportés par la technologie Java Card aux développeurs d'application sont en grande partie les mêmes que ceux qu'a pu apporter Java aux langages de programmation classiques :

- la facilité de développement des applications ;
- la sécurité ;
- l'indépendance par rapport au « hardware » ;
- la capacité de stockage et de gestion de multiples applications ;
- la compatibilité avec les standards existants sur les cartes à puce.

2.2.3 Présentation de la Java Card

Présentation de son architecture

Comme nous l'avons vu précédemment, les configurations mémoires des cartes à puce sont de l'ordre de 1 Ko de RAM, 16 Ko d'EEPROM et de 24 Ko de ROM. Un grand défi de la technologie Java Card a été de s'adapter à ces contraintes pour construire une carte Java tout en conservant assez de place pour les applications.

La solution fut de supporter seulement un sous-ensemble des caractéristiques du langage Java et de découper la machine virtuelle Java (JCVM) en deux parties. Une partie qui s'exécute en dehors de la carte et une partie qui s'exécute sur la carte. La partie de la JCVM embarqué sur la carte comprend principalement l'interpréteur de bytecode. Beaucoup de tâches ne sont plus réalisées à l'exécution mais sont déportées vers la partie de la machine virtuelle hors carte (*e.g.* le chargement de classes, la vérification de bytecode, la résolution de liens, l'optimisation, etc).

Pour palier aux problèmes de sécurité dû à l'absence de vérificateur embarqué, la technologie Java Card a fourni un environnement d'exécution, le JCRE (Java Card Runtime Environment) chargé de fournir des mécanismes sécuritaires qui permettent une séparation claire entre le système de la carte à puce et les applications (*e.g.* le « firewall », etc). C'est le JCRE qui encapsule la complexité fondamentale et les détails du système des cartes à puce (*cf.* FIG. 2.6). Les applets demandent des ressources et des services systèmes au JCRE à travers les APIs.

À cause du découpage de la JCVM, la plateforme est distribuée entre la carte à puce et la machine de développement dans le temps et dans l'espace. Cette architecture de la technologie Java Card est définie par les trois parties de la spécification suivante :

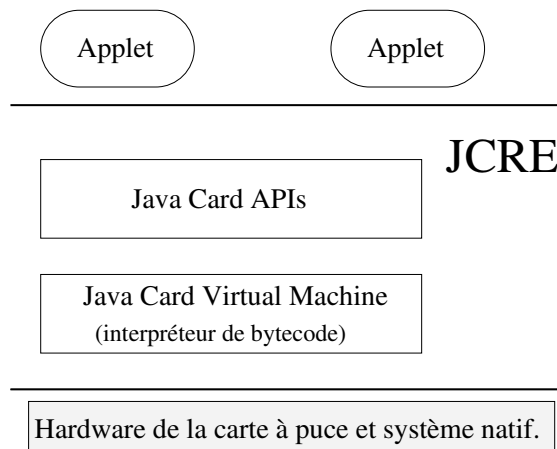


FIG. 2.6 – Architecture de la Java Card

- la Java Card 2.1 Virtual Machine Specification définit un sous-ensemble du langage de programmation Java et la définition de la machine virtuelle nécessaire pour les applications pour des cartes à puce ;
- la Java Card 2.1 Runtime Environment Specification décrit précisément le comportement de l'exécution de la Java Card, comme la gestion de la mémoire, des applets et d'autres caractéristiques ;
- la Java Card 2.1 API Specification décrit l'ensemble du noyau et des extensions des paquetages et des classes Java pour la programmation des cartes à puce.

Le langage Java Card, un sous-ensemble du langage Java

Afin d'optimiser la technologie Java Card pour les environnements à mémoire limitée, seulement un certain nombre de caractéristiques bien choisies de Java ont été spécifiées. Dans le TAB. 2.1 seules les caractéristiques supportées et non supportées importantes ont été incluses.

Caractéristiques Java supportées	Caractéristiques Java non supportées
▷ Type simple de donnée de petite taille : boolean, byte, short ▷ Tableaux à une dimension ▷ Paquetages Java, classes, interfaces et exceptions ▷ Caractéristiques orientées objets : héritage, méthodes virtuelles, surcharge et création dynamique d'objets, contrôle d'accès ▷ Le mot clé int et le support des entiers sur 32 bits sont optionnels	▷ Type simple de donnée de grosse taille : long, double, float ▷ Tableaux à plusieurs dimensions ▷ Caractères et chaînes ▷ Chargement dynamique des classes ▷ Security Manager ▷ Ramasse-miettes et finalisation ▷ Threads ▷ Sérialisation d'objet ▷ Clonage d'objet

TAB. 2.1 – Caractéristiques Java supportées et non supportées

La machine virtuelle de la Java Card : JCVM

La machine virtuelle Java Card (JCVM) a une architecture semblable à celle de la machine virtuelle Java (JVM) à la différence que la JCVM est découpée en deux parties (*cf.* FIG. 2.7) :

- une partie embarquée sur la carte qui inclue l'interpréteur de bytecode ;
- une partie hors carte qui tourne sur une station de travail et comprend le convertisseur.

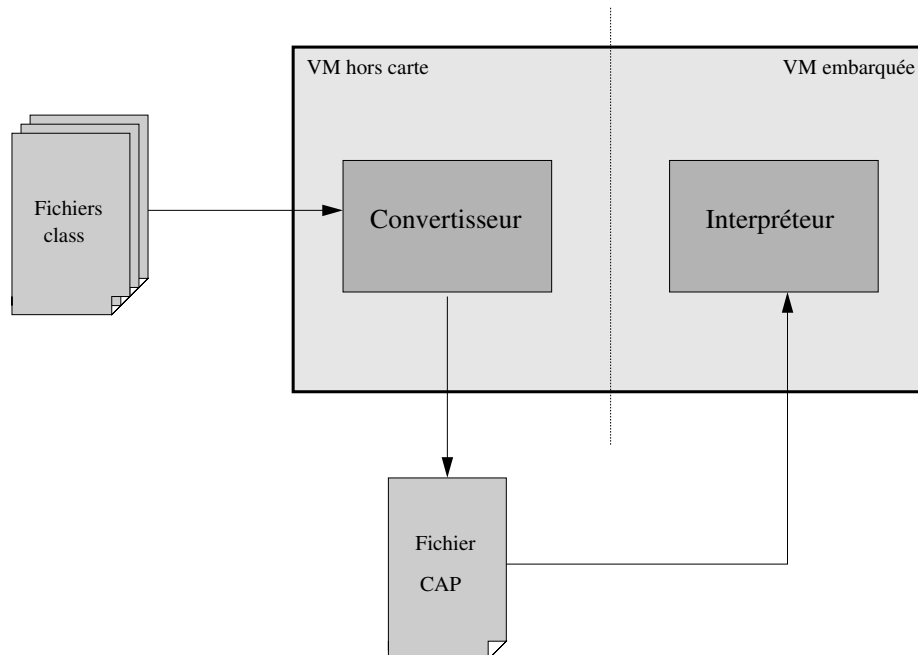


FIG. 2.7 – La machine virtuelle Java Card (JCVM)

Ensemble, ces deux parties implémentent toutes les fonctions d'une machine virtuelle. Le convertisseur charge et traite les fichiers **class** qui composent le paquetage Java et donne en sortie un fichier **CAP** (Converted APplet). En plus de la création du fichier **CAP**, le convertisseur génère un fichier **export** représentant les APIs publiques du paquetage converti. Toute caractéristique non supportée du langage utilisée dans une applet est détectée par le convertisseur. Si la conversion réussie, le fichier **CAP** est alors chargé dans la carte à puce et exécuté par l'interpréteur.

L'environnement d'exécution de la Java Card : JCRE

L'environnement d'exécution de la Java Card (JCRE) consiste en l'ensemble des composants du système Java Card qui tourne à l'intérieur de la carte à puce. Le JCRE est responsable de la gestion des ressources de la carte, communication réseau, exécution des applets, du système de la carte et de la sécurité des applets. Le JCRE est une sorte de système d'exploitation « universel » pour la carte à puce.

En plus du support du modèle d'exécution du langage Java, le JCRE supporte trois autres caractéristiques d'exécution :

- Objets persistants et temporaires.

Par défaut les objets Java Card sont persistents et sont créés dans la mémoire persis-

tante. L'espace et les données de tels objets existent à travers les sessions CAD (Card Acceptance Device). Pour des raisons de sécurité et de performance, les applets peuvent créer des objets en RAM. De tels objets sont appelés temporaires. Les objets temporaires contiennent des données temporaires qui ne persistent pas à travers les sessions CAD.

- Atomicité et transactions.

La JCVM assure que chaque opération d'écriture dans un champ simple d'un objet ou d'une classe est atomique. Le champ mis à jour prend soit la nouvelle valeur soit il est restauré à la valeur précédente. En plus le JCRE fournit des mécanismes de transactions via les APIs. Une applet peut réaliser plusieurs écritures durant une transaction. Soit tout est mis à jour dans une transaction complète, soit si une erreur apparaît au milieu de la transaction, rien n'est réalisé.

- « Firewall » des applets et mécanismes de partage.

Le « firewall » des applets isole les applets. Chaque applet s'exécute à l'intérieur de son espace (contexte). L'existence et les opérations d'une applet n'ont pas d'effet sur les autres applets de la carte. Le « firewall » est renforcé par la JCVM car elle exécute le bytecode. Dans la situation où les applets ont besoin de partager des données ou d'accéder à des services du JCRE, la machine virtuelle autorise de telles possibilités à travers des mécanismes sécurisés de partage.

Les APIs Java Card

Les APIs Java Card sont un ensemble de classes optimisées pour la programmation des cartes à puce en accord avec le modèle ISO 7816. Les APIs contiennent un noyau de trois paquetages (`java.lang`, `javacard.framework` et `javacard.security`) et un paquetage d'extension (`javacardx.crypto`). Beaucoup de classes de la plateforme Java (relatives au réseau, à l'affichage, etc) ne sont pas nécessaires et donc pas disponibles sur la plateforme Java Card.

Les classes dans les APIs Java Card sont compactes. Elles incluent des classes adaptées de la plateforme Java pour fournir un support du langage Java et des services cryptographiques. Elles contiennent aussi des classes créées spécialement pour supporter le standard ISO 7816.

Les applets Java Card

Si les programmes Java Card portent aussi le nom d'« applet » c'est parce qu'ils sont chargeables après la fabrication de la carte et n'ont pas besoin de se trouver dans la ROM. La possibilité de charger les applets confère à la plateforme Java Card un dynamisme qui manque souvent aux systèmes embarqués.

Toute applet Java Card doit hériter de la classe `javacard.framework.Applet`. Cette classe constitue un modèle qui définit les variables et les méthodes pour l'interaction de l'applet avec le JCRE. Une applet exécutable est une instance de la classe `javacard.framework.Applet`. Le cycle de vie d'une applet, comme celui de tout objet persistant, se termine à la fin du cycle de vie de la carte. Le JCRE étant un environnement multi-applicatif, il pourra supporter plusieurs applets sur la même carte.

Le cycle de développement d'une applet peut se décomposer en quatre étapes (cf. FIG. 2.8) :

- La première étape consiste comme pour tout programme Java à compiler les sources afin d'obtenir les fichiers `class`.

- La seconde étape se réduit à tester et à déboguer le comportement général de l'applet sur un simulateur tournant sur une station de travail. L'environnement de simulation utilise la machine virtuelle Java de la station. Ainsi on ne peut pas tester le comportement de l'applet vis-à-vis du « firewall », des objets temporaires et persistants et d'autres caractéristiques.
- La troisième étape est la conversion des paquetages Java afin d'obtenir les fichiers CAPs.
- La quatrième étape permet de tester complètement les applets Java Card sur un émulateur. Un émulateur Java Card émule le JCRE sur une station de travail et comporte une JCVM. Ainsi le comportement de l'applet sera le même sur l'émulateur que sur la carte après son chargement.

La plupart des simulateurs et des émulateurs vont de paire avec un débogueur permettant ainsi au développeur de mettre des points d'arrêts, de suivre les changements d'états, etc.

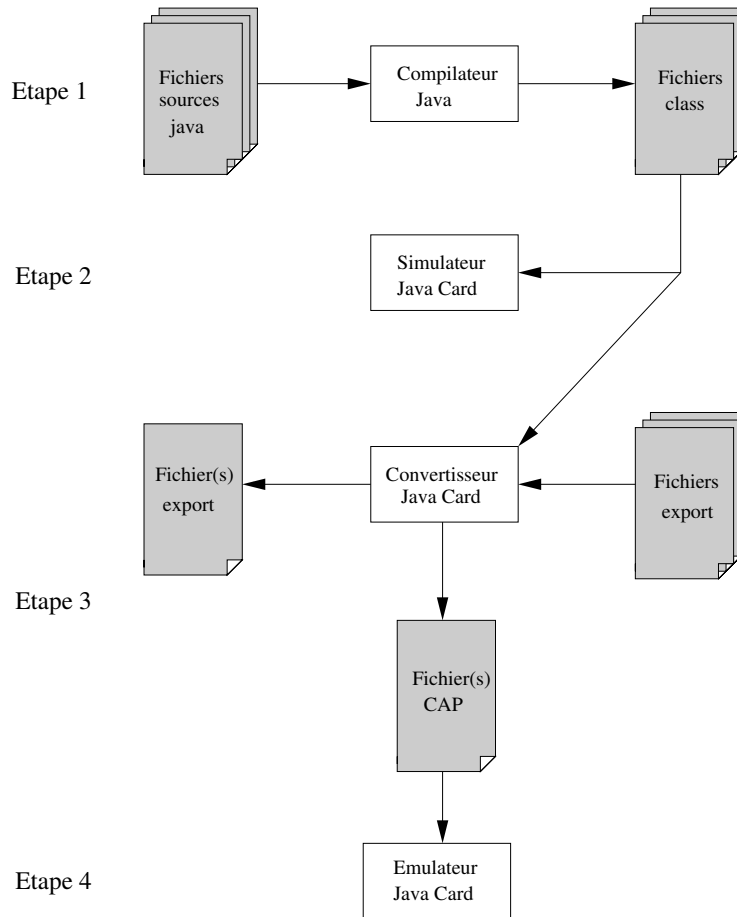


FIG. 2.8 – Les étapes du développement d'une applet Java Card

Lorsque l'applet est finalisée, l'étape suivante sera son installation sur la carte.

Chapitre 3

Les standards et les spécifications des cartes à puce

Dans ce chapitre, nous présentons une vue globale des différents standards et des différentes initiatives de l'industrie pour développer des applications pour les cartes à puce. C'est grâce à ces standards que l'interopérabilité devient possible. Nous verrons d'ailleurs dans le chapitre présentant les différents kits et implémentations Java Cards que les différents produits proposés respectent pour la plupart beaucoup des standards de ce chapitre.

3.1 ISO 7816

Le standard ISO 7816 contient un ensemble de standards qui couvrent différents aspects des cartes à puce avec contact. Ce standard se décline en plusieurs sous-normes que voici :

- partie 1 : caractéristiques physiques ;
- partie 2 : dimensions et position des contacts ;
- partie 3 : signaux électroniques et protocoles de transmission ;
- partie 4 : commandes inter-industrie pour l'échange ;
- partie 5 : identificateur d'application ;
- partie 6 : éléments de données inter-industrie ;
- partie 7 : commandes inter-industrie pour SCQL (Structured Card Query Language) ;
- partie 8 : sécurité de l'architecture et des commandes inter-industrie.

Si vous désirez plus d'informations sur ces points vous pouvez vous reporter au rapport interne de LaBRI RR-1259-01 [3].

3.2 GSM

L'ETSI (European Telecommunications Standards Institute) a publié un ensemble de standards qui couvre l'utilisation des cartes à puce dans les systèmes de téléphonie publique et cellulaire. Le GSM (Global System for Mobile communications) défini par l'ETSI est une spécification pour un système de téléphonie mobile terrestre international. Au départ, cette norme avait pour objectif de couvrir quelques pays d'Europe centrale, puis elle s'est développée pour devenir un standard international pour la téléphonie mobile. Il y a plusieurs standards GSM, et en particulier :

- GSM 11.11 qui spécifie l'interface entre la carte SIM et l'équipement mobile (*e.g.* le téléphone). Une partie de ce standard définit par exemple la taille de la carte SIM.
- GSM 11.14 qui spécifie l'interface entre les outils de développement d'applications SIM pour la carte SIM et l'équipement mobile (*e.g.* le téléphone).
- GSM 3.48 qui spécifie les mécanismes de sécurité des outils de développement d'applications SIM.
- GSM 3.19 qui définit les APIs SIM pour la plateforme Java Card. Ce standard, basé sur GSM 11.11 et GSM 11.14, définit les APIs Java pour le développement d'applications GSM qui s'exécutent sur la plateforme Java Card. Ces APIs sont une extension de APIs Java Card 2.1.

3.3 EMV

En Décembre 1993, Europay International, Mastercard International et Visa International ont créé un groupe de travail, EMV, afin de développer un ensemble de spécifications techniques, basées sur le standard ISO 7816, des cartes à circuit intégré pour l'industrie bancaire. Le but de ces spécifications était de développer l'industrie bancaire avec une base sur laquelle les applications pour les cartes seraient développées. Ils avaient l'intention de définir le minimum de fonctionnalités nécessaires pour supporter une interopérabilité internationale entre les cartes, les terminaux et les périphériques en rapport avec les cartes. Ils cherchaient aussi à établir un niveau de sécurité minimum élevé pour les cartes à circuit intégré et les terminaux afin de décourager la contrefaçon et les autres types de fraude, tout en permettant à chaque produit de fournir des caractéristiques de sécurité supplémentaires.

La dernière version de ces spécifications, EMV 96 version 3.1.1, a été publiée en Mai 1998 et comprend trois livres :

- Integrated Circuit Card Specification ;
- Integrated Circuit Card Terminal Specification ;
- Integrated Circuit Card Application Specification.

La Card Specification se compose de quatre parties :

- La partie 1 est identique aux deux premières parties du standard ISO 7816. Elle définit les caractéristiques électriques et mécaniques de la carte et du terminal.
- La partie 2 définit les éléments de données et les commandes qui s'appliquent à l'échange d'information entre la carte et le lecteur.
- La partie 3 définit la façon dont une application hors-carte peut sélectionner l'application appropriée dans le cas d'une carte multi-applicative.
- La partie 4 rappelle les aspects sécuritaires à respecter dans les étapes définies dans les parties 1 à 3.

La Terminal Specification détaille les conditions obligatoires et optionnelles pour les terminaux utilisés pour les transactions de cartes de crédit, terminaux pour PC, etc. Cette spécification couvre aussi :

- les caractéristiques physiques générales des lecteurs de cartes ;
- l'architecture logicielle comme la gestion des données et des programmes ;
- la sécurité ;
- etc.

L'Application Specification définit les procédures nécessaires pour réaliser un système transactionnel de paiement dans un environnement international d'échange. Ces procédures incluent :

- le mappage des éléments de données sur les fichiers ;
- les flux transactionnels ;
- les erreurs d'exécution ;
- le codage de objets de données spécifiques décrits dans la Card Specification.

La carte à puce MPCOS de Gemplus est compatible EMV et est certifiée par Visa International. Comme il n'y a pas de procédures ou d'outil existant pour vérifier la conformité aux spécifications EMV, toute proclamation de compatibilité EMV est discutable. D'habitude l'affichage d'une compatibilité EMV signifie que le fabricant a implémenté une partie ou plus des spécifications, et plus probablement le mécanisme de sélection des applications car c'est le point le plus intéressant d'EMV.

3.4 Open Platform

Open Platform (OP) définit un environnement intégré pour le développement et les opérations des cartes à puce multi-applicative.

Open Platform consiste en :

- une spécification pour la carte qui définit :
 - les conditions inter-industrie et d'indépendance à la cible pour implémenter Open Platform,
 - les communications hors carte avec le terminal et les communications sur la carte pour la gestion des applications ;
- une spécification pour le terminal qui définit :
 - la partie de l'architecture des applications à l'intérieur du terminal,
 - le terminal pour qu'il soit compatible ISO et EMV.

Les spécifications Open Platform étaient initialement développées par Visa International – c'est pourquoi nous trouvons souvent la notation Visa Open Platform (VOP) – et ont maintenant été transférées vers GlobalPlatform, une organisation chargée de promouvoir une infrastructure globale inter-industrie pour les implémentations de cartes à puce.

Les spécifications actuelles sont :

- Card-Related Chip Technology qui inclue :
 - Open Platform Card Specifications v2.0.1,
 - Open Platform Card Configurator and Script Builder v2.0.2,
 - Open Platform Java Card Applet Compliance Guide v2.0.1,
 - Open Platform Card Applet Developer's Guide v2.0.1 ;
- Terminal-Related Chip Technology qui inclue :
 - Open Platform Terminal Specifications v1.5,
 - Open Platform API Reference Implementation ;
- Systems-Related Chip Technology qui inclue :
 - Multiple Application Smart Card Management Systems Open Platform Functional Requirements v3.3.

3.5 OpenCard Framework

Le projet OpenCard Framework (OCF) a été initié par IBM. Maintenant il appartient et est développé par OpenCard consortium qui regroupe les industriels de premier plan du monde de la carte à puce : 3-G International, American Express Travel Related Services, Bull, First Access, Gemplus, Giesecke & Devrient, IBM, Toshiba Corporation, TOWITOKO, Schlumberger, Siemens, Sun Microsystems, UbiQ Inc., Visa International, XAC Automation.

OCF est un modèle fournissant une interface standard à l'application hôte pour interagir avec le lecteur de carte et les applications sur la carte. C'est donc un standard ouvert qui fournit une architecture et un ensemble d'APIs qui permet aux développeurs d'application et aux fournisseurs de service de construire et déployer des solutions à base de cartes à puce dans un environnement compatible OpenCard. L'architecture de OCF est un modèle structuré qui regroupe des fonctionnalités pour :

- les vendeurs de terminaux ;
- les fournisseurs de systèmes d'exploitation pour carte à puce ;
- les émetteurs de carte.

Le but de OCF est de fournir une indépendance vis-à-vis :

- des vendeurs de terminaux ;
- des fournisseurs de systèmes d'exploitation pour carte à puce ;
- des émetteurs de carte.

OCF est conçu dans l'esprit afin de permettre l'utilisation des cartes à puce dans les réseaux en utilisant le langage de programmation Java.

3.6 PC/SC

Les spécifications PC/SC (Interoperability Specification for ICCs and Personal Computer Systems) appartient et est défini par le PC/SC Workgroup, un consortium industriel qui regroupe Apple, Bull, Gemplus, Hewlett Packard, Infineon, Intel, Microsoft, Schlumberger et Toshiba. PC/SC définit une architecture pour l'utilisation des cartes à puce sur les systèmes PC.

Le PC/SC Workgroup a identifié trois parties à standardiser :

- l'interfaçage du lecteur au PC ;
- une API de haut niveau pour accéder aux fonctionnalités de la carte ;
- les mécanismes autorisant de multiples applications à partager les mêmes ressources telles que la carte ou le lecteur.

Les spécifications PC/SC 1.0 se divise en huit parties :

- La partie 1 fournit un survol de l'architecture système et des composants définits par le groupe de travail.
- La partie 2 détaille :
 - les caractéristiques de compatibilité entre la carte à puce et le lecteur ;
 - les conditions pour l'interopérabilité.
- La partie 3 décrit l'interface pour la compatibilité des lecteurs et les fonctionnalités requises.

- La partie 4 présente des considérations sur la construction des lecteurs. En particulier, elle fournit une implémentation recommandée pour l'intégration du lecteur au clavier PS/2.
- La partie 5 décrit les interfaces et les fonctionnalités supportées par le gestionnaire de ressources de cartes à puce. Le gestionnaire de ressources de cartes à puce est l'élément central du système PC/SC. Il contrôle tous les lecteurs, les ressources du fournisseur de service des cartes et les ressources du fournisseur de services cryptographiques.
- La partie 6 :
 - décrit le modèle de fourniture de service des cartes ;
 - décrit le modèle de fourniture de services cryptographiques ;
 - identifie les interfaces requises ;
 - indique comment cela peut être étendu aux besoins d'une application d'un domaine spécifique.
- La partie 7 décrit, pour les développeurs, des considérations sur la construction d'application et comment utiliser les autres composants.
- La partie 8 décrit les fonctionnalités recommandées pour les cartes qui supportent la cryptographie.

Dans l'architecture PC/SC, les applications hôtes sont construites au dessus d'un ou plusieurs fournisseurs de services et d'un gestionnaire de ressources. Un fournisseur de services encapsule les fonctionnalités attendues par une carte à puce spécifique et la rend accessible à travers des interfaces de programmation de haut niveau. Un gestionnaire de ressources gère les ressources relatives aux cartes à l'intérieur du système pour accéder aux lecteurs et, à travers eux, aux cartes à puce.

PC/SC et OCF ont beaucoup de concepts similaires. Sur une plateforme Windows, OCF peut accéder au lecteur à travers le gestionnaire de ressources PC/SC installé.

3.7 Résumé de l'étendue des divers standards présentés

Nous avons résumé l'étendue des standards et des spécifications des cartes à puce que nous avons présentés (*cf.* TAB. 3.1) afin d'en dégager les spécificités et les différences.

		ISO 7816	GSM	EMV	VOP	OCF	PC/SC
CARTE	Coté Physique	✓	✓				
	Coté Contact	✓	✓				
	Coté Lecteur		✓	✓	✓	✓	✓
INTERFACE	Organisation logique des fichiers	✓	✓				
	ensemble de commandes	✓	✓	✓			✓
API	Application Programming Interface				✓	✓	✓
APPLICATION	Signature numérique	✓					
	Téléphonie numérique		✓				
	Débit/Crédit			✓			

TAB. 3.1 – Étendue des standards et des spécifications des cartes à puce

Chapitre 4

Implémentations et kits Java Cards existants

La plupart des informations que nous indiquons pour les différentes implémentations et les différents kits Java Card existants, proviennent des documentations commerciales des produits – quand elles sont disponibles – et nous devons donc leur faire confiance. Comme nous l’avons déjà précisé la technologie Java Card n’est pas seulement destinée aux cartes à puce, mais à tous les périphériques ayant peu de mémoire. Nous verrons ainsi une implémentation pour les anneaux Java [20].

4.1 L’anneau Java (Java Ring)

Le Java Ring iButton est produit par Dallas Semiconductor et a la forme et la taille d’une pile plate pour montre ou calculatrice. Dallas Semiconductor produit plusieurs types de iButtons, et tous ne sont pas des Java Rings. Par exemple, le badge d’accès du LaBRI est un iButton de type «mémoire» dont la taille mémoire est de 1 Ko.

Le Java Ring est une implémentation des spécifications Java Card 2.0 pour les iButtons. D’ailleurs le iButton a été l’un des premiers périphériques à supporter la technologie Java Card.

L’architecture matérielle du Java Ring iButton se compose :

- d’un processeur rapide ;
- de 32 Ko de ROM pour stocker la machine virtuelle et le système d’exploitation ;
- de 6 Ko à 134 Ko (pour les derniers modèles) de RAM à 100 nanosecondes en écriture pour fournir de bonnes performances pour la JCVM ;
- d’un accélérateur mathématique pour réaliser la cryptographie en RSA ;
- etc.

Les caractéristiques de la technologie Java Card supportées par le Java Ring iButton sont :

- un vrai support des entiers sur 32 bits ;
- un ramasse-miettes ;
- l’ajout et la suppression d’applets de manière sécurisée ;
- les classes cryptographiques standards ;
- etc.

Les documentations ne donnent aucune information sur le support des caractéristiques du JCRE tels que le « firewall », etc.

4.2 Les cartes à puce Java

4.2.1 Le kit de développement de Sun Microsystems, Inc.

La dernière version du kit de développement de Sun Microsystems est très récente puisqu'elle est apparue en Mai 2001. Cette version du kit, la 2.1.2., a subi de nombreuses et fondamentales modifications.

Le kit se présente comme un ensemble d'outils nécessaires au développement d'applications pour les Java Cards. Pour convertir les fichiers sources, le kit utilise toujours le compilateur Java classique pour générer les fichiers `class` (pour plus de détails sur le développement d'une application Java Card cf. 2.2.3). À partir de cette étape deux possibilités s'offrent à nous :

- En effet, la différence fondamentale entre cette version et la précédente est l'ajout d'outils offrant la possibilité de créer des masques normalement destinés à être placés dans la ROM lors de la fabrication. Mais il est possible d'utiliser ces masques dans un environnement d'exécution (*i.e.* un émulateur) fourni avec le kit.
- Comme dans l'ancien kit de développement, nous avons aussi la possibilité de développer les applets ou les paquetages un par un, et de les charger dans l'environnement d'exécution en simulant un échange d'APDUs (sur un port TCP/IP ouvert par l'émulateur) qui symbolise le chargement du fichier `CAP` sur la carte.

Cette nouvelle version du kit permet réellement de tester ses applications de manière sérieuse.

4.2.2 Les kits de développement et les implémentations Java Card

Le kit Odyssey-Lab de Bull

Le kit Odyssey-Lab est un environnement de développement fournissant les outils nécessaires pour optimiser et charger les applets. La dernière carte fournie avec le kit, la carte Odyssey 1.2, est conforme aux spécifications Java Card 2.1 et VOP 2.0. Elle possède 8, 16 ou 32 Ko de EEPROM. Selon Bull, la carte Odyssey 1.2 dispose de solides avantages sécuritaires. La carte possède, en effet, de nouveaux mécanismes de segmentation des zones de la mémoire, empêchant ainsi tout accès illicite entre les secteurs volontairement partitionnés. L'installation des applets peut aussi se faire en mode signé et crypté, ce qui assure une garantie totale de confidentialité, d'intégrité et d'authenticité. En plus la carte supporte les différents types de cryptographie (*i.e.* symétrique et asymétrique). La carte Odyssey offre en standard diverses applications (*e.g.* programmes de fidélité, porte-monnaie électronique, etc).

Le kit GemXpresso RAD (Rapid Applet Development) de Gemplus

Le numéro un de la carte à puce propose toute une gamme de produits. Le plus abouti est le kit de développement GemXpresso RAD 211/PK. Ce kit se compose de deux cartes GemXpresso 211IS, d'un lecteur de cartes et d'un environnement de développement. Gemplus propose plusieurs Java Cards, dont certaines ont un processeur à architecture RISC. La carte

GemXpresso 211IS est une carte conforme aux spécifications Java Card 2.1 et Visa Open Platform 2.0. Elle supporte les APIs Java Card 2.1 et des APIs propriétaires de Gemplus qui fournissent par exemple des routines de gestion plus facile des objets temporaires. Le JCRE de la carte comporte donc le « firewall » et les mécanismes de partage, les objets temporaires, les mécanismes d'atomicité et de transaction, etc. L'environnement de développement fourni par Gemplus permettrait d'accélérer le développement et les tests des applets grâce au simulateur intégré.

Le kit Sm@rtCafé II de Giesecke & Devrient

Nous ne disposons d'aucune information pour le moment sur le kit de développement Sm@rtCafé II.

Le kit MoKard de Incard

Mokard est un kit orienté développement de la technologie Java Card pour le GSM. La seule information disponible sur le kit est sa compatibilité avec le kit 2.1.1 de Sun Microsystems, Inc.

Le kit M-Smart JADE de Motorola

Ce kit, selon Motorola, permet de développer et de tester rapidement les applications pour les cartes à puce dans un environnement Java. En effet ce kit inclut, comme celui de Gemplus, un outil de simulation et de débogage pour le développement des applets Java Card. Le kit permet aussi de générer les fichiers d'installation et de procéder à leur installation sur la carte. Il supporte les Java Cards de Motorola, les cartes M-Smart JUPITER, et toutes les cartes à puce compatible Java Card 2.1. Les cartes M-Smart JUPITER sont équipées de processeur à architecture RISC 32 bits, de 16 Ko d'EEPROM, de 48 Ko de ROM et de 3 Ko de RAM. Elles sont conformes aux spécifications Java Card 2.1 et Visa Open Platform 2.0

Le kit NexSmart Java Card SDK de NexSmart technology

Ce kit se compose de :

- deux NexSmart Java Card 2.1.1, réutilisables pour le SDK seulement ;
- un lecteur de carte PC/SC ;
- un guide utilisateur Java Card ;
- un guide du programmeur Java Card.

Les caractéristiques du kit sont :

- la compatibilité PC/SC ;
- un SDK écrit en Java ;
- un outil pour envoyer des commandes APDUs à la carte et recevoir les réponses APDUs de la carte ;
- des exemples simples ;
- la compatibilité OpenCard Framework pour la gestion du lecteur.

Les cartes NexSmart Java Card possèdent le système d'exploitation Sun Microsystems Java Card 2.1.1. Elles sont équipées de 32 Ko d'EEPROM et sont compatibles avec les standards :

- ISO 7816 1-6 ;
- Java Card 2.1.1 ;
- Visa Open Platform 2.0.1 ;
- EMV 3.1.1.

Le kit GalactIC de Oberthur Card Systems

Presque aucun détail n'est disponible sur le kit. En revanche, nous disposons de quelques informations sur les Java Cards d'Oberthur :

- GalactIC 2.1 est, selon Oberthur Card Systems, la première Java Card compatible avec la technologie Visa Open Platform 2.1. Elle possède 32 Ko d'EEPROM. L'architecture de la carte est compatible avec tous les standards industriels reconnus. Pour rehausser la sécurité, les applications sont placées dans des zones mémoires sécurisées, protégées par une série de « firewalls ».
- CosmopoIIC est dédiée au paiement, à la sécurisation des réseaux et répond aux besoins pour un vrai « e-business » sécurisé. Elle possède 64Ko d'EEPROM.
- SIMphonIC Card V1 est dédiée à la téléphonie mobile. Elle est compatible avec Java Card 2.1, ETSI SIM API et Visa Open Platform.

Le kit Cyberflex Access SDK de Schlumberger

La gamme des Java Cards de Schlumberger porte le nom de Cyberflex. Par exemple, la carte Cyberflex Simera est dédiée à la téléphonie mobile. La carte Cyberflex Palmera est plutôt orientée vers les applications bancaire et financière. Cette carte est compatible avec les standards :

- ISO 7816 ;
- Java Card 2.1.1 ;
- Visa Open Platform 2.0.1 ;
- EMV 3.1.1.

Elle dispose d'un processeur 16 bits, de 64 Ko de ROM et de 32 Ko d'EEPROM. La carte contient la machine virtuelle et le système d'exploitation Solo. Le kit Cyberflex Access SDK V3C contient à peu près les mêmes éléments que celui de Gemplus. Il propose sur le cédérom le Cyberflex Toolkit. C'est un ensemble de programmes destinés à préparer et gérer les applets (ou cardletsTM comme on les nomme chez Schlumberger) de la carte Cyberflex. Le kit posséderait, comme celui de Gemplus, un simulateur. L'outil MakeSolo permet de vérifier l'absence de caractéristiques non supportées dans les fichiers `class` et de gérer les communications APDU pour le chargement et la gestions des fichiers et des clés de la carte.

La carte SmartJ de ST Microelectronics

La SmartJ dispose d'un processeur à architecture RISC 32 bits, très rapide, qui peut exécuter la majorité des bytécodes Java Card 2.1 directement. En plus de ce nouveau cœur on a des mémoires étendues, des architectures de sécurité et les fonctions standards d'entrée/sortie des cartes à puce. ST Microelectronics voit en sa carte l'une des plus sophistiquées du monde.

Chapitre 5

Développement logiciel

Nous présentons ici l'outil de visualisation et de modification d'applications ou de paquets Java Card que nous avons développé dans le cadre de ce mémoire. Une application ou un paquetage Java Card est représenté par un fichier spécial (**CAP** cf. 2.2.3) dont nous décrivons le format.

5.1 Fichier CAP

Dans la technologie Java, le fichier **class** est la pièce centrale de l'architecture Java. Il définit le standard pour la compatibilité binaire de la plateforme Java. À cause des caractéristiques distribuées de l'architecture système de la Java Card, le fichier **CAP** est le format standard de fichier pour la compatibilité binaire de la plateforme Java Card.

Un fichier **CAP** contient une représentation binaire exécutable des classes d'un paquetage Java. Ce fichier **CAP** est un archivage d'un ensemble de composants stockés en utilisant le format de fichier JAR. Chacun de ces composants est stocké comme un fichier individuel dont le nom a pour extension **CAP**. Le format du fichier de chacun de ces composants suit le modèle suivant :

Étiquette	Taille	Données
-----------	--------	---------

TAB. 5.1 – Format de fichier d'un composant du paquetage

Chaque composant décrit un aspect du contenu d'un fichier **CAP**. Il y a dans la spécification 2.1, onze composants officiels dont deux sont optionnels. Voici la liste des composants avec leur étiquette et une brève description :

- header (1) : contient le « nombre magique » et les numéros de versions de l'implémentation de la machine virtuelle Java Card pouvant supporter le paquetage ;
- directory (2) : contient la liste des autres composants avec leur taille ;
- applet (3) (optionnel) : contient la liste des applets avec leur AID et une référence sur la méthode `install()` ;
- import (4) : contient la liste des paquetages référencés dans ce paquetage ;
- constant pool (5) : contient la table de références aux classes, aux méthodes et aux champs, respectivement, dans les composants de classes, de méthodes et de champs ;

- class (6) : contient la table des classes et interfaces ;
- method (7) : contient le code des méthodes des classes ;
- static field (8) : contient les valeurs initiales de tous les champs statiques du paquetage ;
- reference location (9) : contient la liste des références relatives dans le code des méthodes à transformer au chargement en références absolues ;
- export (10) (optionnel) : contient la liste des éléments de ce paquetage que les autres paquetages peuvent référencer directement (méthodes et champs statiques **public** ou **protected** dans les classes publiques) ;
- descriptor (11) : contient des informations de typage (signatures) si ce paquetage est accessible par d'autres paquetages.

5.2 Présentation des caractéristiques de notre outil

Notre outil est donc un éditeur de fichiers **CAP**. Il a été réalisé en utilisant la technologie Java. Son interface a été conçue en utilisant la bibliothèque graphique Swing de Sun Microsystems. La taille totale du code est de 105 Ko.

Il permet l'ouverture, la modification et la sauvegarde de fichiers **CAP**. A notre connaissance aucun outil identique n'était disponible. Avant, la seule solution pour réaliser une modification même infime du fichier **CAP** consistait en l'extraction de l'archive JAR du composant souhaité, puis en sa modification à l'aide d'un éditeur hexadécimal. Enfin il fallait replacer le composant dans l'archive. En plus, la modification d'un composant de l'archive nécessite souvent des modifications dans les autres composants. C'était donc une tâche très fastidieuse que celle de modifier un fichier **CAP**.

Notre outil rend la tâche plus simple car il rend transparent la gestion de l'archive et vous propose directement l'accès aux composants. Pour quatre composants sur les onze que contient un fichier **CAP**, nous proposons une visualisation et une modification facilitée puisque présentée sous une forme graphique.

5.3 Présentation de l'interface de notre outil

Comme le montrent les captures d'écrans (FIG. 5.1, FIG. 5.2, FIG. 5.3, FIG. 5.4 et FIG. 5.5), notre éditeur est plus convivial qu'un simple éditeur hexadécimal. En outre, il dispose d'une aide contextuelle. Il a été construit afin de laisser les mêmes possibilités de modifications du fichier **CAP** qu'en utilisant un éditeur hexadécimal. Néanmoins il simplifie les modifications. Pour cela, notre éditeur n'a volontairement pas été entièrement automatisé et donc l'utilisateur devra avoir une connaissance minimum sur les modifications qu'il veut réaliser et sur leurs conséquences au sein du fichier **CAP**.

Le TAB. 5.2 présente les différentes boîtes de dialogue que l'on peut obtenir depuis le composant Applet (*cf.* FIG. 5.2). Ces boîtes de dialogue correspondent à l'insertion d'une applet, la suppression d'une applet, l'inversion de la position de deux applets et le déplacement d'une applet. Elles se retrouvent de manière presque identiques pour les composants Import et Constant Pool afin réaliser les mêmes opérations sur leur contenu respectif.

Sur la FIG. 5.1, on voit comment l'éditeur affiche les différents champs du composant Header.

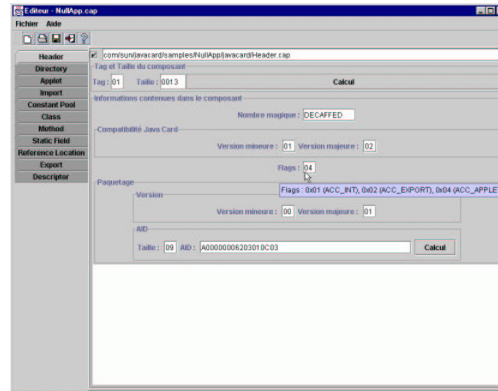


FIG. 5.1 – Affichage du composant Header

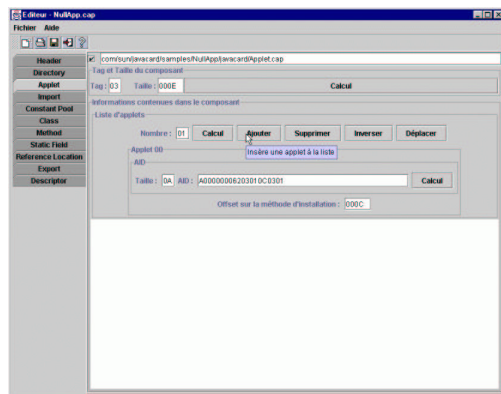


FIG. 5.2 – Affichage du composant Applet

La FIG. 5.2 permet de voir les possibilités de modifications permises par l'interface (cf. TAB. 5.2) :

- insertion d'une applet ;
- suppression d'une applet ;
- inversion de deux applets ;
- déplacement d'une applet.

Cette interface permet de réaliser des modifications plus simplement qu'avec un éditeur hexadécimal.

Sur la FIG. 5.3 on voit qu'on dispose de la même interface pour le composant Import que pour le composant Applet (cf. FIG. 5.2).

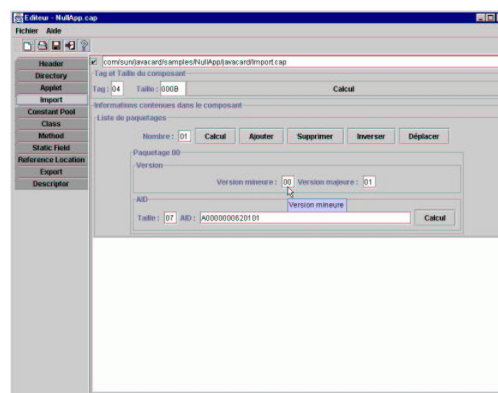


FIG. 5.3 – Affichage du composant Import

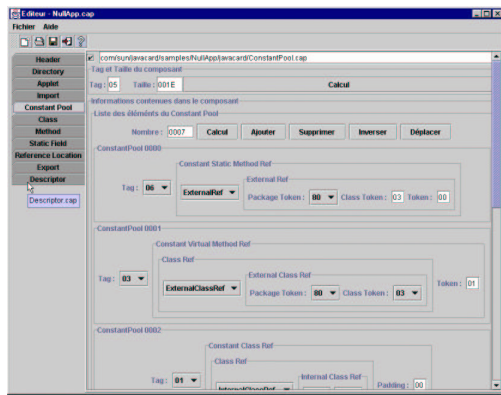


FIG. 5.4 – Affichage du composant Constant Pool

Sur la FIG. 5.5, on peut voir le composant Class qui lui ne dispose que d'un affichage à la manière d'un éditeur hexadécimal. Les composants Directory, Method, Static Field, Reference Location, Export et Descriptor ont le même type d'affichage pour l'instant.

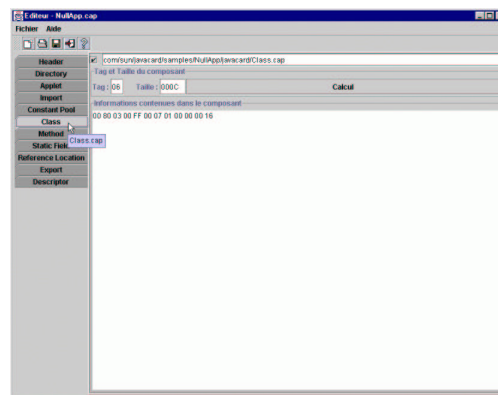
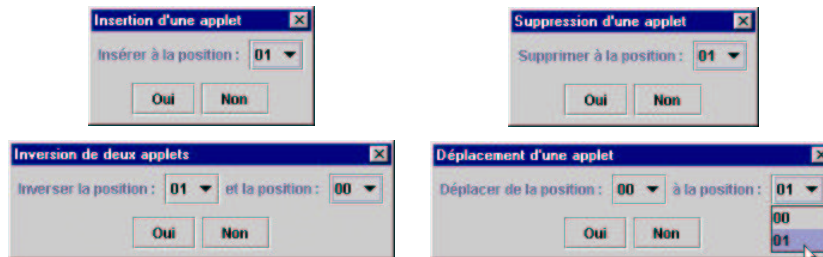


FIG. 5.5 – Affichage du composant Class



TAB. 5.2 – Les boîtes de dialogue disponibles à partir du composant Applet

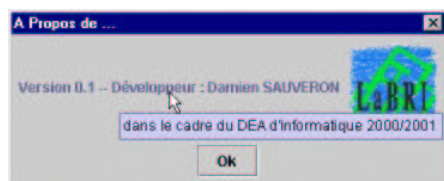


FIG. 5.6 – A Propos de ...

La FIG. 5.6 présente simplement la boîte A Propos de ... qui contient le numéro de version, le nom de l'auteur et le laboratoire dans lequel le développement a eu lieu.

5.4 Objectifs de cet outil

Il est raisonnable de se demander à quoi peut bien servir un tel outil puisque les fichiers **CAP** une fois générés sont destinés normalement à être chargés sur la carte. Pour nous, le développement de cet outil a pour objectif de réaliser des attaques sur la carte. En effet, nous espérons qu'en raison de l'absence de vérificateur embarqué, nous allons pouvoir créer des attaques en modifiant le fichier **CAP**.

5.5 Améliorations à réaliser

Notre éditeur peut encore subir de nombreuses améliorations. Par exemple on pourra réaliser l'affichage graphique des sept composants restants. Il pourrait aussi être intéressant de repenser l'interface afin qu'elle soit encore plus intuitive et moins lourde pour des machines peu puissantes.

Chapitre 6

Perspectives

Comme nous venons de le voir, le monde de la Java Card est en plein essor et de nouveaux produits ne cessent d'apparaître. Afin de valider la technologie Java Card, le LaBRI et SERMA Technologies ont mis en place une collaboration à la hauteur des enjeux. En plus de cette collaboration une thèse CIFRE, intitulée « Étude et réalisation d'un environnement d'expérimentation et de modélisation pour les nouvelles technologies embarquées : application à la sécurité dans un environnement Java Card », devrait voir le jour à la rentrée 2001. Nous allons donc présenter quelques pistes de recherches.

6.1 Le projet entre le LaBRI et SERMA Technologies

Le projet devrait voir le développement, très prochainement d'un émulateur Java Card – donc d'une machine virtuelle Java Card (JCVM) et de l'environnement d'exécution Java Card (JCRE) – afin de tester les attaques mises en place mais aussi pour découvrir d'éventuels problèmes d'implémentation liés à des erreurs dans les spécifications.

Le développement de la JCVM ne devrait pas poser de difficultés grâce à toutes les caractéristiques non supportées (*cf.* 2.2.3). En particulier l'absence de threads et du ramasse-miettes simplifient considérablement le développement.

Le projet verra aussi :

- le développement d'une application Java Card réaliste qui servira de cas test pour réaliser les attaques ;
- le développement de logiciels afin de créer des failles dans les applications chargées dans la carte.

Le projet pourra, par exemple, voir la poursuite du développement de l'éditeur de fichier CAP présenté dans ce mémoire.

Une partie du projet consistera aussi en une modélisation des spécifications de la technologie Java Card.

6.2 La thèse CIFRE

Cette thèse a pour but de réaliser une modélisation formelle de la plateforme Java Card, et plus particulièrement des mécanismes sécuritaires (*e.g.* le « firewall » entre les applets et

les mécanismes de partage, etc).

En fait cette thèse devrait permettre au travers de la collaboration avec SERMA Technologies de modéliser les attaques, d'identifier les problèmes dans les spécifications qui permettent ces attaques et de proposer des corrections. De plus elle s'insère parfaitement au sein de la collaboration et permettra de poursuivre les travaux accomplis une fois la collaboration entre le LaBRI et SERMA Technologies arrivée à terme.

Conclusion

Depuis sa création dans les années 1970 jusqu'à aujourd'hui la carte à puce a subi une lente mais importante évolution. Au fil du temps sa sécurité s'est renforcée et son importance dans la vie quotidienne en Europe s'est accrue. Aujourd'hui elle est incontournable. Malgré tous ces avantages, il lui reste encore un marché énorme à conquérir et en particulier celui des États-Unis. La Java Card saura peut-être grâce à sa technologie séduisante lancer la révolution qui banalisera la technologie de la carte à puce à travers le monde. Mais pour réussir ce pari, il lui faudra prouver qu'elle est parfaitement sûre.

Bibliographie

- [1] Zhiqun CHEN.
Java CardTM Technology for Smart Cards.
Addison-Wesley - ISBN 0-201-70329-7.
- [2] Uwe HANSMANN, Martin SCOTT NICKLOUS, Thomas SCHAECK et Frank SELIGER.
Smart Card Application Development Using Java.
Springer - ISBN 3-540-65829-7.
- [3] Damien SAUVERON.
La technologie Java Card : Présentation de la carte à puce. La Java Card.
Rapport interne du LaBRI RR-1259-01.
- [4] Ludovic ROUSSEAU.
La sécurité des cartes à puce.
Cours de cryptographie et sécurité. Janvier 2001, Bordeaux 1.
- [5] Nathalie FEYT.
La carte à puce.
Séminaire ENSEIRB. Janvier 2001, Bordeaux 1.
- [6] Java Card Technology Home Page.
<http://java.sun.com/products/javacard/>
- [7] International Standards Organisation.
<http://www.iso.ch/>
- [8] Supplément rattaché à Eurêka.
Numéro 51, Janvier 2000.
- [9] Roger GANNE et Brigitte SALOMONI.
La carte à mémoire.
Eyrolles.
- [10] Jorge FERRARI, Robert MACKINNON, Susan POH et Lakshman YATAWARA.
Smart Cards : A Case Study.
IBM. International Technical Support Organization.
<http://www.redbooks.ibm.com/>
- [11] Carine COURBIS.
Simulation d'applications Java Card.
Rapport de stage du DEA d'Informatique de Lyon. 1998.
INRIA de Sophia-Antipolis.
<http://www-sop.inria.fr/oasis/personnel/Carine.Courbis/pubs/>

- [12] Jean-Jacques VANDERWALLE.
Construction d'applications avec la carte à puce.
École d'été IMAG, INRIA, LIFL. Août 1999, Autrans.
<http://www.essi.fr/~riveill/cours/car/08-99-slides-javacard.pdf>
- [13] Sun microsystems.
Java CardTM 2.1.1 Application Programming Interface.
<http://java.sun.com/products/javacard/javacard21.html>
- [14] Sun microsystems.
Java CardTM 2.1.1 Runtime Environment (JCRE) Specification.
<http://java.sun.com/products/javacard/javacard21.html>
- [15] Sun microsystems.
Java CardTM 2.1.1 Virtual Machine Specification.
<http://java.sun.com/products/javacard/javacard21.html>
- [16] Direction Centrale de la Sécurité des Systèmes d'Information.
<http://scssi.gouv.fr/>
- [17] JavaWorld.
Understanding Java Card 2.0.
<http://www.javaworld.com/javaworld/jw-03-1998/jw-03-javadev.html>
- [18] JavaWorld.
Smart Card: A primer.
<http://www.javaworld.com/jw-12-1997/jw-12-javadev.html>
- [19] JavaWorld.
How to write a Java Card applet: developer's guide.
<http://www.javaworld.com/jw-07-1999/jw-07-javacard.html>
- [20] JavaWorld.
An introduction to the Java Ring.
<http://www.javaworld.com/jw-04-1998/jw-04-javadev.html>
- [21] European Telecommunications Standards Institute.
<http://www.etsi.org/>
- [22] Global Platform.
<http://www.globalplatform.org/>
- [23] OpenCard consortium.
<http://www.opencard.org/>
- [24] PC/SC Workgroup.
<http://www.pcscworkgroup.com/>

Table des figures

2.1	Un microprocesseur de carte à puce	10
2.2	La carte à contact	11
2.3	La carte sans contact	11
2.4	Architecture de MULTOS	12
2.5	Architecture de Windows for Smart Card	14
2.6	Architecture de la Java Card	16
2.7	La machine virtuelle Java Card (JCVM)	17
2.8	Les étapes du développement d'une applet Java Card	19
5.1	Affichage du composant Header	33
5.2	Affichage du composant Applet	33
5.3	Affichage du composant Import	33
5.4	Affichage du composant Constant Pool	34
5.5	Affichage du composant Class	34
5.6	A Propos de	34

Liste des tableaux

2.1	Caractéristiques Java supportées et non supportées	16
3.1	Étendue des standards et des spécifications des cartes à puce	26
5.1	Format de fichier d'un composant du packaging	31
5.2	Les boites de dialogue disponibles à partir du composant Applet	34

Glossaire

AID (Application IDentifier) : Nombre unique assigné à une application, à certain type de fichiers, à une instance d'une applet ou à un paquetage Java.

APDU (Application Protocol Data Unit) : Unité de données échangées entre la carte à puce et le terminal au niveau de la couche application. Un APDU contient soit une commande, soit une réponse.

API (Application programming Interface) : Définit les méthodes d'appels d'un programme applicatif au système d'exploitation et à d'autres services.

Applet : Application écrit pour une plateforme Java Card.

ATR (Answer To Reset) : Chaîne d'octets envoyée par la carte après un redémarrage.

CAD (Card Acceptance Device) : Périphérique qui est utilisé pour communiquer avec une carte à puce. Il peut aussi fournir l'alimentation et la fréquence à la carte à puce.

CAP (Converted APplet) : Le format de fichier standard pour la compatibilité binaire de la plateforme Java Card. Un fichier **CAP** contient une représentation binaire exécutable des classes dans un paquetage Java.

Carte à puce Java : carte à puce supportant la technologie Java Card.

COS (Card Operating System) : Le micro-code contenu dans la ROM de la carte à puce et qui est utilisé pour la communication de la carte à puce, pour la gestion de la sécurité et la gestion des données dans les fichiers de la carte à puce.

GSM (Global System for Mobile communications) : (originellement Groupe Systèmes Mobile) le standard adopté par 18 pays Européens afin de développer une compatibilité dans les télécommunications mobiles numériques.

ISO 7816 : Document qui définit le standard des cartes à puce à contact.

JCRE (Java Card Runtime Environment) : Environnement d'exécution pour l'exécution d'applet Java Card.

JCVM (Java Card Virtual Machine) : machine virtuelle Java Card. Elle se compose du convertisseur qui tourne hors carte et de l'interpréteur qui est embarqué sur la carte. Ces deux éléments implémentent toutes les fonctions d'une machine virtuelle.

PIN (Personal Identification Number) : Utilisé pour autoriser les transactions.

SIM (Subscriber Identity Module) : Identifie l'utilisateur du téléphone GSM et fournit les clés de cryptage pour la transmission numérique de la voix.

Index

A

AID 31, 45
APDU 28–30, 45
API 15–18, 45
applet 15–19, 31
ATR 45

C

CAD 45
 session 18
CAP 17, 19, 28, 31, 32, 35, 37, 45
carte
 à contact 10, 11
 à mémoire 9, 41
 à microprocesseur 9
 combi 11
 sans contact 10, 11
carte à puce 9, 41
compilateur 19
convertisseur 17, 19, 45
COS 9, 45
cycle de vie
 d’une applet 18
 de la carte 18

E

EEPROM 10, 15
émulateur 19
EMV 12, 22, 23, 25, 30
ETSI 21, 30
export 17, 19

F

FAT 13

G

GSM 21, 29, 45

I

interpréteur 15–17, 45
ISO 7816 ... 12, 14, 18, 21, 22, 25, 30, 45

J

Java Card 7–9, 14, 15, 27–31, 37
 architecture 15, 16
 convertisseur voir convertisseur
 émulateur 28, 37, voir émulateur
 interpréteur voir interpréteur
 simulateur ... 29, 30, voir simulateur
 spécification 15
JCRE 15–19, 28, 37, 45
JCVM 15–19, 27, 37, 45

M

MEL 12
MULTOS 12
 architecture 12

O

OCF 24, 25, 29
OpenCard Framework voir OCF

P

PC/SC 24, 25, 29

R

RAM 10, 15
ROM 9, 10, 15, 18

S

SIM 22, 30, 45
simulateur 19

V

Visa Open Plateform voir VOP
VOP 23, 25, 28–30

W

WinCard .. voir Windows for Smart Card
Windows for Smart Card 13
 architecture 14