Laboratoire Bordelais de Recherche en Informatique
UMR 5800 – Université Bordeaux 1,
351, cours de la Libération, 33405 Talence CEDEX, FRANCE

Rapport Interne LaBRI RR-1259-01

La technologie Java Card™ [1]

Présentation de la carte à puce

La Java Card

Damien SAUVERON
sauveron@labri.u-bordeaux.fr
http://dept-info.labri.u-bordeaux.fr/~sauveron/

Version : 1.0 Dernière révision : 6 Mai 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.
Nous présenterons les caractéristiques principales des cartes à puces et quelques applications qui en découlent. Ensuite nous verrons 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***. Nous détaillerons donc certaines parties de l'architecture de ***la Java Card*** afin de voir comment la gestion de la carte a été réalisée.
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***. We will show the main features of smart cards and some applications based on these features.
Then we will see how Java technology succeeded in becoming a major technology within smart card to build a multi-applicative card: ***the Java Card***. We will detail some parts of the architecture of ***the Java Card*** in order to see how the management of the card was achieved.
KEYWORDS: *smart card, Java Card, security*

[1]Java 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.

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'utilisations 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 en plus particulièrement grâce à la technologie Java Card.

Première partie

Présentation de la carte à puce

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. Contrairement aux cartes à piste magnétique, la carte à puce n'a pas besoin d'accéder à une base de données distante au moment d'une transaction grâce à sa puissance de calcul et sa capacité de stockage d'informations.

2 Historique

- En 1968, Jürgen Dethloff et Helmut Grötrupp deux inventeurs Allemands sont les premiers à introduire un circuit intégré dans une carte plastique.
- En 1970, indépendamment Kunitaka Arimura de l'institut de technologie Arimura au Japon dépose un brevet sur la carte à puce.
- Entre 1974 et 1978, Roland Moreno dépose 47 brevets dans 11 pays. Il est considéré comme l'inventeur de la carte à puce.
- En 1979, la première carte est créée, assemblée à Toulouse, par Motorola pour Bull CP8 avec 1 Ko de mémoire programmable et un cœur à base de microprocesseur 6805.
- En 1983 apparaissent les premières cartes téléphoniques à mémoire.
- En 1984, le G.I.E carte bancaire adopte la «carte bleue» proposée sur un prototype Bull CP8. Cela aboutit à notre carte bleue actuelle version B0'.
- Entre 1984 et 1987, les normes internationales de l'ISO sur la carte à puce à contact voient le jour sous la référence 7816.
- En 1997, les premières Java Cards apparaissent.

3 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.

Tout le long de cette classification, on gardera présent à l'esprit que la plupart des cartes ont besoin d'une alimentation externe. En effet la taille actuelle des batteries empêche les cartes de correspondre aux standards internationaux dont notamment ISO 7816-1.

3.1 La carte à mémoire

Ces cartes ont été les premiers modèles de cartes à puce et représentaient encore la majorité des cartes vendues dans le monde en 1999.

Comme son nom l'indique, cette carte possède une puce mémoire. 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.

Les avantages de cette carte sont sa technologie simple et son faible coût de revient (1$).

Ses principaux inconvénients sont sa dépendance vis-à-vis du lecteur de carte pour le calcul et la possibilité d'être «assez» facile à dupliquer (mais toujours beaucoup moins que les cartes utilisant la piste magnétique).

En décembre 2000, la société Drexler Technology Corporation a annoncé le développement de cartes à mémoire optique capable de stocker plus de 4 Mo. Pour le moment ces cartes ne correspondent pas au standard international que l'on étudiera plus loin mais constituent sûrement une voie de développement pour les cartes de demain.

3.2 La carte à microprocesseur

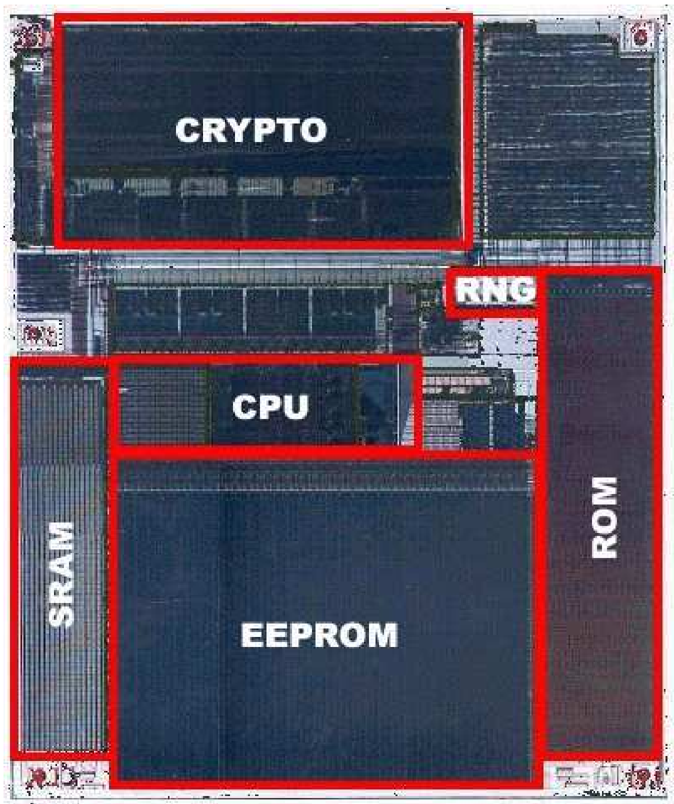

FIG. 1 – Un microprocesseur de carte à puce

Cette carte est composée d'un microprocesseur et de différents types de mémoires (*cf.* FIG. 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. Dans la plupart des cartes actuelles on trouve le Motorola 6805 ou l'Intel 8051 avec une horloge à 5 MHz. On pourrait augmenter les fréquences d'horloges mais le standard que l'on verra plus loin oblige à respecter une certaine plage de fréquence pour rester compatible avec les matériels déjà en place (*e.g.* lecteurs, etc). En revanche il arrive très souvent d'utiliser des multiplicateurs en interne. Il y a ainsi des cartes à microprocesseur dernier cri avec une horloge interne fonctionnant à 40 MHz en utilisant un multiplicateur `x8` permettant ainsi de faire les calculs dans la carte plus rapidement.

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

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. Un problème de cette mémoire est sa durée de vie limitée en nombre de cycles d'écriture (100000 cycles) et en temps (10 ans) ce qui nécessite de changer régulièrement la carte pour éviter des disfonctionnements. Un autre problème est sa lenteur d'accès puisqu'elle est 1000 fois plus lente en écriture que la RAM. 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 persistent, 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.

Il va être difficile d'augmenter de manière significative la taille de ces mémoires en conservant ces technologies. En effet la taille de la puce étant limitée à environ $25mm^2$ les procédés de gravages ont atteint leur limite et ne permettront plus d'accroître les capacités mémoire. De plus les capacités des différentes mémoires sont choisies de façon bien définie pour avoir un système viable car il faut savoir qu'une cellule d'EEPROM prend quatre fois plus de place qu'une cellule de ROM, et qu'une cellule de RAM prend quatre fois plus de place qu'une cellule de EEPROM.

La carte à microprocesseur contient souvent un coprocesseur cryptographique ce qui souvent augmente le prix de la carte. Ce coprocesseur cryptographique est associé à un générateur de nombres aléatoires RNG (Random Number Generator).

L'avantage de cette carte est le microprocesseur qui fournit la sécurité, via le coprocesseur cryptographique, et d'autres mécanismes en interdisant que les données ne soient accessibles de l'extérieur. Le coût est pour un produit aussi sûr tout à fait acceptable puisque situé entre 1$ et 20$. Pour résumer, dans cette carte c'est la triple alliance électronique, informatique et cryptographie qui assure un très haut degré de sécurité.

3.3 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), comme nous le verrons dans le standard ISO 7816.

Les problèmes posés par ces cartes sont multiples. En effet l'insertion et le retrait sont des facteurs d'usure de la carte et même de dégradation totale. Un autre problème est celui de l'orientation de la carte pour son insertion dans le lecteur.

3.4 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. 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). 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

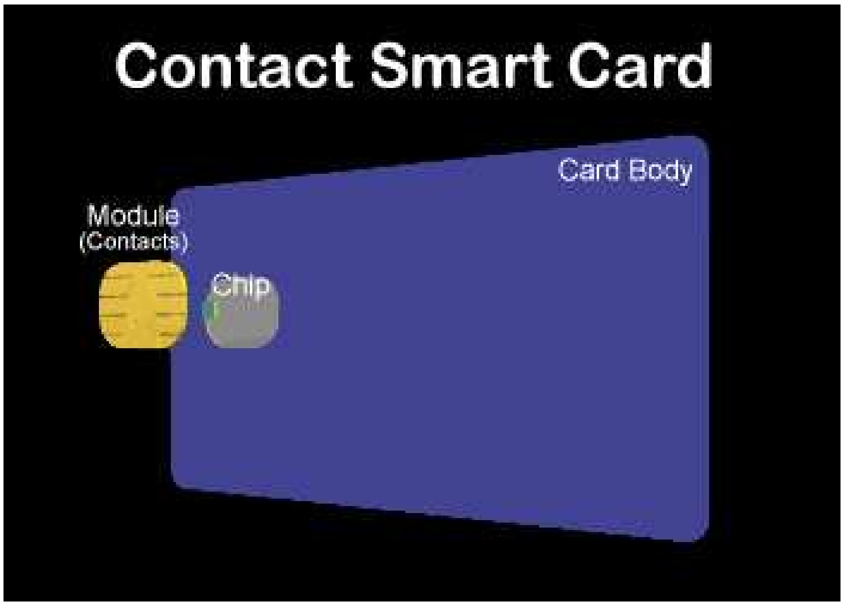

FIG. 2 – La carte à contact

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.

Ces cartes posent aussi quelques problèmes. Par exemple la distance de communication entre la borne de lecture et la carte est limitée (environ 10 cm). Un autre problème est que la carte est souvent en mouvement rapide dans l'espace d'échange et que le temps pour réaliser la transaction est de l'ordre de 200 ms ce qui limite la taille des données à échanger (souvent quelques centaines d'octets). Ces cartes sont aussi beaucoup plus chères que les cartes à contact.

Ces cartes sont surtout intéressantes pour les applications où l'insertion et le retrait de la carte dans un lecteur ne sont pas pratiques.

3.5 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.

4 La fabrication et le cycle de vie de la carte

4.1 Les acteurs

On peut distinguer dans le monde de la carte à puce, quatre principaux acteurs :

- le fabricant de puce (*e.g.* Hitachi, NEC, ST Microelectronics, etc) ;
- le fabricant de carte (*e.g.* Gemplus, Oberthur, Schlumberger, etc) ;
- l'émetteur de carte ou fournisseur de carte («card issuer») (*e.g.* banques, etc) qui fournit la carte à l'utilisateur final ;

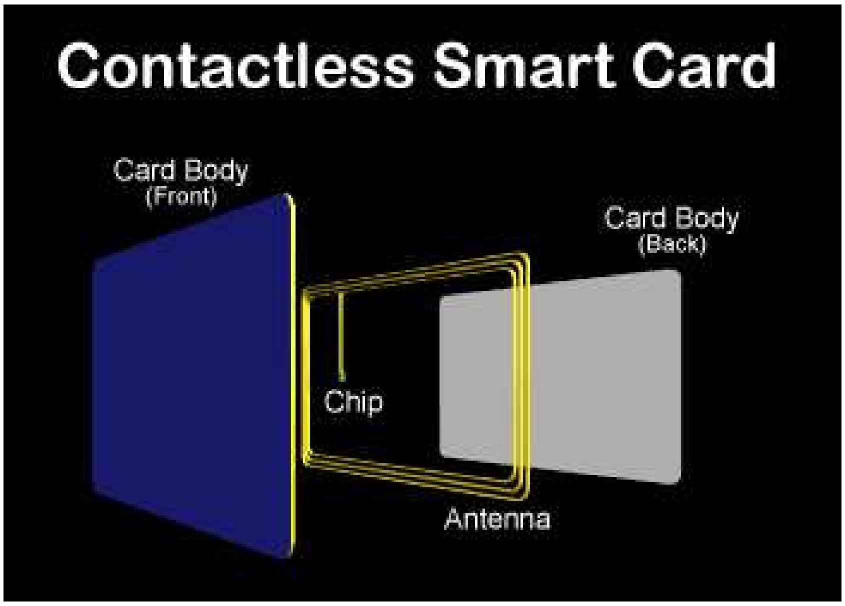

FIG. 3 – La carte sans contact

– le porteur de carte qui est l'utilisateur final.

4.2 La fabrication

Au départ, lors de la fabrication des puces, a lieu l'opération de «masquage». Cette opération consiste à figer le système d'exploitation et certaines données en ROM. Ensuite le fabricant de carte reçoit du fabricant de puce des galettes de silicone («silicon wafer») contenant les puces. Le fabricant de carte doit donc découper les puces (microchips), puis les encapsuler sur les microcontacts afin d'obtenir les micromodules (*cf.* FIG. 4).

Parallèlement il produit le corps de carte à partir de PVC (Poly vinyl chloride), de ABS (Acrylonitrile butadiene styrol) ou de PC (Polycarbonate). Le matériau choisi pour le corps de carte dépendra de la fonction future de la carte. En effet, ces matériaux n'ont pas les mêmes caractéristiques (*e.g.* de longévité, etc), donc pas les mêmes fonctions, ni les mêmes coûts. L'étape suivante consiste à insérer le micromodule sur le corps de carte. On a maintenant une carte vierge qui va pouvoir commencer son cycle de vie.

4.3 Le cycle de vie de la carte

Le cycle de vie de la carte commence avec l'initialisation de la carte qui consiste à inscrire en EEPROM des données communes à l'application et éventuellement à charger d'autres caractéristiques de sécurités. Cette étape, comme la suivante, est toujours réalisée par le fabricant de carte en fonction de la demande de son client émetteur de carte. L'étape suivante est la personnalisation et se fait à deux niveaux :

– au niveau électrique :
  • par l'inscription en EEPROM des données relatives à chaque porteur de carte ;
– au niveau graphique :

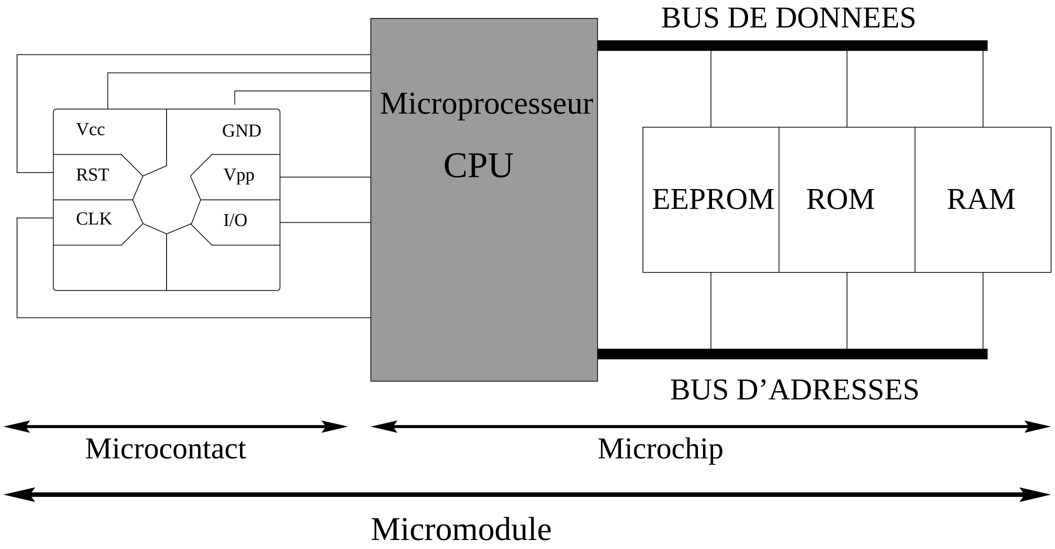

FIG. 4 – L'architecture de la carte à microprocesseur

• par la personnalisation du corps de carte suivant le désir de l'émetteur de carte (*e.g.* son logo),

• par l'embossage de données relatives à chaque porteur de carte.

À cette étape la carte peut maintenant passer du fabriquant de carte au porteur de carte par l'intermédiaire de l'émetteur de carte. Le cycle de vie de la carte se poursuit avec son utilisation par le porteur de carte et se termine soit par invalidation logique, soit par saturation de la mémoire, soit par bris, soit par perte, soit par vol, etc.

5 Le standard 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.

Nous allons détailler dans la suite une par une ces sous-normes mais pour plus de détails vous pouvez consulter [5].

5.1 ISO 7816-1

Cette norme définit comme nous l'avons déjà précisé les caractéristiques physiques des cartes (*cf.* FIG. 5) ayant une interface physique par des contacts électriques. Elle ne définit pas la nature, le nombre et la position des circuits intégrés dans la carte. Elle fait référence aux normes relatives aux cartes d'identification ISO 7810, ISO 7811 parties 1 à 5, ISO 7812 et ISO7813. Les caractéristiques introduites par la norme portent sur :

- la protection contre les ultraviolets et les rayons X ;
- le profil de surface des contacts ;
- la résistance mécanique des cartes et des contacts (au pliage, à la torsion) ;
- la résistance électrique des contacts ;
- l'interférence électromagnétique entre les pistes magnétiques éventuelles de la carte et de ses circuits intégrés ;
- l'exposition de la carte à des champs magnétiques ;
- l'exposition de la carte à une décharge d'électricité statique ;
- la dissipation thermique du circuit intégré.

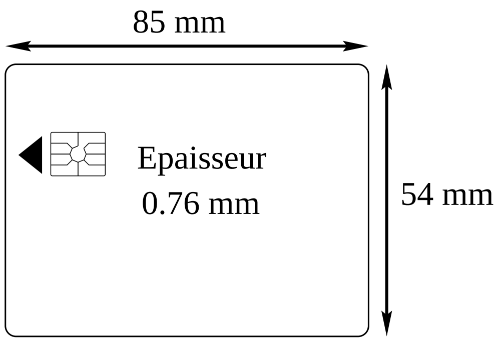

FIG. 5 – Les dimensions de la carte à puce

5.2 ISO 7816-2

La norme ISO 7816-2 définit les aspects électriques et la situation des contacts sur la carte. La carte comporte huit contacts (*cf.* FIG. 6) dont voici l'utilisation :

- Le contact 1 correspond à la tension d'alimentation en lecture (Vcc).
- Le contact 2 correspond au signal Reset (RST) : une tension appliquée sur ce contact déclenche l'initialisation physique et logique du composant.
- Le contact 3 correspond au signal d'horloge (CLK). Il fournit le signal d'horloge externe dont dérivera l'horloge interne.
- Le contact 4 est réservé à une utilisation future (RFU : Reserved for Futur Use).
- Le contact 5 correspond à la masse (GND).
- Le contact 6 correspond à la tension d'alimentation (Vpp) à appliquer, sur demande de la carte, pour programmer la mémoire de données (tension en écriture). Il est seulement utilisé sur des cartes anciennes.
- Le contact 7 est le contact d'entrée/sortie (I/O) par lequel transitent en half-duplex toutes les données échangées entre la carte et le monde extérieur.
- Le contact 8 est réservé à une utilisation future (RFU).

Une norme utilisant les contacts RFU serait en cours de préparation pour l'utilisation de l'USB avec la carte.

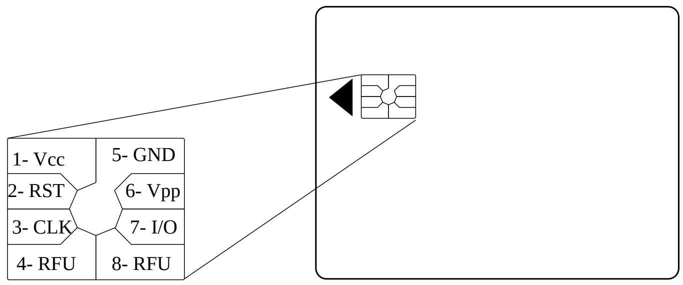

FIG. 6 – Les contacts de la carte à puce

5.3 ISO 7816-3

Cette partie normalise :
- les caractéristiques électriques comme la fréquence d'horloge (entre 1 MHz et 5 MHz), la vitesse des communications (jusqu'à 115200 bauds), etc ;
- les protocoles de transmission (TPDU : Transmission Protocol Data Unit) :
  - T=0, protocole orienté octet,
  - T=1, protocole orienté paquet,
  - T=14, réservé pour les protocoles propriétaires ;
- la sélection du type de protocole (PTS : Protocol Type Selection) si plusieurs protocoles sont supportés ;
- la réponse au reset (ATR : Answer To Reset) qui correspond aux données envoyées par la carte immédiatement après la mise sous tension.

5.4 ISO 7816-4

ISO 7816-4 vise à assurer une inter-opérabilité. Il spécifie :
- le contenu des messages entre la carte et le lecteur (CAD : Card Acceptance Device) qui utilise le protocole APDU (Application Protocol Data Units) pour les commandes et les réponses ;
- les structures des fichiers et des données :
  - l'accès à ces données,
  - l'architecture de sécurité,
  - la sécurisation des communications.

5.4.1 Le protocole APDU

Ce protocole est de niveau application comme son nom l'indique. Il existe deux types de messages APDUs :
- la commande APDU (C-APDU) qui est émise par le CAD en direction de la carte ;
- la réponse APDU (R-APDU) qui elle transite de la carte au CAD.

Le modèle de discussion entre le CAD et la carte est un modèle maître-esclave où le CAD est le maître et la carte est l'esclave. La carte est toujours en attente d'une commande APDU. Une réponse APDU aura toujours lieu en retour d'une commande APDU. Les deux types de structures de communication commande et réponse sont toujours couplés.

La commande APDU est constituée par (*cf* TAB. 1) :

- une entête obligatoire de quatre octets :
  - CLA, l'octet de classe identifie la catégorie de la commande et de la réponse APDUs,
  - INS, l'octet d'instruction spécifie l'instruction de la commande,
  - P1 et P2, deux octets utilisés pour paramétrer l'instruction ;
- un corps optionnel de taille variable :
  - Lc, l'octet qui spécifie la taille du champ de données,
  - le champ de données contient les données envoyées à la carte pour exécuter l'instruction spécifiée dans l'entête,
  - Le, l'octet qui spécifie le nombre d'octets attendu par le CAD pour le champ de données dans la réponse APDU de la carte.

Entête obligatoire				Corps optionnel		
CLA	INS	P1	P2	Lc	Champ de données	Le

TAB. 1 – Structure d'une commande APDU

La réponse APDU est constituée par (*cf* TAB. 2) :
- un corps optionnel de taille variable :
  - Le champ de données de taille Le déterminée dans la commande APDU correspondante ;
- une enqueue obligatoire de deux octets :
  - SW1 et SW2 (Status Word) sont deux champs de un octet précisant l'état de la carte après l'exécution de la commande APDU. Par exemple «0x9000» signifie que l'exécution s'est déroulée complètement et avec succès.

Corps optionnel	Enqueue obligatoire	
Champ de données	SW1	SW2

TAB. 2 – Structure d'une réponse APDU

Comme dans les APDUs certaines parties étant optionnelles il y a donc quatre cas possibles d'échanges APDU (*cf* TAB. 3) :
- cas 1 : Aucune donnée n'est échangée entre le CAD et la carte autre que l'entête de la commande APDU et l'enqueue de la réponse APDU.
- cas 2 : Aucune donnée (dans le champ de données de la commande APDU) n'est envoyée à la carte. Le corps de la commande contient le champ Le qui spécifie le nombre d'octets que la carte lui fournira dans le champ de données de sa réponse APDU.
- cas 3 : Ici des données, de taille Lc octets, sont fournies à la carte (dans le champ de données de la commande APDU) et la carte ne renvoie dans la réponse APDU que son enqueue.
- cas 4 : On a des données échangées dans le champ de données de la commande et de la réponse APDU.

	commande APDU				réponse APDU	
cas 1	entête				SW	
cas 2	entête	Le			données	SW
cas 3	entête	Lc	données		SW	
cas 4	entête	Lc	données	Le	données	SW

TAB. 3 – Les différents échanges de commande et de réponse APDU

5.4.2 Le système de fichiers des cartes à puce

La norme ISO 7816-4 supporte un système de fichiers hiérarchique qui peut contenir trois types de fichiers (*cf.* FIG. 8) :

- «Master File» (MF) : Ce fichier est la racine du système de fichiers. Il peut contenir des «Dedicated Files» et «Elementary Files». Il est unique sur une carte à puce.
- «Dedicated File» (DF) : Ce fichier est en fait un répertoire sur la carte à puce et il peut contenir d'autres «Dedicated Files» et des «Elementary Files». Le MF est un type spécial de DF.
- «Elementary File» (EF) : Ce fichier contient les données et ne peut pas contenir d'autres fichiers. Il en existe quatre types (*cf.* FIG. 7) :
  - «transparent file» est structuré comme des séquences d'octets de données alors que les trois autres types de fichiers sont structurés comme des séquences d'enregistrement individuellement identifiable,
  - «linear fixed» contient des enregistrements de taille fixée,
  - «linear variable» contient des enregistrements de taille variable,
  - «cyclic fixed» contient des enregistrements organisés comme un anneau.

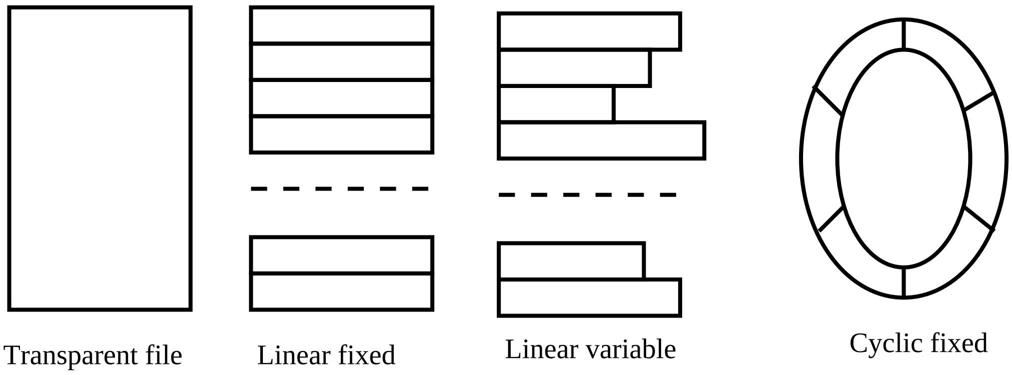

FIG. 7 – Les différents types de fichiers

Chaque fichier est spécifié par un identifiant de deux octets ou par un nom symbolique de taille maximum de seize octets. L'identifiant 3F 00 est réservé pour le MF. La hiérarchie n'a rien à voir avec la numérotation et il peut y avoir des collisions d'identifiants. C'est pourquoi il est recommandé d'utiliser pour :

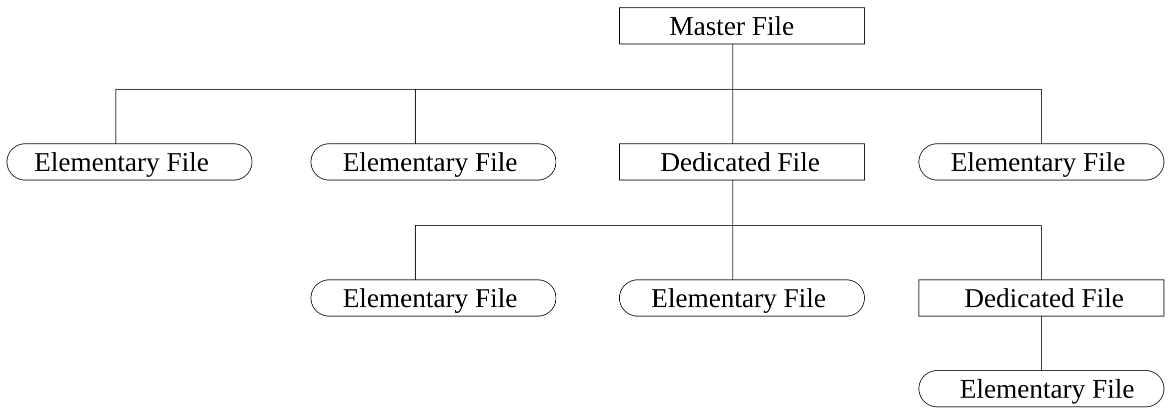

FIG. 8 – Exemple de structure du système de fichiers

- DF, l'identifiant x_1x_2 00 ;
- EF, l'identifiant x_1x_2 y_1y_2 :
  - x_1x_2 identifie le parent DF,
  - y_1y_2 identifie le fichier.

5.5 ISO 7816-5

Cette norme spécifie le système de numérotation et les procédures d'enregistrement des identifiants d'applications (AID pour Application IDentifier). Les AIDs sont utilisés pour l'identification unique d'une application de la carte et de certains types de fichiers. Un AID est un tableau d'octets de taille comprise entre cinq et seize. Il peut être découpé en deux parties distinctes (*cf* TAB. 4) :
- la première partie de taille fixe, qui fait toujours cinq octets, est le RID (Ressource IDentifier) ;
- la seconde partie de taille variant entre zéro et onze octets est désignée par PIX (Proprietary Identifier eXtension).

RID (5 octets)	PIX (0-11 octets)

TAB. 4 – Identifiant d'application : AID

ISO procède à l'assignation aux entreprises des RIDs afin que chacune ait son propre RID. Ensuite chaque entreprise contrôle la gestion des PIXs.

5.6 ISO 7816-6

ISO 7816-6 spécifie les éléments de données inter-industrie tels que le nom du porteur de carte, de la date d'expiration, etc. Les données sont stockées en respectant le patron suivant (*cf* TAB. 5) :
- une étiquette dont la taille varie de un à plusieurs octets ;
- la longueur du champ valeur ;
- le champ valeur.

Étiquette	Longueur	valeur

TAB. 5 – Patron de données inter-industrie

5.7 ISO 7816-7

Cette norme propose des commandes inter-industrie pour SCQL. Elle est destinée à la gestion de base de données à l'intérieur de la carte à puce. Son utilisation actuellement serait très marginale.

5.8 ISO 7816-8

La norme concerne la sécurité de l'architecture et des commandes inter-industrie. Les informations sur cette partie ne nous sont pas accessibles pour le moment.

6 Pourquoi utiliser la carte à puce ?

Les avantages de la carte à puce sont la sécurité, la portabilité, la facilité d'utilisation et la personnalisation.

La carte à puce est résistante aux attaques car elle n'a pas besoin d'être dépendante d'une ressource externe potentiellement vulnérable. La sécurité de la carte à puce se fait à plusieurs niveaux :

- au niveau physique :
  - par des techniques d'impression sur le corps de carte très sophistiquées.
- au niveau hardware :
  - par un numéro de série unique.
  - par l'utilisation de mémoire de type PROM. L'information peut être stockée de manière durable hors-usine et aussi être déplacée par moment à l'intérieur de cette mémoire afin d'éviter des attaques localisées (c'est le software qui gère cette possibilité).
  - par un blindage physique du composant par collage du microchip au micromodule avec des techniques spécialisées et par insertion de couches de métallisation.
  - par un contrôle des accès aux mémoires (EEPROM, ROM).
  - par des détecteurs de conditions anormales telles que la température, l'humidité, la tension d'alimentation, la fréquence d'horloge, etc.
  - par un brouillage des informations dans le composant : dissémination de la mémoire, brouillage du bus, sous-ensembles de leurres.
  - grâce à l'utilisation de témoins d'intégrité disséminés dans la mémoire.
  - grâce à un générateur hardware de nombres aléatoires.
  - grâce à des coprocesseurs cryptographiques qui permettent de calculer le DES en hardware. Les coprocesseurs mathématiques permettent d'implémenter des algorithmes tels que RSA, etc.
- au niveau software :
  - par des contrôles d'accès aux données grâce par exemple à un code secret ou par authentification cryptographique.
  - par un maintien de l'intégrité des données en utilisant la vérification des témoins, les calculs de signatures sur les données internes et l'atomicité des transactions.

    - grâce à des entrées/sorties sécurisées par chiffrement/signature et temps d'exécution des commandes constants.
    - par des commandes effectuant des opérations cryptographiques comme la signature des données et le chiffrement/déchiffrement de données.
- au niveau de l'environnement de production qui doit être physiquement sécurisé :
    - par un contrôle d'accès aux sites et un chiffrement des données clients.

En plus de toutes ces contraintes sécuritaires, la carte à microprocesseur comme certains autres produits commerciaux, subit des évaluations sécuritaires. Les évaluations sont réalisées en France par des Centres d'Évaluation de la Sécurité des Technologies de l'Information (CESTI) agréés par la Direction Centrale de la Sécurité des Systèmes de l'Information (DCSSI), service dépendant du Premier ministre. À l'issue de l'évaluation qui suit des procédures et des critères très stricts, le CESTI délivre une certification correspondant au niveau de sécurité rencontrée. Les critères de l'évaluation sont de plus en plus des critères communs choisis en collaboration avec les autres nations procédant à des évaluations. Ainsi le consommateur qu'il soit un gouvernement, un industriel ou un particulier disposera d'un produit qui correspondra au niveau de sécurité attendue puisque certifiée. Cette méthodologie ne peut que renforcer la confiance dans la sécurité de la carte à puce.

La taille de la carte à puce qui rend celle-ci portable et mobile, est un gros avantage. En effet, elle permet de stocker des informations qui seront disponibles quelque soit l'endroit où se trouve le possesseur de la carte.

La facilité d'utilisation rend la carte à puce incontournable. Pour faire une transaction il suffit d'insérer sa carte dans le lecteur puis de la retirer quand le travail est fini. De plus, les cartes à puce utilisent grâce à la naissance des Java Cards une technologie simple et facile d'accès. Ainsi tout un chacun peut développer sa propre application pour carte à puce très facilement dans un sous ensemble du langage Java. Mais la révolution fournie par les Java Cards, c'est la naissance de cartes multi-applicatives qui permettront d'avoir dans son portefeuille une seule carte contenant toutes les applications. Auparavant aucun produit n'était disponible pour permettre la cohabitation de plusieurs applications sur une même carte, ou seulement pour un nombre limité d'applications développées et installées, en usine, par le fabriquant et non pas par l'utilisateur comme c'est aujourd'hui possible. Les autres produits maintenant disponibles sont MULTOS, proposé par le consortium industriel MAOSCO, et Windows for Smart Card de Microsoft. MULTOS est né en 1996, en même temps que la technologie Java Card. La carte MULTOS dispose d'une machine virtuelle qui exécutent les applications programmées en MEL (MULTOS Enabling Language). Les applications pour la WinCard se développent, elles, en Visual Basic.

Le dernier avantage de la carte est sa possibilité de personnalisation aussi bien pour le porteur de carte que pour le fournisseur de carte (*e.g.* par embossage ou par personnalisation graphique). Ainsi la carte à puce devient un objet marketing.

7 Applications

Les principaux secteurs qui utilisent actuellement la carte à puce sont :
- l'industrie des télécommunications avec par exemple les cartes téléphoniques pré-payées (comme par exemple les cartes téléphoniques classiques qui sont des cartes à mémoires avec contact) ou bien avec les cartes SIMs insérées dans les téléphones GSM ;

- l'industrie bancaire et monétaire avec les cartes de crédits qui sont des cartes à microprocesseur avec contact (*e.g.* Europay, MasterCard, Visa) ;
- le secteur de la santé (*e.g.* la carte Vitale) ;
- l'industrie audiovisuelle avec la télévision à péage, etc.

Mais de nouvelles applications pour la carte à puce sont aujourd'hui à l'étude pour :

- Le porte-monnaie électronique.
- Les transports en commun. Par exemple, l'accès au métro de Séoul se fait via des cartes à puce sans contact [6]. La RATP travaille sur un projet similaire.
- Le contrôle d'accès physique de personnes à des locaux, etc.
- L'identification : à des sites sur l'Internet, etc.
- Les applications de fidélité (*e.g.* l'accumulation de «miles» de voyage en avion gratuits à chaque transaction, etc).
- Les jeux comme le «pocket-gaming» (qui consiste en des terminaux de poche – comme le deviennent les téléphones ou les consoles portables – pour jouer par exemple à des jeux de casino, etc).

Le domaine principal qui permettra à la carte à puce de s'imposer est l'Internet qui au travers des «e-services» exigent de plus en plus une identification et une sécurisation des transactions. Ainsi la carte peut être utilisée pour ces services en stockant par exemple les clefs servant à l'authentification et à la sécurisation. Quelques exemples de «e-services» sont :

- le «e-commerce» ;
- la banque distante ;
- le «e-courrier» ;
- le télétravail ;
- etc.

Deuxième partie

La Java Card

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 Historique

– En Novembre 1996, un groupe d'ingénieurs du centre de production de Schlumberger à Austin au Texas cherche à simplifier la programmation des cartes à puce tout en préservant la sécurité. Le langage de programmation Java fût la solution. Schlumberger devint alors la première entreprise licenciée en proposant un brouillon de quatre pages (la spécification Java Card 1.0).
– En Février 1997, Bull et Gemplus se joignent à Schlumberger pour cofonder le Java Card Forum. Le but de ce consortium industriel est d'identifier et de résoudre les problèmes de la technologie Java Card en proposant des spécifications à JavaSoft (la division de Sun à qui appartient Java Card). Il a aussi pour but de promouvoir des APIs Java Card pour permettre son adoption par l'industrie de la carte à puce. Aujourd'hui, le Java Card Forum regroupe les fabriquants de cartes, Sun et des utilisateurs. À partir de ce moment, avec un large soutien de l'industrie, Sun microsystems va acquérir Integrity Arts afin de développer la technologie Java Card comme plateforme de la technologie Java pour les cartes à puce et les périphériques comportant peu de mémoire. Cette décision va être un gros avantage pour Gemplus qui était spécialisé dans le développement de machine virtuelle et de systèmes d'exploitation pour les cartes à puce.
– En Novembre 1997, Sun présente les spécifications Java Card 2.0 qui consistent en un sous ensemble du langage et de la machine virtuelle Java. Elles définissent des concepts de programmation et des APIs très différentes de celles de la version 1.0. Il n'y a encore rien sur le format des applets téléchargeables.
– En Mars 1999 sort la version 2.1 des spécifications Java Card. Elles consistent en trois spécifications :
  • la Java Card 2.1 API Specification,
  • la Java Card 2.1 Runtime Environment Specification,
  • la Java Card 2.1 Virtual Machine Specification.

  Dans cette nouvelle version, il y a eu quelques modifications des APIs notamment au niveau de la cryptographie et des exceptions. L'environnement d'exécution des applets a été standardisé. La contribution la plus significative de la version 2.1 est la définition explicite de la machine virtuelle de la Java Card (JCVM : Java Card Virtual Machine) et du format de chargement des applets, autorisant ainsi une vraie inter-opérabilité.
– En Mai 2000, est sortie une petite correction qui aboutit à la version 2.1.1 de la spécification (*cf* [11], [12], [13]).
– En Octobre 2000, les entreprises ayant acquis la licence d'exploitation de la technologie Java Card sont : AOL, American Express, CP8, Centra Systems, Citicorp, Dallas Semi-

conductor, Datacard/Platform 7, ETRI, Fujitsu, Gemplus, Giesecke & Devrient, Goldkey, Hitachi, IBM, InCard, Inside Technologies, IPM Group, ITRI, IRIS Automation, Keycorp, Matsushita, Mitsubishi, Motorola, NCT/Advancel, NEC, NTT Corporation, Oberthur, Orga, Schlumberger, Sermepa, Setec, ST Microelectronics, Toppan Printing Co., Ltd., Toshiba, Visa International, Watch Data, Wave Systems et Xilinx.

3 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 grâce :
  • à la programmation orientée objet offerte par Java (contre l'assembleur avant),
  • à la possibilité d'utiliser les environnements de développement existants pour Java,
  • à une plateforme ouverte qui définit des APIs et un environnement d'exécution standardisé,
  • à une plateforme qui encapsule la complexité fondamentale et les détails des systèmes des cartes à puce.

  Les développeurs peuvent ainsi se concentrer sur les applets ou les bibliothèques qu'ils créent, réduisant le temps de développement.
– 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 qui, sinon, pourraient être utilisés dans des programmes malicieux afin d'espionner le contenu de la mémoire,
  • à un «firewall» qui sépare les applets dans la plateforme.

  Grâce à tous ces mécanismes, le système empêche un programme hostile de créer des dommages à une autre partie du système.
– L'indépendance par rapport au hardware réalisée grâce au langage Java, exécutable sur n'importe quel système car son code pré-compilé en bytecode est exécuté par la machine virtuelle Java. Cette portabilité permet d'écrire du code qui fonctionnera sur n'importe quel microprocesseur de carte à puce («Write Once, Run Anywhere»).
– La capacité de stockage et de gestion de multiples applications. En effet les Java Cards peuvent héberger de multiples applications de fournisseurs et de natures totalement différents. Un mécanisme de «firewall» interdit l'accès d'une applet à une autre si celui-ci n'est pas explicitement permis. Une fois la carte fournie à son propriétaire il est encore possible de télécharger des applets. Les applications de la Java Card peuvent ainsi être continuellement mises à jour sans avoir besoin de changer de cartes.
– La compatibilité avec les standards existants sur les cartes à puce. La technologie Java Card est basée sur le standard ISO 7816 autorisant le support d'applications compatibles ISO 7816. Les applets peuvent donc interagir non seulement avec toutes les cartes à puce Java mais aussi avec les CADs existants.

4 Présentation de la Java Card

4.1 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'éxecute sur la carte. La partie de la JCVM embarquée 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. 9). Les applets demandent des ressources et des services systèmes au JCRE à travers les APIs.

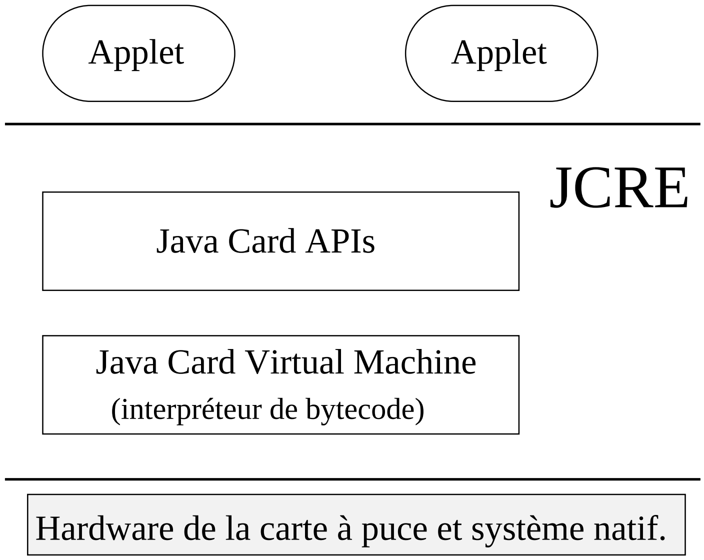

FIG. 9 – Architecture de la Java Card

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 :

- 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 applica-

tions 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.

4.2 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. 6 seules les caractéristiques supportées et non supportées importantes ont été inclues.

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. 6 – Caractéristiques Java supportées et non supportées

Afin de bien comprendre les choix fait pour la technologie Java Card, nous allons détailler certains points du TAB. 6.

4.2.1 Caractéristiques Java non supportées

– Chargement dynamique des classes.
Une implémentation de la plateforme Java Card ne doit pas charger les classes dynamiquement. Les classes sont soit inclues dans le masque à la fabrication soit installées au travers d'un processus d'installation sur la carte après qu'elle ait été fournie. Les programmes qui s'exécutent sur la carte doivent seulement se référer aux classes déjà existantes sur la carte. Il n'y a aucune façon de télécharger des classes durant un exécution normale du code d'une application.
– `Security Manager`.
Le gestionnaire de sécurité sur la plateforme Java Card diffère significativement de celui de la plateforme Java. Dans la plateforme Java, il y a une classe `Security Manager` (`java.lang.SecurityManager`) responsable des caractéristiques sécuritaires d'implémentation. Dans la plateforme Java Card les règles de sécurités du langage sont implé-

mentées par la machine virtuelle. Il n'y a pas de classe `Security Manager` qui fabrique des règles de décision pour autoriser des opérations.

– Ramasse-miettes et finalisation.
  La technologie Java Card ne requière pas de ramasse-miettes et n'autorise pas non plus la désallocation explicite des objets pour garder la contrainte de «pointer-safety». C'est pourquoi le développeur d'application ne peut pas s'assurer que les objets qu'il alloue seront toujours désalloués. L'espace de stockage d'objets inaccessibles ne pourra pas nécessairement être réclamé. Quelques Java Card avancées fournissent un mécanisme de ramasse-miettes qui permet la destruction d'objet.
  La finalisation n'est pas nécessaire. `finalize()` n'est pas nécessairement appelé par la JCVM, et le programmeur ne doit pas dépendre de ce comportement.
– Threads.
  La JCVM ne peut pas supporter de multiple thread de contrôle. Un programme Java Card ne peut pas utiliser la classe `Thread` ou n'importe quel mot clé relatif aux threads du langage Java.
– Clonage d'objet.
  La plateforme Java Card ne supporte pas le clonage d'objet. La classe `Object` de l'API Java Card n'implémente pas la méthode `clone`, et ne fournit pas l'interface `Cloneable`.
– Le contrôle d'accès des paquetages Java.
  Le langage Java Card supporte le contrôle d'accès aux paquetages définit dans le langage Java. Pourtant, il y a des cas non supportés tels que les suivants :
  • Si une classe implémente une méthode avec un accès de visibilité `package`, une sous-classe ne peut pas surcharger la méthode et changer l'accès de visibilité de la méthode en `protected` ou `public`.
  • Une interface définie avec un accès de visibilité `package` ne peut pas être héritée par une interface avec un accès de visibilité `public`.
  • Une méthode `public` ou `protected` d'une classe `public` ne peut pas contenir un paramètre formel de type référence à une classe de visibilité `package`.
  • Une classe de visibilité `package` qui peut être héritée par une classe `public` ne peut pas définir de méthodes `public` et `protected`.

4.2.2 Caractéristiques Java supportées

– Paquetages.
  La plateforme Java Card suit les règles standards des paquetages de la plateforme Java. Les classes des APIs Java Card sont écrites comme des fichiers sources Java qui inclus les désignations des paquetages. Les mécanismes de paquetages sont utilisés pour identifier et contrôler les accès aux classes, aux champs statiques et aux méthodes statiques. Excepter la remarque faite sur le contrôle d'accès, les paquetages sur la plateforme Java Card sont utilisés exactement comme sur la plateforme Java.
– Création dynamique d'objet.
  Les programmes de la plateforme Java Card supporte la création dynamique d'objet, les instances de classes et les tableaux. Cette opération est réalisée comme d'habitude par l'opérateur `new`. Les objets sont alloués en dehors du tas. Comme souligné dans la remarque plus haut, la JCVM n'a pas nécessairement besoin de ramasse-miettes. Tous les objets créés par la machine virtuelle peuvent continuer à exister et à consommer des

ressources même après qu'ils soient devenus inaccessibles.

– Méthodes virtuelles.
  Comme les objets Java Card sont des objets du langage de programmation Java, l'invocation de méthodes virtuelles sur des objets dans un programme écrit pour la plateforme Java Card est exactement la même que celle d'un programme écrit pour la plateforme Java. L'héritage est supporté, incluant l'utilisation du mot clé `super`.
– Interfaces.
  Les classes Java Card peuvent définir ou implémenter des interfaces comme dans le langage de programmation Java. L'invocation de méthodes sur les types d'interfaces fonctionne comme attendu. La vérification de type et l'opérateur `instanceof` fonctionne aussi correctement avec les interfaces.
– Exceptions.
  Les programmes Java Card peuvent définir, lancer et attraper des exceptions, comme dans les programmes Java. La classe `Throwable` et ses sous classes importantes sont supportées. Quelques sous classes de `Exception` et `Error` sont omises, celles dont les exceptions ne peuvent pas apparaître dans la plateforme Java Card.

4.3 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. 10) :
– 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.

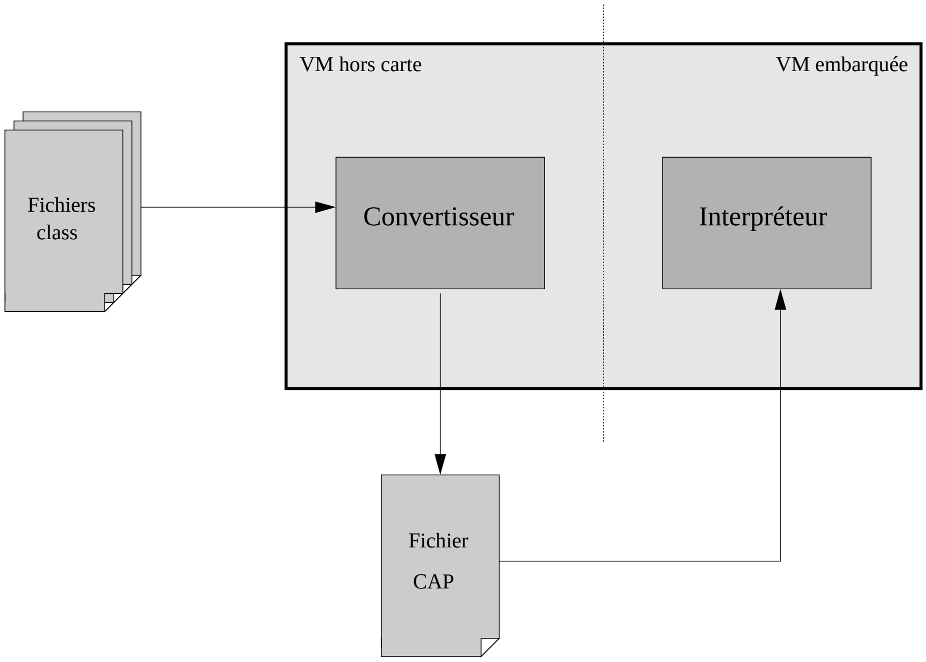

Fig. 10 – La machine virtuelle Java Card (JCVM)

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.

4.3.1 Fichier `CAP` et fichier `export`

La technologie Java Card introduit deux nouveaux formats de fichiers binaires qui permettent l'indépendance de la plateforme de développement, la distribution et l'exécution de logiciels Java Card :

- le fichier `CAP` ;
- le fichier `export`.

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. 7 – 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.

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.

Les fichiers `export` ne sont pas chargés dans la carte et ne sont pas directement utilisés par l'interpréteur. Ils sont produits et utilisés par le convertisseur pour des buts de vérifications et d'édition de liens. Ces fichiers peuvent être vus comme les fichiers entêtes dans la programmation en langage C. Un fichier `export` contient les informations publiques sur les APIs pour un paquetage de classes complet. Il définit l'étendue et le nom des classes et l'étendue et la signature des méthodes et des champs des classes. Un fichier `export` ne contient pas d'implémentation (*i.e.* de bytecode). Ce fichier peut être librement distribué par un développeur d'applet aux utilisateurs potentiels de l'applet sans révéler les détails de l'implémentation interne.

4.3.2 Convertisseur Java Card

Contrairement à la machine virtuelle Java, qui travaille sur une classe à la fois, l'unité de conversion est le paquetage. Les fichiers `class` sont produits par le compilateur Java depuis le code source. Ensuite le convertisseur préprocesse les fichiers `class` qui composent le paquetage Java et convertit le paquetage en un fichier `CAP`.

Durant la conversion, le convertisseur réalise des taches qu'une machine virtuelle Java classique (*i.e.* sur une station de travail) doit réaliser au chargement des classes :
- vérifier que le chargement des images des classes Java sont bien formées ;
- contrôler des violations du langage Java Card ;
- réaliser des initialisations de variables `static` ;
- résoudre les références symboliques aux classes, méthodes et champs dans la forme la plus compact qui peut être traitée efficacement sur la carte ;
- optimiser le bytecode en tirant avantage des informations obtenu au chargement des classes et à l'édition de liens ;
- allouer l'espace et créer les structures de données de la machine virtuelle pour représenter les classes.

Grâce aux taches réalisées par le convertisseur, la partie de la machine virtuelle embarquée sur la carte peut être plus petite et plus performante, car déchargée d'une partie de son travail.

Le convertisseur ne prend pas seulement des fichiers `class` à convertir mais aussi un ou plusieurs fichiers `export`. En plus de la production du fichier `CAP`, le convertisseur génère un fichier `export` pour le paquetage converti. La Fig. 11 montre comment un paquetage est converti. Le convertisseur charge toutes les classes d'un paquetage Java. Si le paquetage importe des classes provenant d'autres paquetages, le convertisseur charge aussi les fichiers `export` de ces paquetages. La sortie produite par le convertisseur est un fichier `CAP` et un fichier `export` pour le paquetage converti.

4.3.3 Interpréteur Java Card

L'interpréteur Java Card fournit le support d'exécution du modèle du langage Java qui autorise une indépendance au hardware du code de l'applet. L'interpréteur réalise les taches suivantes :
- exécute les instructions du bytecode et ultimement les exécutions des applets ;

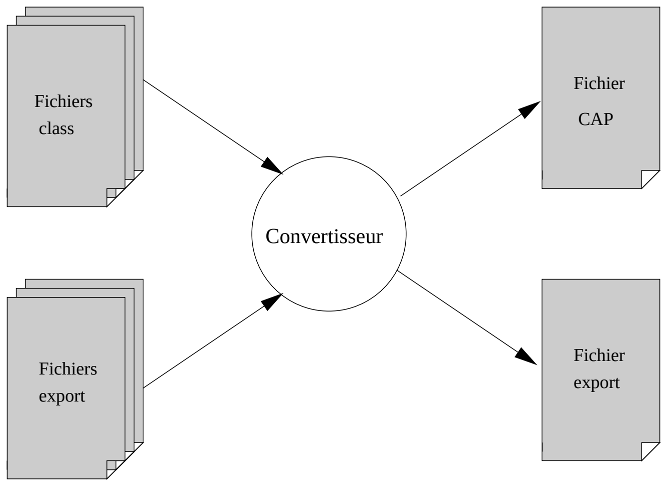

Fig. 11 – Conversion d'un paquetage

– contrôle les allocations de mémoire et les créations d'objets ;
– joue un rôle crucial pour s'assurer de la sécurité lors de l'exécution.

Il nous arrivera d'assimiler dans la suite la JCVM à l'interpréteur.

4.3.4 Installateur Java Card et programme d'installation hors carte

L'interpréteur Java Card ne charge pas le fichier `CAP` lui-même. Il exécute seulement le code du fichier `CAP`. Dans la technologie Java Card, le mécanisme de téléchargement et d'installation du fichier `CAP` est contenu dans une unité appelée l'installateur. L'installateur Java Card réside dans la carte. Il coopère avec un programme d'installation hors de la carte. Ce programme transmet l'exécutable binaire dans un fichier `CAP` à l'installateur qui s'exécute sur la carte via le terminal (CAD). L'installateur écrit le fichier binaire dans la mémoire de la carte à puce et le lie avec les autres classes déjà sur la carte, et créé et initialise les structures de données qui sont utilisées par le Java Card Runtime Environment (JCRE). Ce processus est illustré Fig. 12.

La séparation entre l'interpréteur et l'installateur de fichier `CAP` permet à l'interpréteur de rester petit et fournit une flexibilité des implémentations de l'installateur. Ce dernier peut par exemple être écrit en Java.

4.4 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.

Comme on peut le voir Fig. 13, le JCRE se situe au dessus du hardware de la carte à puce et du système natif. Il se compose :

– des méthodes natives ;
– de la JCVM (l'interpréteur de bytecode) ;

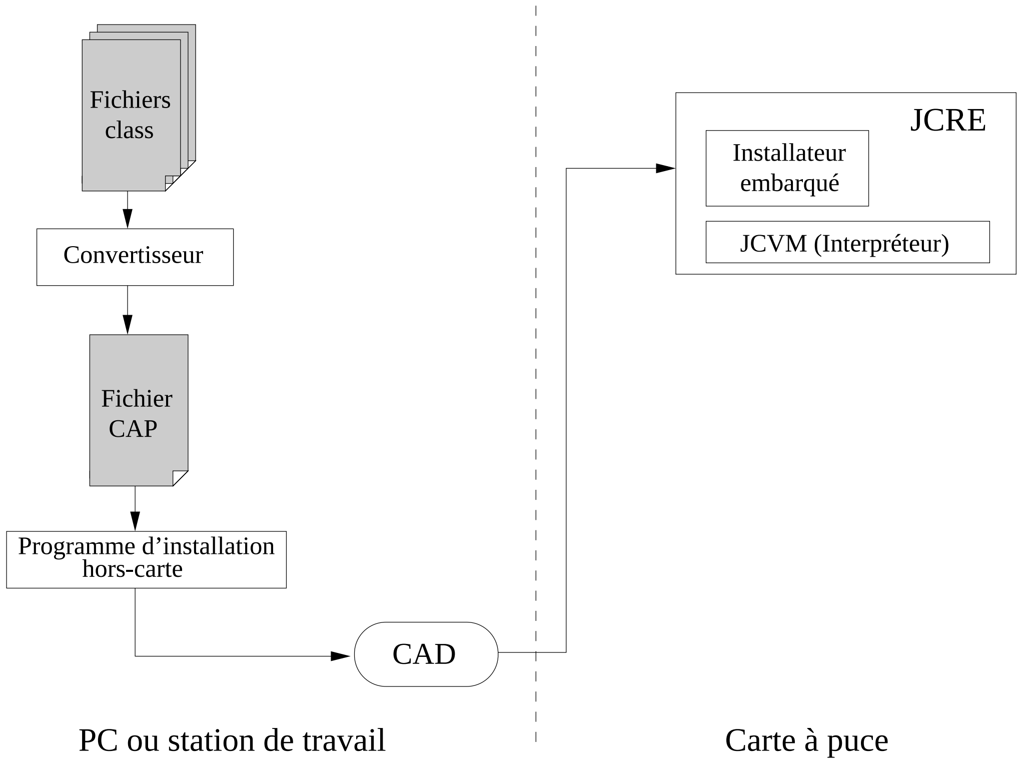

FIG. 12 – Installateur Java Card et programme d'installation hors carte

- des classes systèmes du JCRE ;
- des APIs Java Card ;
- des extensions spécifiques à l'industrie.

Le JCRE permet l'indépendance des applets par rapport aux technologies propriétaires des vendeur de cartes à puce en leur fournissant un système et des APIs standards. Il en résulte que les applets sont plus faciles à écrire et sont portables sur diverses architectures de cartes à puce.

La couche basse du JCRE contient les méthodes natives et la machine virtuelle Java Card. Les méthodes natives fournissent un support à la JCVM et à la couche supérieure que sont les classes systèmes du JCRE. Ces méthodes natives sont responsables du traitement des protocoles de communications bas niveau, de la gestion de la mémoire, du support de la cryptographie, etc.

Les classes systèmes permettent au JCRE d'être actif. Elles sont analogues au noyau d'un système d'exploitation. Elles sont en charge :

- de la gestion des transactions ;
- de la gestion des communications entre les applications hôtes (applications qui tourne sur la machine reliée au terminal) et les applets Java Card ;
- du contrôle de la création, sélection, déselection des applets.

Pour réaliser ces taches, les classes systèmes invoquent en général les méthodes natives.

Les classes des APIs sont compactes et optimisées pour le développement d'applets pour les cartes à puce. Les développeurs peuvent alors concentrer leurs efforts sur les détails de leurs applets plutôt que sur les détails de l'infrastructure du système des cartes à puce. Les applets accèdent aux services du JCRE à travers les APIs.

Les extensions spécifiques à l'industrie sont des bibliothèques qui peuvent fournir des services supplémentaires ou redéfinir la sécurité et le modèle du système. Par exemple Open Platform étend les services du JCRE pour les besoins spécifiques de l'industrie de la finance.

L'installateur permet le téléchargement sécurisé des applets sur la carte après que la carte eu été fournie au client. L'installateur est un composant optionnel du JCRE. S'il est absent, toutes les applets doivent être écrites dans la mémoire de la carte à puce lors de sa fabrication.

Les applets Java Card sont les applications des utilisateurs sur la plateforme Java Card. Elles sont bien entendu écrites dans le sous ensemble du langage Java et sont contrôlées et gérées par le JCRE. Elles sont téléchargeables via le processus d'installation. Les applets peuvent être ajoutées aux cartes à puce Java après qu'elles aient été fabriquées.

4.4.1 Le cycle de vie du JCRE

Dans un PC ou une station de travail, la JVM se comporte comme un système d'exploitation. Les données et les objets sont créés dans la RAM. Quand le système d'exploitation se termine, les applications Java et leurs objets sont automatiquement détruits.

Dans la carte à puce Java, la JCVM tourne à l'intérieur du JCRE. Le JCRE est initialisé à la phase d'initialisation de la carte. L'initialisation du JCRE est réalisée seulement une fois dans le cycle de vie de la carte. Durant cette phase, le JCRE initialise la machine virtuelle et créé les objets nécessaires pour fournir les services du JCRE et la gestion des applets. Comme des applets sont installées, le JCRE créé les instances de ces applets, et les applets créées les objets pour stocker les données.

La plupart des informations sur la carte doivent être préservées même quand la carte

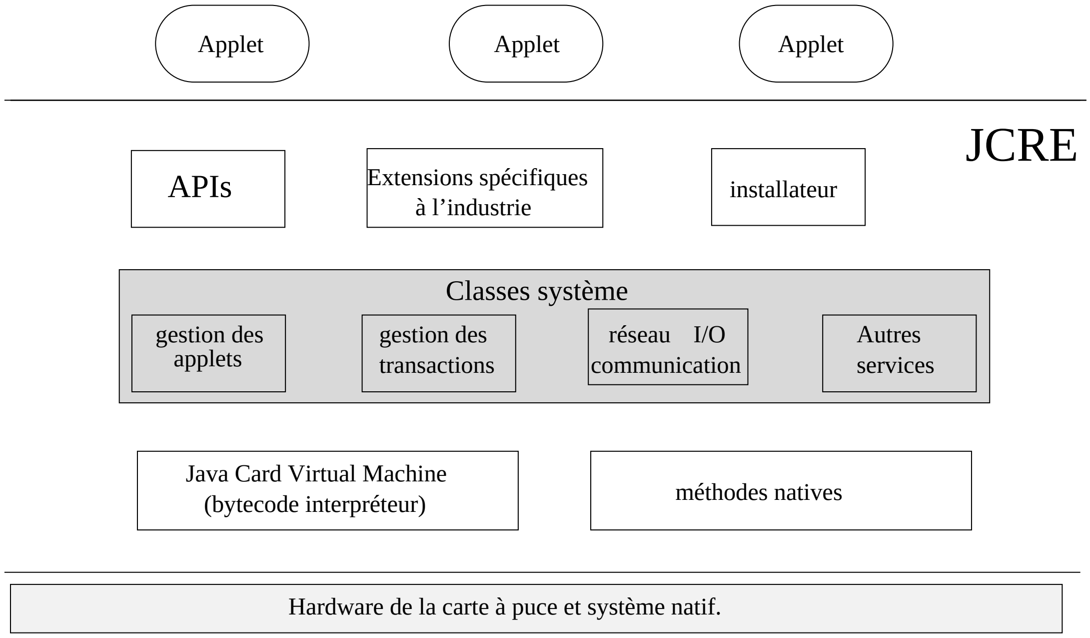

Fig. 13 – L'architecture du système dans la carte

n'est plus sous tension. Cette préservation est réalisée grâce à la mémoire persistante (*e.g.* l'EEPROM). Les objets et les données sont créés dans la mémoire persistante. Le cycle de vie du JCRE est le même que le cycle de vie de la carte. Quand la tension est enlevée, la machine virtuelle est seulement suspendue. L'état du JCRE et des objets créés sur la carte est préservé.

À la prochaine remise sous tension, le JCRE relance l'exécution de la machine virtuelle en chargeant les données depuis la mémoire persistante. Une notion subtile ici, c'est que le JCRE ne relance pas la JCVM exactement au point où la tension a été perdue. La machine virtuelle est redémarrée et s'exécute depuis le début de la boucle principale. Le redémarrage du JCRE diffère de l'initialisation car il préserve les applets et les objets créés sur la carte. Durant le redémarrage, si une transaction n'a pas été terminée, le JCRE réalise tous les nettoyages nécessaires pour remettre le JCRE dans un état cohérent.

4.4.2 Comment le JCRE opère durant les sessions CAD ?

La période entre le moment où la carte est insérée dans le CAD et est mise sous tension jusqu'au moment où la carte est retirée du CAD est appelée la session CAD. Durant une session CAD, le JCRE opère comme une carte à puce typique – il supporte les entrées/sorties d'APDUs avec l'application hôte (*cf.* Fig. 14). Les APDUs sont des données échangées entre l'applet et l'application hôte. Chaque APDU contient soit une commande de l'application hôte vers l'applet, soit une réponse de l'applet vers l'application hôte.

Après un redémarrage du JCRE, le JCRE entre dans une boucle d'attente des commandes APDU de l'hôte. Quand une commande arrive, le JCRE soit sélectionne une applet si l'instruction de sélection est dans la commande soit transmet la commande à l'applet courante. L'applet prend alors le contrôle et exécute la commande APDU. Quand elle a fini, l'applet envoie la réponse à l'application hôte et rend le contrôle au JCRE. Ce processus se répète quand les prochaines commandes arrivent.

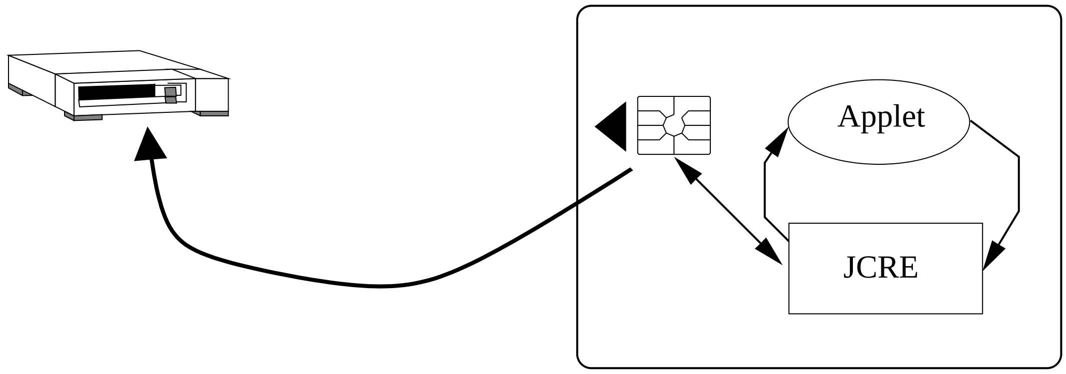

FIG. 14 – Communication APDU

4.4.3 Les caractéristiques du JCRE

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 persitents et sont créés dans la mémoire persistante. L'espace et les données de tels objets existent à travers les sessions CAD. 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 byte-code. 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.

4.5 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.

4.5.1 Le paquetage `java.lang`

Le paquetage Java Card `java.lang` est un sous ensemble strict de son équivalent le paquetage `java.lang` sur la plateforme Java. Ce paquetage supporte les classes `Object`, `Throwable` et `Exception`. C'est lui qui fournit le support fondamental du langage Java. La classe `Object` est définie comme la classe racine de la hiérarchie des classes sur la plateforme Java Card.

4.5.2 Le paquetage `javacard.framework`

Ce paquetage fournit la structure des classes et des interfaces pour le noyau fonctionnel des applets Java Card. La classe la plus importante qu'il définit est la classe de base `Applet` qui fournit la structure de l'exécution de l'applet en interaction avec le JCRE pendant la durée de vie de l'applet. Ce paquetage fournit aussi la classe `APDU` nécessaire à la communication avec l'extérieur. Comme la classe `java.lang.System` de la plateforme Java n'est pas supportée, la plateforme Java Card propose le paquetage `javacard.framework.JCSystem` qui fournit une interface au comportement du système. Ce paquetage fournit d'autres classes dont entre autre la classe `PIN`.

4.5.3 Le paquetage `javacard.security`

Le paquetage `javacard.security` fournit une architecture aux fonctions cryptographiques supportées sur la plateforme Java Card. Sa structure est basée sur le paquetage `java.security` de la plateforme Java.

4.5.4 Le paquetage `javacardx.crypto`

Le paquetage `javacardx.crypto` est un paquetage d'extension. Il contient les classes cryptographiques et les interfaces qui sont sujettes à une demande de régularisation pour l'exportation aux États-Unis.

4.6 Les applets Java Card

Il faudra tout d'abord ne pas confondre les applets Java destinées à être exécutées dans un navigateur web et les applets Java Card qui elles sont destinées à être exécutées au sein du JCRE. Les applets Java Card doivent donc suivre un ensemble des règles nécessaires à leur exécution sur la plateforme 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.

4.7 Convention de nommage des paquetages et des applets Java Card

Alors que sur la plateforme Java un paquetage ou un programme est identifié de manière unique par une chaîne Unicode et un schéma de nommage basé sur les noms de domaine internet, sur la plateforme Java Card un paquetage ou un programme (*i.e.* une applet) est identifié par un AID.

La construction de l'AID d'une applet ou d'un paquetage est la concaténation du RID et du PIX (*cf* Partie I 5.5). Chaque applet et chaque paquetage ont un AID unique. Le RID identifiant le fournisseur de l'applet, les applets définies dans un paquetage partage le même RID que celui-ci. L'AID d'un paquetage et celui par défaut de chaque applet de ce paquetage sont spécifiés dans le fichier `CAP`. On parle d'AID par défaut pour une applet car il est possible de modifier à l'installation l'AID qui était le sien dans le fichier `CAP`. Les AIDs sont fournis au convertisseur lorsque le fichier `CAP` est généré.

4.8 Les étapes du développement d'une applet Java Card

Le cycle de développement d'une applet peut se décomposer en quatre étapes (*cf.* FIG. 15) :

- 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 `CAP`s (*cf* Partie II 4.3.2).
- 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.

Lorsque l'applet est finalisée, l'étape suivante sera son installation sur la carte (*cf* Partie II 4.3.4).

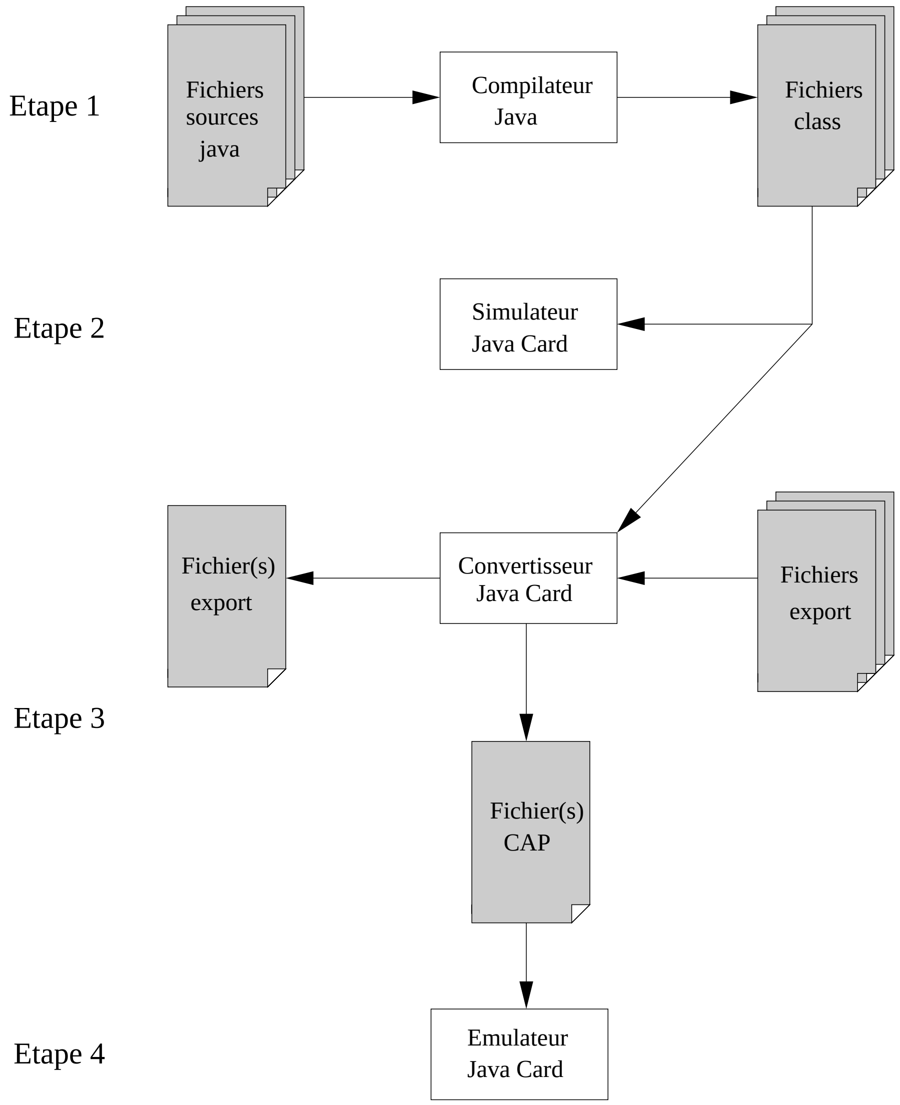

FIG. 15 – Les étapes du développement d'une applet Java Card

Conclusion

Depuis sa création dans les années 1970, jusqu'à aujourd'hui la carte à puce a subit 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 tout à fait sûre.

Références

[1] Zhiqun CHEN.
Java Card^{TM} Technology for Smart Cards.
Addison-Wesley - ISBN 0-201-70329-7.

[2] Ludovic ROUSSEAU.
La sécurité des cartes à puce.
Cours de cryptographie et sécurité. Janvier 2001, Bordeaux I.

[3] Nathalie FEYT.
La carte à puce.
Séminaire ENSEIRB. Janvier 2001, Bordeaux I.

[4] Java Card Technology Home Page.
`http://java.sun.com/products/javacard/`

[5] International Standards Organisation.
`http://www.iso.ch/`

[6] Supplément rattaché à Eurêka.
Numéro 51, Janvier 2000.

[7] Roger GANNE et Brigitte SALOMONI.
La carte à mémoire.
Eyrolles.

[8] Jorge FERRARI, Robert MACKINNON, Susan POH et Lakshman YATAWARA.
Smart Cards : A Case Study.
IBM. International Technical Support Organization.
`http://www.redbooks.ibm.com/`

[9] 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/`

[10] 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`

[11] Sun microsystems.
Java Card^{TM} 2.1.1 Application Programming Interface.
`http://java.sun.com/products/javacard/javacard21.html`

[12] Sun microsystems.
Java Card^{TM} 2.1.1 Runtime Environment (JCRE) Specification.
`http://java.sun.com/products/javacard/javacard21.html`

[13] Sun microsystems.
Java Card^{TM} 2.1.1 Virtual Machine Specification.
`http://java.sun.com/products/javacard/javacard21.html`

[14] Direction Centrale de la Sécurité des Systèmes d'Information.
`http://scssi.gouv.fr`

[15] JavaWorld.
Understanding Java Card 2.0.
`http://www.javaworld.com/javaworld/jw-03-1998/jw-03-javadev.html`

[16] JavaWorld.
Smart Card: A primer.
`http://www.javaworld.com/jw-12-1997/jw-12-javadev.html`

[17] JavaWorld.
How to write a Java Card applet: developer's guide.
`http://www.javaworld.com/jw-07-1999/jw-07-javacard.html`

Table des figures

Liste des tableaux

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

Table des matières