LES JAVA CARDS

Licence d'Informatique 2003-2004

Baptiste BESSON Magid ABERKANE

14 juin 2004

Table des matières

Introduction

L'explosion de l'Internet et des communications sans fil a rapidement changé notre façon de communiquer avec le monde. Les connections devenant de plus en plus nombreuses, les transactions physiques de la main à la main ont laissé place à des transactions en ligne, pouvant être effectuées de chez soi et en quelques clics de souris. L'émergence du business électronique n'ouvre pas simplement de nouvelles voies aux commerces mais permet aussi aux industries un accès plus facile aux clients et l'introduction de services supplémentaires.

Le succès du marché du business électronique repose sur la même confiance que celle instaurée par les compagnies depuis des années lors de transactions physiques. Les smart cards, de part leur sécurité et leur portabilité sont des moyens efficaces, pratiques et fiables de l'e-business et permettent une large étendue de nouvelles applications.

Jusqu'ici, les smart cards étaient programmées en assembleur. Mais les progrès réalisés permettent d'inclure, dans les smart cards, une machine virtuelle capable d'interpréter des instructions d'un sous-ensemble de Java appelé Java Cards, dont le niveau de sécurité et la portabilité permettent des applications de plus en plus nombreuses et variées.

Chapitre 1

Smart Cards

1.1 Qu'est-ce qu'une smart card ?

De la même taille qu'une carte de crédit, une smart card stocke, transmet et traite de l'information grâce aux circuits électroniques en silicium gravés sur sa surface. Contrairement aux simples cartes magnétiques, les smart cards peuvent à la fois posséder de l'information et la traiter. L'information étant directement présente sur la carte, il n'est ainsi pas nécessaire d'accéder à des bases de données externes lorsque l'on effectue une transaction.

1.2 Types de base

Il existe différents types de smart cards. On distingue les cartes à mémoire des cartes à microprocesseur.

Les premières ne sont pas vraiment "smart" puisqu'elles ne possèdent pas de CPU intégré et ne peuvent par conséquent pas traiter d'information. Elles servent simplement d'unité de stockage comme les cartes téléphoniques. Les données déjà présentes sont traitées par des fonctions simples préprogrammées sur la puce. Le circuit ayant un nombre limité de fonctions qui ne peuvent être reprogrammées, les cartes à mémoire ne peuvent pas être réutilisées et sont donc jetées à la fin de leur utilisation.

A l'inverse les cartes à microprocesseur, comme leur nom l'indique, contiennent un processeur. Elles permettent des fonctionnalités bien plus grandes et un niveau de sécurité plus élevé. L'information n'est ainsi pas directement disponible aux applications externes. Le processeur n'autorise l'accès aux données que sous certaines conditions (mot de passe, cryptage,...). Elles sont utilisées dans tous les domaines qui nécessitent un haut niveau de sécurité (banques, téléphones portables ...). Les applications sont multiples et très flexibles et ne sont limitées que par l'espace mémoire du circuit.

Dans les cartes à microprocesseur on peut également faire la distinction entre les cartes à contact et les cartes sans contact. Les cartes à contact doivent être insérées dans un appareil et

dans un sens pariculier (ex : cartes de crédit). Les cartes sans contact, elles, n'ont pas besoin d'être placées dans une machine. Elles communiquent avec l'extérieur grâce à une antenne incorporée dans la carte. L'énergie peut être fournie par une batterie interne ou captée par l'antenne. Elles transmettent alors leurs données à un appareil par ondes électromagnétiques. Ces cartes sont donc très populaires dans des situations nécessitant des transactions rapides (ex : identification à l'entrée d'un site). Toutefois la brièveté de la transaction ne permet de transmettre qu'une quantité limitée de donnée.

Dans la suite nous parlerons de smart cards pour les cartes à microprocesseur avec contact qui sont les plus courantes.

1.3 Architecture

Les Smart Cards possèdent des points de contact, une unité centrale, et divers types de mémoires.

1.3.1 Points de contact

Une Smart Card possède en général huit points de contact :

- Le point Vcc d'alimentation. La tension varie en général de 3 à 5 volts.
- Le point RST utilisé pour réinitialisé le microprocesseur. Cela permet une "réinitialisation chaude"(warm reset) et évite d'avoir à sortir et réintroduire la carte dans l'appareil en cas d'erreur (réinitialisation froide).
- Le processeur ne possédant pas d'horloge interne, celle-ci est située sur le point CLK.
- Le point GND est la masse. Sa tension est zéro.
- Le point Vpp est optionnel et n'est plus utilisé sur les cartes actuelles. Il permettait de contrôler la tension d'entrée.
- Le point I/O est le point d'entrée-sortie permettant de transmettre les données à l'extérieur en mode semi-duplex : un seul sens de communication n'est autorisé à un moment donné.
- Enfin les deux points RFU ne sont là qu'en réserve pour une utilisation future.

1.3.2 CPU

Le microprocesseur présent sur les Smart Cards est le plus souvent un microcontrôleur huits bits utilisant le jeu d'instructions Motorola 6805, ou Intel 8051 avec une vitesse d'horloge à 5MHz. Certaines cartes permettent de multiplier la vitesse d'horloge par 2, 4 ou 8 atteignant une vitesse d'horloge de 40Mhz. Des microcontrôleurs à 16 et 32 bits commencent à apparaitre sur les nouvelles smart cards.

1.3.3 Mémoires des Smart Cards

Les Smart Cards contiennent habituellement trois sortes de mémoires : ROM, EEPROM et RAM.

- La *ROM* (read-only memory) est utilisé pour stocker les programmes fixes de la carte, les routines du microprocesseur et des données permanentes. On ne peut peut ainsi ni les modifier, ni les supprimer. Aucune alimentation n'est nécessaire pour conserver les données.
- L' *EEPROM* (electrical erasable prorammable read-only memory), comme la ROM peut conserver les données quand la mémoire n'est plus alimentée. La différence est que l'on peut modifier les données stockées dans la mémoire. En fait c'est l'équivalent pour les smart cards du disque dur d'un PC.
- La *RAM* (random access memory) est utilisée comme espace temporaire pour modifier et stocker les données. Son contenu n'est pas préserve à l'arrêt de l'alimentation.

D'autres technologies de mémoire sont également en train d'apparaitre sur les Smart Cards comme la mémoire Flash. Celle-ci est en effet plus efficace en espace et en temps que la mémoire EEPROM.

Chapitre 2

La technologie Java Card

Les Smart Cards représentent l'une des plus petite plate-formes informatiques actuelles. La mémoire d'une Smart Card est de l'ordre de 1K de RAM, 16K d'EEPROM, et 24K de ROM. Le plus gros défi de la technologie Java Card est donc d'intégrer les composants java sur une Smart Card tout en gardant assez d'espace pour les applications. On y parvient d'une part en ne gardant qu'un sous-ensemble des caractéritiques du langage Java, et d'autre part en séparant la machine virtuelle java (JVM) en deux parties.

2.1 Le langage Java Card

Dû à la faible capacité de leur mémoire, le langage utilisé sur les plate-formes Java Cards ne peut être qu'une partie du langage Java. Il est adapté à l'écriture de programmes sur des Smart Cards et autres petits composants, mais préserve toutefois les caractéristiques principales de Java et notamment la programmation orienté objet. Les éléments supportés et non supportés sont présentés dans le tableau 2.1.

2.2 La Machine Virtuelle Java Card

La première différence entre la machine virtuelle javacard (JCVM) et la machine virtuelle java (JVM) est que la JCVM est implémentée en deux parties séparées : l'une sur la carte, l'autre sur l'ordinateur destiné à la lire. La partie sur la carte contient l'interpréteur de bytecode alors que le convertisseur est situé sur l'ordinateur. Pris ensemble ils implémentent toutes les fonctions de la machine virtuelle java nécessaire au sous-langage java des java cards.

2.2.1 Fichier CAP et Fichier d'Exportation

La technologie Java Card introduit deux nouveaux formats binaires de fichiers qui permettent l'exécution d'applications Java Card indépendamment de la plate-forme :

TAB. 2.1 – Eléments supportés et non supportés du langage Java

Eléments supportés un	Eléments non supportés
– types primitifs : boolean, byte, short – tableau à une dimension – packages Java, classes, interfaces exceptions – programmation orientéé objet : héritage, méthodes virtuelles, surchargement, création d'objets dynamiques, règles de liaisons – le mot clé int et le type integer sur 32 bit sont optionnels	– types primitifs : long, double, float – caractères et chaînes – tableaux multidimensionnels – chargement de classes dynamiques – Garbage collection et finalisation – Threads – sérialisation d'objets – clonage d'objets

Le fichier CAP qui contient une représentation binaire exécutable de toutes les classes d'un package Java. Le format CAP est le format dans lequel les logiciels sont chargés sur la Java Card. Les fichiers CAP permettent par exemple le chargement dynamique des classes d'applet après la conception de la carte (d'où leur nom "converted applet").

Les fichiers d'exportation ne sont pas chargés sur les smart cards et ne sont donc pas directement utilisés par l'interpréteur. Ils sont produits et consommés par le convertisseur à des fins de vérification. On peut considérer les fichiers d'exportation comme les fichiers .h du langage C. Il ne contiennent en effet que des informations sur les APIs publiques des classes et des signatures de méthodes et de champs. Ils peuvent donc être laissé en accés libre aux utilisateurs de l'applet sans pour autant en révéler son code source.

2.2.2 Le Convertisseur Java Card

Contrairement à la JVM qui ne traite qu'une classe à la fois, l'unité de conversion du convertisseur est un package. Les fichiers .class sont produits par le compilateur java à partir du code source. Ensuite, le convertisseur prétraite tous les fichiers .class contenu dans un package et les convertit en un fichier CAP (Converted Applet). Le fichier CAP est ensuite chargé sur une smart card Java et exécuté par l'interpréteur.

Durant le processus de conversion, le convertisseur exécute des tâches qui sont normalement exécutées par la JVM au moment du chargement des classes :

– Il vérifie que les images des java classes sont bien formées

- Il vérifie qu'il n'y a pas de violations du sous-langage java
- Il initialise les variables statiques
- Il convertit des références symboliques de classes, des méthodes et des champs en une forme plus compacte supportée par les Java Card.
- Il alloue, stocke et crée des structures de données de la machine virtuelle pour représenter les classes.

Le convertisseur ne prend pas seulement en entrée les fichiers .class à convertir en .CAP mais aussi un ou plusieurs fichiers d'exportation. Si le package importe des classes d'autres packages le convertisseur charge aussi les fichiers d'exportation de ces packages. Le convertisseur produit donc un fichier CAP et un fichier d'exportation du package converti.

2.2.3 Interpréteur Java Card

L'interpréteur, en fournissant un support d'exécution du langage Java, permet aux applets d'être indépendantes du matériel (hardware). Il exécute en particulier les tâches suivantes :

- exécution des instructions du bytecode et des applets
- contrôle de l'allocation mémoire et de la création d'objets
- sécurise le temps d'exécution<

La machine virtuelle Java Card a été décrite comme comprenant le convertisseur et l'interpréteur. Elle est toutefois définie informellement comme étant la partie située sur la carte, à savoir l'interpréteur actuellement décrit. Ainsi concernant les Java Cards, on peut faire l'analogie entre l'interpréteur et la machine virtuelle, contrairement à la machine virtuelle Java qui elle, inclue le convertisseur et l'interpréteur.

2.3 Environnement d'Exécution Java Card

L'environnement d'exécution Java Card (JCRE) gère les ressources de la carte, la communication avec le réseau, l'exécution des applets et leur sécurité. En fait c'est le système d'exploitation des smart cards. Le JCRE est situé au sommet de l'architecture d'une smart card. Il consiste en la machine virtuelle Java(l'interpréteur de bytecode), la structure des classes d'applications Java Cards (APIs), des extensions industrielles spécifiques, et les classes systèmes. Le JCRE sépare habilement les applets de la propriété de la technologie des smart cards des vendeurs. Les applets sont ainsi plus facile à écrire et sont indépendantes de l'architecture privé des smart cards.

2.4 Comment fonctionne le JCRE durant une transaction

La période pendant laquelle la carte est insérée dans une machine est appellée session CAD (Card Acceptance Device). Durant une session CAD, le JCRE opère comme une smart card en établissant une communication APDU (application protocol data units)d'entrée-sortie avec un

serveur. Les APDUs sont des paquets de données échangés entre des applets et une applicatin hôte. Chaque APDU contient soit un ordre du serveur à l'applet, soit une requête de l'applet au serveur.

Après l'initialisation du JCRE, le JCRE entre dans une boucle,en attente d'un ordre APDU du serveur. Celui-ci envoie des ordres APDUs à la Java Card par l'intermédiaire d'une communication utilisant les points de contacts d'entrée-sortie.

Quand une commande arrive, le JCRE sélectionne soit une applet à exécuter comme indiqué dans la commande, soit envoie directement la commande à une applet déjà sélectionnée. L'applet selectionnée prend alors le contrôle et exécute la commande APDU.

Une fois la commande exécutée, l'applet envoie une réponse à l'application hôte et redonne le contrôle au JCRE. Le procédé se repète à l'identique quand une nouvelle commande arrive.

2.5 Les APIs Java Card

Les APIs Java Card consistent en un ensemble de classes spécialisées dans la programmation d'applications de smart card conformément à la norme ISO 7816.

Les APIs contiennent trois packages principaux et un package d'extension. Les trois packages principaux sont les packages *java.lang*, *javacard.framework*, et *javacard.security*. Le package d'extension est *javacardx.crypto*.

Les classes dans ces APIs sont compactes et succintes. Elles incluent des classes adaptées à la plate-forme java fournissant un support sur le langage et des services de cryptographie. Elles contiennent également des classes spécialement conçues pour supporter le standard ISO 7816 des smart cards.

2.5.1 Le package java.lang

Ce package est un sous-ensemble du package java équivalent java.lang. Les classes supportées sont *Object*, *Throwable*, et quelques classes d'exeptions reliées à la machine virtuelle. Beaucoup de méthodes java ne sont toutefois pas disponibles dans ces classes. La classe Java Card Object par exemple ne définit qu'un constructeur par défaut et la méthode *equals*.

2.5.2 Le package javacard.framework

Le package *javacard.framework* est essentiel. Il fournit l'architecture des classes et des interfaces pour le fonctionnement basique des applets Java Card. Plus important, il définit la classe de base *Applet* qui donne le cadre d'exécution et d'interaction des applets durant leur fonctionnement. Son rôle est similaire à celui de la classe *Applet* d'un browser internet.

Une autre classe importante de ce package est la classe *APDU* qui sert aux protocoles de transmission.

2.5.3 Le package javacard.security

Ce package est utilisé pour les fonctions de cryptographie de la plate-forme Java Card. Son implémentation est basée sur le package Java du même nom *java.security*.

Il définit notamment la classe *keyBuilder* et des interfaces variées représentant le système cryptographique de clés (DES) et (DSA et RSA).

2.5.4 Le package javacardx.crypto

Comme dit précedemmentm ce package est un package d'extension. Il contient des classes de cryptographie et des interfaces dont les spécifications sont régulées par les Etats-Unis. Il définit la classe abstraite *Cipher* pour l'encryption et la décryption des données.

Chapitre 3

Sécurité des Java Cards

Une des raisons principales de l'utilisation des Smart Cards est de sécuriser les transactions. La sécurité est donc une priorité pour les développeurs de Java Cards. Les caractéristiques fondamentales de sécurité des Java Cards, leur mécanisme et mode de fonctionnement, ainsi que la sécurité du design d'applet sont présentées ici.

3.1 Caractéristiques

La sécurité des Java Cards est basée sur celle du langage Java, tout en lui ajoutant des protections supplémentaires, spécfiques à la plate-forme Java Card.

3.1.1 Sécurité du langage

Le langage Java Card étant un sous ensemble du langage Java, il convient de faire un bref rappel des bases de la sécurité du langage Java.

- Le langage Java est fortement typé. Ainsi on ne peut pas convertir des entiers en pointeurs.
- Les variables doivent être initialisées avant d'être utilisées.
- Le niveau d'accès de toutes les classes, méthodes et champs est strictement controlé. Par exemple, une méthode privé ne peut pas être invoquée à l'extérieur de sa classe.
- Le langage Java n'a pas de pointeur arithmétique. Ainsi on ne peut pas laisser les pointeurs introduire des virus à l'intérieur de la mémoire.

3.1.2 Sécurité supplémentaire des Java Cards

Voici les fonctionnalités supplémentaires que présente la plate-forme Java Card :

- *Objets persistents et temporaires* : Les objets sont stockés par défaut dans la mémoire persistente. Pour des raisons de sécurité et de performance la plate-forme Java Card autorise également des données temporaires comme des clés de session à être stockées dans des objets

temporaires de la mémoire RAM. Quand la carte est retirée ou quand l'applet sélectionnée est libérée ces objets reprennent leur valeur par défaut (zéro, false ou null).

- *Pare-feu d'applets* : La sécurité et l'intégrité du système JCRE réside dans le fait que les Java Cards sont protégées par le pare-feu d'applet. Celui-ci isole l'applet en séparant son espace de celui du système. Les applets ne peuvent pas se voir entre elles à moins d'être définies dans le même package.
- *Partage D'objets* : Voici la façon dont les objets sont partagés dans le système Java Card : Le JCRE est un utilisateur privilégié qui a un accès total aux applets et aux objets crées par celle-ci. Ensuite une applet a accès aux ressources du JCRE à travers les points d'entrée des objets. Enfin les applets peuvent partager des objets qui implémentent des interfaces partagées. Les applets et le JCRE peuvent partager des données à travers des tableaux globaux.

3.2 Mécanismes de sécurité

La sécurité repose sur un ensemble de mécanismes reliés les uns aux autres et dont un seul problème sur l'un d'entre eux peut invalider tout le processus.

3.2.1 Class File Subset Checking

Contrairement à la JVM, la JCVM a une architecture partagée en deux parties : le convertisseur, qui ne s'exécute pas sur la carte mais sur un ordinateur distant, et l'interpréteur, qui lui, s'exécute sur la carte. Le convertisseur est la partie terminale de la machine virtuelle et prend les fichiers .class en entrée. Durant la conversion les fichiers .class sont soumis au même niveau de vérification qu'ils le seraient par le vérificateur de la JVM.

Comme la technologie Java Card est un sous-ensemble du langage Java, le convertisseur doit vérifier également que les fichiers .class n'utilisent que les caractéristiques de ce sous-ensemble. Cette étape est appelée vérification du sous-ensemble (subset-checking). Durant cette étape le convertisseur vérifie la validité des règles : types supportés (types long, double et float), threads, tableaux multidimensionnels.

3.2.2 Vérification du fichier CAP et du fichier d'exportation

Les classes d'une applet sont composées d'un ou plusieurs packages. On a vu précedemment que le convertisseur prenait toutes les classes d'un même package et les convertissait en un fichier CAP. Le fichier CAP était alors chargé sur la Smart Card et exécuté par l'interpréteur. En plus de créer un fichier CAP, le convertisseur génèrait un fichier d'exportation contenant les APIs publiques du package converti. Celui-ci servira plus tard à convertir un autre package qui importe des classes contenu dans le package du fichier d'exportation.

Le fichier CAP est de même importance que les fichiers .class de la plate-forme java. C'est un format binaire permettant de charger des packages Java sur une Java Card.

En pratique on ne peut pas considérer la fiabilité d'un fichier CAP généré à partir de classes vérifiée comme acquise. C'est le rôle du vérificateur de fichier CAP de vérifier sa conformité. Il s'assure que le fichier CAP a un format correct et qu'il répond à certaines contraintes comme la violation de mémoire, les types des champs, la visibilité ou non visibilité des variables et des méthodes. Le vérificateur de fichier CAP ajoute aussi des contraintes supplémentaires par rapport à celui des fichiers .class de Java.

Durant la vérification le vérificateur du fichier CAP s'assure que le fichier CAP est cohérent avec les fichiers d'exportation qu'il importe, et le fichier d'exportation qui représente son API.

3.3 Sécurité des Applets

La sécurité de n'importe qu'elle application est déterminée par celle de la plate-forme sur laquelle elle tourne et par les caractéristiques intrinsèques de l'application. La plate-forme JavaCard a été conçue afin de garantir que les applets peuvent être construites, installées et exécutées de manière hautement sécuritaire.

Les niveaux de sécurité des applications sont programmés dans des applets. Comme les critères de sécurité sont définis dans la plate-forme elle-même, les programmeurs peuvent se concentrer uniquement sur l'élaboration d'une stratégie sécuritaire des applets. Celle-ci comprend plusieurs étapes :

- *Authentification* : L'accès aux fonctions et aux données d'une applet est controlé. L'identité de l'application hôte envoyant les commandes à l'applet doit être authentifiée. L'identité de l'applet cliente est aussi vérifiée lors de partage d'objets.
- *Confidentialité* : Le caractère privé des données de l'applet doit être protégé : les données sécurisées comme numéro de compte et autre ne doivent donnée qu'après une authentification correct. Les données secrètes comme les clés privées de cryptage ne doivent jamais être autorisées à quitter la carte.
- *Intégrité* : L'exactitude des données de l'applet est garantie. Les données ne peuvent être modifiées qu'après authentification.

Bien sûr on n'aura pas besoin du même niveau de sécurité pour toutes les applets. Une applet contenant des données banquaires nécessitera par exemple un degré de sécurité plus élevé qu'une applet de jeu vidéo. La sécurité est également un coût au niveau des performances et de la mémoire des ressources. Ainsi, les développeurs d'applets doivent évaluer les critères de sécurité nécessaires et choisir ceux appropriés à la situation.

Chapitre 4

Ecriture d'une Applet Java Card

On peut séparer l'écriture d'une applet Java Card en plusieurs phases :
- spécifications des fonctions de l'applet
- attribution des identifiants d'application (AIDs) à l'applet et au package contenant ses classes.
- élaboration de la structure des classes de l'applet
- définition de l'interface entre l'applet et l'application hôte.

4.0.1 Spécification des fonctions de l'applet

Nous allons donner l'exemple d'un porte-monnaie électronique qui doit pouvoir traiter les crédits, les débits et d'autre fonctions de vérification. L'utilisateur de la Smart Card doit donc par exemple pouvoir ajouter de l'argent, en retirer et demander le montant de son capital.

L'applet doit également être sécurisée pour empêcher tout autre utilisateur de se servir de la carte. L'utilisateur devra donc entrer un code PIN sur le clavier connecté au CAD. Le code PIN devra être vérifié avant toute transaction.

4.0.2 AIDs de l'applet

Les classes Java de l'applet sont définies dans un seul package. Dans le tableau 4.1 sont définies les AIDs (Application Identifier) de l'applet du porte-monnaie électronique.

Une AID est constituée de deux parties : un RID et un PIX. Le RID du tableau 4.2 est celui de Sun MicroSystems. On doit en demander un à l'International Standards Organization (ISO). Le PIX peut lui être choisi directement par nous-même.

4.0.3 Définitions des fonctions et de la structure de classe

Une classe implémentant une applet Java Card doit étendre la classe *javacard.framework.Applet*. L'applet surcharge une ou plusieurs méthode publique selon ses spécifications.

TAB. 4.1 – AIDs de l'applet du porte-monnaie et du package de l'applet

Champs	Valeurs	Taille
RID	0xa0, 0x00, 0x00, 0x00, 0x62	5 octets
Package PIX	0x03, 0x01, 0x0c, 0x06	4 octets
Applet PIX	0x03, 0x01, 0x0c, 0x06, 0x01	5 octets

La méthode *process* est une méthode abstraite de la classe Applet qui doit être implémentée par l'applet. Dans cette méthode, l'applet interprète chaque commande APDU et exécute la fonction spécifiée par la commande.

La méthode *select* ou *deselect* est invoquée par le JCRE lorsque l'applet est sélectionnée ou libérée. Si l'on ne veut pas rajouter des fonctions durant la sélection ou la libération il n'est pas nécessaire de surcharger ces méthodes.

Deux méthodes *register* et la méthode *selectingApplet* sont des méthodes déclarées protected et final. Elles sont seulement invoquées par l'applet pour enregistrer un de ses attributs avec le JCRE ou détecter une applet et y appliquer la commande SELECT APDU.

4.0.4 Interface de commande entre l'applet et l'application hôte

Une applet s'exécutant sur une Java card communique avec l'application hôte en utilisant les protocoles APDU (application protocol data units). L'interface de communication entre l'hôte et l'applet est donc un ensemble de commande APDUs concordantes et supportées par les deux.

Une applet Java Card doit supporter un ensemble de commandes APDUs comprenant la commande SELECT et une ou plusieurs commandes APDU d'exécution.

- La commande SELECT indique au JCRE de selectionner l'applet de la carte.
- Les commandes d'exécution définissent les commandes que l'applet doit supporter. Elles doivent être définies en fonction du comportement de l'applet.

Pour chaque commande APDU l'applet doit d'abord décoder la valeur de chaque champ dans les commandes d'en-tête (header). Si le champ -optionnel- de donnée (data field) est inclu, l'applet doit alors déterminer le format de donnée et son contenu.

La commande *e*st décrite dans le tableau.

TAB. 4.2 – La commande APDU *Select APDU*

Name	CLA	INS	P1	P2	Lc	Data Field	taille de la réponse
Get Balance	0x0	0xA4	0x04	0x0	0x0A	0xa0,0x00,0x00,0x00,0x62,0x03	N/A

Les commandes d'en-tête (CLA, INS, P1 et P2) doivent être codées comme dans le tableau pour que le JCRE puisse les identifier comme étant la commande *SELECT APDU*. Le champ *data field* contient les AIDs de l'applet. Le JCRE cherche dans son registre interne la table d'octets des AIDs. Si cela correspond, l'applet du porte-monnaie est sélectionnée et la commande *SELECT APDU* est envoyée à la méthode d'exécution (méthode *process*) de l'applet.

L'interface des APDUs varie suivant l'application Java Card. Ainsi une carte de crédit n'aura pas les mêmes commandes qu'une carte vitale qui doit pouvoir fournir des informations sur le patient, le medecin, le traitement...

Dans notre exemple du porte-monnaie électronique nous aurons besoin des commandes d'exécution *Credit*, *Debit*, *Get Balance* et une fonction pour vérifier le code Pin entré par l'utilisateur *Verify Pin*.

Nous allons attribuer à la commande APDU *Get Balance* une instruction 0x80 et 0x30. Cette commande ne nécessite aucun paramètre d'entrée et la réponse est le capital courant donné sur deux octets. Le tableau 4.3 décrit la commande APDU de *Get Balance*.

TAB. 4.3 – La commande APDU *Get Balance*

Name	CLA	INS	P1	P2	Lc	Data Field	Le(taille de la réponse)
Get Balance	0x80	0x30	0	0	N/A	N/A	2

Définissons maintenant la commande APDU *Verify Pin* qui consiste à vérifier le code PIN entré par l'utilisateur. Contrairement à la commande *Get Balance*, celle-ci a besoin d'un paramètre d'entrée. Elle est présentée dans le tableau 4.4.

TAB. 4.4 – La commande APDU *Verify Pin*

Name	CLA	INS	P1	P2	Lc	Data Field	Le(taille de la réponse)
Verify Pin	0x80	0x20	0	0	PIN Len	Pin Value	N/A

- CLA contient la structure de la commande
- L'octet INS(0x20) indique l'instruction VERIFY

– P1 et P2 ne sont pas utilisés et sont tous les deux à 0. Le champ *data field* contient le code PIN.

De la même façon on a les tableaux des commandes *Credit APDU*

TAB. 4.5 – La commande APDU *Credit*

Name	CLA	INS	P1	P2	Lc	Data Field	Le(taille de la réponse)
Credit	0xB0	0x30	0	0	1	Valeur du crédit	N/A

et *Debit APDU*.

TAB. 4.6 – La commande APDU *Debit*

Name	CLA	INS	P1	P2	Lc	Data Field	Le(taille de la réponse)
Debit	0x80	0x20	0	0	1	Valeur du débit	N/A

Conclusion

Les Java Cards joueront un rôle de plus en plus important de la vie quotidienne des gens dans les années à venir. On les rencontre déjà en tant que carte de crédits, cartes de fidélité, carte de santé (carte vitale) et carte d'identité. Leur petite taille, et le niveau de sécurité élevé qu'elles procurent en font l'outil idéal pout tout transport d'information à protéger ou pas.

La technologie Java Card permet aux smart cards d'exécuter des applications (applets) écrites en un sous-ensemble du langage Java. Hautement sécurisée et occupant peut de place mémoire elle intègre tous les avantages du langage Java, tout en n'en gardant que l'essentiel.

Bibliographie

Martin S. Nicklous Thomas Schäck Achim Schneider et Franck Seliger.*Smart Card Application Development Using Java 2nd Edition.*

Zhiqun Chen.*Java Card Technology for Smart Cards.*Addison Wesley, 2000.

http ://java.sun.com/products/javacard/

http ://itmanagement.earthweb.com/ecom/article.php/601661

http ://java.sun.com/products/javacard/

http//itmanagement.earthweb.com/ecom/article.php/601661

http ://www.javaworld.com/javaworld/jw-07-1999/jw-07-javacard.html

http ://developers.sun.com/techtopics/mobility/javacard/articles/javacard2/

http ://developers.sun.com/techtopics/mobility/javacard/articles/intro/

http ://www.javacardforum.org/