SMARTSHELL: UMA FERRAMENTA DE APOIO AO DESENVOLVIMENTO DE APLICAÇÕES JAVA CARD

Crístian Deives dos Santos Viana
Gerência Educacional de Tecnologia da Informação - CEFET-RN
Av. Salgado Filho, 1559, Tirol, CEP 59.015-000, Natal-RN
E-mail: cristiandeives@gmail.com

Lígia Maria Moura e Silva
Gerência Educacional de Tecnologia da Informação - CEFET-RN
Av. Salgado Filho, 1559, Tirol, CEP 59.015-000, Natal-RN
E-mail: ligia.cefet@gmail.com

Jorgiano Márcio Bruno Vidal
Gerência Educacional de Tecnologia da Informação - CEFET-RN
Av. Salgado Filho, 1559, Tirol, CEP 59.015-000, Natal-RN
E-mail: jorgiano@cefetrn.br

RESUMO

Os *smart cards* são cartões de plástico que contêm um processador e possuem capacidade de armazenamento de informações. A manipulação de uma aplicação, incluindo sua instalação, é um processo que exige autenticação com o *smart card* para garantir a segurança desse procedimento. Essa comunicação com o cartão é feita de forma segura, reduzindo a adulteração das informações e evitando sua falsificação através de mecanismos de alta segurança, como criptografia e senhas. A GlobalPlatform é uma organização democrática e completamente independente cuja missão é estabelecer, manter e guiar a adoção de padrões para permitir uma infraestrutura aberta e interoperável para cartões inteligentes e outros dispositivos. Java Card é uma tecnologia mantida pela Sun Microsystems que provê um ambiente seguro para aplicações que executam em *smart cards*. Várias aplicações podem ser instaladas em um único cartão e novas aplicações podem ser adicionadas mesmo depois de o cartão ter sido comercializado. Aplicações escritas na linguagem de programação Java, também chamadas de *applets* ou *cardlets*, podem ser executadas seguramente em cartões de diversos fabricantes. Este artigo tem como objetivo apresentar uma ferramenta, chamada SmartShell, para o apoio à manipulação de aplicações em cartões inteligentes, escritas em Java Card. O objetivo dessa ferramenta é simplificar a interação com o cartão, de forma que o usuário não precise conhecer os detalhes dessa comunicação. Através dela, é possível realizar a autenticação com um cartão, instalação de *applets*, e outras operações para a manipulação dos mesmos.

PALAVRAS-CHAVE: *smart card, APDU, applet*, Java Card, GlobalPlatform

1. INTRODUÇÃO

A instalação de um *applet* em um *smart card* é um processo complexo, pois, previamente, é necessária uma autenticação com o cartão que envolve criptografia, manipulação de comandos APDU (*Application Protocol Data Unit*), entre outras coisas.

Para haver comunicação entre uma aplicação JSE (*Java Standard Edition*) e um *applet* já instalado, é preciso que essa aplicacão trabalhe diretamente com APDUs, assim como fazem os *applets*. Porém, essa é uma tarefa trabalhosa, pois os comandos APDU devem ser montados *byte* a *byte* e cada resposta também deve ser analisada *byte* a *byte*.

A ferramenta proposta, SmartShell, visa simplificar a instalação dos *applets* e sua comunicação com aplicações Java SE.

2. *SMART CARD*

Um *smart card* é um cartão de plástico que contém um circuito integrado embutido, com a finalidade de armazenamento e processamento de informações. A idéia de incorporar um microprocessador em um cartão de plástico foi introduzida por dois inventores alemães, Jürgen Dethloff e Helmut Grötrupp, em 1968. A evolução dos *smart cards* também teve a colaboração do Japão, com o Dr. Kunitaka Arimura, que patenteou o conceito de *smart card*. Na França, Roland Moreno patenteou o cartão com microprocessador, que mais tarde passou a ser chamado de "*smart card*". As primeiras tentativas de utilização de cartões inteligentes ocorreram na França e na Alemanha, no início dos anos 80, com as funcionalidades de cartão de crédito e cartão telefônico [CHEN, 2004].

A cada dia os *smart cards* se tornam mais poderosos, com a evolução da tecnologia dos *chips* e da criptografia, ganhando bem mais funcionalidades, além de cartão de crédito: dinheiro eletrônico, comunicação sem fio, armazenamento de informações médicas, acesso à TV por satélite, entre outras [CHEN, 2004].

2.1. Comunicação

O dispositivo utilizado para fazer a comunicação entre o cartão e outro computador denomina-se CAD (*Card Acceptance Device*). Ele pode ser classificado em leitores e terminais. O leitor é conectado ao computador através de uma porta USB, paralela ou serial. Os terminais são computadores que possuem um leitor de *smart card* como um de seus componentes. As aplicações que se comunicam com o cartão, independentemente de estarem em um computador conectado ao leitor ou estarem em um terminal, são chamadas de aplicações *host*. O caminho de comunicação entre o cartão e o *host* é *half-duplexed*, isto é, os dados podem tanto ser enviados do *host* para o cartão ou do cartão para o *host*, mas não ambos ao mesmo tempo [CHEN, 2004].

Especificado em ISO 7816-4, o APDU é um protocolo de nível de aplicação que está entre o *smart card* e a aplicação *host*. Existem duas estruturas possíveis de APDU. A primeira é utilizada pela aplicação *host* para enviar comandos ao cartão, chamada de *command* APDU. A outra é utilizada pelo cartão para mandar respostas de volta, chamada de *response* APDU. Um *smart card* fica sempre passivo, esperando por um *command* APDU do *host*. Ele então executa a instrução especificada no comando e responde para o *host* com um *response* APDU. *Command* e *response* APDUs são trocados alternadamente entre o cartão e o *host* [CHEN, 2004].

O cabeçalho do *command* APDU possui quatro *bytes*: CLA (classe da instrução), INS (código da instrução) e P1 e P2 (parâmetros 1 e 2). O corpo, que vem após o cabeçalho, varia em tamanho. O Lc especifica, em *bytes*, o tamanho do campo de dados. Esse campo contém os dados que são enviados ao cartão. O último *byte*, Le, indica o número de *bytes* que são esperados como reposta [CHEN, 2004]. A estrutura dos possíveis casos de um *command* APDU está detalhada na figura 1.

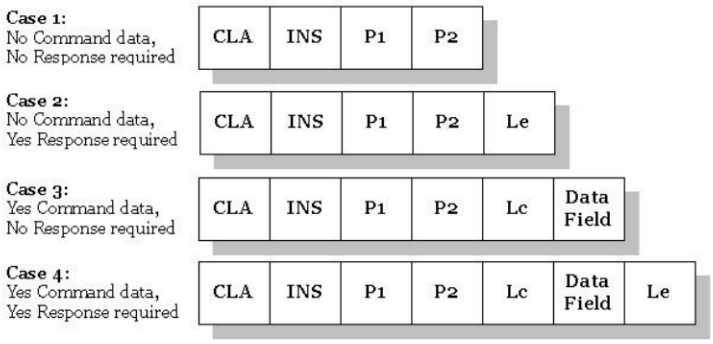

Figura 1: Estrutura de um *command* APDU

O *response* APDU consiste de um corpo opcional e dois campos obrigatórios. O corpo consiste de um campo de dados, cujo tamanho é determinado pelo Le do *command* APDU. A parte final é composta de dois campos, SW1 e SW2, que juntos são chamados de *status word*, que representa o *status* da execução [CHEN, 2004]. A estrutura de um *response* APDU está detalhada na figura 2.

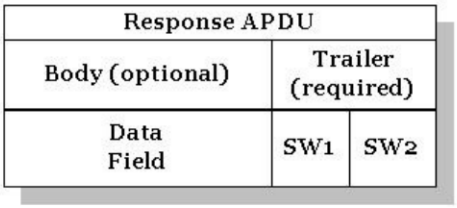

Figura 2: Estrutura de um *response* APDU

3. JAVA CARD

A tecnologia Java Card é uma adaptação da plataforma Java, fornecendo um ambiente seguro para aplicações que rodam em *smart cards*, chamadas de *applets* [SUN MICROSYSTEMS, 2003]. O maior desafio desta tecnologia é fazer com que um software Java seja instalado no cartão, conservando espaço para outras aplicações. Para isso, Java Card suporta apenas um subconjunto dos recursos da linguagem Java e sua máquina virtual é dividida em duas partes, uma que roda no cartão *(on-card)* e uma que roda fora do cartão (*off-card*) [CHEN, 2004].

A plataforma Java Card consiste de três partes:

- *Java Card Virtual Machine* (JCVM): define o subconjunto de Java e as características da máquina virtual necessários para aplicações que rodam em *smart cards*.
- *Java Card Runtime Environment* (JCRE): descreve o comportamento do Java Card em tempo de execução, incluindo gerenciamento de memória e *applets*, entre outros recursos.
- *Java Card Application Programming Interface* (API): descreve os pacotes e classes Java para desenvolver aplicações que rodam em *smart cards* [CHEN, 2004].

3.1. Linguagem Java Card

Devido às restrições de memória, a plataforma Java Card suporta apenas um subconjunto da linguagem Java. Esse subconjunto inclui recursos para desenvolver programas para *smart cards* e outros dispositivos, preservando as características de orientação a objeto da linguagem Java. De acordo com [CHEN, 2004], os recursos suportados pela linguagem são:

- Tipos primitivos: `boolean`, `byte`, `short`;
- *Arrays* unidimensionais;

- Pacotes, classes, interfaces e exceções;
- Recursos de orientação a objeto: herança, métodos virtuais, sobrecarga e criação dinâmica de objetos, entre outros;
- O suporte ao tipo primitivo `int` de 32 *bits* é opcional, dependendo da máquina virtual.

Os recursos não suportados são:

- Tipos primitivos grandes: `long`, `double`, `float`;
- Caracteres e *Strings*;
- *Arrays* multi-dimensionais;
- Carregamento dinâmico de classes;
- Gerenciamento de segurança;
- Coletor de lixo e finalização;
- *Threads*;
- Serialização de objetos;
- Clonagem de objetos;

3.2. Máquina Virtual Java Card

A principal diferença entre a máquina virtual Java Card (JCVM) e a máquina virtual Java (JVM) é que a primeira é implementada em duas partes separadas. A parte *on-card* da JCVM inclui o interpretador de *bytecode* Java Card. O conversor Java Card (*converter*) , executado em um computador, é a parte *off-card* da máquina virtual. O conversor carrega e pré-processa as classes que formam um pacote Java e gera um arquivo CAP (*Converted APplet*). O arquivo CAP é então carregado no cartão e executado pelo interpretador. Além de criar um arquivo CAP, o conversor gera um arquivo *export*, representando as APIs públicas do pacote que está sendo convertido. Quaisquer recursos da linguagem não suportados usados em um *applet* são detectados pelo *converter* [CHEN, 2004].

3.3. *Applets* Java Card

Um *applet* Java Card é um programa Java que adere a um conjunto de convensões que o permitem executar no *Java Card Runtime Environment*. Um *applet* Java Card não é feito para ser rodado em um navegador *web*, e sim em um *smart card* [CHEN, 2004].

Um *applet* deve herdar da classe `javacard.framework.Applet`. Essa classe é a super-classe de todos os *applets* em um Java Card e ela define as variáveis e os métodos de um *applet*. Um *applet* em execução no cartão é uma instância da classe `Applet`. Assim como qualquer objeto persistente, uma vez criado, um *applet* vive no cartão para sempre. O JCRE suporta um ambiente multi-aplicação. Vários *applets* podem coexistir em um único *smart card* Java, e um *applet* pode ter várias instâncias [CHEN, 2004].

4. GLOBALPLATFORM

Com o crescimento da indústria de cartões, em meados dos anos 90, surgiram conflitos na adoção de padrões de comunicação entre cartões inteligentes e aplicações *host*. As três tecnologias liderantes nessa área eram Java Card, Windows for Smart Cards e MULTOS e se fazia necessário o estabelecimento de um padrão de programação que permitisse portabilidade de aplicações entre diferentes fabricantes [GLOBALPLATFORM, 2000].

Através do projeto da GlobalPlatform (anteriormente chamada Open Platform), a Visa desenvolveu um padrão de cartão importante que estava faltando na indústria – um padrão independente de *hardware*, de vendedor e de aplicação. Esse novo padrão oferece uma segurança em comum e uma arquitetura de gerenciamento de cartões que protege o aspecto mais importante de um sistema de cartão: a infraestrutura [GLOBALPLATFORM, 2000].

A GlobalPlatform é uma organização democrática e completamente independente cuja missão é estabelecer, manter e guiar a adoção de padrões para permitir uma infraestrutura aberta e interoperável para cartões inteligentes e outros dispositivos [GLOBALPLATFORM, 2006]. Ela define um padrão flexível e poderoso para fabricantes de cartão criarem sistemas multi-aplicação que satisfaçam suas necessidades de mercado e suas constantes mudanças. Esse padrão permite que eles escolham a tecnologia do cartão que é considerada apropriada para eles hoje, garantindo a possibilidade de migração para outra tecnologia no futuro, sem impactos significantes na infraestrutura [GLOBALPLATFORM, 2000].

5. SMARTSHELL

O objetivo do SmartShell é interagir com um *smart card* de forma simples. Através dele, é possível realizar a autenticação com um cartão, instalação de *applets*, e todas as outras operações definidas pela GlobalPlatform para a manipulação dos mesmos. Além disso, o usuário pode solicitar a execução de vários desses comandos listados em um arquivo de texto.

Para executar uma ação, o usuário deve digitar um comando SmartShell no *prompt*. Para que esse comando seja válido, deve ser composto pelo seu nome – específico do SmartShell – e, opcionalmente, por uma seqüência de parâmetros. Os nomes possíveis para um comando serão definidos na seção 5.3, assim como seus parâmetros, sendo eles separados por espaço, não havendo diferença entre letras maiúsculas e minúsculas.

5.1. Arquitetura

Os detalhes da comunicação entre o SmartShell e o cartão são abstraídos utilizando a API OpenCard Framework (OCF), que é um conjunto de classes Java que interagem com o leitor do cartão, utilizando JNI (*Java Native Interface*).

O núcleo do SmartShell é composto de quatro classes: CAP, Comunicacao, Convert e Criptografia. A classe CAP é responsável por analisar um arquivo CAP e fornecer métodos que retornam informações específicas desse arquivo, como o seu AID, os AIDs dos *applets* (se existirem) , o tamanho total, entre outras. A classe Comunicacao é a principal delas, pois é nela que estão os métodos utilizados na comunicação com o cartão, utilizando o OCF. Para cada comando definido pela GlobalPlatform existe um método correspondente nessa classe, com todos os parâmetros exigidos pela especificação. As outras duas classes, Convert e Criptografia, contêm métodos utilitários para conversão de tipos (de String para *array* de *bytes*, de *array* de *bytes* para String etc) e métodos relacionados a criptografia, utilizados na autenticação do SmartShell com o cartão.

Além do núcleo, existe a possibilidade de utilizar interface gráfica com o usuário (GUI), que pode ser *desktop* ou *web.* No caso do SmartShell, utiliza-se a interface texto, implementada na classe SmartShellText.

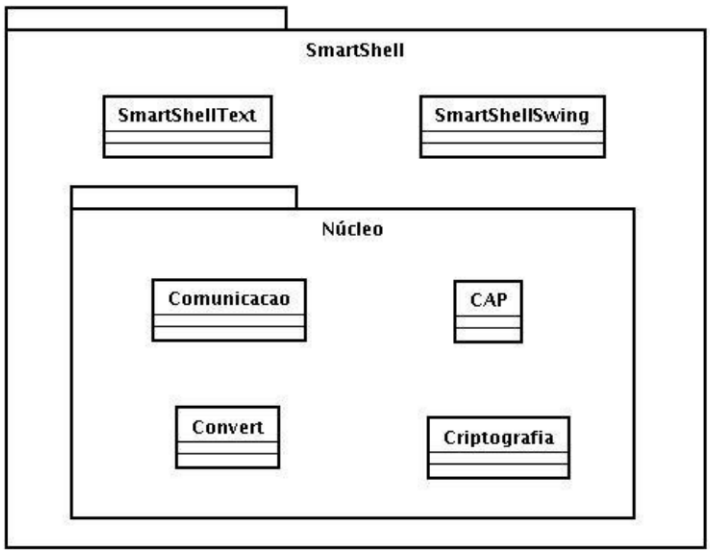

Figura 3: Arquitetura do SmartShell

5.2. Funcionamento

Para cada comando que o usuário deseja executar, o SmartShell realiza uma sequência de operações para que esse

comando seja executado corretamente. Inicialmente, o usuário digita um comando SmartShell e seus respectivos parâmetros, o qual será processado pela ferramenta. Caso esse comando seja válido, o SmartShell irá gerar um ou mais *command* APDUs e enviá-los para o cartão. O *smart card* receberá cada APDU e o *applet* que estiver selecionado irá tratá-lo e gerar um *response* APDU, enviando-o de volta ao SmartShell. Em seguida, a ferramenta irá analisar essa resposta e mostrá-la ao usuário de uma forma amigável. A figura 4 resume o funcionamento do SmartShell.

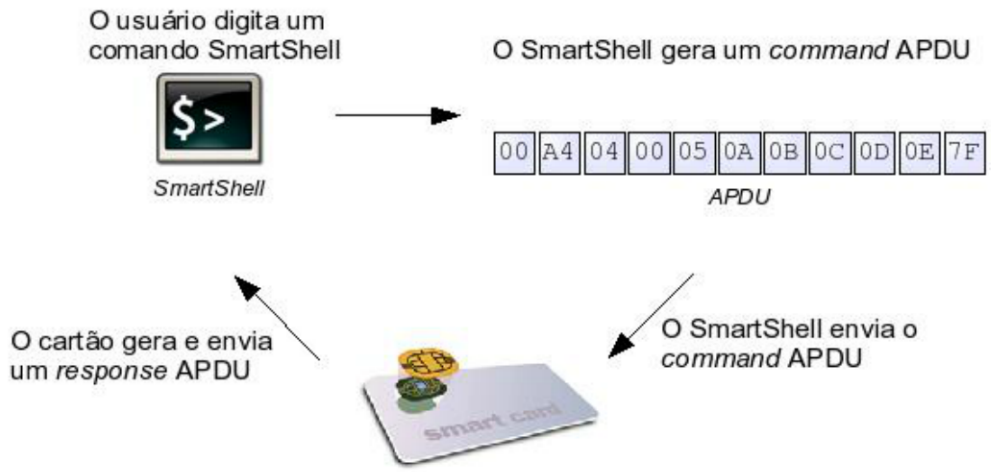

Figura 4: Funcionamento do SmartShell

5.3. Comandos SmartShell

O SmartShell implementa os comandos definidos pela GlobalPlatform de uma forma simplificada. Esses comandos são acessados direta ou indiretamente através das seguintes funcionalidades:

- **Autenticar**: para realizar a autenticação do SmartShell com o cartão, é utilizado o comando `AUTH`, que tem como parâmetro obrigatório apenas o KMC. O KMC é uma chave própria do cartão, utilizada no processo de autenticação. Essa funcionalidade utiliza os comandos `INITIALIZE UPDATE` e `EXTERNAL AUTHENTICATE`, definidos pela GlobalPlatform.
  Ex.: `AUTH CHAVEDOCARTAO`
- **Instalar arquivo CAP**: para executar essa funcionalidade, o usuário deve digitar o comando `INSTALL`, que recebe como parâmetro o endereço completo do arquivo CAP a ser instalado. Caso existam *applets* no arquivo, cada um será instalado automaticamente, utilizando o AID atribuído na criação do CAP. Os comandos da GlobalPlatform utilizados nessa funcionalidade foram `INSTALL` e `LOAD`.
  Ex.: `INSTALL C:\JavaCard\applet.cap`
- **Deletar *applet*/CAP**: depois de instalado, um *applet* ou um arquivo CAP pode ser removido do cartão. Para realizar essa operação, utiliza-se o comando `DELETE`, seguido do AID do *applet* ou do arquivo CAP que se deseja deletar. Para se realizar essa funcionalidade, utiliza-se apenas o comando `DELETE` da GlobalPlatform.
  Ex.: `DELETE 01:02:03:04:05`
- **Listar componentes instalados**: um *smart card* pode conter vários *applets* e arquivos CAP instalados. Para obter o *status* e o AID dos *applets* e dos arquivos CAP instalados no cartão, pode-se utilizar o comando `LIST`. Esse comando possui um parâmetro obrigatório e outro opcional. O parâmetro obrigatório especifica o tipo do componente que será listado, podendo ser apenas um dos três valores seguintes: `APPLETS`, que mostra o AID e o *status* do ciclo de vida dos *applets*; `LOADFILES`, que também mostra o AID e o *status* do ciclo de vida dos arquivos CAP; e `CARDMANAGER`, que mostra o AID do *Card Manager*, seus privilégios e algumas informações específicas do fabricante do cartão. O segundo parâmetro do comando `LIST` é opcional e serve para filtrar os componentes que serão listados, através do AID. Se ele estiver presente, só serão listados os componentes especificados no primeiro parâmetro cujos AIDs iniciem com o AID informado; caso contrário, todos os componentes especificados serão retornados. O comando da GlobalPlatform utilizado para a realização dessa funcionalidade é o `GET STATUS`.
  Ex.: `LIST APPLETS 01:02`

- **Selecionar applet**: um *applet* deve ser selecionado antes que qualquer comando APDU possa ser enviado para ele. Para selecionar um *applet* através do SmartShell, basta apenas digitar o comando `SELECT` – cujo comando correspondente da GlobalPlatform possui o mesmo nome – seguido do seu AID.
  Ex.: `SELECT 01:02:03:04:05`
- **Executar comando APDU**: além dos comandos descritos acima, o usuário pode executar um comando APDU próprio através do comando `EXEC`. Seu único parâmetro é uma seqüência de números hexadecimais separados por ":" (dois-pontos) que formam o comando APDU. Após o envio do comando, é mostrado na tela o *status* e os dados do APDU de resposta.
  Ex.: EXEC 00:A4:04:00:05:01:02:03:04:05:7F
- **Executar seqüência de comandos SmartShell**: caso seja necessário, o usuário pode executar vários comandos SmartShell listados em um arquivo de texto, de tal forma que cada linha contenha apenas um desses comandos. Para utilizar esse recurso, basta apenas digitar EXECFILE seguido do endereço completo do arquivo.
  Ex.: EXECFILE C:\JavaCard\comandos_smartshell.txt

6. CONCLUSÃO

Com o crescente desenvolvimento de aplicações para cartões inteligentes, surgiu a necessidade de um *software* que auxiliasse a manipulação dessas aplicações. O SmartShell foi desenvolvido para suprir essa necessidade, simplificando a interação de uma aplicação *host* com o cartão. Através dele, operações como autenticação com um cartão e instalação de *applets* se tornaram mais fáceis de se realizar, aumentando a produtividade no desenvolvimento de sistemas para *smart cards*.

7. REFERÊNCIAS BIBLIOGRÁFICAS

CHEN, Zhinqun. ***Java Card Technology for Smart Cards – Architecture and Programmer's Guide***. Ed Addison-Wesley, 2004.

GlobalPlatform. ***Open Platform – Card Specification version 2.0.1'***. 2000.

GlobalPlatform. Disponível em <http://www.globalplatform.org/showpage.asp?code=missmstate> Acesso em: 10 de outubro de 2006.

ORTIZ, C. Enrique. ***An Introduction to Java Card Technology***. Disponível em <http://developers.sun.com/techtopics/mobility/javacard/articles/javacard1> Acesso em: 2 de outubro de 2006.