UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE ESTATÍSTICA E INFORMÁTICA
CURSO DE BACHARELADO EM CIÊNCIAS DA COMPUTAÇÃO

JOpenSC: Uma interface gráfica para OpenSC

Eduardo Ruhland

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Ciências da Computação.

Florianópolis - SC
2005 / 1

Eduardo Ruhland

JOpenSC: Uma interface gráfica para OpenSC

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Ciências da Computação

Orientador

Ricardo Felipe Custódio, Dr.
Universidade Federal de Santa Catarina

Banca examinadora

Júlio da Silva Dias, M. Sc.
Universidade Federal de Santa Catarina

Julíbio David Ardigo, Dr.
Universidade do Estado de Santa Catarina

Resumo

Este trabalho tem o objetivo de relatar as atividades realizado por este acadêmico na implementação de uma interface gráfica para a biblioteca e conjunto de aplicações `OpenSC`. Alguns conceitos importantes, como criptografia, padrões PKCS, e smart cards, são brevementes descritos para melhor compreensão do trabalho. Os passos realizados para a conclusão deste trabalho e os problemas encontrados durante a sua execução são relatados para a contribuição com trabalhos futuros.

Palavras-chave: OpenSC, JOpenSC, smart cards.

Abstract

The purpose of this work is to describe the activities done by this student during the implementation of a graphical interface for the smart card library and aplications called `OpenSC`. Some important concepts, like cryptography, PKCS standards, and smart cards, are described for better understanding of this work. The steps taken to conclude this work and the problems that appeared during its execution were reported for contribution with future applications.

Keywords: OpenSC, JOpenSC, smart cards.

Sumário

1 Introdução

Cada vez mais as pessoas e empresas trocam e armazenam informações através de computadores. Esta prática, possivelmente acarretará na substituição do armazenamento e transmissão de informações através do papel para o meio eletrônico.

Neste contexto, mecanismos de proteção de dados e garantia de sua autenticidade fazem-se necessários. Um desses mecanismos é a assinatura digital. Ela permite tanto autenticar o emissor de uma mensagem, isto é, que quem a assinou é o seu emissor, como permite verificar a integridade da mesma.

Na assinatura digital, que é uma seqüência de bits concatenada à mensagem, utiliza-se a tecnologia de chaves públicas e de função resumo.

No processo de assinatura em si cifra-se o resumo da mensagem com a chave privada do autor, que é de conhecimento apenas de seu proprietário. Entretanto, como esta chave é um arquivo, pode ser roubada, permitindo que uma pessoa mal intencionada utilize a chave roubada para assinar documentos em nome de outra pessoa. Umas das formas de minimizar esse problema é a utilização de smart cards.

Os Smart Cards vêm obtendo grande aceitação na utilização de mecanismos de assinatura digital devido a sua segurança, portabilidade e facilidade de uso. Os cartões guardam a chave privada do proprietário em sua memória, tornando-a inacessível diretamente.

Soluções nesta direção estão sendo disponibilizadas tanto por empresas privadas como por comunidades de software livre. Uma das iniciativas em software livre é a OpenSC. Porém, esta ferramenta foi disponibilizada somente com interface de comando de linha. O presente projeto visa implementar uma interface gráfica para esta ferramenta.

1.1 Objetivos

1.1.1 Geral

- Implementar uma ferramenta gráfica para utilização de Smart Cards na realização de assinaturas digitais.

1.1.2 Específicos

- configurar sistema linux para assinatura digital com Smart Card;
- avaliar soluções com Java Card;
- avaliar bibliotecas para interface gráfica;
- implentar inferface gráfica para utilização de ferramentas de assinatura digital com Smart Cards.

1.2 Justificativa

Soluções baseadas em software livre tem sido elaboradas e disponibilizadas para diferentes aplicações. Estas iniciativas necessitam da colaboração da comunidade para seu aperfeiçoamento e disseminação.

Este projeto visa facilitar a utilização da ferramenta OpenSC, que é uma biblioteca e aplicações para Smart Cards baseado em comandos de linha, através da diisponibilização de uma interface gráfica para a mesma.

Como resultado, espera-se contribuir para a disseminação da ferramenta OpenSC, bem como contribuir com a cultura de utilização de Smart Cards para assinatura digital.

1.3 Motivação

A grande demanda por profissionais com conhecimentos em segurança computacional inspirou este acadêmico a aprofundar seus conhecimentos nesta área. Ao analisar os trabalhos desenvolvidos sobre o tema na UFSC, vislumbrou a possibilidade de contribuir com o LabSEC na implementação de soluções baseadas em software livre, tema que estuda a algum tempo. Outra inspiração é a possibilidade de contribuir com a comunidade do software livre no aperfeiçoamento de uma solução disponibilizada.

1.4 Descrição do Documento

Este documento está dividido em 6 capítulos:

Primeiro Capítulo Neste capítulo contextualiza-se a importância dos smart cards, os objetivos deste acadêmico com o presente trabalho, a justificativa e a motivação para a realização do mesmo.

Segundo Capítulo No segundo capítulo faz-se uma rápida introdução aos conceitos básicos de criptografia.

Terceiro Capítulo Nesta parte faz-se uma breve explicação dos Padrões de Criptografia de Chave Pública, mais especificamente dos padrões PKCS #11 e o PKCS #15 que são de maior importância no trabalho desenvolvido.

Quarto Capítulo Este capítulo classifica os smart cards, descreve sua arquitetura, lista seus principais padrões e especificações e descreve sua forma de transmissão de dados.

Quinto Capítulo O quinto capítulo informa os requisitos necessários para a execução do programa, descreveos passos necessários para a instalação dos softwares e hardwares necessários para utilização da interface gráfica da o OpenSC e aaplicação gráfica JOpenSC.

Sexto Capítulo No último capítulo é feita uma análise dos resultados almejados por este acadêmico e dos resultados obtidos pelo mesmo.

2 Criptografia

Para que se possa entender este trabalho é fundamental conceituar alguns tópicos. Neste capítulo descreve-se alguns conceitos básicos de criptografia, algorítmos criptográficos e protocolos criptográficos de 'smart cards".

Criptografia é a arte ou ciência de técnicas matemáticas relacionadas à aspectos da segurança de dados como:

confidencialidade não permitir que pessoas não autorizadas tenham acesso ao conteúdo de determinada informação.

integridade de dados detectar alteração não autorizada de dados.

autenticação identificar entidades ou origem de dados.

não-repúdio evitar que uma entidade negue ações ou obrigações anteriores.

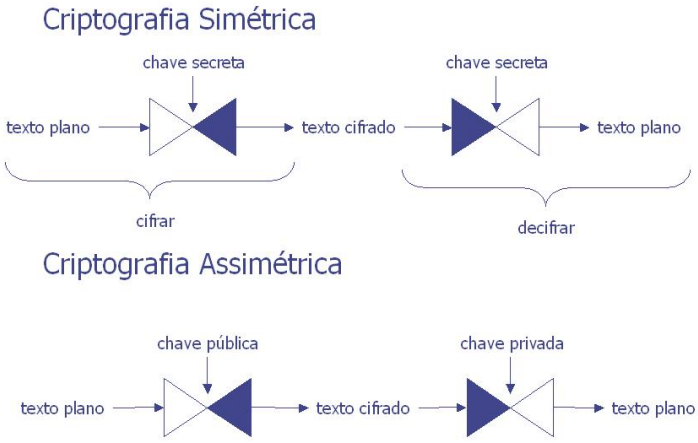
Na terminologia criptográfica, a mensagem inteligível é chamada texto plano ou texto limpo. Codificar o conteúdo da mensagem de maneira que esconda seu conteúdo de terceiros é chamada cifrar. A mensagem cifrada é chamada texto cifrado. O processo de recuperação do texto plano a partir do texto cifrado é chamado decifrar. Ao cifrar e decifrar geralmente utiliza-se uma chave, e o método de codificação é tal que é possível decifrar apenas sabendo a chave apropriada.

Cripto-análise é o estudo de métodos matemáticos que são utilizados na tentativa de quebrar técnicas criptográficas. Criptologia é o estudo da criptografia e da cripto-análise.

O método utilizado para cifrar e decifrar é chamado cifrador. Alguns métodos criptográficos contam com o sigilo dos algoritmos de cifra; estes algoritmos são apenas de interesse histórico e não são adequados paras as necessidades do mundo atual. Ao invés de sigilo do método, todos os algoritmos modernos baseiam sua segurança no uso da chave; uma mensagem pode ser decifrada apenas se a chave utilizada para decifrar combina com a chave utilizada para cifrar.

Existem duas classes de algoritmos baseados em chaves, algoritmos simétricos (ou chave secreta) e assimétricos (ou chave pública). A diferença básica entre eles é que os algoritmos simétricos utilizam a mesma chave para cifrar e decifrar (ou a chave para decifrar é facilmente derivada da chave de cifra), enquanto os algoritmos assimétricos utilizam chaves diferentes (uma para cifrar e outra para decifrar.

Algoritmos simétricos podem ser divididos em cifradores de *stream* ou cifradores de bloco(STALLINGS, 1999). Cifradores de *stream* cifram um único bit de texto plano por vez, enquanto cifradores de bloco tomam um número de bits (tipicamente 64 bits em cifradores modernos), e os cifram como uma única entidade.



Cifradores assimétricos (também chamados algoritmos de chave pública) trabalham com pares de chaves(STALLINGS, 1999). Ao deixar que uma chave seja pública, permite-se que qualquer pessoa cifre com esta chave, entretanto apenas a pessoa que conhece a chave correspontente do par possa decifrar a mensagem. A chave para cifrar é também chamada de chave pública e a chave para decifrar de chave privada. A segurança destes cifradores é baseada em manter a chave privada em segredo.

Geralmente, algoritmos simétricos executam mais rápido em um computador do que os algoritmos assimétricos. Muitas vezes eles são utilizados em conjunto, de forma que o algoritmo de chave pública é utilizado para cifrar um chave de cifra gerada de forma aleatória, e a chave aleatória é utilizada para cifrar a mensagem utilizando algoritmos simétricos.

Um dos algoritmos de criptografia simétrica mais conhecidos é o DES (*Data Encryption Standard*), entretanto a sua versão mais simples não pode mais ser considerado seguro devido ao aumento do poder de processamento dos computadores modernos. Por esse motivo foi padronizado um novo e mais poderosos cifrador chamado AES (*Advanced*

Encryption Standard). O RSA é provavelmente o algoritmo de criptografia assimétrica mais conhecido.

3 PKCS

Os Padrões de Criptografia de Chave Pública são especificações que têm o objetivo de acelerar a distribuição da criptografia de chave pública. Os primeiros documentos PKCS foram publicados em 1991, como resultado de reuniões de um pequeno grupo de pioneiros na utilização da tecnologia de chave pública e hoje são produzidos pelo RSA Laboratories em cooperação com desenvolvedores do mundo inteiro.

O processo para a produção destes padrões ocorre em quatro passos:

1. Publicação de documentos cuidadosamente escritos, descrevendo os padrões PKCS.
2. Solicitação de opiniões e conselhos de desenvolvedores e usuários sobre alterações e extensões necessárias ou úteis.
3. Publicação de padrões revisados quando apropriado.
4. Disponibilizar guias de implementação e/ou implementações de referência.

O RSA Laboratories tem a palavra final em cada documento. Entretanto, quando um documento é aceito como base para um padrão formal, a RSA Laboratories abre mão da propriedade sobre o documento, permitindo o processo de desenvolvimento de padrões abertos.

Serão descritos à seguir dois padrões PKCS relevantes para o mundo dos smart cards: o PKCS #11 e o PKCS #15.

3.1 PKCS #11: Cryptographic Token Interface Standard

Este padrão especifica uma API, chamada Cryptoki, para dispositivos que armazenam informações criptográficas e realizam funções criptográficas. Cryptoki, pronunciada como

crypto-key e forma abreviada para interface de token criptográfico, segue uma simples abordagem baseada em objetos, tratando dos objetivos de independência de tecnologia (qualquer tipo de dispositivo) e compartilhamento de recursos (múltiplas aplicações acessando múltiplos dispositivos), apresentando para as aplicações um visão lógica comum do dispositivo chamado token criptográfico.

3.2 PKCS #15: Cryptographic Token Information Format Standard

PKCS #15 estabelece um padrão que permite usuários utilizem tokens criptográficos para se identificarem para múltiplas aplicações, independente do provedor criptoki (ou outra interface de token) da aplicação.

PKCS #15 cobre dois grupos de dispositivos, os convencionais hardware tokens/-cartões CI e os soft-tokens implementados inteiramente em software. Desde Janeiro de 2004 existe um padrão formal de cartão CI baseado no PKCS #15, chamado ISO/IEC 7816-15. Por esse motivo foi decidido não haver mais desenvolvimento nas partes do PKCS #156 relacionadas com os cartões CI por parte do RSA Laboratories. Entretanto, a ISO continuará seu desenvolvimento.

4 Smart Cards

Smart Cards são dispositivos com um microprocessador embutido em um cartão de plástico e tem a capacidade de guardar dados e executar comandos.
Os cartões de plástico começaram a ser utilizados em 1950 pela Diners Club para permitir que seus clientes pudessem fazer compras à credito. Mais tarde foram feitas melhorias para tornar mais segura a utilização dos cartões como tarjas magnéticas e impressão em alto relevo.
Na década de 70, "os avanços da micro-eletrônica permitiram a integração mecanismos de armazenamento de dados e lógica aritmética em um chip de silício medindo alguns milímetros quadrados"(RANKL; EFFING, 2001). Em 1968 os inventores alemães Dethloff e Grötrupp patentearam a idéia embutir um circuito integrado em um cartão de plástico e em 1970 Kunitaka Arimura do Japão patenteou independentemente o smart card. A área apresentou grandes progressos com as pesquisas de Roland Moreno e foi comercializada pela primeira vez pela empresa CII-HoneyWell-Bull. A utilização dos cartões iniciou-se na França e na Alemanha nos anos 80. Atualmente os smart cards são utilizados em todo o mundo em aplicações como telefonia, redes de computadores, aplicações financeiras e na área da saúde.

Um smart card é um computador portátil. Geralmente tem o tamanho e o formato de um cartão de crédito, um smart card possui um microprocessador ou um chip de memória embutido. Um smart card pode ser fisicamente à prova de alterações maliciosas. Por esse motivo ele pode ser utilizado como dispositivo de armazenamento seguro para todo tipo de informação como chaves secretas. Segurança, portabilidade e conveniência são motivos para se utilizar um smart card.

4.1 Classificação

Existem dois tipos de classificação básico em relação aos tipos de smart cards; pela presença ou não de microprocessador e pela necessidade de contato ou não com o leitor

de smart cards ou terminal.

4.1.1 Cartões de memória e cartões com microprocessador

Os cartões de memória possuem apenas a capacidade de armazenar dados e sua memória pode ser protegida, desprotegida ou com lógica segura. Os circuitos dos cartões com memória desprotegida podem ser acessados diretamente pelos contatos do cartão; nos cartões com memória protegida exigem um código de segurança para permitir acesso a seus circuitos; a segura além de possui a característica dos cartões com memória protegida, restringe como a leitura e a escrita pode ser feita por uma aplicação externa.

Cartões com microprocessador não permitem que os dados sejam diretamente acessados por uma aplicação externa; todos os dados enviados ou recebidos devem passar pelo processador do cartão.

No restante deste trabalho, exceto quanto informado, quando se falar em smart cards, estará se referindo aos cartões com microprocessadores, pois serão os cartões utilizados no projeto devido a sua capacidade de realizar a assinatura digital.

4.1.2 Cartões com contato e cartões sem contato

O padrão ISO 7816 parte 1 define um conjunto de 6 a 8 contatos para que os cartões de contato comuniquem-se com o mundo exterior. Através dos contatos é possível trocar informações e comandos entre o cartão e o computador host. Os cartões sem contato comunicam-se com o computador host através de transmissão sem fio.
A possibilidade de realizar comunicação sem fio permite ao cartão sem contato realizar transações rápidas e são mais fáceis de usar que os cartões com contato já que é necessário conectar os cartões de contato corretamente no leitor. A falta de contato, porém limita a capacidade do processador e pode não ser eficiente quando uma grande quantidade de informações precisa ser transmitida.
Neste trabalho, quando se refere a smart cards, se está falando dos cartões com contatos, pois serão os cartões utilizados no projeto devido a sua maior capacidade em processar grande quantidade de informações.

4.2 Arquitetura

Os smart cards são constituídos de um cartão plástico, pontos de contato, um processador embutido, vários tipos de memória e opcionalmente um co-processador matemático.

4.2.1 Contatos

A posição e o tamanho dos contatos é definido pelo padrão ISO 7816 parte 2. Oito contatos são definidos: Vcc (provê energia ao cartão), RST (sinal para resetar o microprocessador), CLK (fornece sinal de clock externo), GND (voltagem de referência), Vpp (utilizado em cartões antigos para fornecer voltagem de programação em dois níveis), o ponto de E/S (transferência de dados e comandos) e dois pontos RFU (reservados para o futuro).

4.2.2 Unidade Central de Processamento

Os processadores mais utilizados são os micro-controladores de 8 bits Motorola 6850 ou Intel 8051. Podem funcionar em até 40 MHz. Existem cartões mais avançados com processadores de até 32 bits (CHEN, 2000).

4.2.3 Memória

Os smart cards podem possuir os mais variados tipos de memória: ROM, PROM, Flash, EEPROM e RAM.

4.2.4 Co-processador

São utilizados em aplicações de segurança permitindo que o cartão realize operações aritméticas voltadas a criptografia como aritmética modular e cálculos com grandes números inteiros.

4.3 Transmissão de Dados em Smart Cards

A transmissão digital de dados entre um cartão e um terminal é realizada alternadamente entre as duas partes em modo de transmissão half-duplex. Apesar de não estar implementado para smart cards, o modo de transmissão full-duplex entre o cartão e o

terminal seria tecnicamente possível, já que o processadores dos cartões possuem duas portas de E/S, e dois dos oito contatos são reservados para aplicações futuras.

A comunicação com o cartão é sempre iniciada pelo terminal. Este é um relacionamento mestre-escravo, onde o terminal é o mestre e o cartão é o escravo.

Após a inserção de um cartão em um terminal, seus contatos são mecanicamente conectados aos contatos do terminal. Os contatos são habilitados eletricamente, e em seguida, o cartão executa um *power-on reset* e envia um *Answer to Reset (ATR)* ao terminal. O terminal analisa o ATR para verificação dos parâmetros relacionados ao cartão e a transmissão de dados para então enviar o primeiro comando. O cartão processa o comando e gera a resposta, que é enviada de volta para o terminal. Este processo de comando-resposta continua até o cartão ser desativado.

Antes do primeiro comando, o terminal pode enviar ao cartão um *Protocol Parameter Selection (PPS)*, para configurar parâmetros para o protocolo de transmissão do cartão.

A comunicação entre o cartão e o terminal é serial e assíncrona. É muito comum que a transmissão de dados seja controlada por software. Cartões mais novos possuem uma UART embutida para cuidar da comunicação pela interface serial, reduzindo o *overhead* de software na transmissão de dados(RANKL; EFFING, 2001).

4.4 ANSWER TO RESET (ATR)

Esta string de dados contém até 33 bytes, e possui vários parâmetros relacionados ao protocolo de transmisão e ao cartão. Normalmente a ATR possui poucos bytes, sendo muito raro ela possuir o número máximo de bytes permitidos(RANKL; EFFING, 2001). O ATR precisa ser curto principalmente em aplicações onde o cartão precisa estar pronto para o uso logo após a seqüência de ativação.

A string de dados e elementos de dados da ATR são definidos e descritos em detalhes no padrão ISO/IEC 7816-3. Os primeiros dois bytes, denominados TS e T0, definem vários parâmetros de transmissão fundamentais e indicam a presença dos bytes subseqüentes. Os caracteres de interface especificam parâmetros de transmissão especiais para o protocolo, que são importantes para as transmissões de dado subseqüentes. Os caracteres históricos (historical characters) descrevem a extensão das funções básicas do cartão. O caracter de verificação (check caracter), que é uma checksum dos bytes anteriores, pode opcionalmente ser enviado como último byte da ATR, dependendo do protocolo de transmissão utilizado.

4.4.1 Caracteres do ATR

O caracter inicial (initial character)

Este byte, denominado 'TS', especifica a convenção utilizada para todos os dados na ATR e subseqüentes processos de comunicação. Este primeiro byte é uma parte obrigatória da ATR e deve sempre ser enviado.

O caracter de formato (format character)

O segundo byte, T0, contém um campo de bits que indica que caracteres de interface o seguem na ATR. Também indica o número de caracteres históricos posteriores aos caracteres de interface. Como o TS, este byte deve estar presente em toda ATR.

Os caracteres de interface

Especificam todos os parâmetros de transmissão do protocolo em uso. Os caracteres são os bytes TAi, TBi, TCi e TDi. Eles são opcionais. Já que os valores padrão são definidos para todos os parâmetros do protocolo de transmissão, os caracteres de interface são geralmente desnecessários na ATR para um processo de comunicação normal.

Podem ser classificados como caracteres de interface globais e caracteres de interface específicos. Os globais especificam parâmetros básicos do protocolo de transmissão, como o divisor, que aplicam-se a todos os protocolos subseqüentes. Os específicos especificam os parâmetros para um protocolo de transmissão em particular. O 'work waiting time' para o T = 0 é um exemplo típico para este tipo de parâmetro.

Inicialmente, os caracteres globais de interface aplicam-se a todos os protocolos, entretanto, por razões históricas, muitos desses caracteres são relevantes apenas para o protocolo T = 0.

Os caracteres históricos

Por não terem sido definidos, por um longo período, por algum padrão, os caracteres históricos contém uma grande variedade de informações, dependendo do sistema operacional.

Muitas empresas utilizam os bytes disponíveis para identificar o sistema operacional e o número de versão da máscara ROM. Geralmente é codificada em ASCII, sendo fácil de interpretá-lo. Caracteres históricos não precisam estar presentes na ATR, portanto podem ser ommitidos. Em algumas situações, isto pode ser útil, já que torna a ATR menor e, portanto, mais rápida de ser enviada.

O caracter **check**

O último byte na ATR é o caracter **check** (TCK), que contém a checksum XOR dos bytes de T0 até o último byte anterior ao caracter **check**. A checksum pode ser utilizada em adição ao teste de paridade para verificar a correção da transmissão da ATR. Entretanto, apesar da aparente simplicidade da estrutura e computação desta checksum, existem muitas diferenças significativas entre os vários protocolos de transmissão.

4.5 Protocolos de Transmissão de Dados

Após o envio da ATR, o smart card espera pelo primeiro comando do terminal. O processo subseqüente sempre corresponde ao príncipio de meste-escravo, com o terminal no papel de mestre e o cartão como escravo. Em termos concretos, o terminal envia um comando ao cartão, e o último o executa e subseqüentemente retorna a resposta. Este processo de comandos e respostas nunca muda.

Existem várias formas da comunicação com um smart card ser estabelecida. Também existe uma grande quantidade de métodos diferentes para ressincronizar comunicações se algum distúrbio ocorrer. A implementação exata dos comandos, as respostas correspondentes e os procedimentos utilizados durante o evento de erros de transmissão são definidos na forma de protocolos de transmissão.

Um total de 15 protocolos de transmissão foram definidos em termos de suas funcionalidades básicas. Estes protocolos, que são chamados 'T =' mais um número seqüencial(RANKL; EFFING, 2001).

Dois destes protocolos são muito utilizados; o protocolo T = 0 e o protocolo T = 1. O primeiro é um protocolo assíncrono, half-duplex, orientado a byte, que foi especificado no padrão ISO/IEC 7816-3. O segundo é um protocolo assíncrono, half-duplex, orientado a bloco, que foi especificado no padrãp ISO/IEC 7816-3 Amd. 1.

4.6 APDUs

A unidade de troca de dados entre os cartões e os terminais é a *application protocol data unit* (APDU) (HANSMANN et al., 2000).

As informações e comandos são trocados em forma de application protocol data units (APDUs). A aplicação externa envia uma APDU com uma command APDU (C-APDU)

que é respondida pelo smart card com uma response APDU (R-APDU). O formatos das APDUs são especificados no padrão ISO 7816 parte 4. As APDUs que são compatíveis com este padrão, são independentes do protocolo de transmissão (HANSMANN et al., 2000).

4.6.1 Estrutura da C-APDU

É composta de um cabeçalho e um corpo. Estas duas seções da APDU de comando serão descritas à seguir.

Existem quatro elementos no cabeçalho: o byte de classe (`CLA`), o byte de instrução (`INS`) e dois bytes de parâmetros (`P1` e `P2`). O byte de classe é utilizado para identificar aplicações e seu conjunto de comandos. O byte de instrução, codifica o comando que o smart card deve realizar dentro da aplicação. Os bytes de parâmetro trazem mais informações sobre o comando referenciado no byte de instrução.

O corpo tem a função de especificar o tamanho dos dados enviados para o cartão (campo `Lc`), o tamanho esperado da resposta do cartão (campo `Le`) e contém os dados para os comandos enviados ao cartão.

Existem quatro casos possíveis de C-APDU:

- apenas o cabeçalho
- o cabeçalho e o campo `Le`
- o cabeçalho, o campo `Lc` e os dados
- o cabeçalho, o campo `Lc`, os dados e o campo `Le`

4.6.2 Estrutura da R-APDU

A APDU de resposta consiste em um corpo opcional e um *trailer* mandatório. O corpo é um campo de dados de tamanho definido no byte `Le` da C-APDU antecedente. O campo de dados do corpo pode ser zero, mesmo que o byte `Le` indique o contrário, caso o processamento do comando enviado resulte em um erro. Isto é indicado nas palavras de status `SW1` e `SW2` contidas no *trailer*.

O *trailer* deve ser sempre enviado pelo cartão. Os bytes `SW1` e `SW2` são conhecidos como 'código de retorno'.

4.7 Sistemas operacionais

O tipo mais comum sistema operacional (S.O.) em smart cards é o baseado em sistema de arquivos baseados no padrão 7816 parte 4. Nesse tipo de cartão, a separação entre aplicação e S.O. não é muito clara. Existem sistemas operacionais mais avançados que possuem separação mais clara entre as camadas e a capacidade de download de aplicações.

Dentre os S.O. mais avançados, os mais conhecidos são:

- Java Card - cartões que vem com uma Java Virtual Machine (JVM) com algumas restrições. Sua especificação é feita pela Sun;
- MULTOS - Especificado consórcio Maosco. Os programas são implementados na linguagens Multos Executable Language (MEL) e executam Application Abstract Machine (AAM);
- Smart Card for Windows - fornecido pela Microsoft. Os desenvolvedores criam um sistema operacional escolhendo os componentes desejados em um kit de desenvolvimento para Visual Basic ou C++. A máscara criada é carregada no cartão(HANSMANN et al., 2000).

4.8 Padrões e especificações

Dos diversos padrões e especificações para desenvolvimento de sistemas de smart cards, cito:

ISO 7816 padrão publicado pela International Organization for Standardization (ISO), "Identification cards - Integrated circuit cards with contacts", que define características físicas, localização dos contatos, protocolos de transmissão e sinais eletrônicos, identificadores de aplicação entre outros. Este padrão possui 9 partes;

ISO 7816-1 define as características físicas,

ISO 7816-2 define as dimensões e a localização dos contatos;

ISO 7816-3 define os sinais eletrônicos e os protocolos de transmissão;

ISO 7816-4 Comandos entre indústrias

GSM define padrões para um sistema telefônico terrestre móvel internacional;

EMV baseado no padrão ISO 7816, especifica características proprietárias para utilização dos smart cards em aplicações financeiras;

Open Platform define um ambiente integrado para desenvolvimento de operação de sistemas de smart cards multi-aplicações;

OpenCard Framework É um framework para desenvolvimento de aplicações dos computadores hosts fornecendo uma interface padrão de interação entre leitores e aplicações no cartão. Voltada para computadores em redes;

PC/SC Mesma filosofia do OpenCard Framework, porém voltada para sistemas em computadores pessoais.

5 Aplicação

Este capítulo tem como objetivo descrever a aplicação, JOpenSC, desenvolvida por este acadêmico.

JOpenSC é uma aplicação gráfica desenvolvida com o objetivo de facilitar e disseminar a utilização do OpenSC.

5.1 Requisitos

Para utilizar a JOpenSC são necessários:

- Linux
- OpenCT
- OpenSC
- Java Runtime Enviroment (JRE)
- Leitor de Smart Card
- Smart Card suportado pelo OpenSC

Para desenvolver a aplicação foi necessário instalar uma plataforma com capacidade interagir com smart cards. O sistema operacional escolhido foi o Linux Fedora Core 3, e os softwares OpenCT 0.6.2 e OpenSC 0.9.4 foram instalados para a comunicação com a tríade smart card, leitor e aplicação. O leitor utilizado foi um Towitoko CHIPDRIVE micro 130 v4.30 e o cartão um STARCOS SPK 2.3 com 32KB.

5.2 Instalação da Plataforma

Nesta seção, serão explicados as ações realizadas para a instalação da plataforma que permite a utilização da interface gráfica JOpenSC. A ordem de instalação dos componentes da plataforma é relativamente rígida, sendo recomendado instalá-los na seguinte ordem: Linux, OpenCT, OpenSC, JRE e JOpenSC. Para funcionar, o OpenSC necessita do OpenCT. O OpenSC e o JRE são pré-requisitos para o JOpenSC.

5.2.1 Linux

A Red Hat é uma empresa que produz uma das distribuições Linux mais populares. A empresa identificou dois tipos diferentes de usuários: o cliente corporativo e o "pioneiro". O primeiro deseja software estável com suporte e está disposto a pagar pelo suporte. Este software evolui lentamente, porque significa menos tempo em upgrades e menos treinamento. O segundo adora experimentar e utilizar as mais novas e avançadas funcionalidades do Linux. O pioneiro não se importa em fazer upgrades freqüentemente, não utiliza sistemas críticos e não quer pagar pelo suporte.

A empresa decidiu separar sua linha de produtos para satisfazer os dois tipos de clientes. A linha Red Hat Enterprise Linux foi criada para satisfazer os clientes corporativos e a linha Fedora Project para atender os desejos dos pioneiros.

Com a ajuda de uma interface gráfica, a instalação do Fedora é relativamente simples. Cada passo é explicado com explicações detalhadas. A instalação da distribuição Fedora é organizada de forma a satisfazer diferentes usuários, oferecendo as opções de servidor, estação de trabalho e desktop pessoal. Estas opções instalam softwares pré-selecionados, como ervidores de FTP e e-mail na opção de servidor e softwares de desenvolvimento e de escritório na versão de estação de trabalho. É possível ainda customizar a instalação, escolhendo os softwares que serão instalados.

Para instalar o sistema operacional e as aplicações são necessários os seguintes passos:

- Iniciar a instalação
- Selecionar as opções de instalação
- Criar partições
- Configurar o carregador de boot

- Configurar a rede
- Configurar o suporte a línguas
- Configurar o horário do sistema
- Definir a senha de root
- Selecionar os pacotes
- Instalar os pacotes
- Criar o disco de boot
- Completar a instalação

Geralmente a instalação do Linux ocorre sem problemas, entretanto erros e falhas de funcionamento podem ocorrer durante a instalação de um sistema operacional causando perda de dados. É recomendado que se faça backup de todos os dados do sistema e que se determine que o backup está livre de erros. Também é aconselhado a criação de discos de boot para realizar o boot do sistema mesmo que as informações de boot contidas no disco rígido estejam danificadas.

5.2.1.1 Red Hat Package Manager (RPM)

O RPM é uma ferramenta que facilita a instalação, desisntalação e a atualização de software e um sistema Linux Red Hat.

Um pacote RPM é uma arquivo que contém programas executáveis, scripts, documentação e outros arquivos necessários a uma aplicação ou unidade de software. Existe uma convenção para a nomeação de um RPM que permite a determinação do nome do pacote, da versão do software, o número do release do software, e a arquitetura do sistema para o qual a aplicação foi feita. Toma-se como exemplo um dos pacotes necessários para a utilização da JOpenSC: opensc-0.9.4-4.rpm. O nome do pacote é opensc, a versão do software é 0.9.4, o release do software é 4 e a arquitetura é i386.

Para instalar um pacote, efetue o login como root e digite o seguinte comando: "`rpm -ivh <pacote>`", onde em `<pacote>` se especifica o nome do arquivo que contém o pacote. É possível especificar múltiplos pacotes, sendo necessário separá-los por um espaço entre seus nomes.

As opções utilizadas para instalar o pacote foram as seguintes:

-i especifica que o RPM deve instalar o pacote ou pacotes informados como arqumentos.

-h especifica que o RPM deve imprimir marcas de hash (#) durante a instalação do pacote como uma indicação visual do progresso da instalação.

-v especifica que o RPM deve imprimir mensagens que resumem suas ações e seu progresso.

Podem ocorrer erros durante a instalação de um pacote especificado. O RPM pode informar que um pacote já está instalado, que um arquivo do pacote está em conflito com um arquivo de outro pacote, e que uma dependência falhou.

5.2.2 OpenCT

Para instalar o software OpenCT, foi utilizado o seguinte comando:

```
rpm -ivh openct-0.6.2-5.rpm
```

Se todas as dependências desta versão do OpenCT estiverem satisfeitas, a instalação será concluída com sucesso.

Este acadêmico encontrou dificuldades na instalação do software OpenCT. O arquivo de configuração deste software necessitava de uma diretiva de configuração no mínimo curiosa. Para conseguir o funcionamento do leitor USB da Towitoko foi necessário informar o dispositivo serial encontrava-se na porta USB. A diretiva incluída no arquivo */etc/openct.conf* foi:

```
device = serial:/dev/ttyUSB0;
```

Com a modificação efetuado, o arquivo ficou da seguinte forma:

Início do arquivo */etc/openct.conf* >>>

```
# Set debug level
debug   = 0;
#
# Enable hot plugging
hotplug = yes;
#
# Path to ifdhandler
ifdhandler = /usr/sbin/ifdhandler;
```

```
# Configure static, non-hotplug aware readers here
#
# For a list of drivers try command 'ifdhandler -i', please
# notice that not all drivers have serial device capability.

reader towitoko {
       driver = towitoko;
       device = serial:/dev/ttyUSB0;
};

#
# Hotplug IDs
driver  egate {
        ids = {
                usb:0973/0001,
        };
};
driver  etoken {
        ids = {
                usb:0529/050c,
                usb:0529/0514,
        };
};
driver  eutron {
        ids = {
                usb:073d/0005,
        };
};
driver  ikey2k {
        ids = {
                usb:04b9/1202,
        };
};
driver  ikey3k {
```

```
        ids = {
                usb:04b9/1300,
        };
};
driver  cardman {
        ids = {
                usb:076b/0596,
        };
};
```

Fim do arquivo */etc/openct.conf* <<<

5.2.3 OpenSC

Para instalar o software OpenCT, foi utilizado o seguinte comando:

```
rpm -ivh opensc-0.9.4-4.rpm
```

Se todas as dependências desta versão do OpenSC estiverem satisfeitas, a instalação será concluída com sucesso.

5.2.4 JRE

A linguagem de programação Java é uma linguagem de alto nível que pode ser definidade pelas seguintes características:

Simples é simples de aprender e simples de implementar. Sua sintaxe é muito parecida com C++.

Orientada à Objetos suporta objetos, herança, e polimorfismo. Este paradigma de programação ajuda a diminuir a lacuna entre a maneira que a máquina pensa e a maneira que um ser humano pensa.

Distribuída a conectividade entre componentes é uma característica básica da linguagem de programação Java desde a sua concepção.

Robusta a linguagem apresenta uma biblioteca GUI que rivaliza com as funcionalidades dos melhores sistemas operacionais, é uma solução de desenvolvimento Web

excepcional, possui um conjunto de bibliotecas para entrada e saída de dados, multithreading, manipulação avançada de imagens, e mais.

Segura existem diversas regras que definem o que pode e o que não pode ser feito. Vários componentes de linguagem operam em ambientes seguros diferentes.

Arquitetura neutra por ser executada em uma máquina virtual, a linguagem não depende do sistema operacional ou hardware.

Portável como as aplicações são compiladas em byte-code, que são instruções para a máquina virtual, é possível executar este byte-code em qualquer sistema operacional que possua uma máquina virtual compatível.

Interpretada Java não é compilada em código específico de uma máquina até que seja executada. Código interpretado geralmente executa mais lentamente do que um código compilado, entretanto Java é parcialmente compilada (byte-code).

Threads múltiplas permite que uma aplicação execute diferentes operações ao mesmo tempo.

Antes que seja possível compilar ou executar uma aplicação Java, é necessário instalar um ambiente Java. Existem vários compiladores Java, e o compilador padrão é distribuído no site da Sun. É possível fazer o download do Java 2 JRE, Standard Edition em http://java.sun.com/j2se/.

O JRE (Java Runtime Enviroment) é utilizado para executar programas java, entretanto não pode ser utilizado para compilar estas aplicações.

Existem duas versões de instalação para lInux: um arquivo binário com auto-extraçõa, e um arquivo binário contendo o pacotes RPM.

5.2.4.1 Arquivo Binário com Auto-Extração

Para instalar este arquivo são necessários os seguintes passos:

1. copiar o arquivo para o local de instalação do Java.
2. dar ao arquivo privilégios de execução com o comando "`chmod a+x filename.bin`".
3. executar o arquivo com o comando "`./filename.bin`".
4. seguir as instruções na tela

5.2.4.2 Arquivo Binário com Pacotes RPM

O arquivo RPM é muito fácil de instalr e necessita de pouca interação do usuário. Para instalá-lo, são necessários os seguintes passos:

1. fazer downaload do JRE
2. dar ao arquivo privilégios de execução com o comando "`chmod a+x filename.rpm.bin`". O resultado desta operação será a criação de um arquivo RPM.
3. executar o arquivo rpm com o comando: "`rpm -ivh filename.rpm`"

5.2.5 Leitor de Smart Cards

Apesar de existir um driver para o leitor Towitoko CHIPDRIVE micro 130, o mesmo não foi necessário. Apenas com a instalação dos softwares OpenCT e OpenCT, foi possível a comunicação entre a aplicação e um smart cart conectado ao leitor Towitoko. Portanto para instalar o leitor, foi necessário somente conectar o leitor à porta USB.

5.2.6 Smart Card

É necessário conectar o cartão ao leitor de smart cards. Como já foi dito anteriormente, o cartão utilizado foi o cartão um STARCOS SPK 2.3., entretando qualquer cartão suportado pela `OpenSC` pode ser utilizado.

5.3 OpenSC

OpenSC é um conjunto de aplicações e biblioteca para smart cards com suporte para cartões compatíveis com PKCS #15. As funcionalidades básicas funcionam com cartões compatíveis com o padrão ISO 7816-4, enquanto as funções criptográficas utilizando chaves privadas no smart card são possíveis apenas em cartões compatíveis com o padrão PKCS #15.
Sua biblioteca central é a biblioteca opensc (`libopensc`), que permite o acesso aos smart cards através do *middleware* PC/SC Lite. Além desta biblioteca existe ainda um módulo PKCS #11, um módulo PAM, dois *engines* para OpenSSL e outras ferramentas para testar e utilizar estas ferramentas e bibliotecas.

5.3.1 Camadas da biblioteca opensc

A biblioteca `libopensc` possui três camadas de código, sendo elas: `reader`, `card`, `pkcs15init`. Cada uma dessas camadas é implementada através de um ou mais *drivers*.

5.3.1.1 Camada reader

Existem vários software que permitem que uma aplicação comunique-se com os leitores de cartões e com os smart cards conectados a estes leitores. Na camada reader existe um módulo para cada aplicação deste tipo.
O módulo `pcsc reader module` permite a comunicação com o conhecido PC/SC Lite, mas existem alternativas suportadas pelo OpenSC, como por exemplo o OpenCT.

5.3.1.2 Camada card

Muitos smart cards não seguem seguem à risca o padrão ISO 7816, aceitando comandos e enviando respostas, diferentes dos padronizados. A camada `card` possui módulos para implementar os diferentes comandos que um determinado cartão possa ter.

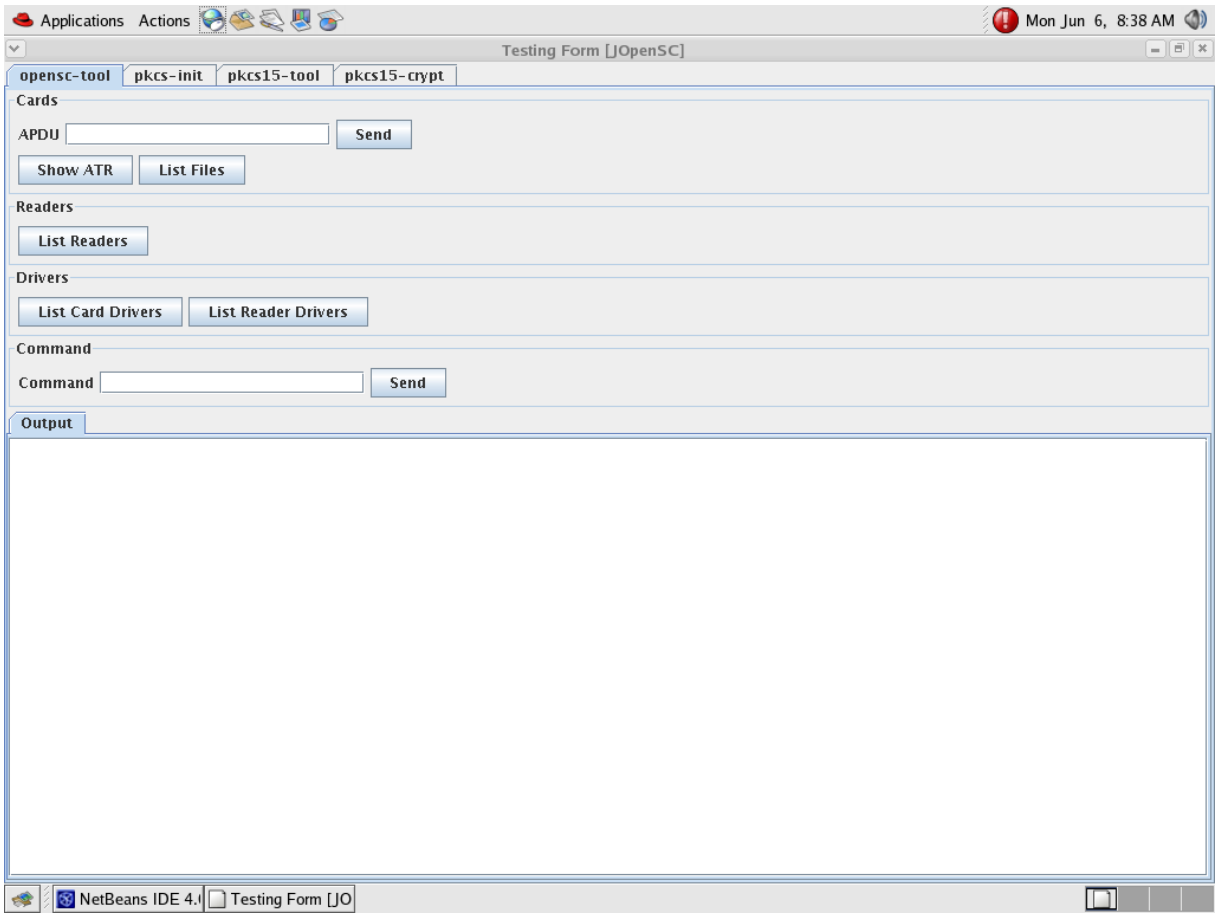
5.3.1.3 Camada pkcs15init

Esta camada implementa os comandos para, de forma segura, formatar cartões, criar diretórios e objetos, e definir permissões. Cada módulo nesta camada implementa os diferentes comandos e modelos de segurança para realizar aquelas operações.

5.4 JOpenSC

A aplicação oferece suporte gráfico a quatro aplicações de comando de linha disponíveis no OpenSC: `opensc-tool, pkcs15-init, pkcs15-tool e pkcs15-crypt`.

5.4.1 Aba opensc-tool



Interface gráfica do utilitário genérico para smart cards `opensc-tool`.

5.4.1.1 Painel Cards

No painel `Cards` é possível imprimir a ATR do cartão conectado ao leitor, enviar APDUs para o cartão e listar os arquivos armazenados no cartão de forma recursiva.

Para imprimir a ATR do cartão é necessário clicar o botão `Show ATR`. Ao executar esta ação, o comando executado será: "`opensc-tool --atr`".

Para enviar uma APDU para o cartão é necessário digitá-la na caixa de texto `APDU` e clicar no botão `Send`. Como no utilitário de comando de linha `opensc-tool` a APDU deve estar no formato AA:BB:CC:DD:FF(...). Ao executar estas ações, o comando executado será: "`opensc-tool --send-apdu <apdu>`", onde `<apdu>` é a APDU que se deseja enviar para o cartão.

Para listar os arquivos armazenados no cartão é necessário clicar o botão `List Files`. Ao executar esta ação, o comando executado será: "`opensc-tool --list-files`".

5.4.1.2 Painel Readers

Neste painel é possível listar todos os leitores de cartão configurados.

Para listar os leitores é necessário clicar o botão `List Readers`. Ao executar esta ação, o comando executado será: "`opensc-tool --list-readers`".

5.4.1.3 Painel Drivers

Duas ações são possíveis neste painel: listar todos os drivers de cartão instalados e listar todos os drivers de leitores de cartão instalados.

Para listar os drivers de cartão é necessário clicar o botão `List Card Drivers`. Ao executar esta ação, o comando executado será: "`opensc-tool --list-drivers`".

Para listar os drivers de leitores de cartão é necessário clicar o botão `List Reader Drivers`. Ao executar esta ação, o comando executado será: "`opensc-tool --list-rdrivers`".

5.4.1.4 Painel Command

Este painel tem o objetivo de remediar algumas limitações da aba `opensc-tool`. É possível digitar um comando na caixa de texto `Command` para utilizar opções do comando `opensc-tool` que não estão disponíveis na aba `opensc-tool`.

5.4.1.5 Janela Output

Nesta janela são impressos os resultado das ações executadas na aba `opensc-tool`.

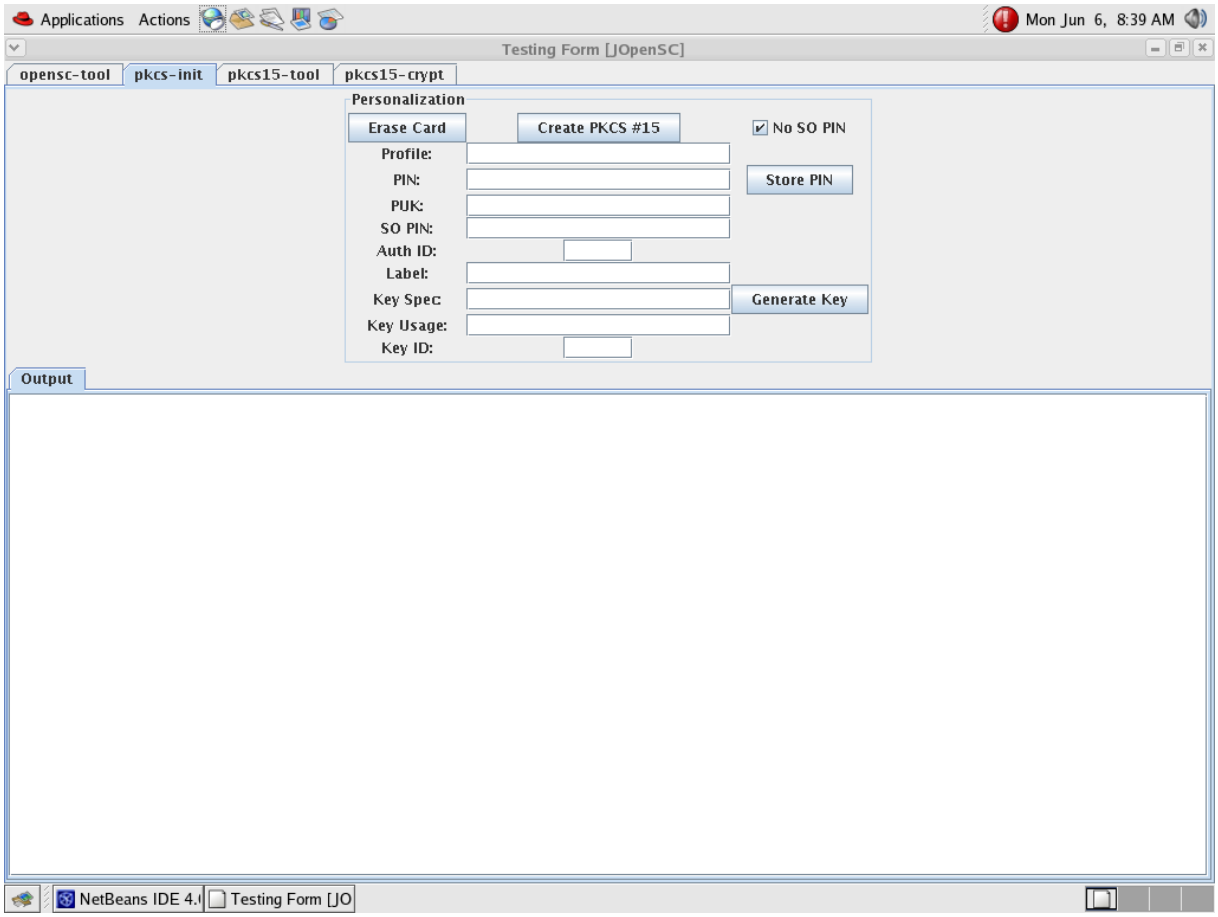
5.4.1.6 Limitações

São limitações conhecidas da aba `opensc-tool`:

- escolher qual o leitor será acessado durante a execução do comandos. Na ferramenta de commando de linha `opensc-tool` isto é feito com a opção `--reader <num>`, onde `<num>` é o leitor que se deseja acessar.
- escolher qual o driver de cartão será utilizado durante a execução de comandos. Na ferramenta de commando de linha `opensc-tool` isto é feito com a opção `--card-driver <driver>`, onde `<driver>` é o driver que se deseja utilizar.

- tornar as saídas das ações executadas mais descritivas. Na ferramenta de commando de linha `opensc-tool` isto é feito com a opção `--verbose`.

5.4.2 Aba pkcs15-init



Interface gráfica da ferramenta de personalização de smart cards `pkcs15-init`.

5.4.2.1 Painel Personalization

No painel `Personalization` é possível remover o conteúdo do cartão, criar a estrutura PKCS #15 inicial no cartão, criação de PINS para proteger os objetos armazenados no cartão e geração de chaves públicas e privadas.

Para remover o conteúdo do cartão é necessário clicar o botão `Erase Card`. Ao executar esta ação, o comando executado será: "`pkcs15-init --erase-card`".

Para criar a estrutura PKCS #15 no cartão é necessário clicar o botão `Create PKCS #15`. Ao executar esta ação, o comando executado será: "`pkcs15-init --create-pkcs15`".

Existem duas formas de criar PINs no cartão; a diferença entre as duas formas reside na quantidade de informações fornecidas pelo usuário. Na primeira, o usuário informa o PIN (caixa de texto `PIN`), o PUK (caixa de texto `PUK`) e o ID do PIN (caixa de texto `Auth ID`). Na segunda, o usuário informa o PIN, o PUK, o ID do PIN e um rótulo para o PIN (caixa de texto `Label`).

Ao executar uma destas ações, o comando executado será: "`pkcs15-init --store-pin --pin <pin> --puk <puk> --auth-id <auth-id> --label <label>`", onde `<pin>` é o PIN desejado pelo usuário, `<puk>` é o PUK desejado pelo usuário, `auth-id` é o ID do PIN e `<label>` é o rótulo do PIN. A opção do rótulo será incluída apenas se o mesmo for incluso pelo usuário.

Para criar as chaves no cartão, como na criação de PINs, existem duas formas, e a diferença está na inclusão ou não de um rótulo para as chaves. São necessárias as seguintes informações: a especificação da chave (caixa de texto `Key Spec`), a ID da chave (caixa de texto `Key ID`), o PIN (caixa de texto `PIN`, o ID do PIN caixa de texto `Auth ID`) e opcionalmente o rótulo (caixa de texto `Label`).

Ao executar uma destas ações, o comando executado será: "`pkcs15-init --generate-key <key-spec> --id <key-id> --key-usage <key-usage> --pin <pin> --auth-id <auth-id> --label <label>''`, onde `<key-spec>` descreve o algoritmo e o tamanho da chave a ser criada, `<key-id>` é o ID da chave a ser criada, `<key-usage>` é o propósito da chave a ser criada (ex.: autenticação), `<pin>` é o PIN que irá "proteger" a chave, `<auth-id>` é o ID do PIN e `<label>` é o rótulo do PIN. A opção do rótulo será incluída apenas se o mesmo for incluso pelo usuário.

5.4.2.2 Janela Output

Nesta janela são impressos os resultado das ações executadas na aba `pkcs15-init`.

5.4.2.3 Limitações

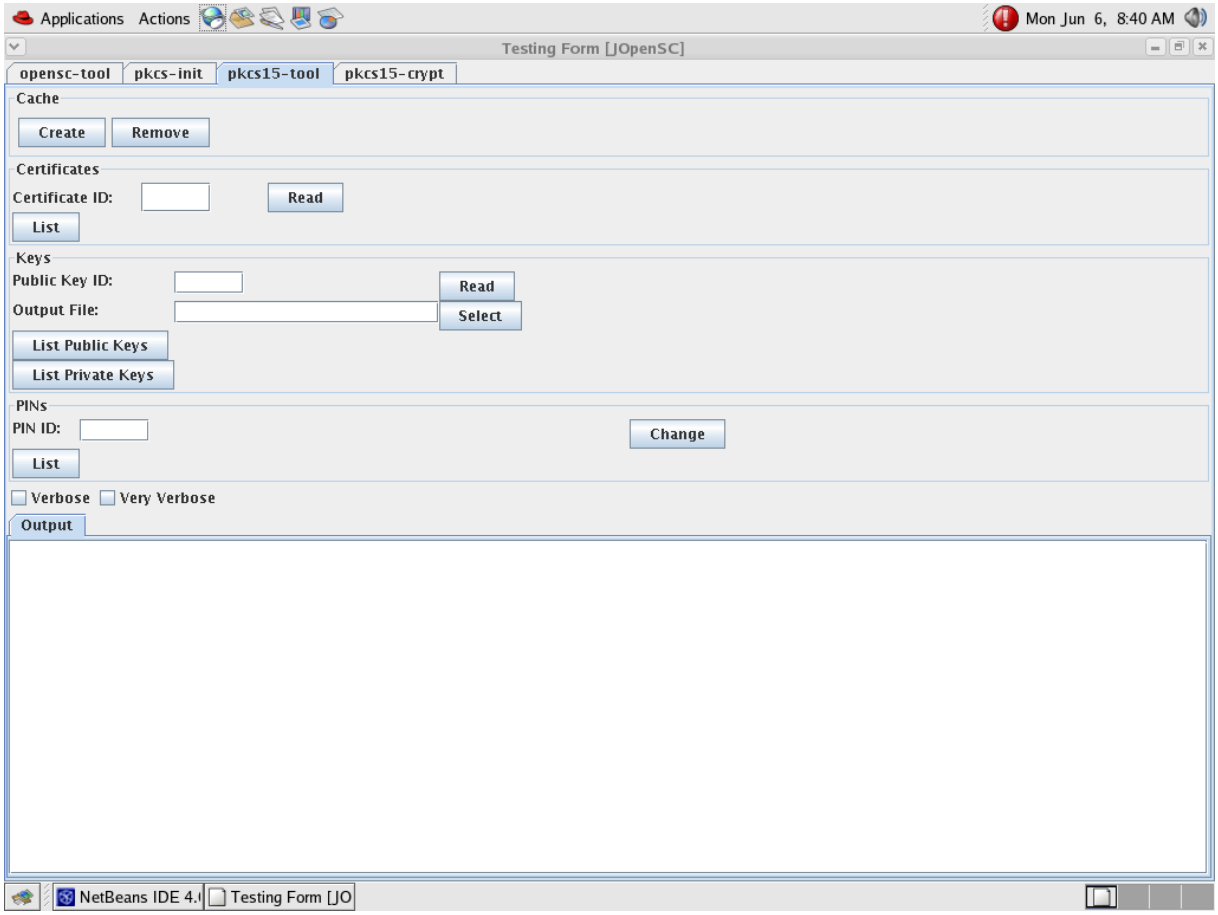
Por ser uma ferramenta de comando de linha com muitas opções, a `pkcs15-init` realiza diversas operações e possui variadas formas de realizá-la. A aba `pkcs15-init` da `JOpenSC` ainda não implementa todas as operações e suas variações. As limitações principais são:

- download de chave privada para o cartão. Na ferramenta de commando de linha

`pkcs15-init` isto é feito com a opção `--store-private-key`.

- download de chave pública para o cartão. Na ferramenta de commando de linha `pkcs15-init` isto é feito com a opção `--store-public-key`.
- download de certificado digital para o cartão. Na ferramenta de commando de linha `pkcs15-init` isto é feito com a opção `--store-certificate`.

5.4.3 Aba pkcs15-tool



Interface gráfica do utilitário `pkcs15-tool` para manipulação de estruturas PKCS #15 em cartões.

5.4.3.1 Painel Certificates

O painel `Certificates` permite a leitura de um certificado armazenado no cartão e a listagem de todos os certificados armazenados no token.

Para ler um certificado armazenado no cartão é necessário digitar o ID do certificado na caixa de texto `Certificate ID` e clicar no botão `Read`. Ao executar esta ação, o

comando executado será: "`pkcs15-tool --read-certificate <cert>`", onde `<cert>` é o ID do certificado que o usuário deseja ler.

Para listar os certificados armazenados no cartão é necessário clicar o botão `List`. Ao executar esta ação, o comando executado será: "`pkcs15-tool list-certificates`".

5.4.3.2 Painel Keys

A leitura, listagem e exportação para arquivo de chaves públicas, e listagem de chaves privadas, são as operações possíveis neste painel.

A visualização do conteúdo de uma chave pública é feita através da digitação do ID da chave na caixa de texto `Public Key ID` e do clique no botão `Read`. Ao executar esta ação, o comando executado será: "`pkcs15-tool --read-public-key <id>`", onde `<id>` é o ID da chave pública que o usuário deseja ler.

Para exportar uma chave pública são necessários os seguintes passos: escolher o arquivo onde a chave pública será escrita através do botão `Select`, digitar o ID da chave na caixa de texto `Public Key ID` e finalmente clicar o botão `Read`. Ao executar estas ações, o ocmando executado será: "`pkcs15-tool --read-public-key <id> --output <output-file>`", onde `<id>` é o ID da chave pública que o usuário deseja ler e `<output-file>` é o arquivo onde a chave pública será escrita.

Para listar as chaves públicas que estão armazenadas no cartão é necessário clicar o botão `List Public Keys`. Ao executar esta ação, o comando executado será: "`pkcs15-tool --list-public-keys`".

Para listar as chaves privadas que estão armazenadas no cartão é necessário clicar o botão `List Private Keys`. Ao executar esta ação, o comando executado será: "`pkcs15-tool --list-keys`".

5.4.3.3 Painel PINs

O painel `PINs` permite a alteração de PINs e a listagem de todos PINs armazenados no cartão.

Para alterar o valor de um PIN, é necessário digitar o ID do PIN que o usuário deseja alterar e depois clicar o botão `Change`. Ao executar estas ações, o comando executado será: "`pkcs15-tool --change-pin --pin-id <pin>`", onde `<pin>` é o ID do PIN que deseja-se alterar.

A listagem do PINs armazenados no cartão é feita com o clique do botão `List`. Ao executar esta ação, o comando executado será: "`pkcs15-tool --list-pins`".

5.4.3.4 Janela Output

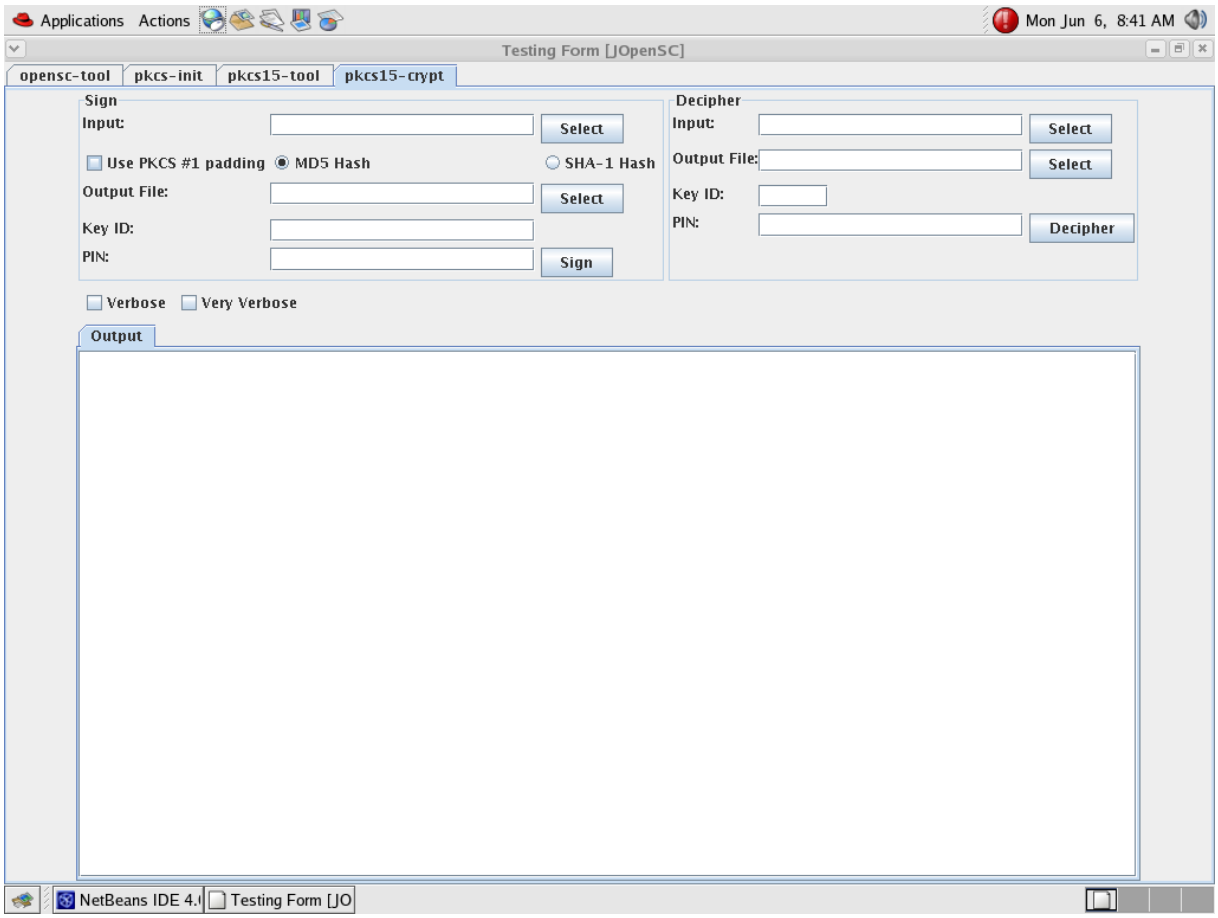
Nesta janela são impressos os resultado das ações executadas na aba `pkcs15-tool`.

5.4.3.5 Limitações

As limitações conhecidas da aba `pkcs15-tool` são:

- criação e remoção de cache dos dados do cartão. Na ferramenta de comando de linha `pkcs15-tool` isto é feito com a opção `--learn-card` para criar um cache e `--no-cache` para desabilitar o cache.
- alterar o valor de um PIN. Para realizar esta operação, é preciso que a `JOpenSC` realize operações interativas com o usuário e com o utilitário `opensc-tool`, e atualmente a `JOpenSC` não realiza nenhuma operação interativa deste utilitário.
- tornar as saídas das ações executadas mais descritivas. Na ferramenta de commando de linha `opensc-tool` isto é feito com a opção `--verbose`.

5.4.4 Aba pkcs15-crypt



Interface gráfica do utilitário `pkcs15-crypt` que permite a realização de operações criptográficas em cartões PKCS #15.

5.4.4.1 Painel Sign

O painel `Sign` permite a realização de assinaturas digitais de dados lidos de um arquivo.

Para assinar digitalmente determinado dado faz-se necessário a realização das seguintes operações: escolher o arquivo que contém o dado a ser assinado clicando o botão `Select` que fica ao lado da caixa de texto `Input`, selecionar se o dado contido no arquivo é um hash MD5 ou um hash SHA-1, selecionar o arquivo onde deve ser escrito o resultado da operação de assinatura clicando no botão `Select` que fica ao lado do campo de texto `Output File`, informar no campo `Key ID` o ID da chave privada que será utilizada na assinatura e o PIN que "protege" esta chave privada no campo `PIN`. Após esta seleção, clicar no botão `Sign`. Ao executar esta ação, o comando executado será: "`pkcs15-crypt --input <input-file> <hash-algorithm> --output`

<output-file> --key <key-id> --pin <pin>", onde <input-file> é o arquivo que contém o hash que será assinado, <hash-algorithm> é o tipo de hash contido no arquivo de entrada, <output-file> é o arquivo onde será gravada a assinatura digital, <key-id> é o ID da chave privada que será utilizada na realização da assinatura digital e <pin> é o PIN que permitirá a utilização da chave privada na assinatura.

5.4.4.2 Painel Decipher

O painel Decipher permite utilizar um smart card para decifrar o conteúdo de um arquivo.

Para decifrar os dados de um arquivo, o usuário deve realizar os seguintes passos: escolher o arquivo que contém o dado a ser decifrado clicando o botão Select que fica ao lado da caixa de texto Input, selecionar o arquivo onde deve ser escrito o resultado da operação clicando no botão Select que fica ao lado do campo de texto Output File, informar no campo Key ID o ID da chave que será utilizada na operação e o PIN que "protege" esta chave no campo PIN. Após a realização destes passos, clicar no botão Decipher. Ao executar esta ação, o comando executado será: "pkcs15-crypt --input <input-file> --output <output-file> --key <key-id> --pin <pin>", onde <input-file> é o arquivo que contém os dados a serem decifrados, <output-file> é o arquivo onde serão gravados os dados decifrados, <key-id> é o ID da chave que será utilizada na operação e <pin> é o PIN que "protege" a chave.

5.4.4.3 Janela Output

Nesta janela são impressos os resultado das ações executadas na aba pkcs15-crypt.

5.4.4.4 Limitações

A limitação conhecida da aba pkcs15-crypt é:

- tornar as saídas das ações executadas mais descritivas. Na ferramenta de commando de linha opensc-tool isto é feito com a opção --verbose.

6 Conclusão

Este capítulo tem o objetivo de relatar os resultados obtidos por este acadêmico em confronto com os resultados que buscava-se alcançar.

A instalação do sistema linux para assinatura digital com smart cards foi realizada com sucesso, já que a instalação do sistema operacional é atualmente, intuitiva, e estão disponíveis RPMs dos softwares necessários para o trabalho, como OpenCT e OpenSC. Houve dificuldade, já relatada anteriormente, no momento da configuração do leitor de smart cards.

Foram avaliadas algumas soluções com Java Cards para definir a possibilidade de executar assinaturas digitais com estes cartões. Esta alternativa é viável, entretanto possui um curva de aprendizado acentuada. Muitos do maiores fabricantes de smart cards, como Gemplus e Axalto, possuem ferramentas que ajudam no desenvolvimento das soluções Java Card. Tais ferramentas apresentam um preço elevado, entretanto existe um plugin para a IDE Eclipse, disponibizado pela IBM, que possue um preço acessível. Além deste plugin, chamado JCOP Tools, existe a família de cartões JCOP que, em alguns de seus membros, possue inclusive suporte à biometria.

Para implementar a aplicação foram avaliadas algumas bibliotecas para interface gráfica. Forma elas: GTK+ e GNOME, Mono e Java. A opção GTK+ e GNOME é uma solução viável. Necessita de conhecimento em programação na linguagem C e possui algumas ferramentas que facilitam o desenvolvimento da interface, porém funciona apenas em plataformas UNIX e GNU/Linux (WARKUS, 2004), e por este motivo foi descartada. A segunda alternativa, Mono (http://www.mono-project.com/), é uma implementação *open source* da framework .NET da Microsoft. É portável para diversas plataformas e possui uma boa ferramenta de desenvolvimento,a IDE MonoDevelop (http://www.monodevelop.com). Esta opção não foi utilizada por requerer deste acadêmico o aprendizado de uma nova linguagem de programação. A terceira opção, Java, é a mundialmente consagrada linguagem de programação. Conhecida pela sua portabili-

dade, possui excelentes ferramentas de desenvolvimento como as IDEs Eclipse e NetBeans. Esta foi a alternativa escolhida por este acadêmico.

Após a escolha e instalação da plataforma de desenvolvimento, foi necessário conhecer o OpenSC. Após a definição das funções do OpenSC que seriam implementadas, iniciou-se o desenvolvimento da interface gráfica. A portabilidade almejada não foi alcançada por motivos já mencionados no capítulo anterior.

De maneira geral, os objetivos propostos por este acadêmico foram atingidos. Como resultado do seu trabalho, obteve-se uma interface gráfica para o OpenSC, que permitirá a disseminação de sua utilização, contribuindo para a comunidade de software livre. A interface gráfica implementada facilita a utilização do OpenSC por permitir uma utilização mais intuitiva, sem a necessidade de digitação de comandos, e por acabar com a necessidade de decorar os comandos e suas respectivas opções.

Este acadêmico acredita que este trabalho pode ser melhor desenvolvido em trabalhos futuros. É necessária melhoria da aplicação de forma que a mesma consiga comportar as atualizações constantes na ferramenta OpenSC. Existem funções no OpenSC que ainda não foram implementadas na aplicação JOpenSC, e muitas outras que virão. Sugere-se ainda que se implemente uma solução baseada totalmente em um cartão Java Card, sem a necessidade de utilização da OpenSC.

Referências

STALLINGS, W. *Cryptography and Network Security*: principles and practice. 2nd. ed. Estados Unidos: Prentice-Hall, 1999.

RANKL, W.; EFFING, W. *Smart Card Handbook*. 2nd. ed. Inglaterra: John Wiley & Sons, 2001.

CHEN, Z. *Java Card technologhy for smart cards*: architecture and programmer's guide. 1st. ed. Estados Unidos: Addison-Wesley Pub Co, 2000.

HANSMANN, U. et al. *Smart Card development using Java*. Alemanha: Springer, 2000.

WARKUS, M. *The Officical GNOME 2 Developer's Guide*. Estados Unidos: No Starch Press, 2004.

Anexo

JOpenSC: Uma interface gráfica para OpenSC

Eduardo Ruhland

Laboratório de Segurança em Computação
Universidade Federal de Santa Catarina
Florianópolis, SC, 88049-970, Caixa Postal 476, Brasil

Resumo

Este trabalho tem o objetivo de relatar as atividades realizado por este acadêmico na implementação de uma interface gráfica para a biblioteca e conjunto de aplicações `OpenSC`. Alguns conceitos importantes, como criptografia, padrões PKCS, e smart cards, são brevemente descritos para melhor compreensão do trabalho. Os passos realizados para a conclusão deste trabalho e os problemas encontrados durante a sua execução são relatados para a contribuição com trabalhos futuros.

Abstract

The purpose of this work is to describe the activities done by this student during the implementation of a graphical interface for the smart card library and aplications called `OpenSC`. Some important concepts, like cryptography, PKCS standards, and smart cards, are described for better understanding of this work. The steps taken to conclude this work and the problems that appeared during its execution were reported for contribution with future applications.

Palavras-chave: OpenSC, JOpenSC, smart cards.

1 Introdução

Cada vez mais as pessoas e empresas trocam e armazenam informações através de computadores. Esta prática, possivelmente acarretará na substituição do armazenamento e transmissão de informações através do papel para o meio eletrônico.

Neste contexto, mecanismos de proteção de dados e garantia de sua autenticidade fazem-se necessários. Um desses mecanismos é a assinatura digital. Ela permite tanto autenticar o emissor de uma mensagem, isto é, que quem a assinou é o seu emissor, como permite verificar a integridade da mesma.

Na assinatura digital, que é uma seqüência de bits concatenada à mensagem, utiliza-se a tecnologia de chaves públicas e de função resumo.

No processo de assinatura em si cifra-se o resumo da mensagem com a chave privada do autor, que é de conhecimento apenas de seu proprietário. Entretanto, como esta chave é um arquivo, pode ser roubada, permitindo que uma pessoa mal intencionada utilize a chave roubada para assinar documentos em nome de outra pessoa. Umas das formas de minimizar esse problema é a utilização de smart cards.

Os Smart Cards vêm obtendo grande aceitação na utilização de mecanismos de assinatura digital devido a sua segurança, portabilidade e facilidade de uso. Os cartões guardam a chave privada do proprietário em sua memória, tornando-a inacessível diretamente.

Soluções nesta direção estão sendo disponibilizadas tanto por empresas privadas como por comunidades de software livre. Uma das iniciativas em software livre é a OpenSC. Porém, esta ferramenta foi disponibilizada somente com interface de comando de linha. O presente projeto visa implementar uma interface gráfica para esta ferramenta.

2 Criptografia

Criptografia é a arte ou ciência de técnicas matemáticas relacionadas à aspectos da segurança de dados como:

confidencialidade não permitir que pessoas não autorizadas tenham acesso ao conteúdo de determinada informação.

integridade de dados detectar alteração não autorizada de dados.

autenticação identificar entidades ou origem de dados.

não-repúdio evitar que uma entidade negue ações ou obrigações anteriores.

Cripto-análise é o estudo de métodos matemáticos que são utilizados na tentativa de quebrar técnicas criptográficas. Criptologia é o estudo da criptografia e da cripto-análise.

Alguns métodos criptográficos contam com o sigilo dos algoritmos de cifra; estes algoritmos são apenas de interesse histórico e não são adequados paras as necessidades do mundo atual. Ao invés de sigilo do método, todos os algoritmos modernos baseiam sua segurança no uso da chave; uma mensagem pode ser decifrada apenas se a chave utilizada para decifrar combina com a chave utilizada para cifrar.

3 PKCS

Os Padrões de Criptografia de Chave Pública são especificações que têm o objetivo de acelerar a distribuição da criptografia de chave pública. Os primeiros documentos PKCS foram publicados em 1991, como resultado de reuniões de um pequeno grupo de pioneiros na utilização da tecnologia de chave pública e hoje são produzidos pelo RSA Laboratories em cooperação com desenvolvedores do mundo inteiro.

O RSA Laboratories tem a palavra final em cada documento. Entretanto, quando um documento é aceito como base para um padrão formal, a RSA Laboratories abre mão da propriedade sobre o documento, permitindo o processo de desenvolvimento de padrões abertos.

4 Smart Cards

Smart Cards são dispositivos com um microprocessador embutido em um cartão de plástico e tem a capacidade de guardar dados e executar comandos.

Um smart card é um computador portátil. Geralmente tem o tamanho e o formato de um cartão de crédito, um smart card possui um microprocessador ou um chip de memória embutido. Um smart card pode ser fisicamente à prova de alterações maliciosas. Por esse motivo ele pode ser utilizado como dispositivo de armazenamento seguro para todo tipo de informação como chaves secretas. Segurança, portabilidade e conveniência são motivos para se utilizar um smart card.

Existem dois tipos de classificação básico em relação aos tipos de smart cards; pela presença ou não de microprocessador e pela necessidade de contato ou não com o leitor de smart cards ou terminal.

Os smart cards são constituídos de um cartão plástico, pontos de contato, um processador embutido, vários tipos de memória e opcionalmente um co-processador matemático.

A transmissão digital de dados entre um cartão e um terminal é realizada alternadamente entre as duas partes em modo de transmissão half-duplex. Apesar de não estar implementado para smart cards, o modo de transmissão full-duplex entre o cartão e o terminal seria tecnicamente possível, já que o processadores dos cartões possuem duas portas de E/S, e dois dos oito contatos são reservados para aplicações futuras.

A comunicação com o cartão é sempre iniciada pelo terminal. Este é um relacionamento mestre-escravo, onde o terminal é o mestre e o cartão é o escravo. Esta comunicação é serial e assíncrona. É muito comum que a transmissão de dados seja controlada por software. Cartões mais novos possuem uma UART embutida para cuidar da comunicação pela interface serial, reduzindo o *overhead* de software na transmissão de dados [Rankl e Effing 2001].

O tipo mais comum de sistema operacional (S.O.) em smart cards é o baseado em sistema de arquivos baseados no padrão 7816 parte 4. Nesse tipo de cartão, a separação entre aplicação e S.O. não é muito clara. Existem sistemas operacionais mais avançados que possuem separação mais clara entre as camadas e a capacidade de download de aplicações.

5 Aplicação

JOpenSC é uma aplicação gráfica desenvolvida com o objetivo de facilitar e disseminar a utilização do OpenSC. OpenSC é um conjunto de aplicações e biblioteca para smart cards com suporte para cartões compatíveis com PKCS #15.

Para desenvolver a aplicação foi necessário instalar uma plataforma com capacidade interagir com smart cards. O sistema operacional escolhido foi o Linux Fedora Core 3, e os softwares OpenCT 0.6.2 e OpenSC 0.9.4 foram instalados para a comunicação com a tríade smart card, leitor e aplicação. O leitor utilizado foi um Towitoko CHIPDRIVE micro 130 v4.30 e o cartão um STARCOS SPK 2.3 com 32KB.

A aplicação oferece suporte gráfico a quatro aplicações de comando de linha disponíveis no OpenSC: `opensc-tool, pkcs15-init, pkcs15-tool e pkcs15-crypt`.

6 Conclusão

Foram avaliadas algumas soluções com Java Cards para definir a possibilidade de executar assinaturas digitais com estes cartões. Esta alternativa é viável, entretanto possui um curva de aprendizado acentuada. Muitos do maiores fabricantes de smart cards, como Gemplus e Axalto, possuem ferramentas que ajudam no desenvolvimento das soluções Java Card. Tais ferramentas apresentam um preço elevado, entretanto existe um plugin para a IDE Eclipse, disponibizado pela IBM, que possue um preço acessível. Além deste plugin, chamado JCOP Tools, existe a família de cartões JCOP que, em alguns de seus membros, possue inclusive suporte à biometria.

De maneira geral, os objetivos propostos por este acadêmico foram atingidos. Como resultado do seu trabalho, obteve-se uma interface gráfica para o OpenSC, que permitirá a disseminação de sua utilização, contribuindo para a comunidade de software livre. A interface gráfica implementada facilita a utilização do OpenSC por permitir uma utilização mais intuitiva, sem a necessidade de digitação de comandos, e por acabar com a necessidade de decorar os comandos e suas respectivas opções.

Este acadêmico acredita que este trabalho pode ser melhor desenvolvido em trabalhos futuros. É necessária melhoria da aplicação de forma que a mesma consiga comportar as atualizações constantes na ferramenta OpenSC. Existem funções no OpenSC que ainda não foram implementadas na aplicação JOpenSC, e muitas outras que virão. Sugere-se ainda que se implemente uma solução baseada totalmente em um cartão Java Card, sem a necessidade de utilização da OpenSC.

Referências

[Chen 2000]CHEN, Z. *Java Card technologhy for smart cards*: architecture and programmer's guide. 1st. ed. Estados Unidos: Addison-Wesley Pub Co, 2000.

[Dreyfus 1998]DREYFUS, H. *Smart Cards*: a guide to building and managing smart card applications. Estados Unidos: John Wiley & Sons, 1998.

[Feghhi, Feghhi e Williams 1999]FEGHHI, J.; FEGHHI, J.; WILLIAMS, P. *Digital Certificates*: applied internet security. Estados Unidos: Addison-Wesley Pub Co, 1999.

[Hansmann et al. 2000]HANSMANN, U. et al. *Smart Card development using Java*. Alemanha: Springer, 2000.

[Hendry 2001]HENDRY, M. *Smart Card security and applications*. 2nd. ed. Estados Unidos: Artech House Publishers, 2001.

[Rankl e Effing 2001]RANKL, W.; EFFING, W. *Smart Card Handbook*. 2nd. ed. Inglaterra: John Wiley & Sons, 2001.

[Stallings 1999]STALLINGS, W. *Cryptography and Network Security*: principles and practice. 2nd. ed. Estados Unidos: Prentice-Hall, 1999.

[Warkus 2004]WARKUS, M. *The Officical GNOME 2 Developer's Guide*. Estados Unidos: No Starch Press, 2004.