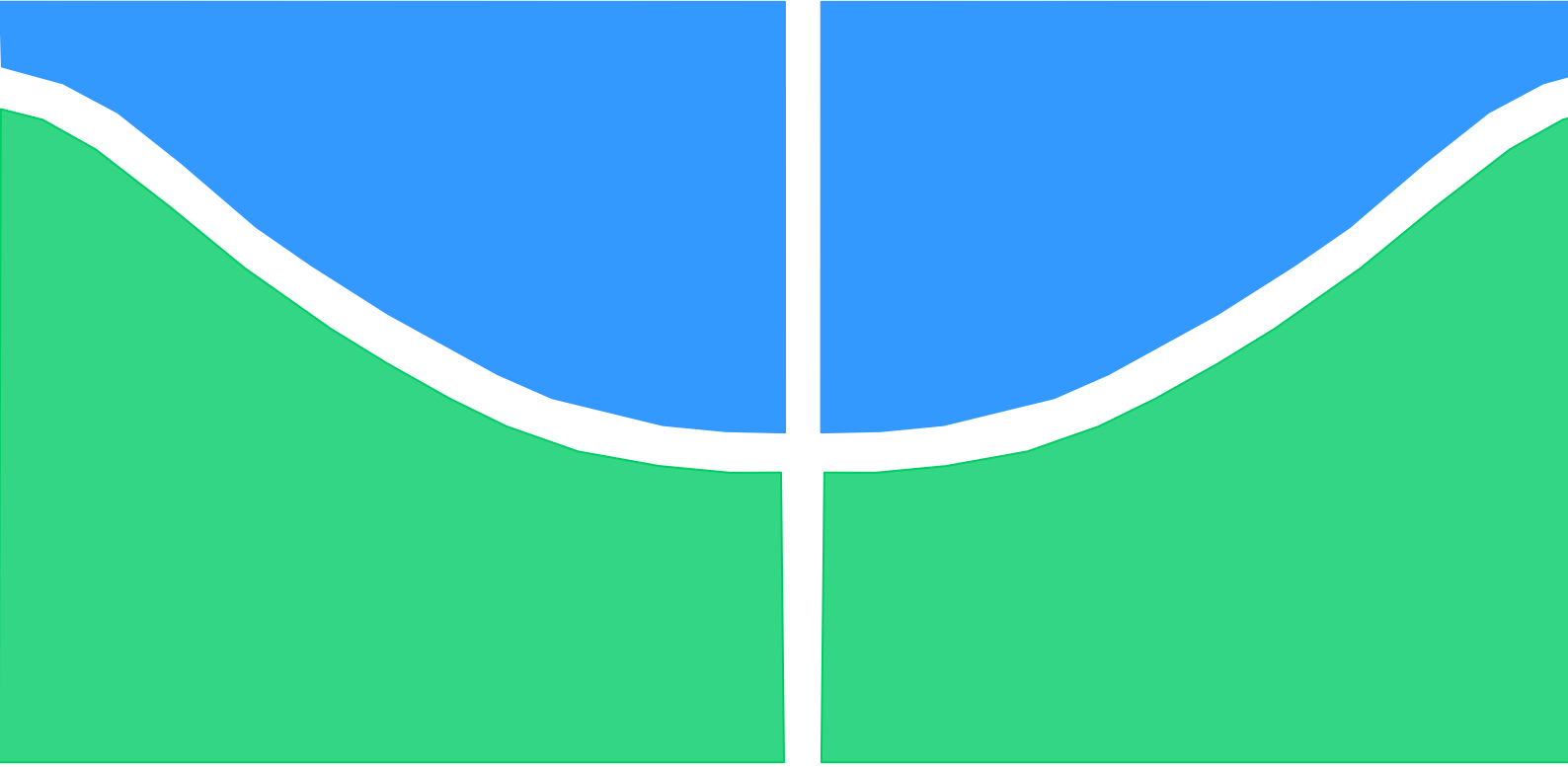

APLICAÇÃO DE SEGURANÇA ELETRÔNICA COM JAVA CARDS: O CASO DE UM PROTOCOLO PARA REGISTRO *ONLINE* E SEM ANONIMATO EM CARTÕES CRIPTOGRAFICAMENTE INTELIGENTES

JOSE MARIA LEOCADIO

DISSERTAÇÃO DE MESTRADO EM ENGENHARIA ELÉTRICA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

UNIVERSIDADE DE BRASILIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASILIA

FACULDADE DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

APLICAÇÃO DE SEGURANÇA ELETRÔNICA COM JAVA CARDS: O CASO DE UM PROTOCOLO PARA REGISTRO *ONLINE* E SEM ANONIMATO EM CARTÕES CRIPTOGRAFICAMENTE INTELIGENTES

JOSE MARIA LEOCADIO

ORIENTADOR: ANDERSON CLAYTON ALVES NASCIMENTO

DISSERTAÇÃO DE MESTRADO EM ENGENHARIA ELÉTRICA

PUBLICAÇÃO: PPGENE.TD-383/09

BRASÍLIA/DF: MARÇO – 2009

FICHA CATALOGRÁFICA

LEOCADIO, JOSE MARIA.

Aplicação de segurança eletrônica com Java Cards: o caso de um protocolo para registro *online* e sem anonimato em cartões criptograficamente inteligentes [Distrito Federal] 2009.

122 p., 210 x 297 mm (ENE/FT/UnB, Mestre, Dissertação de Mestrado – Universidade de Brasília. Faculdade de Tecnologia.

Departamento de Engenharia Elétrica

1. Cartões Inteligentes 2. Java Card

3. Pagamento eletrônico 4. Smart Card

I. ENE/FT/UnB II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

LEOCADIO, J. M. (2009). Aplicação de segurança eletrônica com Java Cards: o caso de um protocolo para registro *online* e sem anonimato em cartões criptograficamente inteligentes. Dissertação de Mestrado em Engenharia Elétrica, Publicação PPGENE.TD-383/09. Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 122p.

CESSÃO DE DIREITOS

AUTOR: José Maria Leocádio.

TÍTULO: Aplicação de segurança eletrônica com Java Cards: o caso de um protocolo para registro *online* e sem anonimato em cartões criptograficamente inteligentes.

GRAU: Mestre ANO: 2009

AGRADECIMENTOS

A presente dissertação de mestrado foi feita sob a supervisão do professor Anderson Clayton Alves Nascimento. Gostaria de agradecê-lo pelo constante apoio, direcionamento e ensinamentos na área de matemática e criptografia que foram essenciais para a conclusão do projeto de pesquisa que deu origem à dissertação escrita.

Agradeço também aos demais membros da banca de avaliação por terem despendido seu tempo nesta avaliação e pelas importantes contribuições para a minha revisão, professores Ricardo Staciarini Puttini e Jacir Luiz Bordim.

Durante o último ano, este projeto de pesquisa também foi apoiado pelo pesquisador do Departamento de Engenharia de Redes, Fabio Kaiser Rauber, que muito contribuiu com sua visão de engenharia de produto para a criação e implementação do novo protocolo proposto.

Também gostaria de agradecer aos professores Francisco Assis de Oliveira Nascimento e Geovany Araújo Borges, no Grupo de Processamento Digital de Sinais do Departamento de Engenharia Elétrica da UnB, pelo apoio na criação do projeto de pesquisa e cessão dos laboratórios de eletrônica.

Por fim e mais importante, gostaria de agradecer à minha família pelo apoio, carinho e paciência demonstrados durante a realização desta pós-graduação *stricto sensu*. Esta dissertação é dedicada a Maria Iolanda e Cecilia.

RESUMO

APLICAÇÃO DE SEGURANÇA ELETRÔNICA COM JAVA CARDS: O CASO DE UM PROTOCOLO PARA REGISTRO *ONLINE* E SEM ANONIMATO EM CARTÕES CRIPTOGRAFICAMENTE INTELIGENTES

Autor: José Maria Leocádio

Orientador: Anderson Clayton Alves Nascimento

Programa de Pós-graduação em Engenharia Elétrica

Brasília, mês de março (2009)

Esta dissertação propõe e implementa um novo protocolo de registro para pagamento eletrônico online e sem anonimato o qual é uma variante do protocolo cSET. Diferentemente do cSET, este novo protocolo proposto não exige nenhum tipo de confiança na leitora de cartões inteligentes em uso, possibilitando uma maior abrangência de aplicações.

ABSTRACT

APPLICATION OF E-SECURITY WITH JAVA CARDS: THE CASE OF AN ONLINE AND NON ANONYMOUS REGISTER PROTOCOL FOR CRYPTOGRAPHIC SMART CARDS

Author: José Maria Leocádio

Supervisor: Anderson Clayton Alves Nascimento

Programa de Pós-graduação em Engenharia Elétrica

Brasília, month of march (2009)

This dissertation describes a novel protocol for online electronic transaction based on smart cards and its implementation. This protocol is based on the cSET protocol but, differently from cSET, in this novel protocol no degree of trust is needed from the smart card reader, this increases the range of applicability.

Sumário

1. Introdução 1
2. Preliminares 3
   2.2. Algumas Definições Básicas de Segurança eletrônica 3
   2.3. Conceitos de Criptografia 5
   2.4. Revisão da Teoria dos Números 7
   2.5. Criptografia simétrica ou de chave secreta 10
      2.5.1 Cifras de Bloco e Cifras de Fluxo 10
      2.5.2 Funções de *hash* 13
      2.5.3 Criptografia assimétrica ou de chaves públicas 13
      2.5.4 Criptosistema RSA 14
      2.5.5 Assinatura digital 17
      2.5.6 Infraestrutura para criptografia de chaves públicas 18
      2.5.7 Certificado digital 20
      2.5.8 Identificação de usuário por biometria 22
   2.6. Avaliação da segurança eletrônica de um esquema criptográfico 23
   2.7. Modelo Adversarial 25
3. Cartões criptograficamente inteligentes 26
   3.2. Áreas de aplicação 27
      3.1.1 Governo 27
      3.1.2 Finanças 28
      3.1.3 Telecomunicações 29
   3.3. Arquitetura de cartões inteligentes 30
   3.4. Padrões para cartões inteligentes 38
   3.5. Criação de um applet Java Card 40
   3.6. Cliente Java 42
4. Protocolos para pagamento eletrônico 43
   4.2. Estado da arte 45
      4.2.1 Anonimato e não rastreabilidade 45
      4.2.2 Possibilidade de uso de leitora de cartões maliciosa 46
   4.3. Projeto de protocolos confiáveis 47
   4.4. Protocolos para pagamento eletrônico com cartões inteligentes 53
      4.4.1 Pagamento *offline*, não rastreável e com adição/remoção de anonimato 54

JDK – Java Development Kit

JRE – Java Runtime Environment

JVM – Java Virtual Machine

KVM – K Virtual Machine

MDC – Máximo Divisor Comum

MIPS – Millions of Instructions Per Second

MIT - Massachusetts Institute of Technology

NIST – National Institute of Standards and Technology

NFC – Near Field Communication

NPU – Numeric Processor Unit

NSA – National Security Agency

NSF – National Smartcard Framework

OAEP - Optimal Asymmetric Encryption Padding

OSI - Open Systems Interconnection

OTP – One Time Pad

PC/SC - Interoperability Specification for Integrated Circuits and Personal Computer System

PDA – Personal Digital Assitant

PIN – Personal Identification Number

PKCS – Public Key Cryptography Standards

PKD – Public Key Directory

PKGC - Private Key Generator Center

PKI – Public Key Infrastructure

PKIX - Public Key Infrastructure X.509

PSS - Probabilistic Signature Scheme

RA – Registration Authority

RAM – Read Access Memory

RFID – Radio Frequency Identification

ROM – Read Only Memory

RSA – Rivest Shamir Adleman

SIM - Subscriber Identity Module

WEP - Wired Equivalent Privacy

WOT – Web Of Trust

XOR – Exclusive Or

4.4.2 Pagamento *online* e não anônimo – *framework* iKP 55
4.4.2.1 1KP 58
4.4.2.2 2KP 58
4.4.2.3 3KP 59
4.4.3 Pagamento *online* e não anônimo do tipo cartão de débito – cSET 60
4.4.3.1 Arquitetura 60
4.4.3.2 Protocolo de Registro 62
5. Adaptação do protocolo de registro do cSET 65
5.2. Descrição do Protocolo de Registro 65
5.3. Análise de Segurança 69
5.4. Teclado virtual 70
6. Implementação 71
6.2. Laboratório – configuração de *hardware* e *software* 71
6.3. Descrição da implementação 73
6.3.1 Applet Java Card 74
6.3.2 Cliente Java 84
6.4. Testes e Simulação 90
6.5. Eficiência 90
7. Conclusão 92
Referências bibliográficas 95
Apêndices 99

Lista de Tabelas

Tabela 2-1– Mecanismos de defesa do padrão X.800 4
Tabela 2-2– Serviços de defesa do padrão X.800 5
Tabela 3-1– Estatísticas Java. 27
Tabela 3-2– Mercado brasileiro de cartões. 29
Tabela 3-3– Mercado brasileiro de celulares, por tecnologia. 29
Tabela 3-4 – Codificação para as *status words* de retorno SW1 e SW2. 36
Tabela 3-5– Categorias de comandos e respostas APDU´s. 37
Tabela 3-6– Exemplos de ATR´s para cartões inteligentes. 38
Tabela 3-7 – Características linguagem Java Card. 41
Tabela 4-1 – Comparação dos protocolos iKP. 59

Lista de Gráficos

Gráfico 2-1 – Representação de um canal de comunicação. 6
Gráfico 2-2 – Modo ECB. 11
Gráfico 2-3 – Modo CBC 12
Gráfico 2-4 – Assinatura DSS. 18
Gráfico 2-5 – Verificação da assinatura DSS. 18
Gráfico 2-6– Representação de um modelo de segurança de comunicação. 19
Gráfico 2-7 – Arquitetura PKIX. 20
Gráfico 3-1 – Classificação para cartões inteligentes. 26
Gráfico 3-2 – SIM card GSM – cartão com contato, para setor de telecomunicações. Fabricante Gemalto. 30
Gráfico 3-3 – Bilhete de transporte com chip – cartão com contato, para setor de transporte. Fabricante Sagem Orga. 31
Gráfico 3-4 – Passaporte eletrônico – cartão sem contato, para setor Governo. Fabricante GD Burti. 31
Gráfico 3-5 – Cartão de identidade eletrônico – cartão dual, para setor de Governo. Fabricante Oberthur. 32
Gráfico 3-6 – Representação dos oito contatos de um circuito integrado de cartões inteligentes. 33
Gráfico 3-7 – Dimensões de um *smart card*. 33
Gráfico 3-8 – Esquemático básico de um microcontrolador de um cartão inteligente 34
Gráfico 3-9 – Modelo físico de comunicação para cartões inteligentes 35
Gráfico 3-10 – Comandos e respostas APDU´s, saída na console do Eclipse. 37
Gráfico 3-11 – Arquitetura Java Card. 40
Gráfico 3-12 – Criação de um applet Java Card. 42
Gráfico 3-13 – Componentes Java 6. 42
Gráfico 4-1 – Atores de um sistema de pagamento eletrônico 44
Gráfico 4-2 – Exemplo de comunicação explícita 48
Gráfico 4-3 - Exemplo para o caso de dar nomes 49
Gráfico 4-4 – Exemplo para o caso de cifrar apenas quando necessário. 50
Gráfico 4-5 – Exemplo para o caso de assinar antes de cifrar. 51
Gráfico 4-6 – Exemplo para o caso de geração de números aleatórios. 52
Gráfico 4-7 – Exemplo para o caso de números previsíveis. 52

Gráfico 4-8 – Atores de um pagamento eletrônico no iKP 56
Gráfico 4-9 – Fluxo de mensagens no iKP 57
Gráfico 4-10 – Fluxo de mensagens no 1KP 58
Gráfico 4-11 – Fluxo de mensagens no 2KP 58
Gráfico 4-12 – Fluxo de mensagens no 3KP 59
Gráfico 4-13 – Arquitetura geral do cSET 62
Gráfico 4-14 – Procedimento de Registro no cSET 64
Gráfico 6-1 – Leitora ECO 5000 72
Gráfico 6-2 – Leitora CardMan 5321 72
Gráfico 6-3 – Cartão inteligente TOP IM GX4 73
Gráfico 6-4 – Facção do protocolo proposto, usado na implementação 74
Gráfico 6-4 – Processamento do comando APDU pela JCRE de um applet 76
Gráfico 6-5 – Inicializar Cartão 77
Gráfico 6-6 – Verificar PIN 78
Gráfico 6-7 – Recuperar Chave Pública para Cifrar 79
Gráfico 6-8 – Recuperar Chave Pública para Assinar 79
Gráfico 6-9 – Upload de Dado Assinado 80
Gráfico 6-10 – Assinar RSA 81
Gráfico 6-11 – Upload Certificado Digital 82
Gráfico 6-12 – Recuperar Certificado Digital 83
Gráfico 6-13 – Inicialização do cartão inteligente 85
Gráfico 6-14 – Verificação do PIN 86
Gráfico 6-15 – Gravação do Certificado Digital no Cartão 87
Gráfico 6-16 – Recuperação de Certificado Digital 88
Gráfico 6-17 – Recuperação de Chaves Públicas 89
Gráfico 6-18 – Assinatura no Cartão 90

LISTA DE SÍMBOLOS, NOMENCLATURA E ABREVIAÇÕES

3G – Third Generation Cell Phone

AES – Advanced Encryption Standard

APDU – Application Data Unit

API – Application Programming Interface

ATM – Automatic Teller Machine

ATR – Answer to Reset

CA – Certification Authority

CCA – Chosen Ciphertext Attack

CCA2 – Adaptative Chosen Ciphertext Attack

CPU – Central Processing Unit

CRL – Certificate Revocation List

DEMA – Differential Electromagnetic Attack

DFA – Differential Fault Analysis

DH – Diffie-Hellman

DNS – Domain Name System

DPA – Differential Power Analysis

DSS – Digital Signature Standard

ECC – Elliptic Curve Cryptography

EEPROM - Electrically-Erasable Programmable Read-Only Memory

EMV – Europay Mastercard Visa

GNFS - Generalized Number Field Sieve

GP - GlobalPlatform

GPL – General Public License

GSM - Global System for Mobile Communications

ICAO – International Civil Aviation Organisation

ICP – Infraestrutura de Chaves Públicas

IEC - International Electrotechnical Commission

ISSO – International Standard Organisation

ITU – International Telecommunication Union

JCAPI – Java Card Application Programming Interface

JCRE – Java Card Runtime Environment

JCVM – Java Card Virtual Machine

1. INTRODUÇÃO

Um sistema de pagamento eletrônico pode ser definido como um conjunto de relações totalmente desmaterializadas que os agentes econômicos têm com respeito uns aos outros se valendo de primitivas criptográficas para se obter segurança eletrônica. Há duas grandes classificações para este sistema, a primeira é baseada no estado da conexão de rede e privacidade durante o pagamento, e a segunda é baseada na temporalidade em que o fluxo financeiro do pagamento ocorre.

Em primeiro lugar, os sistemas de pagamento podem ser do tipo:

- *online* e não-anônimo;
- *online* e anônimo;
- *offline* e não-anônimo;
- *offline* e anônimo.

A característica de anonimato é utilizada para a garantia da privacidade dos atores envolvidos na transação. Em muitos países existem barreiras legais (do Estado) e administrativas (das Instituições Bancárias) para a proibição de operações eletrônicas totalmente anônimas, estas barreiras visam intimidar operações criminosas tais como lavagem de dinheiro. Desta forma um sistema de pagamentos *online* e perfeitamente anônimo é impraticável nos dias atuais.

Em segundo lugar, os sistemas de pagamento podem ser do tipo:

- cartão de crédito;
- cartão de débito;
- cartão de dinheiro eletrônico.

De forma geral, um pagamento baseado em cartão de crédito é chamado de *pay later*, ou seja, o pagamento é feito posteriormente à obtenção de uma mercadoria. O pagamento baseado em cartão de débito é chamado de *pay now*, ou seja, o pagamento é feito simultaneamente à obtenção da mercadoria. Finalmente, o pagamento com dinheiro eletrônico é chamado de *pay before* uma vez que há a necessidade de se introduzir, eletronicamente, uma quantia monetária no cartão antes da realização de qualquer compra.

Um dos sistemas de pagamentos mais práticos de se implementar, que pode usar mecanismos de criptografia com segurança comprovada, que possui uma personalização que inibe a lavagem de dinheiro e que, portanto, está de acordo com o modelo de negócio das Instituições Bancárias, é o sistema *online* e não-anônimo do tipo cartão de débito ou crédito.

De acordo com a pesquisa bibliográfica realizada, os protocolos de pagamento eletrônico existentes comercialmente não são públicos, no entanto, um protocolo que se encontra aberto é o cSET o qual está descrito no livro *Protocols for Secure Electronic Commerce* (Sherif (2003, p. 343)). A presente dissertação implementa uma modificação na função de registro do protocolo cSET de modo que o novo protocolo continue seguro mesmo que a leitora de cartões seja maliciosa.

Adicionalmente, outra preocupação da presente dissertação é o desenvolvimento de técnicas baseadas em padrões abertos para interoperabilidade e integração. Cartões inteligentes são notoriamente conhecidos por serem dispositivos "caixa preta" com parca documentação e com alguns fabricantes mundiais usando estratégias de segurança pela obscuridade. A implementação realizada tem fundação em ferramentas não proprietárias obtidas em comunidades de usuários como o *Java Card Development Kit* e o *Global Platform Shell*, procurando criar uma implementação com o menor grau de acoplamento com o *hardware* possível no momento atual, ou seja, implementando-se as funcionalidades criptográficas para o pagamento a partir de rotinas de *software* embarcadas.

Esta dissertação organiza-se, após esta introdução, com a apresentação inicial de um capítulo com as preliminares matemáticas e de primitivas criptográficas que constituem a base para um correto entendimento de um protocolo de pagamento eletrônico com os respectivos custos computacionais envolvidos. O terceiro capítulo descreve a arquitetura de cartões inteligentes utilizados como dispositivos invioláveis em protocolos de pagamentos eletrônicos os quais são introduzidos no capítulo quatro. Uma adaptação do protocolo cSET que resiste a ataques de uma leitora maliciosa é proposto em seguida, no capítulo cinco, sendo que o mesmo é implementado em Java Card no capítulo seis. Finalmente, apresenta-se um capítulo com as conclusões e propostas de trabalhos futuros.

2. PRELIMINARES

Neste capítulo apresentam-se alguns conhecimentos básicos que são necessários para um melhor entendimento do restante desta dissertação. Descrevem-se, de maneira sucinta, conceitos básicos de criptografia e de segurança eletrônica, bem como o problema da definição do modelo de segurança.

É importante frisar que não é necessário o entendimento do funcionamento interno das primitivas criptográficas usadas para a compreensão do protocolo de pagamento eletrônico proposto e implementado neste trabalho. No entanto, de modo a propiciar ao leitor uma melhor compreensão das dificuldades de implementação e dos custos computacionais envolvidos no uso destas primitivas, apresenta-se nas seções seguintes uma descrição mais detalhada de alguns criptosistemas de chaves públicas (os que apresentam maior demanda por poder computacional por parte da plataforma de implementação), bem como uma breve revisão da matemática necessária.

2.2. ALGUMAS DEFINIÇÕES BÁSICAS DE SEGURANÇA ELETRÔNICA

Antes de se definir semanticamente a palavra criptografia ou de se buscar uma definição matemática formal é necessário entender alguns conceitos mais gerais oriundos da segurança eletrônica.

Em ITU (1991, p. 4) são abordadas, três grandes áreas relativas à segurança: ataques, mecanismos de defesa e serviços de defesa.

Ataques: qualquer ação que comprometa a confiabilidade de uma determinada informação. Há 2 tipos principais, os **ataques passivos** que são aqueles em que o ataque é feito apenas observando-se a informação que está sendo transmitida e os **ataques ativos** em que o ataque envolve alterações na informação que está sendo transmitida.

Mecanismos de defesa: quaisquer processos para se detectar um ataque, prevenir-se de um ou recuperar-se após um ataque. A Tabela 2.1 traz um resumo da recomendação X.800 para os mecanismos de defesa dividindo-os em mecanismos de defesa específicos

para determinada camada do protocolo *Open Systems Interconnection* (OSI) e em mecanismos de defesa pervasivos por mais de uma camada.

Tabela 2-1– Mecanismos de defesa do padrão X.800

Fonte: ITU (1991, p. 10)

Mecanismos de defesa específicos
Ciframento: é o mecanismo de transformação da informação, via criptografia, de modo a se produzir um objeto diferente do objeto original.
Assinatura digital: é o mecanismo que permite ao ator receptor da informação verificar a identidade do ator emissário da informação.
Controle de acesso: é o mecanismo que previne o uso não autorizado de um determinado recurso incluindo a prevenção do seu uso de maneira inadequada.
Integridade da informação: é o mecanismo usado para garantir que a informação não foi alterada ou destruída de maneira não autorizada.
Troca de autenticação: é o mecanismo usado para garantir a identidade de um ator por meio da troca de informações entre atores.
***Traffic padding*:** é o mecanismo de inserção de bits em trechos que apareçam no fluxo da informação de modo a evitar-se tentativa de análise de tráfego.
Controle de roteamento: é o mecanismo usado para permitir que determinadas informações classificadas possam ter diferentes rotas especialmente quando se desconfia de falha na segurança em alguma rota.
Cartório: é o apoio em uma terceira parte confiável para a garantia da acurácia de certas características da informação.
Mecanismos de defesa pervasivos
Funcionalidade confiável: é o mecanismo que permite que uma funcionalidade seja percebida como correta, por exemplo, de acordo com a política de segurança.
Selo de segurança: é o mecanismo de aplicação de uma marca ou selo a um determinado recurso que tipifica os atributos de segurança dele.
Detecção de evento: é o mecanismo disponível para detecção de eventos relevantes de segurança.
Trilha de auditoria: é o mecanismo para se coletar informação de modo a permitir uma inspeção em registros e atividades a ser realizada por um ator independente segundo uma política de segurança.
Recuperação de segurança: é o mecanismo que lida com requisições de atores e gera ações de recuperação.

Serviços de defesa: quaisquer serviços que são providos por um sistema para dar um determinado tipo de proteção aos recursos do sistema, estes serviços usam políticas de segurança e são implementados por mecanismos de segurança. A Tabela 2.2 traz um resumo da recomendação X.800 para os serviços de defesa.

Tabela 2-2– Serviços de defesa do padrão X.800

Fonte: ITU (1991, p. 8)

Autenticidade	Serviço para garantia de que os atores no processo de comunicação são, de fato, aqueles que eles dizem ser.
Confidencialidade	Serviço para garantia de que a informação não ficará disponível ou revelada para atores não autorizados.
Integridade	Serviço para garantia de que a informação recebida é a mesma que foi enviada.
Não repúdio	Serviço de proteção contra a negação de participação, por qualquer ator, em um processo de troca de informações.
Controle de acesso	Serviço de proteção contra o acesso não autorizado a um determinado recurso.

2.3. CONCEITOS DE CRIPTOGRAFIA

A criptografia apresenta-se como um dos blocos fundamentais para a obtenção de sistemas de informação seguros. Nesta seção apresenta-se, de maneira breve, alguns conceitos básicos porém importantes de criptografia que serão utilizados no decorrer desta dissertação.

Na literatura, existe uma profusão de técnicas criptográficas que podem ser aplicadas em um sistema de pagamento eletrônico. Dois grandes grupos se destacam, a criptografia de chave secreta e a criptografia de chave pública.

A palavra “criptografia” é oriunda dos termos gregos *kryptós* e *gráphein* que significam respectivamente "escondido" e "escrita", o que sugere uma definição para criptografia, na eletrônica, como sendo o ato de ofuscar o conteúdo dos dados em um processo de comunicação. Para Menezes et al (1996, p. 4) a criptografia é o estudo das técnicas matemáticas relacionadas a aspectos de segurança da informação tais como integridade de dados, autenticação de entidades e autenticação de origem. Já Stinson (2006, p. 1) cita que o principal objetivo da criptografia é permitir que dois atores comuniquem-se de forma segura através de um canal inseguro, de tal forma que um adversário não entenda o que está sendo comunicado.

Uma definição mais operacional de criptografia, baseada em sua aplicabilidade, pode ser retirada de ITU (1991, p. 4), "***Cryptography*** *is the discipline wich embodies principles, means and methods for the transformation of data in order to hide its information content, prevente its undetected modification and/or prevente its unauthorized use.*".

O modelo de comunicação usualmente estudado em criptografia é apresentado no Gráfico 2.1. Neste modelo, Stinson (2006, p. 1), define 2 atores, aqui chamados como Alice (referência à ponta A) e Bob (referência à ponta B), que podem comunicar-se através de um canal, não necessariamente seguro, como a Internet, de tal forma que um adversário, aqui chamado como Eva (referência à *Eavesdropping*), não possa entender a comunicação. Este canal pode ser uma rede de computadores, uma linha telefônica, entre outros. A informação que Alice envia para Bob é chamada *plaintext* e pode ser um texto, uma imagem ou qualquer outra estrutura arbitrária. Alice criptografa o *plaintext* usando uma chave pré-compartilhada (através de um canal seguro) e envia o resultado, chamado *ciphertext*, via qualquer canal de comunicação (seguro ou inseguro). Eva, observando um canal inseguro, pode interceptar o *ciphertext*, porém não pode determinar o *plaintext* a partir do *ciphertext* pois não possui a chave. Apenas Bob, que pré-compartilhou a chave com Alice, poderá decriptografar o *ciphertext* e reconstruir o *plaintext*.

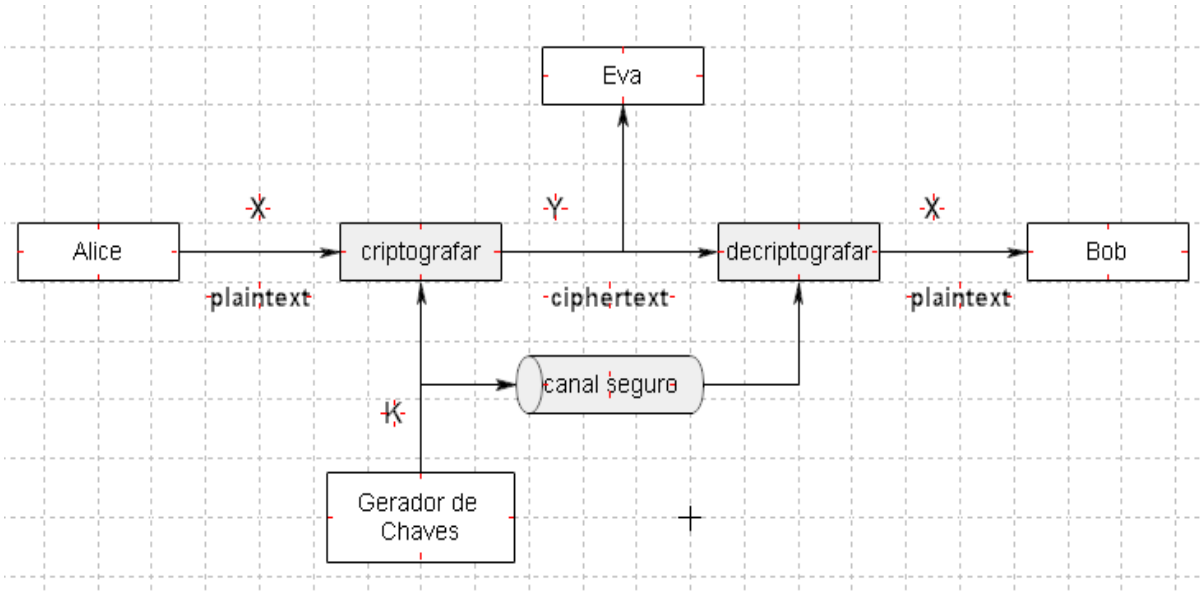

Gráfico 2-1 – Representação de um canal de comunicação.

Fonte: Stinson (2006, p. 2)

Matematicamente, supõe-se um criptosistema como uma quíntupla (α, β, γ, ε, δ) nas

seguintes condições:

1. Seja α um conjunto finito de possíveis *plaintexts*;
2. Seja β um conjunto finito de possíveis *ciphertexts*;
3. Seja γ um conjunto finito de possíveis chaves (*keyspace*);
4. Para cada $K \in \gamma$ existe uma regra de criptografia $E_k \in \varepsilon$ e a correspondente regra de decriptografia $D_k \in \delta$, onde $E_k : \alpha \rightarrow \beta$ e $D_k : \beta \rightarrow \alpha$ são funções tal que:

   $D_k(E_k(x)) = x$ para cada *plaintext* $x \in \alpha$.

Como exemplo, suponha que Alice e Bob escolham uma chave aleatória $k \in \gamma$. Isto é feito quando Alice e Bob encontram-se em um mesmo local e Eva não está observando ou a chave k é transmitida por um canal seguro.

Posteriormente, Alice e Bob desejam transmitir, via canal não necessariamente seguro, $x = x_1 x_2 \ldots x_n$ para algum inteiro $n \geq 1$ onde cada $x_i \in \alpha$, $1 \leq i \leq n$. Cada x_i é criptografado usando-se uma regra de criptografia E_k. Alice então computa $y_i = E_k(x_i)$, $1 \leq i \leq n$, resultando na *ciphertext* $y = y_1 y_2 \ldots y_n$ que é enviada pelo canal.

Quando Bob recebe $y = y_1 y_2 \ldots y_n$ ele decriptografa usando a função D_k e obtém a mensagem original $x = x_1 x_2 \ldots x_n$.

Há dois grandes grupos de primitivas criptográficas comumente utilizados em sistemas eletrônicos diversos, a criptografia simétrica ou de chave secreta e a criptografia assimétrica ou de chaves públicas. Antes de iniciar a discussão destas primitivas criptográficas é necessária uma breve revisão da teoria dos números.

2.4. REVISÃO DA TEORIA DOS NÚMEROS

Os criptosistemas de chaves públicas são usualmente baseados na álgebra moderna e na teoria dos números. Nesta seção faz-se uma revisão de alguns pontos destas duas disciplinas, de forma sintética, visando permitir o entendimento do formalismo do RSA, o qual será utilizado no protocolo de pagamento implementado nesta dissertação. Uma

análise mais abrangente da teoria dos números, aplicada à computação, pode ser encontrada no livro *Number Theory for Computing*, Yan (2002).

Conforme descrito em Yan (2002, p. 89), em 1801, Gauss introduziu as bases da moderna teoria dos números. A aritmética modular, que interessa em criptosistemas de chaves públicas como o RSA, tem suas bases na definição de congruência. Stinson (2006, p. 3) traz a seguinte definição: suponha a e b inteiros e m um inteiro positivo. Então podemos escrever $a \equiv b \ (mod\ m)$ se m divide (b - a). A expressão $a \equiv b \ (mod\ m)$ é chamada congruência e deve ser lida da seguinte forma: a é congruente com b módulo m. O resultado de b mod m também pode ser visualizado como o resto da divisão de b por m.

Seja $\mathbb{Z}_m = \{0, 1, 2, \ldots\ldots, \mathrm{m} - 1\}$, observa-se que este conjunto, munido de uma adição módulo m e de uma multiplicação módulo m, possui as seguintes propriedades, conforme descrito em Stinson (2006, p. 3):

1. $a, b \in \mathbb{Z}_m \rightarrow a + b\ mod\ m \in \mathbb{Z}_m$
2. Existe um elemento neutro 0 na adição $| \forall\ a \in \mathbb{Z}_m : a + 0 = 0 + a\ (mod\ m)$
3. $\big((a + b) + c\big)\ mod\ m = \big(a + (b + c)\big)\ mod\ m, para\ a, b\ e\ c \in \mathbb{Z}_m$
4. $\forall\ a \in \mathbb{Z}_m, \exists$ -a $\in \mathbb{Z}_m \mid a + (-a) = 0\ (mod\ m)$

Um conjunto de elementos munido de uma operação binária onde as propriedades de 1 a 4 são satisfeitas é chamado de **Grupo**. Caso o conjunto de elementos seja finito diz-se que temos um **Grupo Finito**.

5. $\forall\ a, b \in \mathbb{Z}_m, a + b = b + a\ \ (mod\ m)$

Um Grupo que obedece à propriedade 5 é chamado **Grupo Comutativo ou Grupo Abeliano**.

6. $\forall\ a, b \in \mathbb{Z}_m, a * b\ mod\ m \in \mathbb{Z}_m$
7. $\exists\ 1 \in \mathbb{Z}_m \mid \forall\ a \in \mathbb{Z}_m : a * 1 = 1 * a\ (mod\ m)$
8. $\forall\ a, b, c \in \mathbb{Z}_m, (ab)c = a(bc)\ (mod\ m)$
9. $\forall\ a, b, c \in \mathbb{Z}_m, a * (b + c) = a * b + a * c\ (mod\ m)$
10. $\forall\ a, b \in \mathbb{Z}_m, a * b = b * a\ (mod\ m)$

11. $\forall\, a \in \mathbb{Z}_m$ e para alguns valores de m, $\exists\, a^{-1} \in \mathbb{Z}_m \mid a * a^{-1} = 1\ (mod\ m)$

Um conjunto $\mathbb{Z}_m$ munido de adição e multiplicação modulares satisfazendo as propriedades de 1 a 11 é chamado de **Corpo Finito ou Corpo de Galois**. O cálculo do inverso multiplicativo (quando ele existir) é efetuado através do **Algoritmo Estendido de Euclides**.

Conforme pode ser visto em Stinson (2006, p. 9), sejam a e b elementos de $\mathbb{Z}_m$, a congruência ax ≡ b (mod m) possui uma solução única x ∈ $\mathbb{Z}_m$, se e somente se, o máximo divisor comum **MDC(a, m) = 1**. Suponha agora que a ≥ 1 e m ≥ 2 sejam ambos inteiros, se o MDC(a,m) = 1 então diz-se que a e m são coprimos ou primos relativos. A quantidade de número inteiros em $\mathbb{Z}_m$ que são coprimos a m é frequentemente denominado de **função de Euler** $\emptyset(n)$, em outras palavras, $\emptyset(n)$ equivale à cardinalidade de $\mathbb{Z}_m$. O conjunto de números inteiros positivos menores que m e coprimos com m é denominado $\mathbb{Z}_m^*$.

A função de Euler é utilizada no **Teorema de Euler** que diz que $a^{\emptyset(n)} \equiv 1\ (mod\ n)$ para valores de n que sejam inteiros positivos tal que MDC(a,n) = 1. O Teorema de Euler é importante para a demonstração de que, no RSA, as operações de ciframento e deciframento se cancelam.

O **Teorema do Resto Chinês** é ilustrado em Stinson (2006, p. 170) e é utilizado em algumas situações para acelerar o processo de decriptografia e assinatura RSA. Para este caso, suponha uma mensagem m_1, ..., m_r com MDC(m_i, m_j) = 1 para i ≠ j. Sejam ainda a_1 ..... a_r inteiros. Então a congruência x ≡ a_i (mod m_i), 1 ≤ i ≤ r tem uma solução única modulo m = m_1 * .... m_r que é dada por:

$$x = \sum_{i=1}^{r} a_i M_i y_i \bmod m$$

onde: $M_i = {}^{M}/_{m_i}\ M_i = \frac{M}{m_i}$ e $y_i = M_i^{-1}\ (mod\ m)$ para 1≤ i ≤ r.

Uma representação comum de $\mathbb{Z}_m$ é para o caso de m ser um número primo e daí representa-se por $\mathbb{Z}_p$. No **Pequeno Teorema de Fermat**, descrito em Stinson (2006, p. 171), supõe-se que p seja primo e b ∈ $\mathbb{Z}_p$, então a congruência b^p ≡ b (mod p) é válida. Também é válida a forma, b^{p-1} ≡ 1 (mod p). O Pequeno Teorema de Fermat é a base do

teste probabilístico que verifica as chances de um número ser primo.

2.5. CRIPTOGRAFIA SIMÉTRICA OU DE CHAVE SECRETA

De modo geral, divide-e o estudo de criptografia simétrica em três grandes áreas: as cifras de bloco, as cifras de fluxo e as funções de *hash*.

2.5.1 Cifras de Bloco e Cifras de Fluxo

O princípio fundamental das cifras de bloco e das cifras de fluxo é que a chave utilizada para decriptografar seja de trivial obtenção (ou idêntica) a partir da chave utilizada para criptografar (ou que a chave seja pré-compartilhada entre as partes). Em termos matemáticos:

Criptografia: $E_k(m) = c$

Decriptografia: $D_k(c) = m$

onde: m é a mensagem original;
c é a mensagem criptografada;
k é a chave pré-compartilhada.

No caso das cifras de bloco, como o próprio nome sugere, o ciframento ocorre sempre em blocos. Logo, deve-se dividir a mensagem a ser cifrada em blocos de tamanho igual ao tamanho usado na cifra. Caso isto não seja possível, deve-se completar o texto em claro com símbolos adicionais (*padding*). Este preenchimento deve ser feito de maneira tal que a sua retirada possa ser efetuada de maneira não ambígua. Em seguida, deve-se escolher um modo de operação a ser adotado na cifra de bloco, os dois modos mais utilizados são o *Electronic Code Book* (ECB) e o *Cipher Block Chaining* (CBC).

No modo ECB, a criptografia e a decriptografia são aplicadas em cada um dos blocos sucessivos da mensagem original e da mensagem cifrada, respectivamente, conforme ilustrado no Gráfico 2.2. Observa-se que se ocorrer, no Gráfico 2.2, P1 = P2 então C1 = C2, este fato ilustra a fragilidade deste modo de ciframento pois, dado um bloco cifrado previamente, pode-se deduzir como será o bloco cifrado futuro.

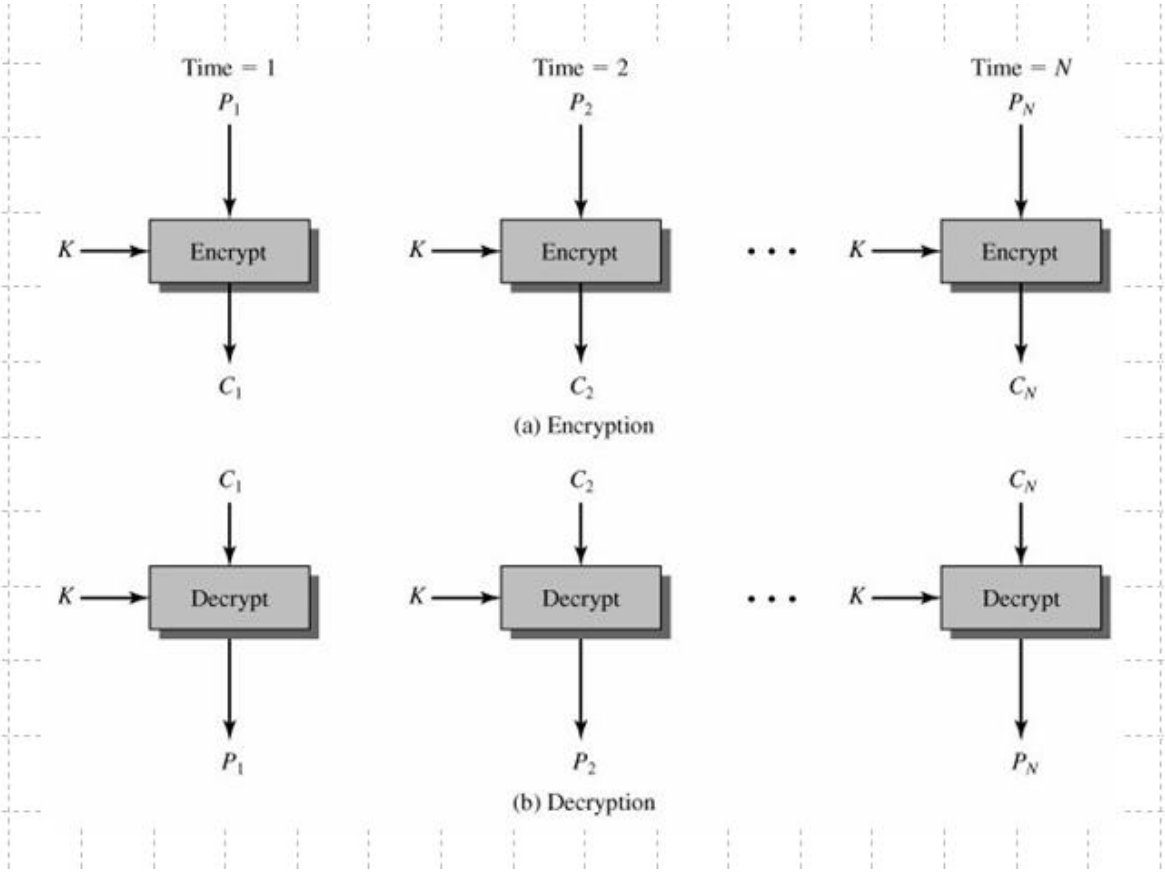

Gráfico 2-2 – Modo ECB.

Fonte: Stallings (2006, p. 182)

Já no modo CBC, ilustrado no Gráfico 2.3, percebe-se que as deficiências do ECB foram corrigidas com a introdução de um fator, não determinístico, chamado de IV (Vetor de Inicialização). Desta forma, para dois blocos de entrada iguais, se o vetor de inicialização for não previsível, a saída cifrada também será imprevisível, dirimindo-se a deficiência encontrada no modo ECB. Assim, pode-se dizer que o ECB é um modo não seguro devendo-se dar preferência ao modo CBC nas cifras de bloco.

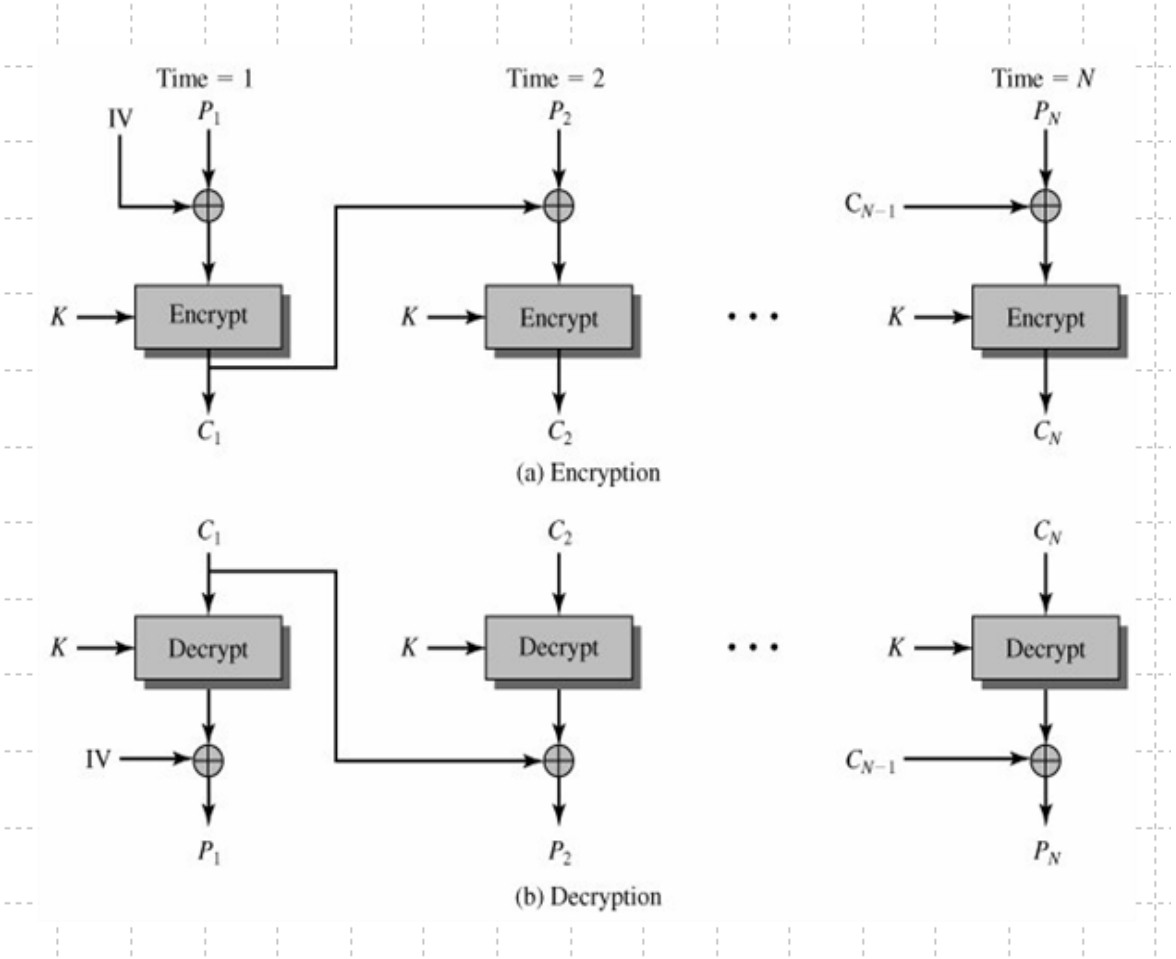

Gráfico 2-3 – Modo CBC

Fonte: Stallings (2006, p. 183)

Já as cifras de fluxo caracterizam-se pela criptografia de uma mensagem um byte por vez, por exemplo, pode-se usar a função XOR para produzir um cifra de fluxo da seguinte forma:

$$\begin{array}{rll} & 11001100 & mensagem \\ \oplus & \underline{01101100} & \underline{chave} \\ & 10100000 & cifrado \end{array}$$

A cifra de fluxo da forma acima é muito similar a um criptosistema chamado *one-time pad*, porém este criptosistema necessita de uma chave gerada verdadeiramente aleatória enquanto que uma cifra de fluxo pode usar uma chave gerada de forma pseudo-aleatória.

Um exemplo, bem sucedido, de uma cifra de fluxo é a RC4 proposta por Rivest em 1987 e, segundo cita Stallings (2006, p. 191), é utilizada em protocolos de redes sem fio 802.11

WPA e também nos padrões SSL/TLS por ter bom desempenho e ser extremamente simples de se implementar.

2.5.2 Funções de *hash*

Segundo Stallings (2006, p. 334), uma função de *hash* é aquela que aceita, como entrada, uma mensagem de tamanho variável e produz na saída uma mensagem de tamanho fixo. Uma função que tenha esta propriedade tem inúmeras aplicações de segurança como, por exemplo, servir de base para a construção de assinaturas digitais, MAC, protocolos de autenticação, entre outros.

Diz-se que uma boa função de *hash* (H) tem as seguintes características:

1. ***Preimage resistant***: dada uma saída y, é impossível encontrar-se uma entrada x tal que H(x) = y;
2. ***Second preimage resistant***: dada uma entrada x, é impossível encontrar-se uma outra entrada z tal que H(x) = H(z);
3. ***Collision resistant***: é impossível encontrar-se duas entradas diferentes, x e w, tal que H(x) = H(w).

Como regra geral, a grande maioria dos esquemas de criptografia simétrica é projetada de modo heurístico, não havendo uma base matemática rigorosa que dê garantias acerca da intrabilidade computacional envolvida na quebra das cifras em questão. Logo, no caso de criptografia simétrica, é vital que sejam escolhidas primitivas de uso amplo e difundido e que sofreram o escrutínio da academia. Como exemplo de criptografia simétrica obedecendo a estas características pode-se citar a função de *hash* SHA2 com 256 bits e, como exemplo de criptografia simétrica comprovadamente segura, a cifra de bloco AES com 256 bits.

2.5.3 Criptografia assimétrica ou de chaves públicas

A criptografia de chaves públicas introduz o conceito de um par de chaves para cada ator participante de um processo de comunicação de dados (chave privada ou chave secreta (S_k) e chaves públicas (P_k)), de modo que deva ser impossível deduzir a chave privada a partir do conhecimento da chave que é pública. Este conceito foi proposto inicialmente por Diffie e Hellman, em 1976, na Universidade de Stanford, mas sem uma solução matemática que

suportasse a impossibilidade de inversão da chave pública em chave privada. Posteriormente surgiram esquemas como RSA e El Gamal, que, com as devidas alterações, podem ser provados seguros em um modelo formal. Em termos matemáticos:

Criptografia: $E_{PKB}(m) = c$ $E_{PKB}(m) = c$

Decriptografia: $D_{SKB}(c) = m$ $D_{SKB}(c) = m$

onde: m é a mensagem original;

c é a mensagem criptografada;

PK_B é a chaves públicas de B;

SK_B é a chave privada de B.

A técnica utilizada para impedir o descobrimento da chave privada a partir da chave pública consiste em se valer da intratabilidade de solução de certos problemas matemáticos. Os dois conjuntos de problemas matemáticos mais usados em criptografia de chaves públicas são a fatoração e o logaritmo discreto.

2.5.4 Criptosistema RSA

O algoritmo mais popular para criptografia de chaves públicas baseado na intratabilidade da fatoração é o RSA, criado em 1977 no MIT, e nomeado com as iniciais dos nomes dos seus inventores: Rivest, Shamir e Adleman. O RSA é definido como se segue:

- considere dois números primos grandes e aleatórios p e q. Atualmente recomenda-se um tamanho de, no mínimo, 1024 bits.
- calcula-se n = p * q
- calcula-se $\emptyset(n) = (p-1) * (q-1)$
- escolhe-se um número aleatório "e", com $1 < e < \emptyset$; MDC(e, Ø) = 1 (diz-se que "e" e "Ø" são coprimos)
- calcula-se d tal que 1 < d < Ø ; e * d ≡ 1 (mod Ø)
- **(n, e) é chamado chave pública**
- **d é chamado chave privada**
- a criptografia de uma mensagem m é definida, então, por $c = m^e \, mod \, n$
- a decriptografia é definida $m = c^d mod \, n$

O RSA funciona uma vez que as operações de ciframento e deciframento se cancelam

(operações inversas) conforme pode ser visto na prova abaixo. Na prova para $m \in \mathbb{Z}_m^*$, tenta-se provar que $c^d = m^{ed} = m \ (mod \ n)$.

$$ed - 1 = k * \emptyset(n), k \ inteiro$$

$$ed = k * \emptyset(n) + 1$$

$$m^{e\,d} = m^{k\emptyset(n)+1} \ (mod \ n) = m^{k\emptyset(n)} * m \ (mod \ n)$$

A partir do Teorema de Euler tem-se que:

$$m^{k\emptyset(n)} * m \ (mod \ n) = 1 * m \ (mod \ n) = m \ (mod \ n)$$

O RSA puro como descrito acima é inseguro, pois é determinístico, porém existem soluções baseadas no acréscimo de *padding* e de aleatoriedade, transformando o RSA em um esquema probabilístico. Uma técnica de destaque é o *Optimal Asymmetric Encryption Padding* (OAEP) o qual foi proposta por Bellare e Rogaway, em 1994.

Segundo Stallings (2006, p. 279), a empresa RSA Security Inc recomenda a modificação do RSA através do procedimento OAEP e, para assinatura digital, é indicado o uso do procedimento *Probabilistic Signature Scheme* (PSS). Estas recomendações estão descritas na especificação PKCS#1 e um desenho ilustrativo do OAEP é mostrado no Gráfico 2.3. A idéia do OAEP é inicialmente adicionar um *pad* à mensagem M juntamente com o *hash* de um conjunto de parâmetros opcionais, representado por P, e gerar uma saída chamada DB. O próximo passo é adicionar uma aleatoriedade ao procedimento através da geração de uma semente aleatória (*seed*). Esta semente aleatória passa pela função de hash MGF e feito um XOR bit a bit com DB resultando na saída maskedDB. De forma similar, a variável maskedDB passa pela função *hash* MGF e é feito um XOR bit a bit com a entrada seed gerando a saída maskedSeed. A concatenação de maskedSeed e maskedDB forma a mensagem codificada EM. Resumindo-se, a variável EM contém a mensagem original com o *padding* mascarada pela semente aleatória e esta semente mascarada pela variável temporária maskedDB. O processo de criptografia RSA utiliza como mensagem de entrada a variável EM e não a mensagem inicial, a ser cifrada, M.

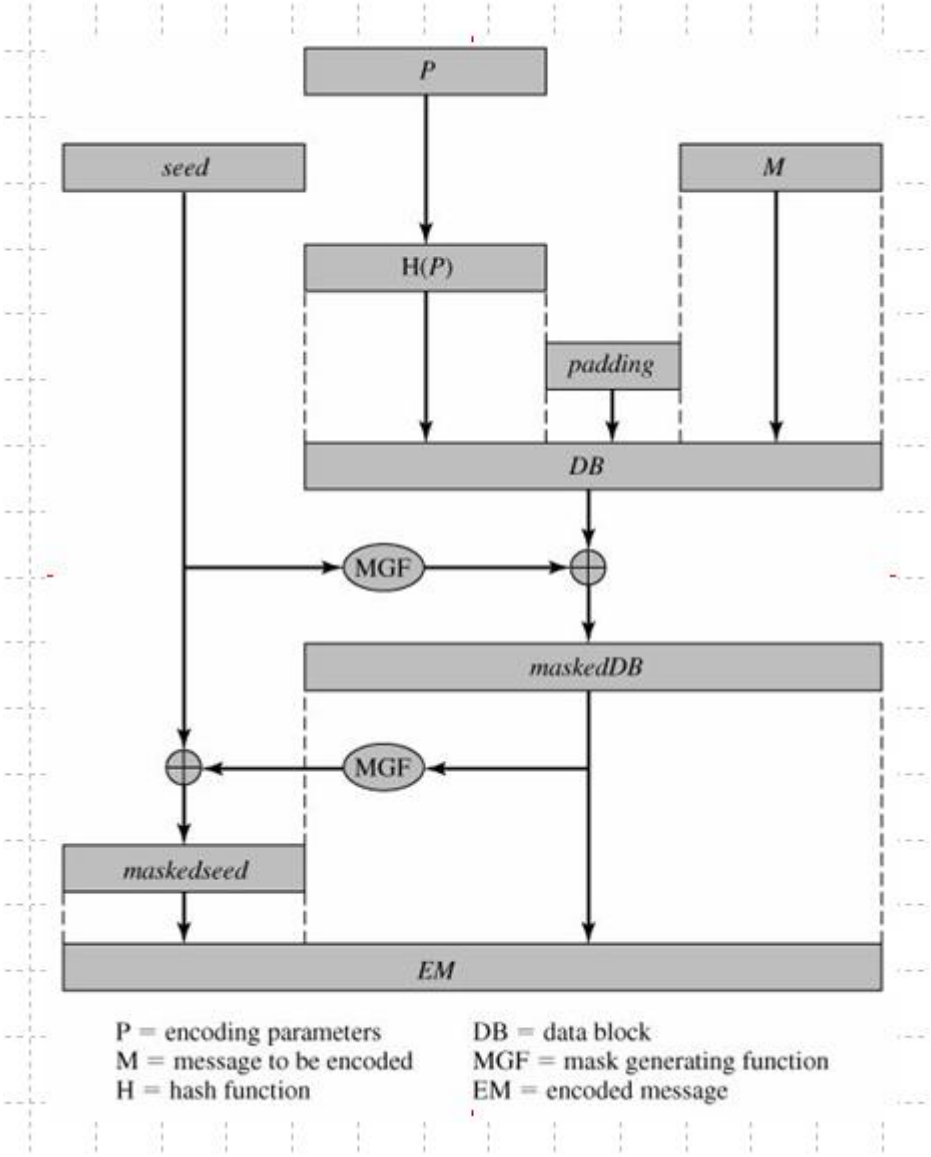

Gráfico 2-3 – *Optimal Asymmetric Encryption Padding* (OAEP).

Fonte: Stallings (2006, p. 280)

2.5.5 Assinatura digital

O conceito de assinatura digital é importante em um sistema de pagamento eletrônico para se evitar o repúdio de transações efetuadas e garantir, via assinatura da autoridade certificadora, que determinada chave pública em um diretório pertença, de fato, a um determinado participante da transação (certificados digitais de pessoas e de *hardwares*). A idéia básica é que uma assinatura digital em um objeto possa ser criada apenas por um ator, mas que possa ser verificada por qualquer um. Um esquema de assinatura pode ser probabilístico ou determinístico, este significa que o cômputo de uma assinatura sempre dará o mesmo resultado enquanto que aquele dará um resultado diferente. Do ponto de vista de uma assinatura à mão o esquema probabilístico é o mais real, pois quando se escreve com uma caneta em papel dificilmente se tem o exato formato para as letras, embora seja possível identificar uma assinatura falsa.

Ilustra-se a seguir os passos para uma assinatura RSA, ou seja, baseada no problema matemático da **fatoração**:

- Considere que a chave pública é (n,e) e a chave privada é d;
- Para assinar uma mensagem m: $s = m^d \bmod n$;
- Para verificar uma assinatura s de uma mensagem m:

  $$s^e \bmod n = (m^d)^e \bmod n = m;$$

- Na prática, ao esquema acima, também é adicionado um fator probabilístico.

Conforme se pode ver, a assinatura digital e a decriptografia são a mesma operação no RSA, sendo que a assinatura é realizada com o auxílio da chave privada e a verificação da assinatura através da chave pública.

Um exemplo de assinatura digital baseada no problema matemático do **logaritmo discreto** é a *Digital Signature Standard* (DSS). Ilustra-se a seguir os passos do DSS para assinatura, no Gráfico 2.4, e para verificação da assinatura, no Gráfico 2.5. A chave privada do usuário é x, onde $0 < x < q$ e a chave pública é $y = g^x \bmod p$.

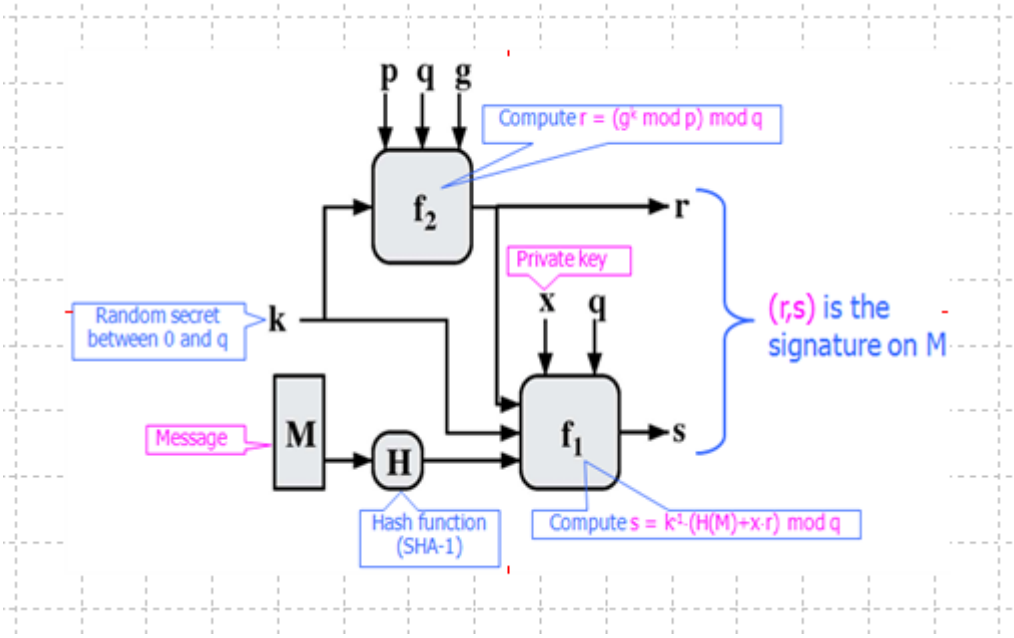

Gráfico 2-4 – Assinatura DSS.

Fonte: Nascimento (2008, p. 23)

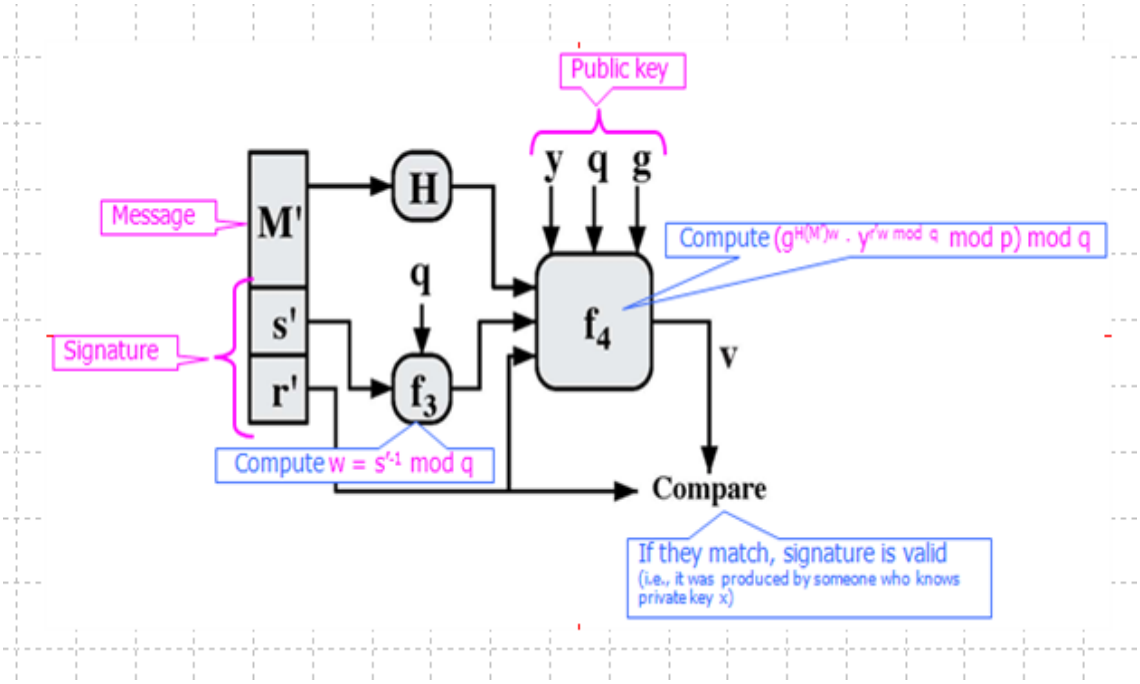

Gráfico 2-5 – Verificação da assinatura DSS.

Fonte: Nascimento (2008, p. 24)

2.5.6 Infraestrutura para criptografia de chaves públicas

A utilização de criptografia de chaves públicas comumente está associada a uma

infraestrutura de chaves públicas (PKI) onde existe apenas uma parte confiável no topo da hierarquia (a Autoridade Certificadora), a qual assina digitalmente certificados digitais que seguem algum padrão, por exemplo, o X.509 (PKIX). O Gráfico 2.6 ilustra uma rede de comunicação com uma terceira parte confiável.

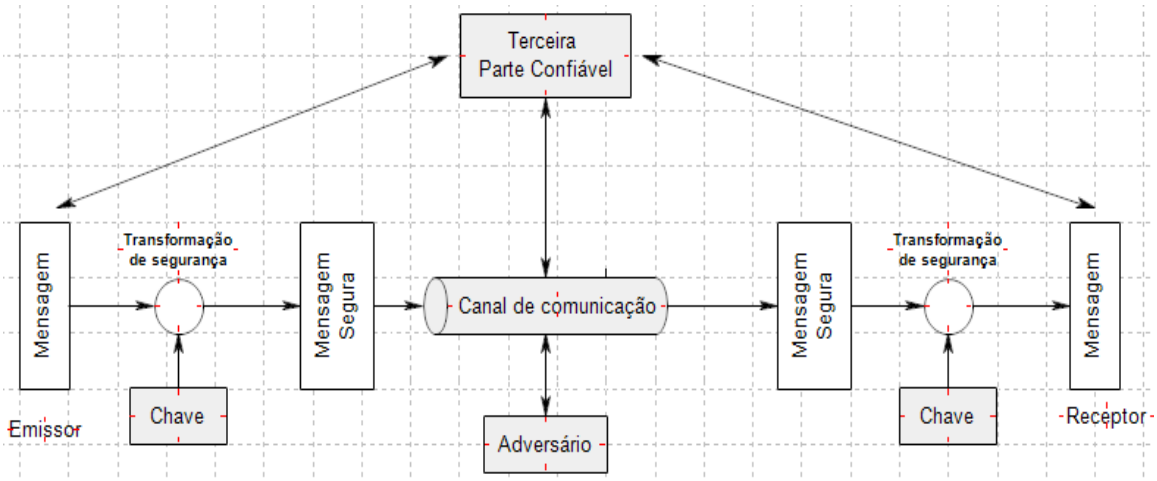

Gráfico 2-6– Representação de um modelo de segurança de comunicação.
Fonte: Adaptado de Stallings (2006, p. 22)

Para o foco de estudo desta dissertação será utilizada a alternativa de infraestrutura de chaves públicas X.509, conforme definido pelo Governo brasileiro na ICP-Brasil. Adams e Lloyd (2002, p. 28) conseguiram sintetizar o conceito de uma *Public Key Infrastucture* (PKI) da seguinte forma: "*PKI is the basis of a pervasive security infrastructure whose services are implemented and delivered using public-key concepts and techniques.*". Stinson (2006, p. 457) cita que o maior desafio da criptografia de chaves públicas é garantir a confiabilidade das chaves e que uma infraestrutura de chaves públicas é um sistema seguro usado para gerencial e controlar certificados digitais.

O Gráfico 2.7 representa a arquitetura do modelo PKIX onde os seguintes elementos compõe a infraestrutura de chaves públicas: certificado digital, autoridade certificadora, autoridade registradora, lista de certificados revogados (CRL) e diretório de chaves públicas (PKD).

Autoridade Certificadora (CA): é o emissor dos certificados e usualmente da lista de certificados revogados (CRL).

Autoridade Registradora (RA): é um componente opcional que pode assumir funções administrativas para a CA.

Lista de Certificados Revogados (CRL): é uma lista de publicação periódica contendo a identificação de certificados revogados. A periodicidade de publicação da lista é ajustada de acordo com políticas que variam de país a país. Esta lista normalmente é disponibilizada *online* e *offline*.

Diretório de chaves públicas (PKD): é um repositório online para todos os certificados emitidos pela CA, ou seja, um diretório que armazena todas as chaves públicas emitidas pela CA.

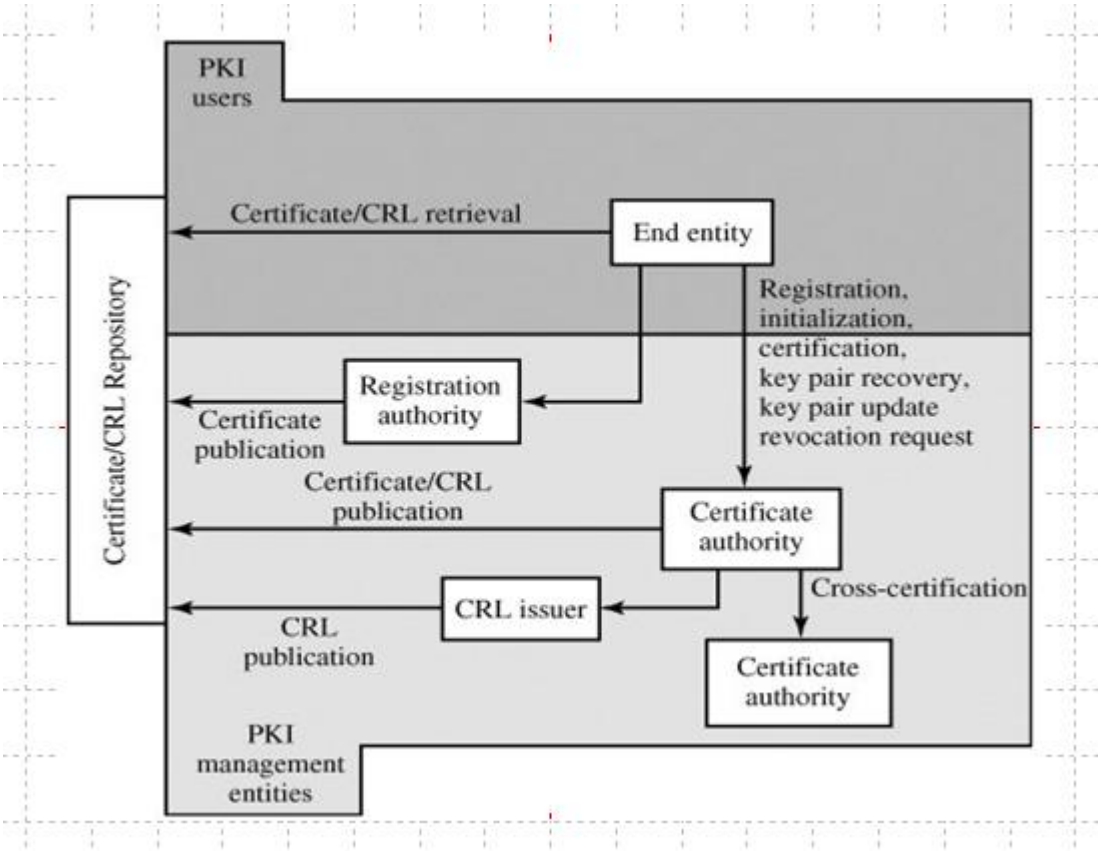

Gráfico 2-7 – Arquitetura PKIX.

Fonte: Stallings (2006, p. 429)

2.5.7 Certificado digital

Segundo Stinson (2006, p. 461) o conceito de certificados digitais foi introduzido em 1978

por Kohnfelder no MIT. Um certificado digital é o instrumento que liga a identidade de um ator à sua chave pública através de uma terceira parte confiável, a autoridade certificadora (CA), a qual assina digitalmente a informação contida no certificado de um ator qualquer. Assume-se que a chave pública da CA é conhecida de qualquer entidade que queira verificar a assinatura digital de um certificado, de modo a garantir que o conteúdo do certificado não foi maliciosamente adulterado. Um formato popular para certificados digitais é o X.509 que se encontra atualmente na versão 3, porém existem outros formatos como SPKI, OpenPGP e SET, este último também é um protocolo para pagamento eletrônico que estendeu o X.509, de maneira própria, de modo que o certificado só faz sentido para o protocolo SET. De todos os certificados citados, o mais usado em nível empresarial e governamental tem sido o X.509, inclusive a infraestrutua de chaves públicas brasileira, a ICP-Brasil, o utiliza. A estrutura do X.509 é definida por 11 campos:

1. Número da versão do certificado;
2. Número de série do certificado;
3. Identificador do algoritmo da assinatura digital;
4. Nome único do emissor do certificado;
5. Validade do certificado;
6. Nome único do dono do certificado;
7. Identificador do algoritmo das chaves públicas e Número da chaves públicas;
8. Campo opcional: Identificador único da autoridade certificadora;
9. Campo opcional: Identificador único do dono do certificado;
10. Extensões opcionais;
11. Assinatura digital da CA dos campos de 1 a 10 acima.

A infraestrutura PKIX está intimamente ligada ao algoritmo RSA e ao padrão PKCS (*Public Key Cryptography Standards*) que é composto por 15 áreas, conforme extraído de Sherif (2000, p. 121):

- **PKCS#1**: define as bases matemáticas e propriedades do RSA para criptografia e assinatura digital, tais como a orientação para se usar o RSA de forma não determinística, na forma provida pelos esquemas propostos por Bellare e Rogaway: RSA OAEP (*Optimal Asymmetric Encryption Padding)* para criptografia e RSA PSS (*Probabilistic Signature Scheme*) para assinatura. Ao PKCS#1 foram englobados os padrões PKCS#2 e PKCS#4.
- **PKCS#3**: define o protocolo de troca de chaves através do algoritmo Diffie-

Hellman.

- **PKCS#5**: descreve um método para criptografar informações usando uma chave secreta derivada de uma senha.
- **PKCS#6**: é a sintaxe de certificados X.509.
- **PKCS#7**: define a sintaxe de mensagens criptografadas e assinaturas usando as regras ASN.1 (*Abstract Syntax Notation* 1) em uma PKI.
- **PKCS#8:** descreve um formato para envio de informações relativas a chaves privadas.
- **PKCS#9:** define atributos opcionais que podem ser adicionados pelos outros padrões como PKCS#6, PKCS#7, PKCS#8 e PKCS#10.
- **PKCS#10:** descreve a sintaxe de uma requisição de Certificado digital feita a uma autoridade certificadora a qual assinará a CSR (*Certificate Signing Request*).
- **PKCS#11:** define uma API, chamada Cryptoki (*Cryptographic Token Interface*) para comunicação com *hardware*.
- **PKCS#12:** descreve a sintaxe para o armazenamento e transporte de chaves públicas e certificados digitais.
- **PKCS#13:** em desenvolvimento, descreverá um sistema criptográfico utilizando curvas elípticas.
- **PKCS#14:** em desenvolvimento, descreverá um gerador de números pseudo-aleatórios.
- **PKCS#15:** descreve um formato para permitir a interoperabilidade de chaves, certificados, senhas e PIN´s entre aplicações e *hardware*. Com relação a circuitos integrados (cartões inteligentes), a RSA orienta a referência do padrão ISO 7816-15 e retirou do PKCS#15 as especificações equivalentes.

2.5.8 Identificação de usuário por biometria

Um modo pouco utilizado, mas muito eficiente de se identificar um usuário em um pagamento eletrônico é a verificação biométrica. A biometria menos invasiva, com as técnicas atuais, é a imagem da face. É perfeitamente viável armazenar-se uma fotografia em cartões inteligentes associada aos dados biográficos do dono do cartão. A título de ilustração, em documentos de viagem internacionais a *International Civil Aviation Organization* (ICAO) recomenda uma única biometria a ser armazenada nos chips de

passaportes e carteiras de identidade: a imagem da face. Multi-aplicações embarcadas em cartões podem ser compostas de aplicativos de identificação pessoal e de pagamento eletrônico em um mesmo cartão inteligente. O armazenamento de dados biométricos em cartões inteligentes de pagamento eletrônico ainda não é algo comum e não existem dispositivos específicos para leitura, porém o uso de dispositivos com telas de vídeo como PDA´s, celulares e PC´s é comum em pontos de venda.

Alguns *hardwares* biométricos, como câmeras digitais, têm um ciclo de vida relativamente curto e constantemente têm descontinuada a sua linha de produção. A norma ISO/IEC 19784 define um conjunto de API´s para produtos biométricos e especificamente a primeira parte desta norma, a ISO 9899, é conhecida popularmente como BioAPI. Jain et al (2008, p. 513) ilustra que a BioAPI possui um *framework* que suporta, via *services providers*, a instanciação de componentes de diversos fornecedores, minimizando o impacto do ciclo de vida curto dos *hardwares* biométricos e possibilitando a troca de fornecedores, se necessário.

Nesta dissertação, as mesmas funções descritas para se armazenar um certificado digital poderão ser utilizadas para se armazenar uma imagem da face em um cartão inteligente adequando-se os tamanhos dos variáveis multidimensionais.

2.6. AVALIAÇÃO DA SEGURANÇA ELETRÔNICA DE UM ESQUEMA CRIPTOGRÁFICO

De acordo com Goldreich (2001, p. 60), há duas abordagens para se definir a segurança de um esquema criptográfico: no contexto da teoria da informação ou no contexto da complexidade computacional.

Para o contexto da teoria da informação, Claude Shannon, em 1949, escreveu um artigo chamado *Communication Theory of Secrecy Systems* no *Bell System Technical Journal*, o qual influenciou enormemente o estudo científico da criptografia por ter sido o pioneiro na prova de um sistema incondicionalmente seguro independentemente do poder computacional e do tempo disponível para um adversário. Antes deste artigo, em 1948, Shannon já havia publicado as bases da disciplina teoria da informação no artigo *A Mathematical Theory of Communication*, também publicada pelo *Bell System Technical*

Journal, onde o problema de engenharia central era a transmissão de informação em cima de um canal ruidoso. A definição de segurança para Shannon diz que se o texto cifrado contém alguma correlação estatística com o texto em claro, o esquema criptográfico é considerado inseguro. Esta abordagem oferece as maiores garantias possíveis de segurança, no entanto, normalmente mostra-se impraticável. No caso do problema de ciframento, por exemplo, pode-se provar que para obtermos segurança incondicional, é necessário o uso de chaves totalmente aleatórias do mesmo tamanho da mensagem a ser transmitida, requisitos extremamente difíceis de se obter na prática.

Para o contexto da complexidade computacional, a definição de segurança de um esquema criptográfico diz que não importa se o texto cifrado contenha alguma informação sobre o texto em claro, porém, importa se a informação pode ser extraída. Em outras palavras, não importa se é possível, mas se é computacionalmente viável. Esta abordagem obviamente resulta em criptosistemas mais realizáveis do que a segurança incondicional.

Menezes et al (1996, p. 42) classifica os modelos de avaliação de segurança em: segurança incondicional, segurança baseada em complexidade teórica, segurança comprovada, segurança computacional e segurança *ad hoc*. Stinson (2005, p. 45) classifica os modelos de avaliação de segurança em segurança computacional, segurança comprovada e segurança incondicional.

Segurança incondicional: acontece quando um criptosistema não pode ser quebrado independentemente do poder computacional ou tempo disponíveis para um adversário. Exemplo: *One-Time Pad*.

Segurança computacional: acontece quando a melhor técnica para se quebrar um criptosistema requer um número muito grande de operações. Exemplo: SHA2 com chave de 256 bits.

Segurança comprovada: acontece quando a quebra de um criptosistema de acordo com um rigoroso e preciso modelo adversarial pode ser formalmente reduzida a resolução de um problema matemático tido como de difícil solução, por exemplo, a dificuldade de fatorar números inteiros. A segurança comprovada ainda é uma área de intensa pesquisa, no entanto, já impacta de maneira decisiva as nossas tarefas cotidianas. Por exemplo, o

RSA com o preenchimento OAEP (que possui segurança comprovada) é encontrado descrito em Stallings (2006, p. 280).

2.7. MODELO ADVERSARIAL

Genericamente, ataques em esquemas criptográficos tentam obter o texto original a partir do texto cifrado, ou mais drasticamente, tentam deduzir a chave de criptografia. Stinson (2005, p. 26) relaciona quatro tipos de ataques em esquemas de criptografia: *ciphertext only attack, known plaintext attack, chosen plaintext attack, chosen ciphertext attack*. Já Menezes et al (1996, p.41) descreve que os tipos de ataques em esquemas criptográficos podem ser: *ciphertext only attack, known plaintext attack, chosen plaintext attack, adaptative chosen plaintext attack, chosen ciphertext attack e adaptative chosen ciphertext attack.*

- ***Ciphertext only attack***: um adversário tenta deduzir a chave criptográfica ou o texto original apenas observando o texto cifrado.
- ***Known plaintext attack***: um adversário possui uma quantidade de texto original e correspondentes textos cifrados.
- ***Chosen plaintext attack***: um adversário obtém acesso temporário à máquina de ciframento. Daí ele pode escolher textos de entrada e construir o correspondente texto cifrado para utilização posterior.
- ***Adaptative chosen plaintext attack***: é um ataque do tipo *chosen plaintext attack* em que a escolha do texto de entrada pode depender de respostas anteriores.
- ***Chosen ciphertext attack* (CCA)**: um adversário obtém acesso temporário à máquina de deciframento. Daí ele pode escolher um texto cifrado e construir o correspondente texto de entrada para utilização posterior.
- ***Adaptative chosen ciphertext attack* (CCA2)**: é um ataque do tipo *chosen ciphertext attack* em que a escolha do texto cifrado pode depender de respostas anteriores.

É interessante notar que, conforme anteriormente comentado, o criptosistema RSA proposto inicialmente é vulnerável a *chosen ciphertext attacks* (CCA´s) e não é utilizado na prática, mas soluções para tornar o RSA, CCA seguro, existem. Conforme citado anteriormente, e descrito em Stallings (2006, p. 279), a técnica probabilística OAEP é

uma alternativa para se introduzir a segurança CCA no criptosistema RSA puro.

3. CARTÕES CRIPTOGRAFICAMENTE INTELIGENTES

Para Chen (2000, p. 3), o termo cartão inteligente (*smart card*), refere-se a qualquer cartão que contenha um chip embutido no mesmo para realizar o armazenamento e processamento de informações por circuitos eletrônicos. Além disto, o termo cartão **criptograficamente** inteligente refere-se a cartões inteligentes acrescidos de coprocessadores especializados em alguns criptosistemas específicos, como RSA e ECC. Rankl e Effing (2004, p. 18) ilustram no Gráfico 3.1 uma classificação para cartões inteligentes.

Uma das características desejáveis em um sistema de pagamento eletrônico é que se possa confiar em algum dispositivo inviolável. Nesta dissertação, assume-se que esta característica é provida por cartões criptograficamente inteligentes.

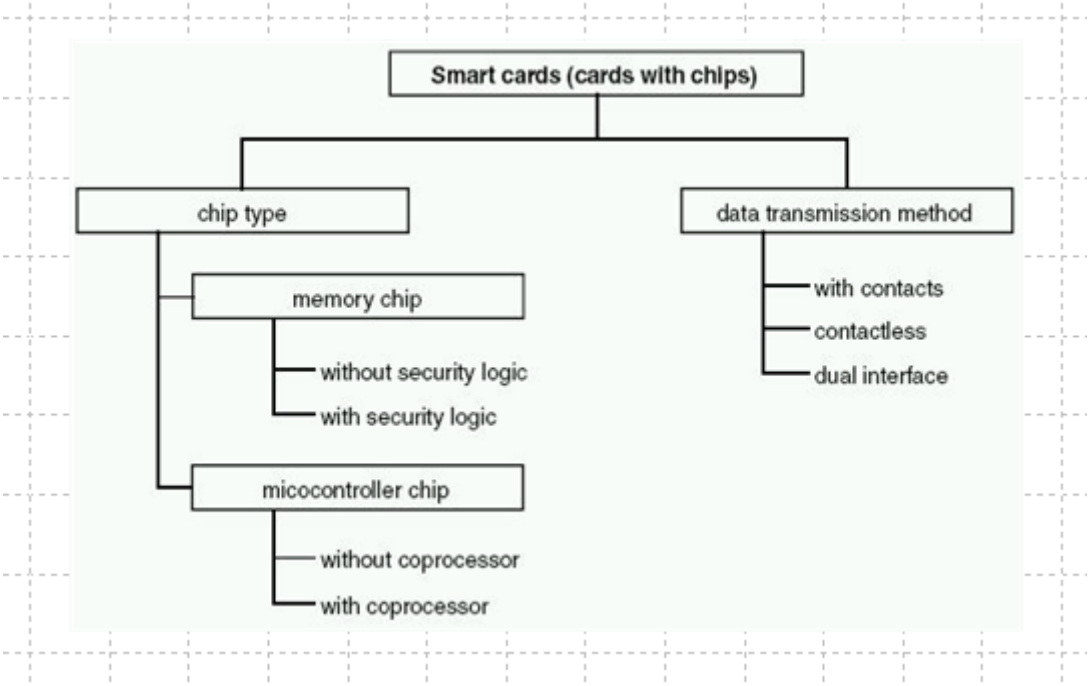

Gráfico 3-1 – Classificação para cartões inteligentes.

Fonte: Rankl e Effing (2004, p. 18).

3.2. ÁREAS DE APLICAÇÃO

A Smart Card Alliance, SCA1 (2008) classifica sete grandes áreas de aplicações de cartões inteligentes: identificação corporativa, finanças, Governo, saúde, identidade de pessoas, telecomunicações e transporte. Para o presente trabalho, pode-se aglutinar pontos semelhantes, sem perda de conteúdo, e classificar três grandes áreas da indústria: Governo, finanças e telecomunicações.

Especificamente no foco desta dissertação, existe uma estatística da Sun, disponível em Sun3 (2008), a qual aponta um total de aproximadamente 6 bilhões de dispositivos com suporte a Java no mercado mundial. A Tabela 3.1 ilustra os números da tecnologia Java, segundo a Sun.

Tabela 3-1– Estatísticas Java.
Fonte: Sun3 (2008)

Dispositivos móveis com Java
20 milhões de set-top-boxes para TV digital com Java
2,1 bilhões de telefones celulares e PDAs com Java
3,5 bilhões de Java Cards

3.1.1 Governo

A utilização de cartões inteligentes como um dos instrumentos para o Estado prestar serviços aos cidadãos de forma confiável e evitando-se fraudes ainda é incipiente no Brasil. Uma experiência de Governo, citada em ICP1 (2008), foi a construção, no ano de 2001, de uma infraestrutura de chaves públicas brasileira chamada ICP-Brasil que possibilita armazenar-se o par de chaves de cada Certificado Digital em um dispositivo inviolável, como um cartão inteligente ou um *token*, o qual fica de posse do dono do Certificado Digital.

Alguns países apresentam uma padronização no tema relacionado a cartões inteligentes para o âmbito de Governo Eletrônico. Como exemplo, o Governo da Austrália, criou o

National Smartcard Framework que, conforme pode ser visto em NSF (2008) cita que "*smartcards are emerging as an important technology in a variety of online, offline and hybrid applications in the public and private sectors. They have the potential to enhance service delivery, improve citizens' privacy and increase security against identity theft and fraud.*".

Várias áreas de aplicação, como a verificação da segurança de documentos em papel tais como carteiras de identidades, passaportes, cartões de benefícios sociais, selos, certidões, entre outros, necessitam, em princípio, de uma verificação *online* em algum sistema de informação central. Uma das vantagens de se transformar estes documentos de papel em cartões inteligentes é a possibilidade de se executar algumas funções eletrônicas sem o custo da rede de comunicação como, por exemplo, em localidades internacionais sem acesso à rede. Em aplicações internacionais como é o caso de esquemas de pagamento eletrônico e passaportes eletrônicos, é inviável o acesso *online* e tempestivo a todos os serviços de um país e um protocolo relevante deve suportar a característica de operar *offline* em algumas situações. Além de viabilizar alguns procedimentos, a execução *offline* serve como contingência em caso de indisponibilidade da rede. Evidentemente que este procedimento deve restringir-se às tarefas relativas à verificação de segurança, uma vez que um cartão inteligente não deve ser utilizado como um banco de dados, visto que há questões de privacidade que são de difícil manuseio, por exemplo, o caso do anonimato.

3.1.2 Finanças

A Associação Brasileira das Empresas de Cartões de Crédito e Serviços (ABECS) vem realizando um acompanhamento do mercado de cartões no Brasil e conforme ilustra a Tabela 3.2 abaixo, o número de cartões acumulados até outubro de 2008 está em 489 milhões de cartões. Do total de cartões em circulação no Brasil, uma pesquisa da Frost & Sullivan citada em Computerworld1 (2006) mostrou que em 2005 haviam quase 80 milhões de cartões inteligentes em circulação e como, pela pesquisa da ABECS existiam em circulação 338 milhões de cartões naquele ano, dá uma proporção de aproximadamente 25% da base instalada no ano de 2005. Expandindo-se estes números até out/2008, chega-se a algo em torno de 120 milhões de cartões inteligentes, em números atuais.

Tabela 3-2– Mercado brasileiro de cartões.

Fonte: ABECS (2008)

Qtde de cartões (em milhões)	**2002**	**2003**	**2004**	**2005**	**2006**	**2007**	**2008** (out)
CRÉDITO	42	45	53	68	79	93	**107**
DÉBITO	82	105	138	171	187	201	**214**
FIDELIDADE	59	71	86	99	118	144	**167**
TOTAL	182	222	277	338	384	438	**489**

3.1.3 Telecomunicações

O mercado para o Serviço Móvel Celular é acompanhado pela Anatel (2008) e conforme ilustra a Tabela 3.3 abaixo, a participação do uso de celulares GSM e 3G com cartões inteligentes até outubro de 2008 já chega a aproximadamente 128 milhões de celulares, em um mercado que hoje conta com uma base instalada de quase 145 milhões de aparelhos, o que faz com que celulares com cartões inteligentes representem em torno de 89 % da base instalada.

Tabela 3-3– Mercado brasileiro de celulares, por tecnologia.

Fonte: Anatel (2008)

TECNOLOGIA	**OUT/2008**	
	QUANTIDADE	**PARTICIPAÇÃO**
AMPS	12.304	0,01%
TDMA	1.882.473	1,30%
CDMA	14.020.102	9,68%
GSM	**127.078.665**	87,76%
3G	**1.262.576**	0,87%
CDMA2000	466.469	0,32%
Term. Dados	73.029	0,05%
TOTAL	**144.795.618**	100,00%

3.3. ARQUITETURA DE CARTÕES INTELIGENTES

Historicamente, conforme cita Anderson (2008, p. 500), as primeiras patentes relacionadas a cartões inteligentes datam do final dos anos 60 e o primeiro grande impulsionador de cartões inteligentes na indústria foi, no final dos anos 80, o surgimento *do Subscriber Identity Module* (SIM) para celulares do tipo GSM. Na área financeira, o trabalho pioneiro no uso de cartões inteligentes veio da França, a partir de 1994, enquanto que no resto da Europa a utilização só veio a ganhar impulso a partir de 2005. Basicamente, um cartão criptograficamente inteligente é um microcontrolador composto de microprocessador, coprocessador criptográfico, memória e interface de entrada/saída serial integrados em um único chip. Este chip fica encapsulado em um cartão, podendo ser acessado através de contato, sem contato com radiofreqüência (RFID) ou mesmo de forma dual com ambas as interfaces. Como exemplos práticos, o Gráfico 3.2 representa um SIM card para celular GSM ou 3G e o Gráfico 3.3 um bilhete de transporte, ambos com contato. O Gráfico 3.4 ilustra um cartão inteligente RFID que é utilizado em passaportes eletrônicos enquanto que o Gráfico 3.5 ilustra um cartão dual, ou seja, com as interfaces com contato e sem contato.

Com relação ao microprocessador, os cartões inteligentes podem usar um conjunto de instruções *Reduced Instruction Set Computer* (RISC) ou *Complex Instruction Set Computer* (CISC). Exemplificando, os cartões da família SmartMX da Philips utilizam microprocessadores CISC de 8 bits enquanto que os cartões da família SLE88 da Infineon utilizam microprocessadores RISC de 32 bits.

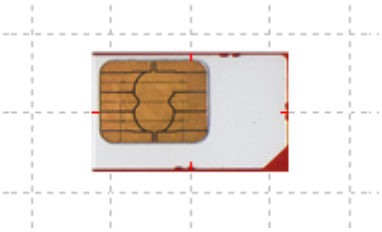

Gráfico 3-2 – SIM card GSM – cartão com contato, para setor de telecomunicações. Fabricante Gemalto.

Gráfico 3-3 – Bilhete de transporte com chip – cartão com contato, para setor de transporte. Fabricante Sagem Orga.

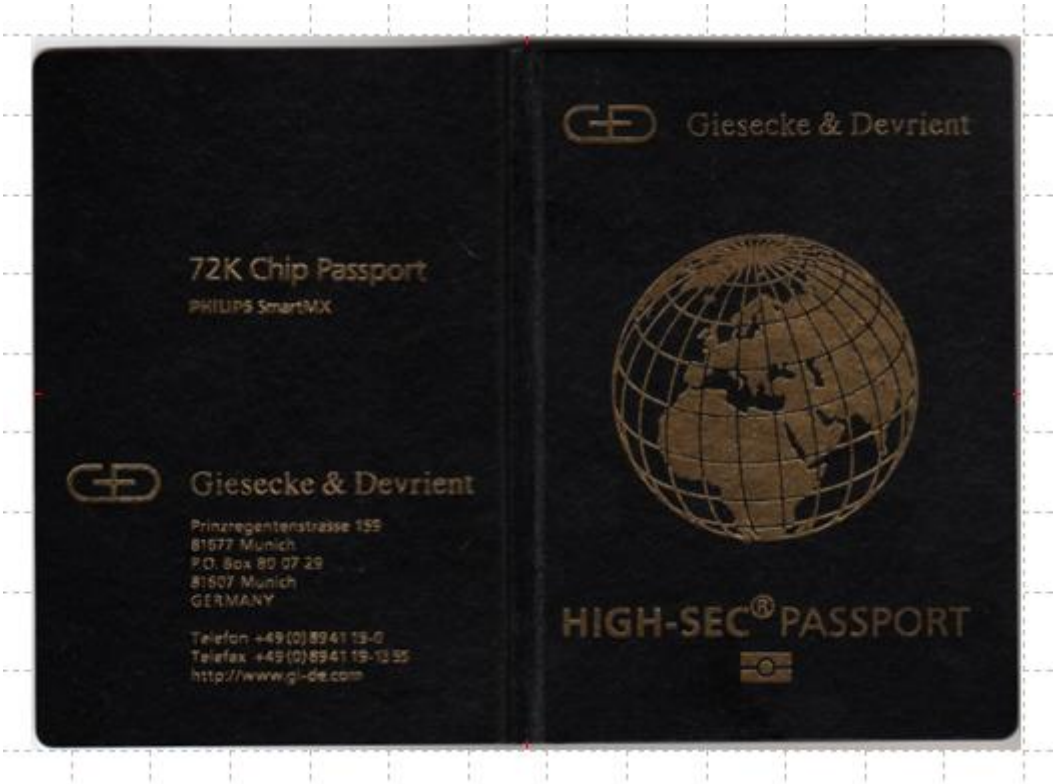

Gráfico 3-4 – Passaporte eletrônico – cartão sem contato, para setor Governo. Fabricante GD Burti.

Gráfico 3-5 – Cartão de identidade eletrônico – cartão dual, para setor de Governo. Fabricante Oberthur.

O Gráfico 3.6 ilustra os contatos de um circuito integrado que compõe um cartão inteligente o qual segue o padronizado na norma ISO 7816. Os oito contatos têm as seguintes funcionalidades:

- Vcc: tensão que provê a alimentação do chip do cartão com valores aproximados entre 3 e 5 V.
- *Reset*: este ponto de contato tem como objetivo promover um *reset* a quente, ou seja, ocorre o envio de um sinal de *reset* ao chip. Já um *reset* a frio ocorre quando se desliga a fonte de alimentação, como por exemplo, quando se retira e insere um cartão na leitora.
- *Clock*: caso os cartões inteligentes não tenham gerador de *clock* interno no chip, há a necessidade de um sinal de *clock* externo a partir do qual o *clock* interno é derivado.
- GND: é o ponto de contato que serve de tensão de referência, normalmente esta tensão é de 0 V.
- Vpp: é um ponto opcional e utilizado em cartões inteligentes antigos, era utilizado para programar a EEPROM, é mantido para compatibilidade.
- I/O: é um ponto *half-duplex* para entrada ou saída de dados do cartão.
- Contatos disponíveis: reservados para uso futuro.

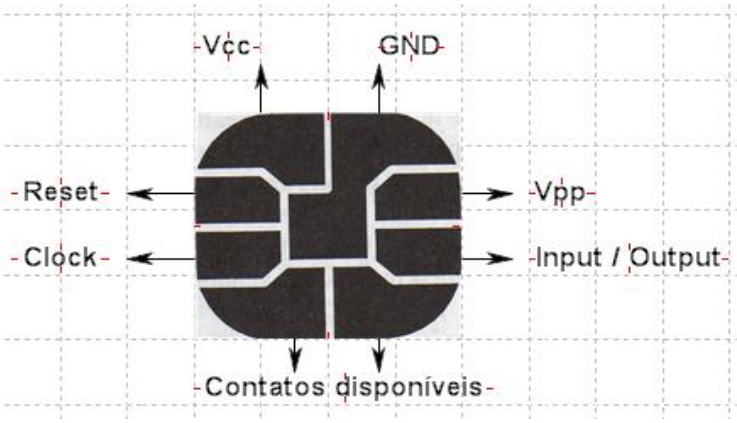

Gráfico 3-6 – Representação dos oito contatos de um circuito integrado de cartões inteligentes.

Fonte: Sherif (2000, p. 396).

O Gráfico 3.7 ilustra os tipos de um cartão inteligente com relação às dimensões padronizadas pela ISO 7816. O formato ID-000 é comumente utilizado em celulares, o ID-1 é preferido em cartões bancários enquanto que o formato ID-00 é o menos utilizado.

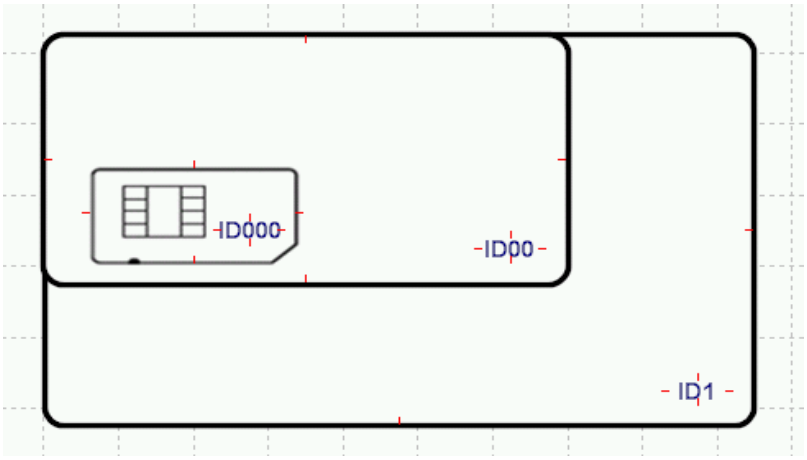

Gráfico 3-7 – Dimensões de um *smart card*.

Fonte: Adaptado de Rankl e Effing (2004, p. 31).

A unidade central de processamento (*Central Processing Unit* - CPU) dos cartões é, de modo geral, de 8 bits a 32 bits e usa normalmente um conjunto de instruções baseado no Motorola 6805 ou mais frequentemente no Intel 8051/80C51 podendo ser RISC ou CISC. Normalmente, o *clock* interno varia de 1 MHz a 30 MHz sendo que o mesmo pode ser obtido a partir de um *clock* externo o qual, em geral, está na faixa de 1 MHz a 10

MHz. Além desta unidade central de processamento, os cartões mais modernos também possuem um ou mais coprocessadores criptográficos (*Numerical Processing Unit* – NPU) com especializações para diminuir o tempo de execução de algoritmos como RSA e AES.

Com relação à memória, em um cartão inteligente há três tipos de necessidades: memória não volátil e não gravável, memória não volátil e gravável, e memória volátil e gravável, chamadas respectivamente de ROM, EEPROM e RAM. Como regra geral, programas criados por usuários e variáveis estáticas, são gravados na EEPROM, o sistema operacional é gravado na ROM enquanto que objetos temporários ficam gravados na RAM. O Gráfico 3.8 traz um esquemático que ilustra um microcontrolador de um cartão inteligente, o microcontrolador contém CPU, NPU, memórias e as interfaces de entrada/saída com o mundo externo.

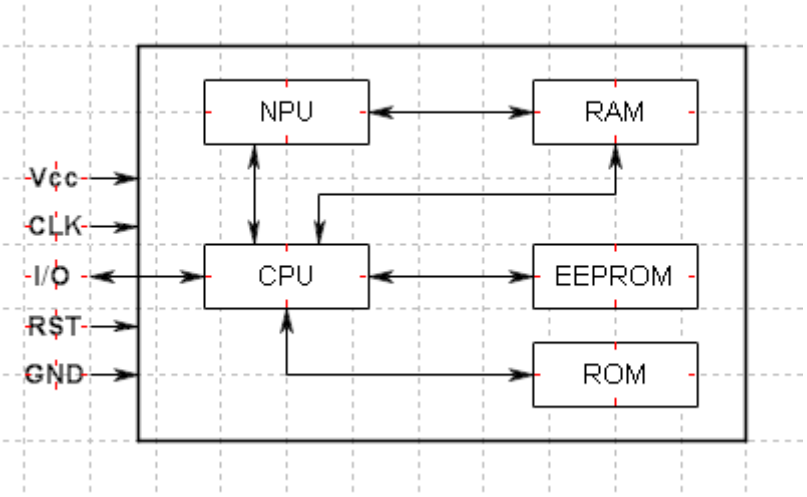

Gráfico 3-8 – Esquemático básico de um microcontrolador de um cartão inteligente.

Fonte: Rankl e Effing (2004, p. 20).

Para se comunicar com um leitor de cartões inteligentes e conseqüentemente com um computador existe um protocolo chamado *Application Protocol Data Unit* (APDU) que permite a comunicação em modo *half duplex*, ou seja, os dados só podem ser enviados em um sentido por vez. O Gráfico 3.9 ilustra este modelo de comunicação fisicamente.

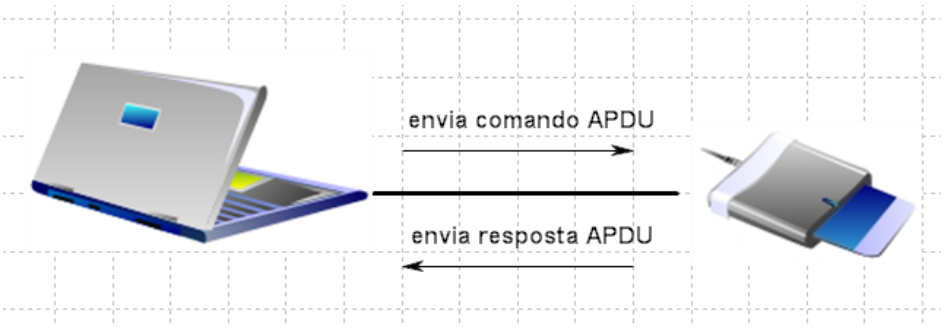

Gráfico 3-9 – Modelo físico de comunicação para cartões inteligentes

O protocolo APDU é especificado na ISO 7816-4 e tem a seguinte estrutura extraída de ISO78164 (1998):

- Comando APDU:

Header obrigatório				**Campos opcionais**		
CLA	INS	P1	P2	Lc	Dados	Le

Onde:

CLA – classe de instrução, 1 byte;

INS – código de instrução, 1 byte;

P1 – parâmetro de entrada, 1 byte;

P2 – parâmetro de entrada, 1 byte;

Lc – comprimento do campo “Dados”, 1 byte;

Dados – comandos, Lc bytes;

Le – comprimento esperado na resposta, 1 byte.

- Resposta APDU:

Campo opcional	**Trailer obrigatório**	
Dados	SW1	SW2

Onde:

Dados: resposta do cartão inteligente, Le bytes;

SW1: indica se o comando foi executado corretamente, 1 byte;

SW2: indica se o comando foi executado corretamente, 1 byte;

A ISO 7816-4 padroniza os códigos SW1 e SW2 conforme traz a Tabela 3.4.

Tabela 3-4 – Codificação para as *status words* de retorno SW1 e SW2.

Fonte: ISO78164 (1998).

SW1 SW2	**Significado**
9000	**Processamento normal**
61XX	
62XX	**Processamento com avisos (*warnings*)**
63XX	
64XX	**Erros de execução**
65XX	
66XX	
6700	**Erros de verificação**
68XX	
69XX	
6AXX	
6B00	
6CXX	
6D00	
6E00	
6F00	

Os comandos APDU´s podem ser organizados em 4 categorias conforme a Tabela 3.5. Um exemplo destes comandos e respostas pode ser visto no Gráfico 3.10, onde, para um cartão, foi enviada o comando APDU 80 50 00 00 08 48 6F 73 74 52 61 6E 64 1C que objetiva que o cartão retorne um número aleatório, que foi a resposta APDU, 4D 00 40 82 D5 13 09 11 26 AF 01 01 C9 98 E4 95 1E 98 3D 50 24 00 0D B3 EB 55 2C A4 90 00. Para este comando APDU percebe-se que ele se enquadra na categoria 4 da Tabela 3.5 conforme ilustrado abaixo.

Comando APDU:

Header	Lc	Dados	Le
80 50 00 00	08	48 6F 73 74 52 61 6E 64	1C

Resposta APDU:

Dados	SW1	SW2
4D 00 40 82 D5 13 09 11 26 AF 01 01 C9 98 E4 95 1E 98 3D 50 24 00 0D B3 EB 55 2C A4	90	00

Tabela 3-5– Categorias de comandos e respostas APDU´s.

Fonte: ISO78164 (1998).

Cate-goria	**Comando APDU**				**Resposta APDU**		
1	Header					SW1	SW2
2	Header	Le			Dados	SW1	SW2
3	Header	Lc	Dados			SW1	SW2
4	Header	Lc	Dados	Le	Dados	SW1	SW2

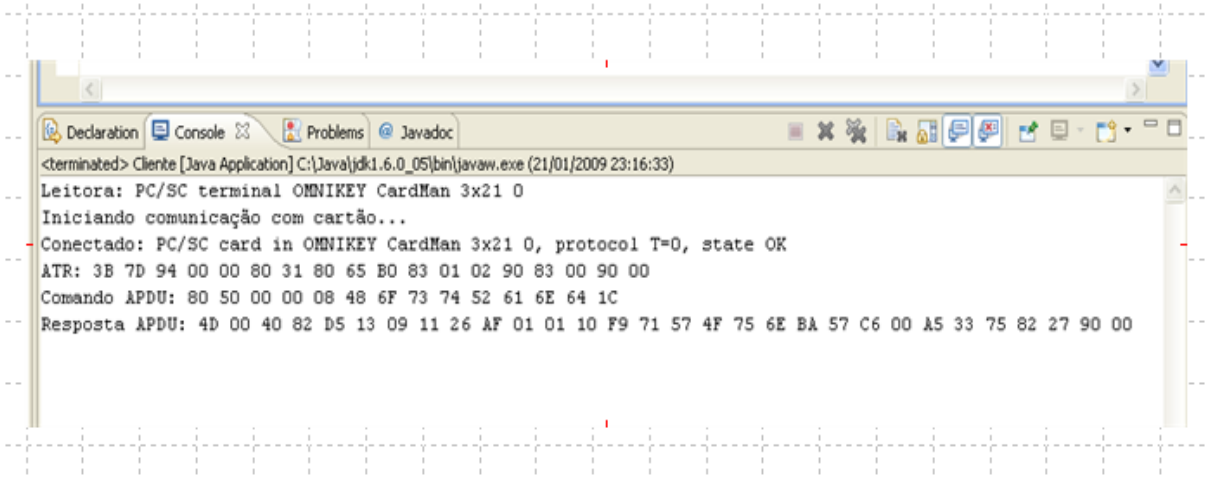

Gráfico 3-10 – Comandos e respostas APDU´s, saída na console do Eclipse.

Normalmente cada fabricante produz cartões com um *Answer to Reset* (ATR) diferente conforme pode ser visto na Tabela 3.6. O parâmetro ATR, portanto, é uma forma de identificação de um mesmo modelo de cartão inteligente pois modelos iguais têm um mesmo ATR.

Tabela 3-6– Exemplos de ATR´s para cartões inteligentes.

Fonte: Rousseau (2008).

Fabricante / modelo	**ATR**
Brazilian NET Digital (Cable TV provider) - Nagra Vision "NASP110 RevA01"	3F FF 95 00 FF 91 81 71 FE 47 00 44 4E 41 53 50 31 31 30 20 52 65 76 41 43 33 65
Sagem Orga SCOSTA 4 K	3F 86 40 FA 80 81 01 52 01 00
Gemalto TOP IM GX4 72 kB	3B 7D 94 00 00 80 31 80 65 B0 83 01 02 90 83 00 90 00
JCOP30 "OP-DI 16k VISA v2 (JCOP30) ORGA"	3B 66 00 FF 4A 43 4F 50 33 30
Schlumberger Cyberflex Access 32K	3B 76 11 00 00 00 9C 11 01 02 02
Oberthur Card Systems: Cosmo 64 RSA V5.4 (ISK Key Set: 404142 .. 4E4F)	3B 7B 18 00 00 00 31 C0 64 77 E9 10 00 01 90 00
German Passport (issued Apr 2007)	3B 89 80 01 00 64 04 15 01 02 00 90 00 EE
Mifare card with 4k EEPROM	3B 8F 80 01 80 4F 0C A0 00 00 03 06 03 00 02 00 00 00 00 69
Gemplus GemXplore 3G USIM	3B 9F 95 80 1F C3 80 31 E0 73 FE 21 1B B3 E2 01 74 83 0F 90 00 88
Cingular GSM SIM Card	3B 16 94 71 01 01 00 27 00

3.4. PADRÕES PARA CARTÕES INTELIGENTES

Padrões foram criados prioritariamente para integração e interoperabilidade. Pode-se dizer que os padrões para cartões inteligentes podem ser classificados em 2 grandes grupos: horizontais e verticais. Os padrões horizontais são aqueles utilizados por várias aplicações diferentes enquanto que os padrões verticais são específicos para uma determinada classe de aplicação. Em um sistema de pagamento eletrônico *online* sem anonimato é importante conhecer, no mínimo, os seguintes padrões horizontais: PC/SC, GlobalPlatform e Java Card.

A especificação PC/SC significa *Interoperability Specification for Integrated Circuits and Personal Computer System* e define uma arquitetura de propósito geral para permitir a comunicação entre cartões inteligentes e computadores pessoais.

A especificação GlobalPlatform (GP) foi criada em 1999 pela Visa para manter e evoluir um padrão chamado OpenPlatform, também criado pela Visa em meados dos anos 90. O objetivo da GlobalPlatform é especificar um ambiente integrado para o desenvolvimento e operação de cartões inteligentes multi-aplicação. Atualmente a GP é constituída de três partes: especificação do cartão, especificação da leitora e especificação do software do sistema.

A especificação Java Card é um conjunto de especificações que possibilita a escrita de programas através do uso da linguagem Java para serem executados em cartões inteligentes e outros dispositivos inteligentes com baixa capacidade de memória e processamento. Há três especificações Java Card, que podem ser vistas no Gráfico 3.11.

- Java Card *Virtual Machine* (JCVM): especificação que define o subconjunto da linguagem Java e a máquina virtual adequada para cartões inteligentes.
- Java Card *Runtime Environment* (JCRE): especificação que descreve o comportamento em modo de execução dos applets tais como gerenciamento de memória e gerenciamento do applet.
- Java Card *Application Programming Interface* (JCAPI): especificação que define o subconjunto de classes Java para a programação de cartões inteligentes.

O Gráfico 3.11 também ilustra que os cartões inteligentes têm seu próprio sistema operacional e este varia de acordo com o fabricante dos cartões.

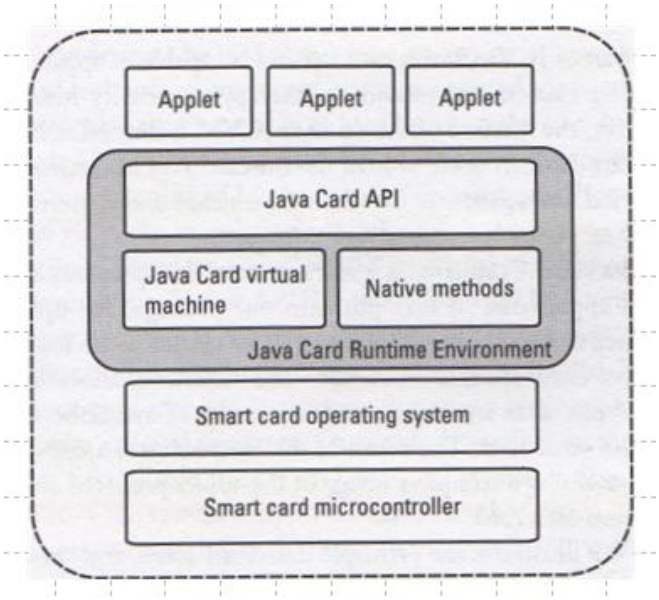

Gráfico 3-11 – Arquitetura Java Card.

Fonte: Hassler (2002, p. 81).

3.5. CRIAÇÃO DE UM APPLET JAVA CARD

Devido às características de restrição de memória e capacidade de processamento dos cartões inteligentes, a linguagem Java Card é um subconjunto da linguagem Java com algumas características não suportadas, conforme ilustrado na Tabela 3.7.

Tabela 3-7 – Características linguagem Java Card.

Fonte: Chen (2000).

Suporta as seguintes características do Java	Tipos boolean, byte e short
	Arrays unidimensionais
	Mesmo compilador
	Pacotes, classes, interfaces e exceções
	Orientação a objeto
Não suporta as seguintes características do Java	Tipos long, double, float, character e string
	Arrays multidimensionais
	Carregamento dinâmico de classe
	Gerenciador de segurança (porém pode-se implementar diretamente na JCVM)
	Garbage collection
	Multiple threads
	Serialização de objetos

O Gráfico 3.12 representa o processo de geração de um applet Java Card para ser armazenado em um cartão. Um código fonte Java (.java) é compilado via Java Development Kit (JDK) com ligação para as bibliotecas Java Card criando classes Java (.class). O Java Card Kit da Sun possui um utilitário de conversão que gera *converted* applets (.cap). Os applets .cap podem ser armazenados no cartão a partir do uso de alguma ferramenta de carregamento como, por exemplo, o GPShell. Este procedimento de carga do applet em um cartão inteligente depende do fabricante do cartão.

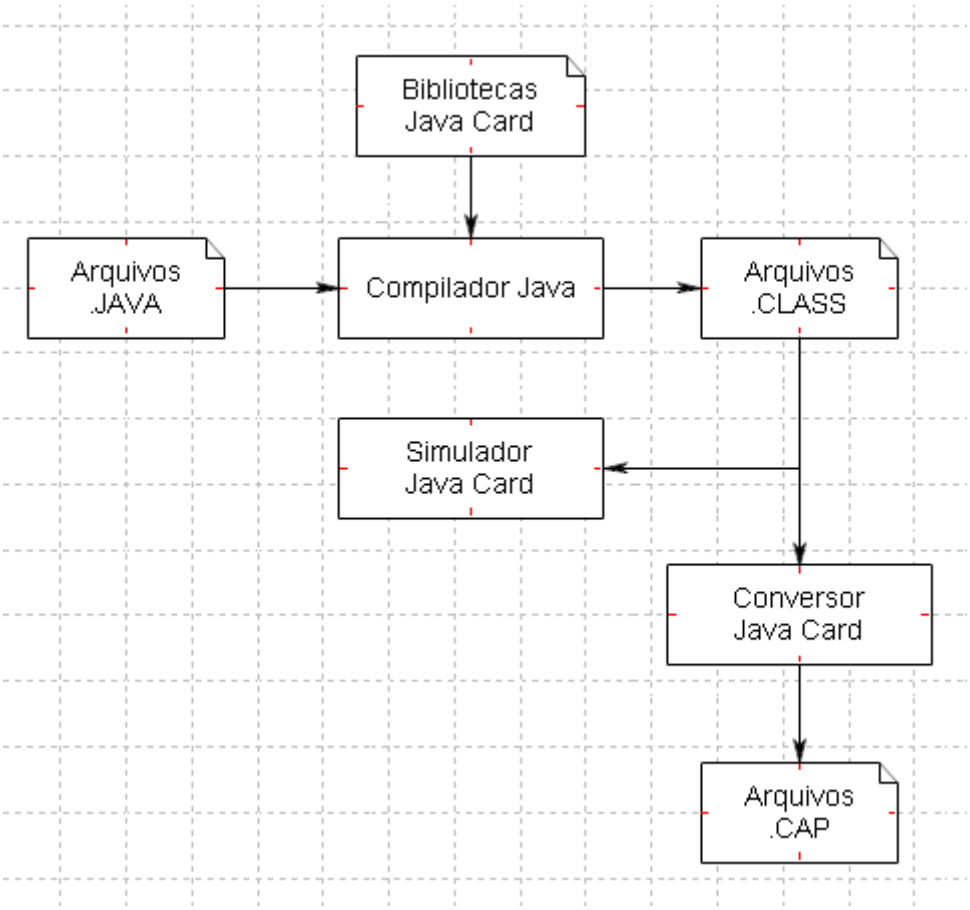

Gráfico 3-12 – Criação de um applet Java Card.

Fonte: Sun1 (2006).

3.6. CLIENTE JAVA

A versão 6 da linguagem Java possui uma API para comunicação com cartões inteligentes chamada *Smart Card* IO e um provider de segurança chamada SunPCSC. A presença destes elementos na especificação Java evita a necessidade do uso da Java *Native Interface* (JNI) para prover comunicação entre as classes Java e algum código nativo escrito em outra linguagem.

4. PROTOCOLOS PARA PAGAMENTO ELETRÔNICO

Para se entender o conceito de pagamento eletrônico é necessário inicialmente entender o que é comércio eletrônico. O termo comércio eletrônico, utilizado por Law et al (1996, p. 3) refere-se a qualquer transação financeira que envolva a transmissão de pacotes de informação na forma eletrônica. Para Sherif (2000, p. 2) comércio eletrônico é *"the set of relations totally dematerialized that economic agents have with respect to each other"*. Já **pagamento eletrônico** é um tipo particular de comércio eletrônico onde o pagamento é realizado através de um comando emitido por uma terceira parte (por exemplo, uma autorização de um Banco). Além disto, pode-se definir dinheiro eletrônico como um tipo específico de esquema de pagamento eletrônico guiado por certas propriedades criptográficas (Law et al (1996, p.3)).

Um modelo mais genérico é mostrado no Gráfico 4.1 com os seguintes atores:

- comprador: ator que dispende o dinheiro;
- vendedor: ator que recebe o dinheiro;
- instituição financeira com interação com o consumidor: ator que emite o dinheiro eletrônico para o consumidor a partir do dinheiro físico;
- instituição financeira com interação com o vendedor: ator que emite o dinheiro físico para o vendedor a partir do dinheiro eletrônico;
- árbitro: ator que resolve disputas no sistema de pagamento;
- terceira parte confiável: ator que pode atuar como autoridade certificadora em que funcionalidades cartoriais, anonimato e geração de recibos podem ser viabilizadas.

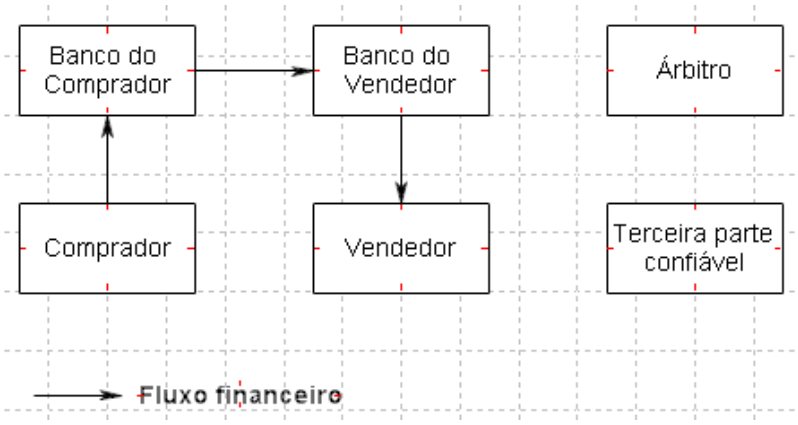

Gráfico 4-1 – Atores de um sistema de pagamento eletrônico

Fonte: Adaptado de Abad-Peiro et al (1996, p. 2)

Há duas grandes classificações para estes sistemas, a primeira é baseada no estado da conexão de rede e privacidade durante o pagamento, e a segunda é baseada na direção em que o fluxo financeiro do pagamento ocorre.

Inicialmente, os sistemas de pagamento podem ser do tipo:

- *online* e não-anônimo;
- *online* e anônimo;
- *offline* e não-anônimo;
- *offline* e anônimo.

Um sistema *online* é aquele em que há a necessidade premente de se ter uma comunicação de rede de dados no momento da execução de uma transação, enquanto que um sistema *offline* permite a comunicação em um momento posterior.

A característica de anonimato é utilizada para a garantia da privacidade dos atores envolvidos em uma transação, porém operações eletrônicas totalmente anônimas podem ser prejudiciais a uma economia, pois podem permitir operações criminosas tais como lavagem de dinheiro. Um protocolo que implemente anonimato deve prever a possibilidade da remoção do anonimato em caso de necessidade ou exigência judicial.

Em segundo lugar, os sistemas de pagamento podem ser do tipo:

- cartão de crédito;
- cartão de débito;

- cartão de dinheiro eletrônico.

De forma simplificada, um pagamento baseado em cartão de crédito é chamado de *pay later*, ou seja, o pagamento é feito posteriormente à obtenção de uma mercadoria. O pagamento baseado em cartão de débito é chamado de *pay now*, ou seja, o pagamento é feito simultaneamente à obtenção da mercadoria. Finalmente, o pagamento com dinheiro eletrônico é chamado de *pay before* uma vez que há a necessidade de se introduzir uma quantia monetária no cartão antes da realização de qualquer compra.

4.2. ESTADO DA ARTE

O período de maior aparecimento de novas propostas para protocolos de pagamento eletrônico foi durante o crescimento da bolha da Internet, de 1997 a 2001. Neste período surgiram algumas propostas de protocolos abertos e predominantemente não abertos que, de certa forma, permitiram que o comércio eletrônico de bens pudesse ser implementado em uma rede não confiável como a Internet. Para isto estes protocolos basearam-se na presunção de dispositivos invioláveis como cartões inteligentes e leitoras de cartões com teclado numérico. Uma vez que o modelo de protocolo para pagamentos eletrônicos foi estabilizado e atendeu às necessidades das instituições financeiras, os estudos acadêmicos se focaram em propostas de protocolos para pagamentos que pudessem prover um certo grau de anonimato e não rastreabilidade bem como propostas de protocolos que permitam a utilização de redes não necessariamente seguras e de leitoras de cartões mais simples como é o foco de estudo desta dissertação.

4.2.1 Anonimato e não rastreabilidade

O protocolo *Compact E-Cash*, proposto por Camenisch et al (2006), propõe que o usuário seja capaz de retirar moedas de uma bolsa com 2L moedas e que o espaço necessário para armazená-las seja apenas L e não 2L bem como a complexidade do protocolo de retirada seja proporcional a L e não 2L. Além disto, este protocolo preenche os requisitos de anonimato e não rastreabilidade. Um problema deste protocolo é que o número de moedas retiradas durante o uso e os valores das moedas é predeterminado pelo sistema. Outro problema é a não divisibilidade dos pagamentos.

Tendo um protocolo eficiente para retirada de várias moedas, o próximo passo é ter-se um protocolo eficiente para o protocolo de compra. O primeiro protocolo divisível para moeda eletrônica foi proposto por Okamoto e Ohta (1992). O protocolo de compra deste esquema requer 20 exponenciações modulares feitas pelo usuário e provê anonimato mas a rastreabilidade não é provida. A eficiência deste protocolo foi melhorado por Chan et al (1998) para 10 exponenciações modulares porém não provê anonimato.

O primeiro anônimo, não rastreável e divisível protocolo de moeda eletrônica foi proposto por Nakanishi e Sugiyama (2002). Porém este protocolo necessita de computação elevada e de uma terceira parte confiável para detecção do usuário em caso de *double spending*. Mais recentemente, Canard e Gouget (2007) apresentaram um protocolo *offline* anônimo, não rastreável e divisível sem a necessidade de uma terceira parte confiável. Eles propuseram uma construção genérica e depois aplicaram-na à construção de Nakanishi e Sugiyama. Este protocolo é o primeiro a prover anonimato ao usuário tal que é impossível para alguém fazer algum relacionamento entre a compra e a retirada da *wallet*. Além disto, ele não requer uma terceira parte confiável para revogar o anonimato de algum usuário que gastou mais de uma vez a mesma moeda. O ponto fraco deste protocolo é o custo computacional uma vez que quando o usuário gasta múltiplas moedas de uma vez, o protocolo usa provas de dupla exponenciação.

Outra área de interesse na criptografia é a de transferência de moeda eletrônica entre dispositivos pervasivos de pagamentos como celulares ou cartões inteligentes para, por exemplo, servir de empréstimo monetário entre indivíduos. Chaum e Pedersen (1998) criaram um protocolo para transferência, porém apresenta dois grandes problemas: o usuário receptor de uma moeda deve ter, anteriormente, retirado uma moeda vazia de uma instituição financeira para receber a moeda do usuário emissor; qualquer usuário que pagar com esta moeda poderá identificá-la posteriormente dentro da cadeia de pagamentos feita com esta moeda.

4.2.2 Possibilidade de uso de leitora de cartões maliciosa

Bellare et al (2000) descreveram uma implementação, chamada zIP, para o *framework* proposto em Bellare et al (1995), chamado iKP, o qual cabalmente ilustra uma forma de transação segura utilizando-se certificados digitais para os atores envolvidos em um

pagamento eletrônico. Neste caso, como todo o pagamento é realizado com a premissa de que os atores possuem um certificado digital, podem-se utilizar os benefícios da infraestrutura de chaves públicas para obter não repúdio, distribuição de chaves de forma confiável, criptografia, decriptografia e assinatura digital. Três diferentes tipos de ameaças são consideradas defensáveis no iKP: ataque passivo (*eavesdropper*), ataque ativo e ataque a partir de um dos atores (*insiders*). Desta forma, é possível utilizar-se uma rede não confiável como a Internet e até mesmo leitoras maliciosas desde que uma infraestrutura de chaves públicas seja utilizada pelos participantes do protocolo de comunicação.

4.3. PROJETO DE PROTOCOLOS CONFIÁVEIS

Como um sistema de pagamento eletrônico utiliza determinadas primitivas criptográficas, um protocolo para pagamento eletrônico deve ser elaborado por pessoas que conheçam criptografia para se evitar erros grosseiros como os que aconteceram no famoso protocolo *Wired Equivalent Privacy* (WEP) de redes Wi-Fi IEEE 802.11. Conforme citado por Anderson (2008, p. 666), um dos erros mais flagrantes deste protocolo foi a utilização de um vetor de inicialização que permitia colisões ou repetições, ou seja, não razoavelmente aleatório. Usar um vetor de inicialização em um protocolo é uma técnica cujo objetivo é fazer que um protocolo originalmente determinístico transforme-se em probabilístico, obviamente que se há repetições no vetor de inicialização este não cumpre o seu papel.

Anderson e Needham, em 1995, escreveram um artigo chamado *Robustness principles for public key protocols*, onde os autores expõem oito princípios conforme ilustrado abaixo:

1 - Assinar primeiro e cifrar posteriormente, Anderson e Needham (1995, p. 2).

2 - Uso de chaves diferentes de acordo com a função, Anderson e Needham (1995, p. 4).

3 - Não ser um oráculo para um adversário, Anderson e Needham (1995, p. 5).

4 - Monitorar todos os bits, Anderson e Needham (1995, p. 6).

5 - Não assumir segredos de terceiros, Anderson e Needham (1995, p. 7).

6 – Não assumir determinado formato prévio para uma mensagem recebida, Anderson e Needham (1995, p. 8).

7 – Ser explícito quanto aos parâmetros criptográficos, Anderson e Needham (1995, p. 8).

8 – Ter transparência do projeto do protocolo, Anderson e Needham (1995, p.

9).

Aproximadamente no mesmo período, Abadi e Needham, em 1994, escreveram um artigo mais abrangente chamado *Prudent Engineering Practice for Cryptographic Protocols* que contêm uma série de boas práticas para o projeto de protocolos.

Princípio 1 – Comunicar de forma explícita, Abadi e Needham (1994, p. 2): cada mensagem deve ter o significado auto-contido, ou seja, a interpretação da mensagem deve depender apenas no seu conteúdo. Para exemplificar o Princípio 1, suponha o Gráfico 4.2 abaixo com as seguintes mensagens:

1- I_A, I_B

2- $C[K_{AB}, t]K_{AT}$, $C[K_{AB}, t]K_{BT}$

3- $C[K_{AB}, t]K_{BT}$

Onde:

- a chave K_{AB} é gerada no tempo t e enviada de A para B.
- as chaves K_{AT} e K_{BT} são pré-compartilhadas entre A e T e entre B e T respectivamente.

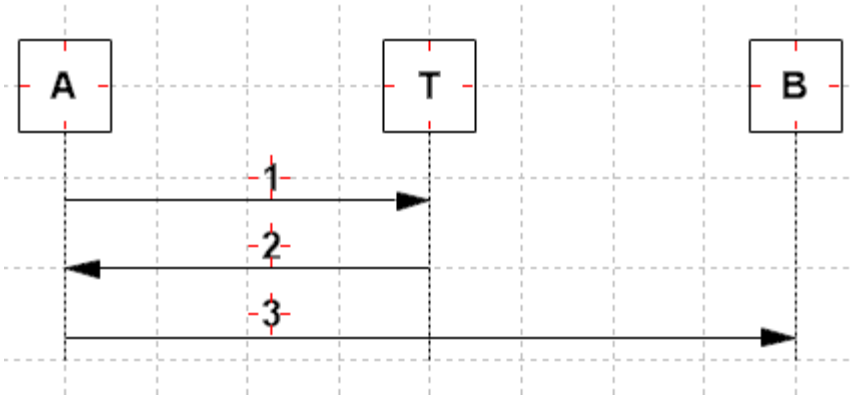

Gráfico 4-2 – Exemplo de comunicação explícita

As mensagens no passo 2 do protocolo exemplo violam o Princípio 1 pois não dependem somente do conteúdo delas, há necessidade de informação de contexto. Uma possível solução seria:

1- I_A, I_B

2- $C[I_A, I_B, K_{AB}, t]K_{AT}$, $C[I_A, I_B , K_{AB}, t]K_{BT}$

3- $C[I_A, I_B, K_{AB}, t]K_{BT}$

Princípio 2 – Agir de forma apropriada, Abadi e Needham (1994, p. 3): deve ser possível que alguém faça uma revisão no projeto do protocolo para verificar se ele é aceitável ou não, ou seja, o protocolo deve ser transparente o suficiente para permitir uma revisão.

Princípio 3 - Dar nomes, Abadi e Needham (1994, p. 5): deve-se explicitar a identidade de quem envia a mensagem.

Para exemplificar o princípio 3, suponha o Gráfico 4.3 abaixo com as seguintes mensagens:

1- $C[N_A, I_A]K_B$

2- $C[N_A, N_B]K_A$

3- $C[N_B]K_B$

Onde:

- K_B e K_A são as chaves públicas de B e A respectivamente
- N_A e N_B são números aleatórios usados apenas 1 vez ("*nonce*")

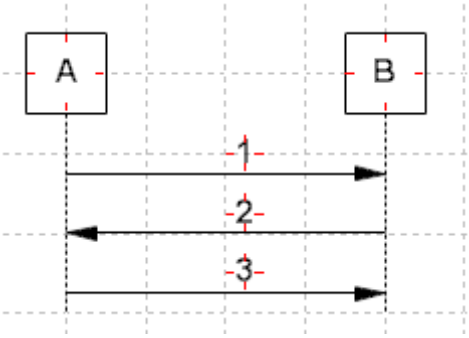

Gráfico 4-3 - Exemplo para o caso de dar nomes.

As mensagens no passo 2 do protocolo exemplo ferem o princípio 3 pois não há informação do nome de quem envia a mensagem, no caso a parte B. Uma possível solução seria:

1- $C[N_A, I_A]K_B$

2- $C[I_B, N_A, N_B]K_A$

3- $C[N_B]K_B$

Princípio 4 - Cifrar apenas quando necessário, Abadi e Needham (1994, p. 9): o projeto de protocolos deve considerar claramente o porquê de se usar criptografia nos passos, pois esta não é sinônimo de segurança. Deve considerar também que a criptografia de chaves públicas provoca um overhead de cerca de 1000 vezes sobre a criptografia simétrica.

Para exemplificar o Princípio 4, suponha o Gráfico 4.4 abaixo com as seguintes mensagens:

1- m, I_A, I_B, $C[N_A, m, I_A, I_B]K_{AT}$
2- m, I_A, I_B, $C[N_A, m, I_A, I_B]K_{AT}$, $C[N_A, m, I_A, I_B]K_{BT}$
3- m, $C[N_A, K_{AB}]K_{AT}$, $C[N_B, K_{AB}]K_{BT}$
4- m, $C[N_A, K_{AB}]K_{AT}$

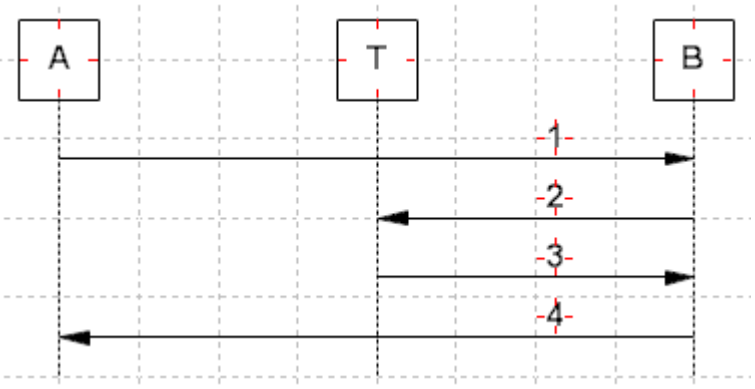

Gráfico 4-4 – Exemplo para o caso de cifrar apenas quando necessário.

As mensagens do passo 1 mostram que o uso da criptografia não está relacionado à confidencialidade de m, I_A e I_B pois elas são transferidas em branco. A criptografia serve para fazer forte a relação entre N_A, m, I_A e I_B. Similarmente acontece com o passo 2 porém nos passos 3 e 4 a criptografia é utilizada para prover confidencialidade.

Princípio 5 – Utilizar a sequência assinatura e ciframento posterior, Abadi e Needham (1994, p. 11): a sequência assinatura e depois ciframento é a que expõe menos bit´s de informação.

Para exemplificar o Princípio 5, suponha o Gráfico 4.5 abaixo com as seguintes mensagens:

1- I_A, T_A, N_A, I_B, X_A, $sig_A[T_A, N_A, I_B, X_A, C[K_{AB}]K_B]$

Onde:

- T_A é um *timestamp* informado por A.
- sig_A é a assinatura com a chave privada de A

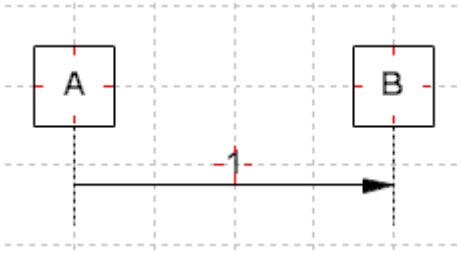

Gráfico 4-5 – Exemplo para o caso de assinar antes de cifrar.

As mensagens no passo 1 do protocolo exemplo ferem o princípio 5 pois, para evitar ataques do tipo *man in the middle*, o correto seria assinar e posteriormente cifrar. Uma possível solução seria:

1- I_A, T_A, N_A, I_B, X_A, $C[K_{AB}, sig_A[T_A, N_A, I_B, X_A, K_{AB}]]$

Princípio 6 – Usar números aleatórios, Abadi e Needham (1994, p. 13): um número aleatório utilizado em um protocolo deve basicamente ter três características: aleatório, não previsível, garantidamente novo.

Para exemplificar o Princípio 6, suponha o Gráfico 4.6 abaixo com as seguintes mensagens:

1- I_A, N_A
2- $C[N_A, K'_{AB}]K_{AB}$
3- $C[N_A]K'_{AB}$
4- N'_B

Onde:

- K'_{AB} é uma nova chave gerada para a sessão.

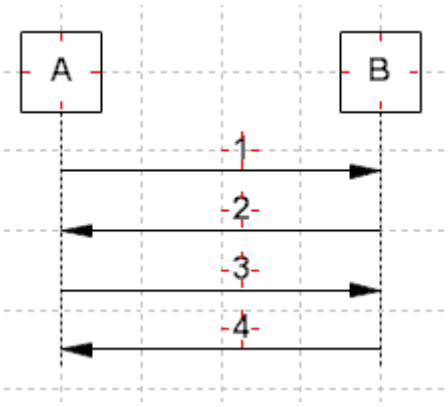

Gráfico 4-6 – Exemplo para o caso de geração de números aleatórios.

Princípio 7 – Não usar números previsíveis, Abadi e Needham (1994, p. 14): deve-se considerar que números seqüenciais são previsíveis e não podem evitar ataques do tipo *replay*.

Para exemplificar o Princípio 7, suponha o Gráfico 4.7 abaixo com as seguintes mensagens:

1- I_A, N_A
2- $C[T_B, N_A]K_{AB}]$

Onde:
- N_A é um número previsível
- T_B é um timestamp informado por B

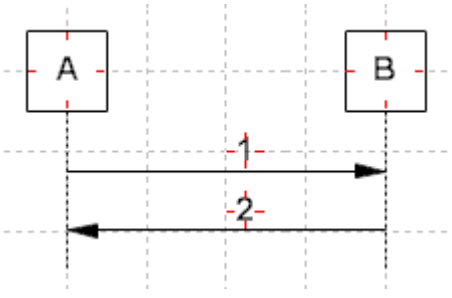

Gráfico 4-7 – Exemplo para o caso de números previsíveis.

Os passos 1 e 2 seriam adequados se o número N_A fosse aleatório, não previsível e não reutilizado. Para o caso de um número seqüencial e previsível, uma possível solução para as mensagens 1 e 2 seria:

1- I_A, $C[N_A]K_{AB}$
2- $C[T_B, C[N_A]K_{AB}]K_{AB}$

Princípio 8 – Usar carimbo de tempo, Abadi e Needham (1994, p. 15): para ser eficiente o carimbo de tempo tem de ser baseado em um relógio confiável.

Princípio 9 – Ter renovação de chaves, Abadi e Needham (1994, p. 16): um uso recente de uma chave não significa que ela não seja parte de um ataque de *replay*, ou seja, uma chave antiga. Assim deve-se periodicamente gerar novas chaves.

Princípio 10 – Usar codificação explícita, Abadi e Needham (1994, p. 18): caso seja utilizado algum procedimento de codificação para apresentar o conteúdo de uma mensagem, deve ser possível explicitar qual método de codificação foi utilizado.

Princípio 11 – Gerenciar a relação de confiança, Abadi e Needham (1994, p. 19): um projetista de protocolo deve ter claramente identificadas todas as relações de confiança entre os atores envolvidos na comunicação e o porquê de elas serem necessárias.

4.4. PROTOCOLOS PARA PAGAMENTO ELETRÔNICO COM CARTÕES INTELIGENTES

Conforme citado anteriormente, os pagamentos eletrônicos com cartões podem ser do tipo cartão de crédito, cartão de débito e cartão de dinheiro eletrônico. Uma boa introdução a protocolos para pagamento eletrônico pode ser vista em Law et al (1996, p. 8) onde estes autores descrevem uma sequência de passos simplificada para pagamentos com cartão de dinheiro eletrônico, na modalidade *online* e *offline* e com a introdução / remoção de anonimato.

Como se sabe, existem algumas características da segurança financeira que impõem restrições a pagamentos na forma eletrônica, por exemplo, a não rastreabilidade de um

pagamento poderá permitir a lavagem de dinheiro, que é um crime na legislação brasileira. Um pagamento *online*, apesar de não simular todas as características possíveis de um dinheiro em papel, como transferência e anonimato, é um esquema muito utilizado atualmente por estar aderente ao modelo de negócios de instituições bancárias. De, Law et al (1996, p. 12), obtem-se o protocolo simplificado para pagamento *offline* descrito abaixo.

4.4.1 Pagamento *offline,* não rastreável e com adição/remoção de anonimato

Retirada:

. Alice cria uma moeda eletrônica incluindo informações de identificação [1];

. Alice oculta (*blinding*) a moeda com o auxílio de um fator aleatório [2];

. Alice envia um pedido de retirada para o Banco e envia juntamente a moeda oculta;

. Banco verifica se as informações de identificação estão presentes;

. Banco assina digitalmente a moeda oculta [3];

. Banco envia a moeda oculta assinada para Alice e debita a conta bancária de Alice;

. Alice remove o fator aleatório que ela introduziu para ocultação da moeda.

Pagamento:

. Alice envia uma moeda para Bob;

. Bob verifica se a assinatura digital da moda está válida;

. Bob envia um desafio aleatório para Alice;

. Alice envia uma resposta a Bob revelando uma parte (de duas existentes) das informações de identificação;

. Bob verifica a resposta;

. Bob dá a mercadoria que está sendo comprada à Alice.

Depósito:

. Bob envia a moeda, o desafio e a resposta para o Banco [4];

. Banco verifica se a assinatura digital da moeda está válida;

. Banco verifica se a moeda já não foi gasta anteriormente (*double spending*);

. Banco cria os registros para a moeda que está sendo gasta, o desafio e a resposta [5];

. Banco credita a conta bancária de Bob.

[1] a informação de identificação é formada de tal forma que existem duas partes. A informação de uma parte é insuficiente para se identificar Alice, somente quando se juntam

as duas partes é que se identifica Alice.

[2] a ocultação ocorre com o auxílio de uma quantidade aleatória (chamada, em inglês, de *blinding factor*).

[3] como a moeda está oculta, o Banco não sabe o que está assinando. Neste caso há 2 alternativas:

- ter-se mais de uma chave para cada valor informado no pedido de retirada, por exemplo, uma chave para assinar $10, outra para assinar $50 e assim por diante.

- as duas partes (Alice e Banco) incluem uma quantidade aleatória na moeda a ser assinada.

[4] observar que o Banco não consegue relacionar a moeda de Alice com a moeda que Bob enviou por conta do "*blinding factor*" que provê a não rastreabilidade.

[5] em caso de *double spending* serão recebidas duas respostas (duas partes das informações de identificação) que podem assim quebrar o anonimato.

Como pode ser visto acima, este exemplo de protocolo de pagamento permite a remoção do anonimato em caso de se gastar duas vezes a mesma moeda.

4.4.2 Pagamento *online* e não anônimo – *framework* iKP

Um dos primeiros trabalhos em protocolos de pagamentos eletrônicos utilizando uma infraestrutura de chaves públicas, ou seja, baseado em uma conexão *online* e não anônima, foi produzido no IBM *Research Labs* pela equipe de Bellare et al (1995) conforme pode ser visto na família de protocolos iKP (1KP, 2KP e 3KP). O *framework* iKP foi prototipado pela IBM na forma de um produto chamado ZiP (Zurich iKP *Prototype*) conforme descrito em Bellare et al (2000, p. 621). O iKP assume a premissa de que um ou mais atores possuem um par de chaves idealmente gerado e armazenado em um dispositivo inviolável, como um cartão inteligente, e uma chaves públicas depositada em um PKD de uma terceira parte confiável.

Três diferentes tipos de ameaças são consideradas defensáveis no iKP: ataque passivo (*eavesdropper*), ataque ativo e ataque a partir de um dos atores (*insiders*).

O Gráfico 4.8 ilustra os atores envolvidos no iKP, Banco do Comprador, Banco do

Vendedor, Comprador e Vendedor, evidentemente que as características de segurança do Banco do Comprador podem ser análogas às do Banco do Vendedor, por isto a linha pontilhada no Gráfico. As instituições bancárias podem também serem representadas um *Gateway* de Pagamentos que transaciona com várias instituições. No iKP o i é o indicador da quantidade de atores que possuem certificação digital, assim convenciona-se que o 1KP é o caso em que apenas o *Gateway* possui certificado, o 2KP é caso em que o *Gateway* e o Vendedor possuem certificados e o 3KP é o caso onde todos os participantes possuem certificado digital.

O *framework* iKP possui sete passos, conforme ilustrado no Gráfico 4.9, com mensagens envolvendo o Comprador, o Vendedor e o *Gateway*: *initialization, invoice, payment, cancelation, auth-request, auth-response* e *confirmation*.

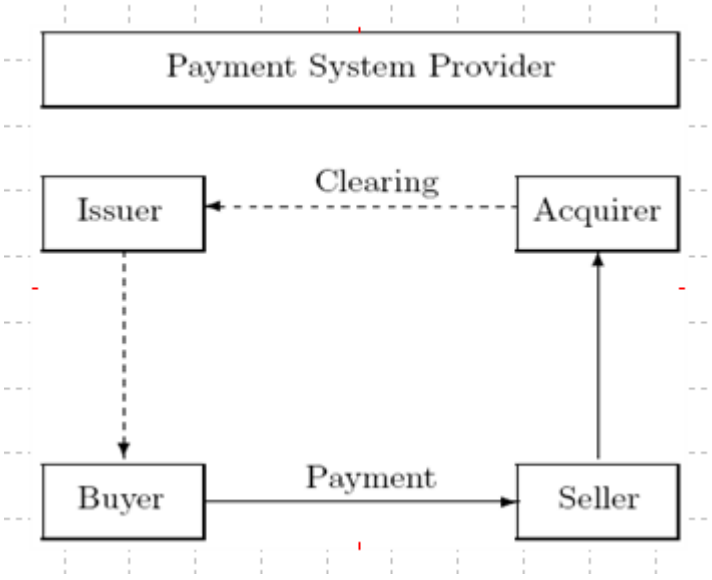

Gráfico 4-8 – Atores de um pagamento eletrônico no iKP

Fonte: Bellare et al (2000, p. 3)

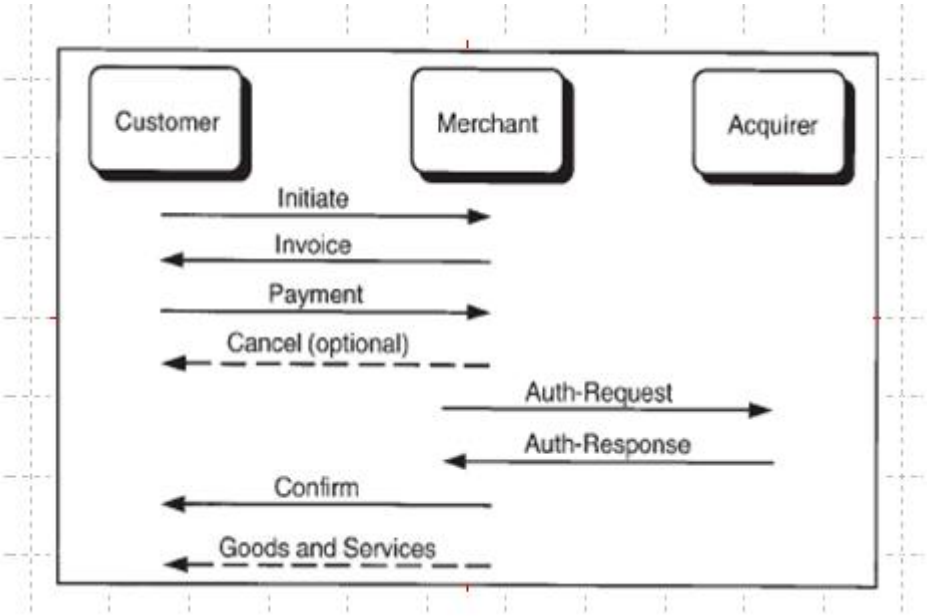

Gráfico 4-9 – Fluxo de mensagens no iKP

Fonte: Sherif (2000, p. 191)

Requisitos de segurança para o *Gateway* de pagamentos

A1. Prova de autorização de transação dada pelo Comprador.

A2. Prova de autorização de transação dada pelo Vendedor.

Requisitos de segurança para o Vendedor

M1. Prova de autorização de transação dada pelo *Gateway*.

M2. Prova de autorização de transação dada pelo Comprador.

Requisitos de segurança para o Comprador

C1. Impossibilidade de realizar um pagamento não autorizado.

C2. Prova de autorização de transação dada pelo Gateway.

C3. Certificação e Autenticação do Vendedor.

C4. Recibo dado pelo Vendedor.

Requisitos de segurança opcionais

C5. Privacidade.

C6. Anonimato.

4.4.2.1 1KP

No protocolo 1KP a criptografia de chaves públicas é utilizada por todos os atores mas somente o *Gateway* possui um certificado, ou seja, o Vendedor e o Comprador possuem uma cópia da chaves públicas do *Gateway*. O Gráfico 4.10 representa o fluxo de mensagens no 1KP.

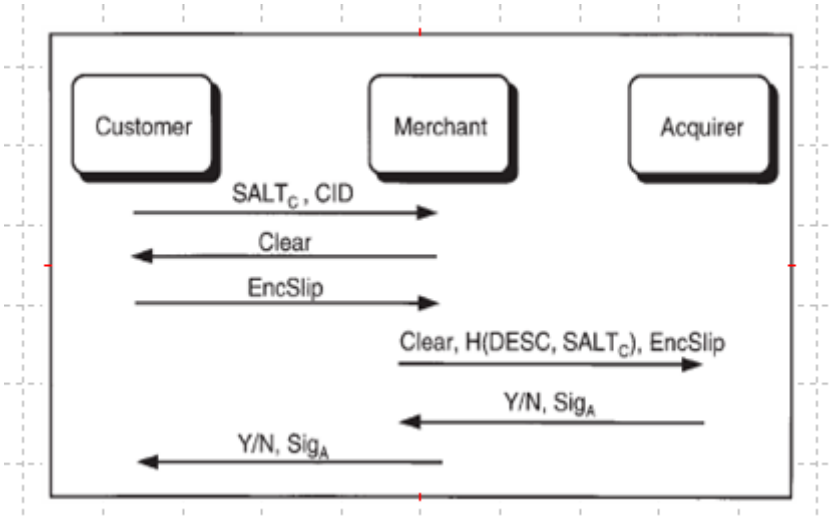

Gráfico 4-10 – Fluxo de mensagens no 1KP

Fonte: Sherif (2000, p. 193)

4.4.2.2 2KP

O protocolo 2KP corrige algumas fraquezas do protocolo 1KP e possui essencialmente o mesmo fluxo. Adicionalmente ao *Gateway*, no 2KP o Vendedor também possui um certificado digital conforme ilustrado no fluxo do Gráfico 4.11.

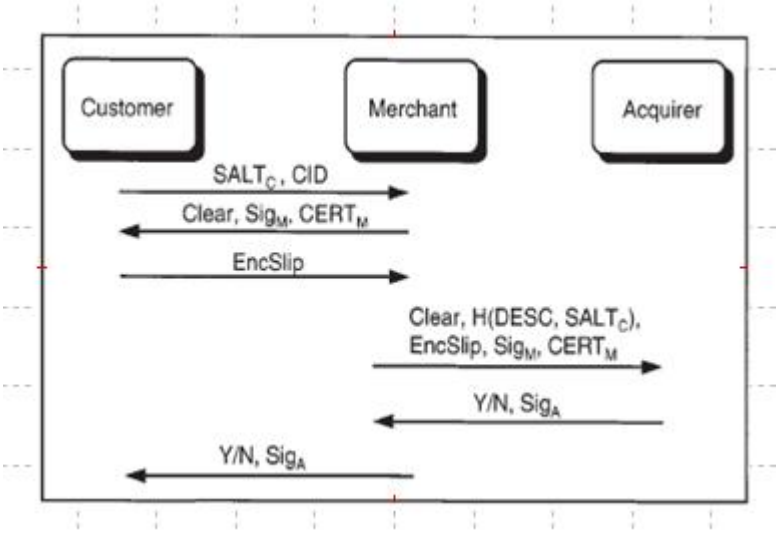

Gráfico 4-11 – Fluxo de mensagens no 2KP

Fonte: Sherif (2000, p. 196)

4.4.2.3 3KP

No protocolo 3KP, todos os três atores possuem certificados digitais. O Gráfico 4.12 representa o protocolo.

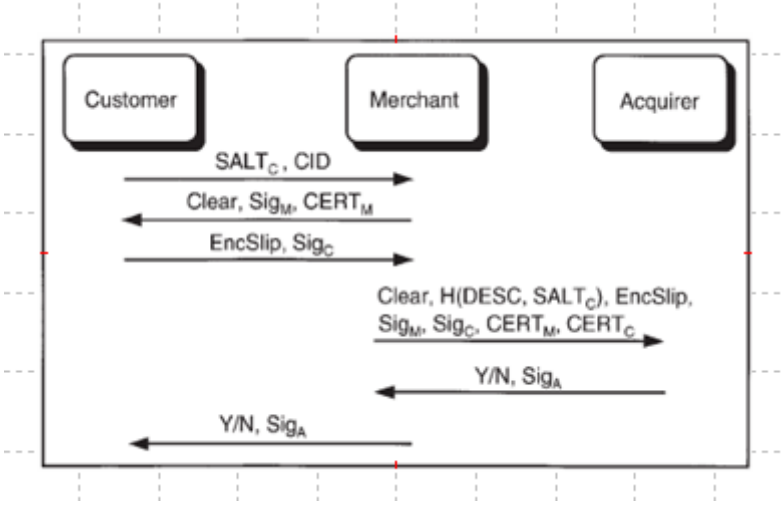

Gráfico 4-12 – Fluxo de mensagens no 3KP

Fonte: Sherif (2000, p. 196)

A Tabela 4.1 ilustra uma comparação entre os protocolos da família iKP onde se percebe que o 3KP é o protocolo que apresenta máxima aderência aos requisitos de segurança estipulados.

REQUIREMENTS/PROTOCOLS	1KP	2KP	3KP
Issuer/Acquirer			
A1. Proof of Transaction Authorization by Buyer	√	√	√√
A2. Proof of Transaction Authorization by Seller		√√	√√
Seller			
S1. Proof of Transaction Authorization by Acquirer	√√	√√	√√
S2. Proof of Transaction Authorization by Buyer			√√
Buyer			
B1. Unauthorized Payment is Impossible	√	√	√√
B2. Proof of Transaction Authorization by Acquirer	√√	√√	√√
B3. Certification and Authentication of Seller		√√	√√
B4. Receipt from Seller		√√	√√

Tabela 4-1 – Comparação dos protocolos iKP.

Fonte: Bellare et al (2000, p. 12)

O *framework* iKP foi proposto pensando-se em uma implantação incremental, ou seja, do 1KP para o 3KP. Com isto consegue-se que um maior grau de segurança seja possibilitado pelo fortalecimento da infraestrutura de acordo com a necessidade descrita nos casos de negócio. O *framework* iKP, devido a sua flexibilidade, pode ser estendido para suportar novos requisitos como processamento *batch*, micropagamentos, anonimato, entre outros.

O iKP segue princípios de segurança de protocolos robustos e foi projetado de forma a se evitar os ataques mais comuns de forma eficiente. A sua flexibilidade e robustez são excelentes pontos de partida para a construção de extensões do *framework* iKP que construam, por exemplo, *providers* de anonimato e privacidade que não existem na proposta original.

4.4.3 Pagamento *online* e não anônimo do tipo cartão de débito – cSET

Existe um protocolo baseado no protocolo *Secure Electronic Transaction* (SET), que está descrito em Sherif (2000, p. 285) com a denominação de cSET (*chip-secured electronic transaction*). Tanto o protocolo SET quanto o cSET baseiam-se no trabalho original do iKP e são moldados para a utilização de certificados digitais.

O cSET foi desenvolvido na França, em 1997, através de em uma parceria entre a Gie Cartes Bancaires (que representava cerca de 200 bancos franceses) e a Europay conforme citado em Sherif (2000, p.285). A segurança do cSET é baseada na verificação da senha do usuário (PIN) em conjunto com um cartão inteligente cujo certificado digital está armazenado na memória EEPROM.

O cSET utiliza tanto criptografia simétrica quanto assimétrica para prover a segurança necessária: o número do cartão é protegido com criptografia DES de 56 bits, a autenticação do Comprador, Vendedor, Gateway de pagamento, *Software* de pagamento e *Software* da leitora é protegida com RSA de 1024 bits. Além disso, o *hash* da assinatura digital, quando necessário, é feito com SHA.

4.4.3.1 Arquitetura

O Comprador necessita digitar um PIN em uma leitora de cartões segura para autorizar

um pagamento, sendo que esta senha substitui a assinatura à mão, amplamente utilizada em cartões não inteligentes. Popularmente, cartões inteligentes que exigem a digitação de um PIN são chamados “chip and PIN”. O *software* do cartão bem como o *software* no Gateway executam os procedimentos necessários para o pagamento.

A figura 4.13 ilustra a arquitetura geral do cSET. A leitora de cartões segura, aprovada pela administradora de cartões, deve ter um teclado para entrada do PIN, uma tela para orientar o usuário e um módulo criptográfico. A leitora deve ser inviolável, não deve armazenar nenhum tipo de informação secreta e o PIN não deve passar pelo computador, indo diretamente ao cartão. O Servidor de Registro do cSET tem a função de emitir certificados digitais para os participantes do protocolo e o Servidor de Distribuição de Software é um agente do Gateway que garante a instalação e distribuição de software de maneira segura.

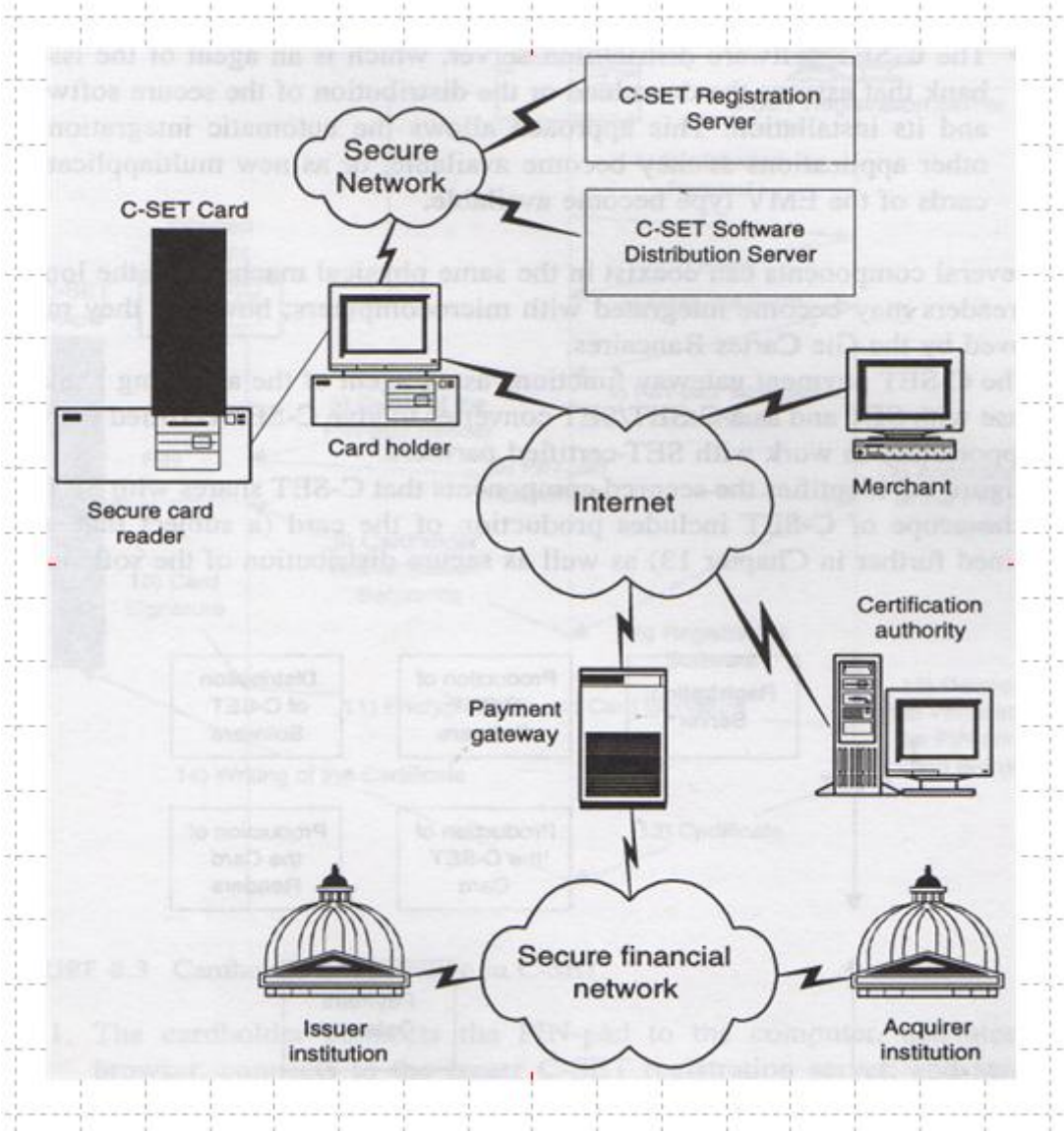

Gráfico 4-13 – Arquitetura geral do cSET

Fonte: Sherif (2000, p. 287)

O cSET é composto por três protocolos: o de Registro, que emite os certificados digitais para os Compradores, o de Distribuição do software de pagamento, o qual garante a distribuição segura do *software* utilizado no pagamento e o de Compra que é responsável pelas transações do cSET.

4.4.3.2 Protocolo de Registro

Em Sherif (2000, p. 289) há a descrição do protocolo, ou sequência de passos, que é realizada para o Registro do Dono do cartão e a conseqüente emissão do certificado

digital utilizado nas transações. O Gráfico 4.14 ilustra o protocolo.

1. O Dono do cartão conecta a leitora ao computador e comunica-se com o servidor de Registro cSET. Há um envio de uma requisição de registro.
2. O servidor de registro envia um formulário para o Dono do cartão.
3. O Dono do cartão preenche o formulário e o envia de volta ao servidor de registro.
4. O servidor de Registro verifica todas as informações necessárias, baixa o *software* de verificação da leitora para o computador do Dono do cartão e o ativa.
5. O *software* de verificação de leitoras inspeciona as portas do computador e identifica a marca e modelo das leitoras.
6. O *software* de verificação envia os identificadores da leitora para o servidor de Registro.
7. O servidor de Registro verifica se a leitora conectada ao computador do Dono do cartão está na sua lista de leitoras aprovadas. Então, ele baixa para o computador do Dono do cartão um *software* de registro correspondente ao tipo de leitora conectada e o ativa.
8. O *software* de registro pede ao usuário que insira seu cartão na leitora, e então envia para a leitora uma seqüência de instruções (seqüência de autenticação).
9. A leitora recebe a seqüência de autenticação e, depois de fazer as verificações necessárias, executa a seqüência de autenticação. Então, ela pede ao Dono do cartão que insira seu PIN. Se a verificação do PIN é feita corretamente, ela envia um comando ao cartão para que este assine digitalmente a resposta.
10. A leitora recebe a resposta do cartão, a qual inclui a assinatura digital do cartão.
11. A leitora criptografa a resposta do cartão com criptografia de chaves públicas e envia a mensagem criptografada para o *software* de registro no computador. Este, por sua vez, repassa a mensagem ao servidor de Registro.

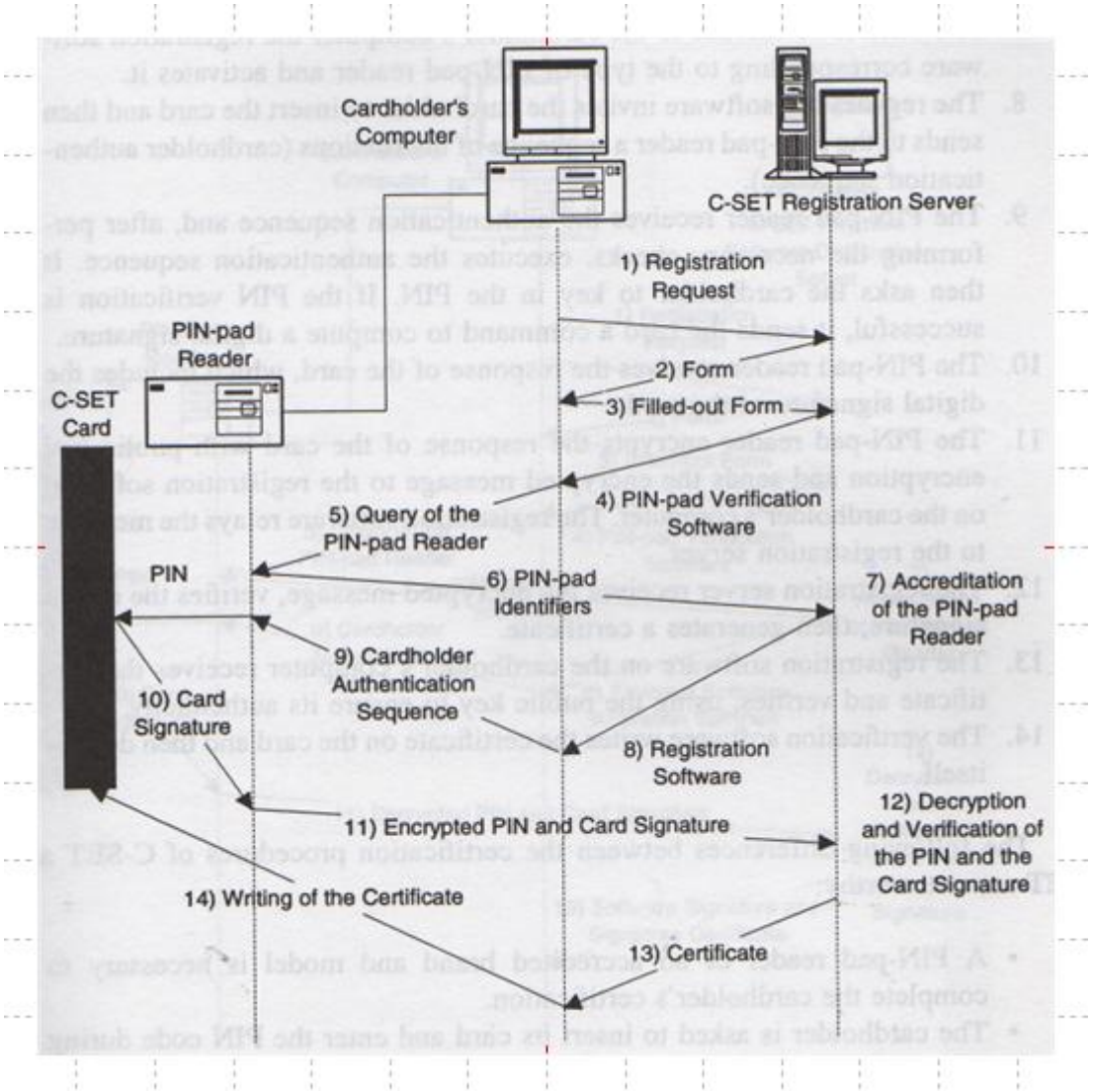

Gráfico 4-14 – Procedimento de Registro no cSET

Fonte: Sherif (2000, p. 289)

12. O servidor de Registro recebe a mensagem criptografada, verifica a assinatura do cartão e então gera um certificado.
13. O *software* de Registro no computador do usuário recebe o certificado e o verifica, utilizando a chaves públicas, para certificar-se de sua origem.
14. O *software* de verificação escreve o certificado no cartão e então se apaga do computador local.

5. ADAPTAÇÃO DO PROTOCOLO DE REGISTRO DO CSET

Diferentemente do que ocorre no c-SET, o novo protocolo proposto utiliza leitoras de cartão sem teclado numérico, sem módulo criptográfico embarcado e sem display local. Não há qualquer esquema de aprovação de leitoras ou homologação prévia das mesmas.

O novo protocolo assume que a leitora de cartões pode ser maliciosa e a rede pode ser não confiável, por isto, todas as informações que possam ser úteis a um adversário são assinadas e criptografadas.

5.2. DESCRIÇÃO DO PROTOCOLO DE REGISTRO

O Gráfico 5.1 ilustra o protocolo que é descrito abaixo.

1. O usuário do cartão conecta uma leitora compatível com o padrão PC/SC ao computador, entra em seu navegador web, conecta com o servidor de registro e envia uma requisição de registro.
2. O servidor de registro envia um formulário para o usuário do cartão.
3. O usuário do cartão preenche o formulário com seus dados e o envia de volta ao servidor de registro. Estas comunicações podem ser protegidas através de SSL/TLS, ou seja, por criptografia.
4. O servidor de registro envia para o computador do usuário um *software* de registro que utiliza as bibliotecas do padrão PC/SC para comunicação com o cartão. Este *software* de registro é devidamente assinado pelo servidor.
5. O *software* de registro pede o PIN que o usuário deseja que o cartão possua (Novo PIN), e o PIN Padrão que é fornecido juntamente com o cartão. A *interface* gráfica de entrada dos valores dos PIN´s é um teclado virtual, que torna a posição de cada número na tela aleatória, de forma a prevenir a captura do PIN através do *buffer* do teclado. Esta solução de teclado virtual tem que levar em conta a possível existência de *trojans* que capturam informação mesmo com o teclado virtual.
6. O Novo PIN é enviado para o servidor de registro criptografado com a chaves públicas do servidor.
7. O servidor de registro decifra o Novo PIN, seleciona de sua base de dados o número do cartão do usuário e a data da expiração e faz um *hash* com os três

valores. Este *hash* é persistido na base de dados e é a informação de contexto que evita um ataque do tipo *replay*.

8. O servidor de registro cifra o *hash* e o envia para o computador do usuário.
9. O *software* de registro pede então, ao usuário, que insira seu cartão na leitora e envia para o cartão um comando para inicializar os pares de chaves RSA de assinatura e criptografia, o que é feito com a verificação do PIN padrão. Este procedimento é conhecido como INIT Fase 1, e não pode ser repetido.
10. O cartão checa o PIN padrão e, se ele bater com o pré-gravado em sua memória, gera os pares de chaves. Quando concluído, o cartão envia uma mensagem INIT Fase 1 OK para o computador.
11. Tão logo receba a mensagem INIT Fase 1 OK, o computador pede a chaves públicas de criptografia PKc ao cartão, que será utilizada para criptografar todas as comunicações entre o computador e o cartão.
12. O *software* de registro no computador concatena o Novo PIN do usuário com o *hash* recebido do servidor de registro, criptografa os dois com a chaves públicas do cartão (PKc) e envia o resultado para o cartão (INIT Fase 2).
13. O cartão recebe e decifra a mensagem, calcula o *hash* do Novo PIN recebido concatenado com o número de cartão e data de expiração (pré-gravados em sua memória pela administradora de cartões), compara este *hash* com o *hash* recebido do computador do usuário e altera o PIN padrão para o Novo PIN, se os *hashes* forem idênticos. Então, envia para o computador uma mensagem INIT Fase 2 OK. Além disso, o cartão impede que sejam feitos novos INIT Fase 2 e grava o *hash* em sua memória para comparações posteriores.
14. O *software* no computador verifica que foi completado o INIT Fase 2 e envia uma requisição para checagem de PIN ao cartão, com o Novo PIN criptografado com PKc. O cartão verifica o PIN e retorna uma mensagem PIN Check OK. Este procedimento se faz necessário pois toda operação subseqüente necessita que o PIN tenha sido verificado anteriormente.
15. O computador pede ao cartão sua chaves públicas de assinatura. O cartão a retorna para o computador se o PIN tiver sido verificado.
16. O computador então seleciona a chave pública de assinatura do cartão, encapsulando-a em uma Requisição de Assinatura de Certificado (CSR – *Certificate Signing Request*) juntamente com informações sobre o certificado a ser pedido à Autoridade Certificadora e com informações de identificação do dono do

certificado posteriormente gerado. Esta nova proposta de protocolo, assim como o cSET, utiliza um apelido para identificação do usuário, que é o *hash* calculado anteriormente pelo Servidor de Registro: *hash* (Nº cartão | Data Expiração | PIN) que é codificado em Base64 e gravado no campo CName da CSR.

17. O *software* de registro envia para o cartão uma Requisição de Assinatura de CSR, contendo a CSR criptografada com a chaves públicas de criptografia do cartão (PKc).
18. O cartão então decifra a CSR, verifica se ela está no padrão ASN.1, extrai dela o identificador CName, decodifica-o de Base64 para binário e o compara com o *hash* gravado na memória no passo 13, assinando a CSR somente se os valores forem idênticos. Caso afirmativo, envia a assinatura da CSR para o computador do usuário.
19. O computador recebe a assinatura da CSR, inclui a mesma na CSR gerada anteriormente e codifica-a em Base64.
20. O *software* de registro envia a CSR assinada para a Autoridade Certificadora.
21. A Autoridade Certificadora verifica a assinatura do cartão e a identificação do certificado com o *hash* armazenado no passo 7 em sua base de dados. Também verifica as informações enviadas anteriormente no formulário de registro. Se todas as verificações estiverem OK, o certificado é emitido.
22. O *software* de registro requisita à Autoridade Certificadora o Certificado Digital do usuário, e este é retornado ao computador.
23. O *software* no computador faz o *upload* do Certificado Digital (em binário e criptografado com PKc) para o cartão, que o aceita desde que o PIN tenha sido verificado. O cartão decifra os dados e confirma que os bytes foram recebidos.
24. O *software* de registro envia um comando de “Fim de Certificado” para o cartão.
25. O cartão verifica se o certificado está no padrão ASN.1, se o CName decodificado para binário bate com o *hash* gravado na memória e somente então marca os bytes recebidos como prontos. Além disso, o cartão grava em sua memória que um certificado foi gravado e não permite a operação novamente.
26. O cartão envia para o computador uma mensagem indicando que o certificado foi aceito.

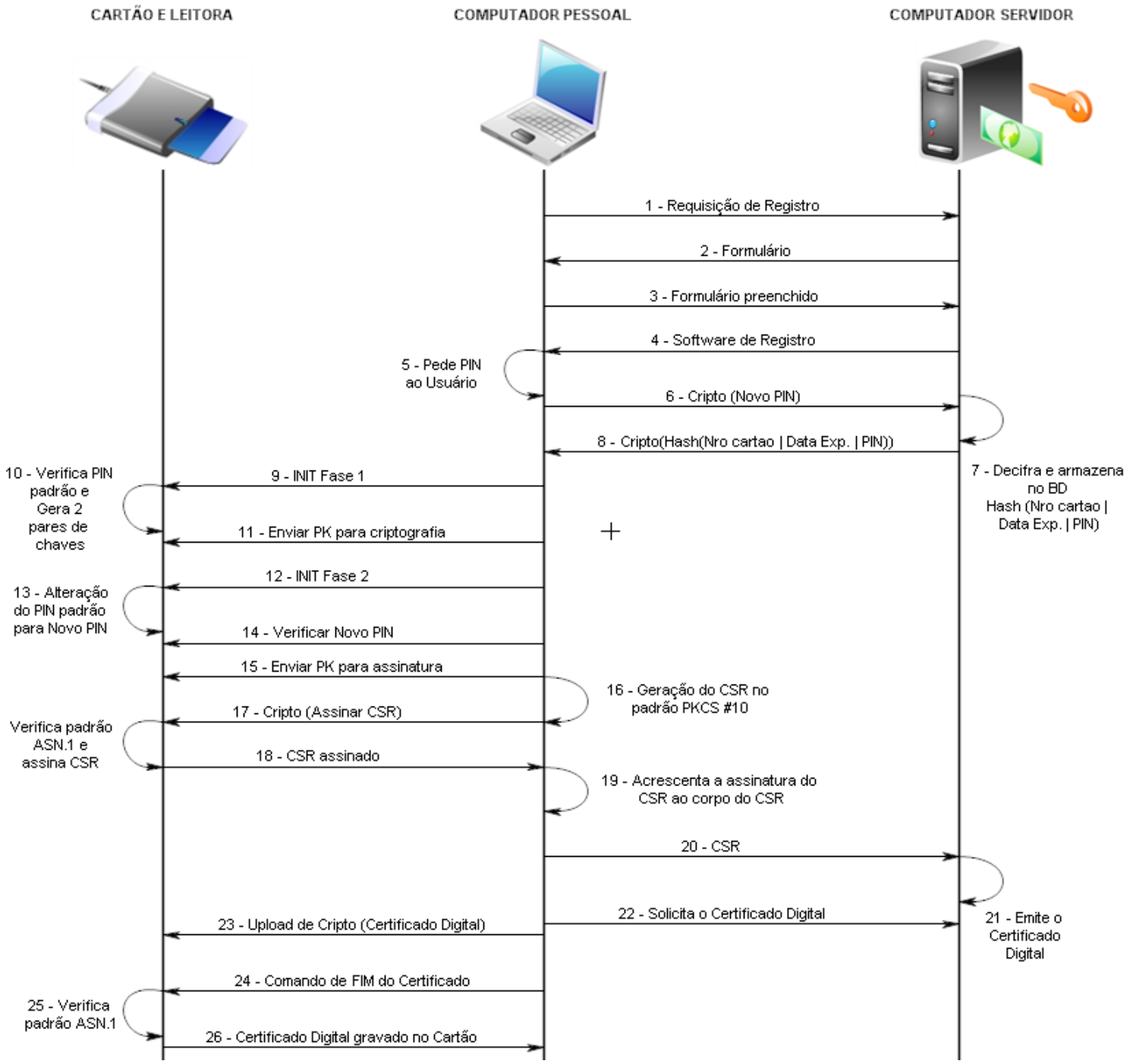

Gráfico 5-1 – Procedimento de Registro no protocolo proposto

5.3. ANÁLISE DE SEGURANÇA

A nova proposta de protocolo permite que, em um pagamento eletrônico, seja explorado o benefício de uma infraestrutura de chaves públicas e a inviolabilidade de um cartão inteligente. Abaixo são apresentados oito pontos importantes relativos a este protocolo no que tange à comunicação entre o cartão inteligente e o computador pessoal.

1. Geralmente os cartões inteligentes vêm de fábrica com um PIN de transporte. O Usuário do cartão deve, necessariamente, trocar este PIN padrão por um novo PIN que será o segredo individual do dono do cartão. Esta operação pode ser feita, inclusive, em terminais confiáveis de alguma empresa Administradora dos cartões ou em alguma *Automatic Teller Machine* (ATM) de Instituição Bancária.
2. O PIN nunca é enviado em claro, tanto na Internet quanto entre o computador do Usuário e o cartão.
3. Após dez tentativas de digitação errada do PIN, o cartão é bloqueado e um novo cartão tem que ser gerado pois não há uma função para reiniciar o contador de tentativas erradas após um limite de dez tentativas. Cada método do cartão executa, primeiramente, a verificação de que o PIN foi validado.
4. Somente o Usuário do cartão saberá seu PIN uma vez que ele não será gerado pela Administradora, mas escolhido pelo usuário e cifrado em todas as comunicações.
5. A geração de chaves de criptografia (passo 10) e assinatura é feita no próprio cartão sendo que não existe maneira de acessar as chaves privadas geradas no cartão. Estas chaves são geradas apenas uma vez, o que significa que se forem comprometidas, um novo cartão deverá ser emitido.
6. Para a troca do PIN (passo 13) o cartão exige que sejam passados o próprio PIN e o *hash* passado anteriormente pelo servidor de registro, *hash* que é feito a partir da concatenação do número do cartão, data de expiração e PIN. O cartão tem as informações de número expiração gravadas em sua memória, o que torna possível o recálculo do *hash* para comparar o PIN recebido com o PIN vindo do servidor de registro. Ou seja, não é possível para um *software* ou leitora maliciosa atacarem o cartão, pois um adversário não teria acesso aos dados do cartão nem ao *hash*, que sempre trafega de forma cifrada. Não é possível trocar o PIN após este passo. Se for comprometido, um novo cartão deverá ser emitido.
7. O cartão apenas assina requisições de certificado ASN.1 e que sejam para o apelido correto *hash*(Nº cartão | Data Expiração | PIN) codificado em Base 64, e apenas

com a verificação do PIN anteriormente (passo 18). Ou seja, o cartão não assinará qualquer informação. Além disso, este procedimento só poderá ser feito uma vez.

8. O cartão apenas grava em sua memória Certificados Digitais codificados em ASN.1 gerados para o apelido correto e após verificação do PIN, procedimento que também é permitido apenas uma vez (passo 25).

5.4. TECLADO VIRTUAL

O teclado virtual utilizado no protocolo de registro permite que uma leitora de cartões sem teclado seja utilizada neste novo protocolo de registro proposto. No entanto, somente o artifício do teclado virtual não evita todos os ataques possíveis como *trojans* que capturam informações do mesmo. Um ataque com *trojan* ao teclado virtual do Citibank, explorando vulnerabilidades do sistema operacional Microsoft Windows e do navegador Internet Explorer, é ilustrado em Zdnet (2008).

Evidentemente que na implementação completa do protocolo c-SET é necessário que se forneça a senha em momentos posteriores ao registro, por exemplo, no momento de confirmação de uma compra. Assim, no restante do protocolo c-SET, a utilização do teclado virtual é de vital importância e uma forma possível de defesa contra *trojans* é o uso do cartão inteligente operando conjuntamente com o teclado virtual.

6. IMPLEMENTAÇÃO

Para demonstrar o funcionamento do novo Protocolo de Registro, foi implementado um applet em Java Card para o cartão TOP IM GX4 e uma aplicação cliente em Java 6 que serve como *software* de Registro, localizado no computador do usuário. De acordo com o foco do estudo, a implementação foi direcionada para as operações do protocolo que envolvem a leitora de cartões.

De acordo com o manual do cartão TOP IM GX4, disponível em Gemalto (2008, p.1), este cartão suporta os padrões GlobalPlatform 2.1.1 e Java Card 2.2.1. O applet foi desenvolvido utilizando-se funções de segurança e criptografia contidas nos pacotes javacard.security.∗ e javacardx.crypto.∗, enquanto que o *software* cliente foi desenvolvido utilizando-se, para comunicação com o *hardware*, a bliblioteca javax.smartcardio.∗.

6.2. LABORATÓRIO – CONFIGURAÇÃO DE *HARDWARE* E *SOFTWARE*

Os experimentos desta dissertação foram realizados no laboratório do Grupo de Processamento Digitais de Sinais (GPDS) da Universidade de Brasília (UnB). Para implementação do protótipo utilizou-se a seguinte plataforma de *hardware* e *software* relativa aos cartões inteligentes.

- Leitora de cartões com contato. Fabricante Sagem Orga, modelo ECO 5000, ilustrada no Gráfico 6.1.
- Leitora de cartões dual (com contato e sem contato RFID). Fabricante Omnikey, modelo CardMan 5321, ilustrada no Gráfico 6.2.
- Cartão inteligente com suporte a Java Card, com contato. Fabricante Gemalto, modelo TOP IM GX4, ilustrado no Gráfico 6.3.
- *Software* de comunicação GlobalPlatform GPShell 1.4.2.
- Java Card *Development* Kit (JCDK) 2.2.2 para desenvolvimento do applet.
- Java Card *Development* Kit (JCDK) 2.1.1 para gravação no chip.
- Java *Development* Kit (JDK) 1.6.0 para desenvolvimento do cliente.
- Java *Development* Kit (JDK) 1.4.2 para gravação no *chip*.

Gráfico 6-1 – Leitora ECO 5000

Gráfico 6-2 – Leitora CardMan 5321

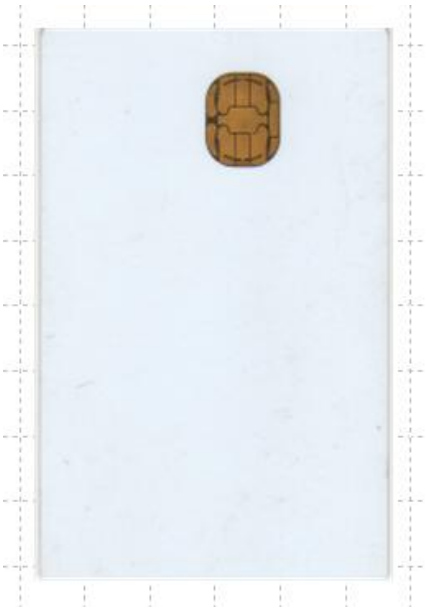

Gráfico 6-3 – Cartão inteligente TOP IM GX4

6.3. DESCRIÇÃO DA IMPLEMENTAÇÃO

O código fonte completo do applet e do cliente encontram-se nos Apêndices B e C, respectivamente. Este código é correspondente a uma facção do protocolo descrito anteriormente e que está ilustrado no Gráfico 6.4. O foco da implementação está nas interações que ocorrem entre o cliente Java e o applet Java Card.

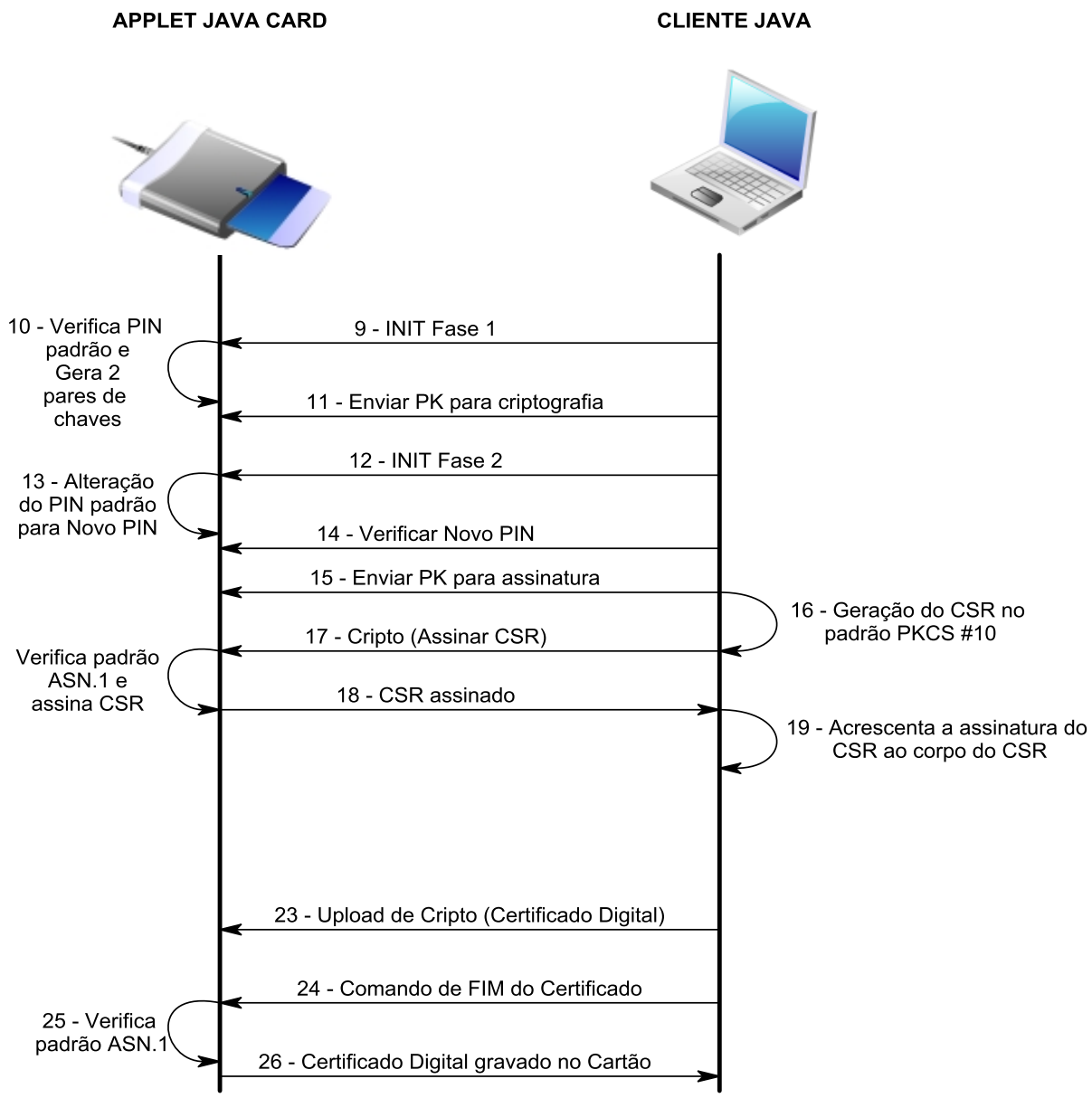

Gráfico 6-4 – Facção do protocolo proposto, usado na implementação

6.3.1 Applet Java Card

O gráfico 6.5 ilustra o fluxograma geral do applet que executa na JCRE do cartão. O applet é totalmente extensível para ser utilizado em outros protocolos pois é composto de funções reutilizáveis em nível de um cartão inteligente, como verificação de PIN, criptografia RSA, assinatura digital RSA, geração de par de chaves, recuperação de chaves públicas, *upload* de Certificado Digital, recuperação de Certificado Digital e bloqueio de cartão após tentativas erradas de digitação do PIN. Muitas destas tarefas são encontradas em módulos comerciais que acompanham cartões, fornecidos pelos

fabricantes, chamados *Cryptographic Service Providers* (CSP's). Assim, o applet também pode ser extensível para a construção de um CSP multiplataforma.

Um applet Java Card estende uma classe abstrata de nome "Applet" que é localizada no pacote javacard.framework. A classe Applet contém métodos necessários para desalocação, instalação, processamento de APDU´s, seleção e registro do applet, estes comandos são denominados *deselect*, *install*, *process*, *select* e *register*. O excerto de código abaixo, extraído do Apêndice B, exemplifica a instanciação da classe abstrata Applet e seus métodos estruturadores.

```
public class ucSETApplet extends Applet {
 (...)
      protected ucSETApplet(byte[] bArray, short bOffset, byte bLength)
          register();
      public static void install(byte[] bArray, short bOffset, byte bLength)
      public void process(APDU apdu)
      public boolean select()
 (...)
```

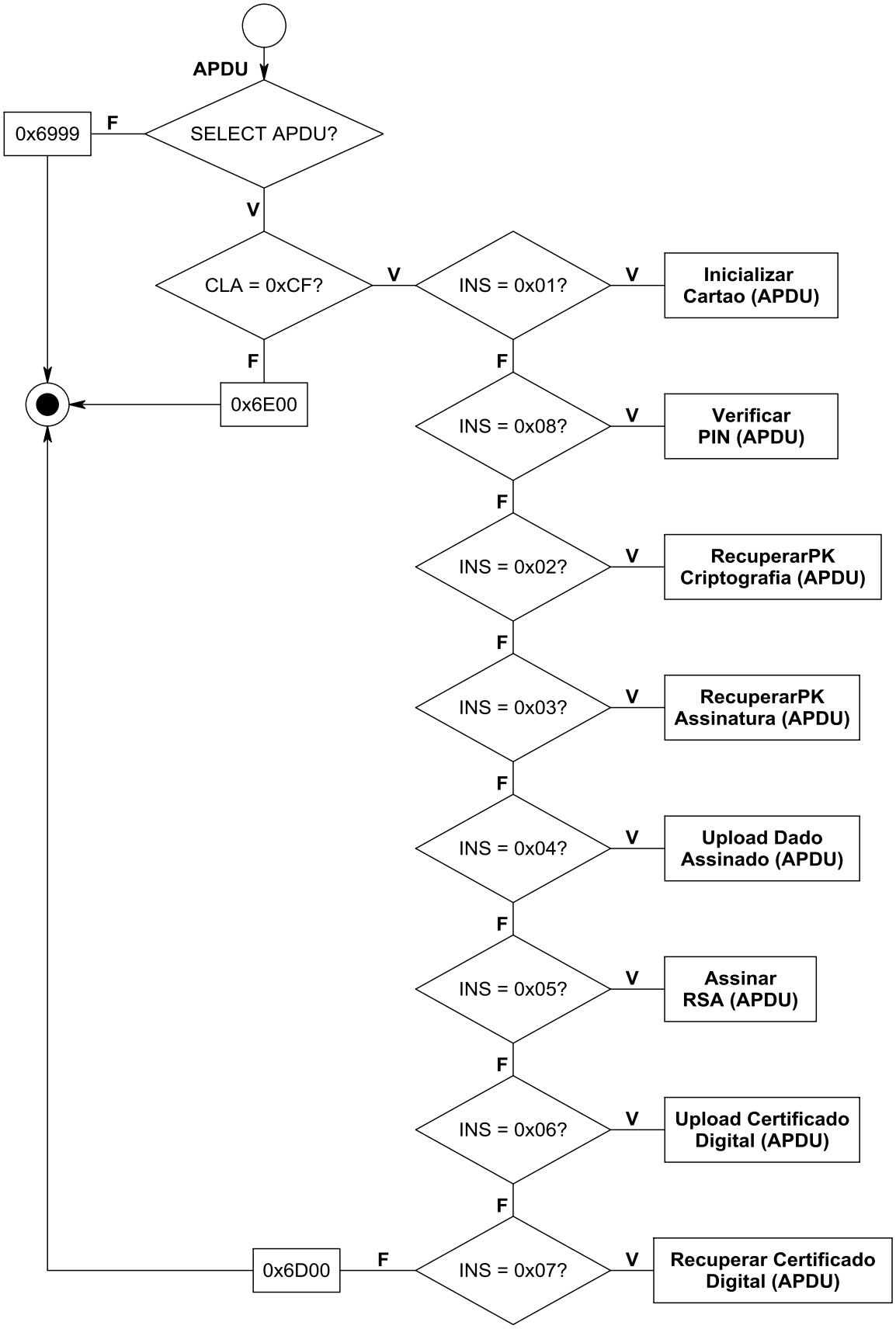

Gráfico 6-5 – Processamento do comando APDU pela JCRE de um applet

O Gráfico 6.6 ilustra o procedimento de inicialização do cartão o qual é implementado, em Java Card, na função InicializarCartao(APDU apdu) constante do Apêndice B. O applet gera o seu próprio par de chaves (pública e privada) internamente. Como as chaves são armazenadas em variáveis estáticas internas ao applet e o mesmo ocupa uma memória EEPROM (não volátil), as chaves não são perdidas mesmo quando se retira a alimentação elétrica do cartão. Particularmente, somente as chaves públicas devem ser expostas e qualquer método de exposição de chaves privadas pode ser considerado uma *backdoor* no projeto. Uma forma simples de se alterar a chave privada de um cartão é, portanto, executar-se novamente a função GerarParChaves( ) e sobrescrever as variáveis estáticas. Outra forma de se alterar a chave privada é apagar-se o applet e carregá-lo novamente com o mesmo AID ou carregar o applet com um novo AID e manter mais de um applet no cartão.

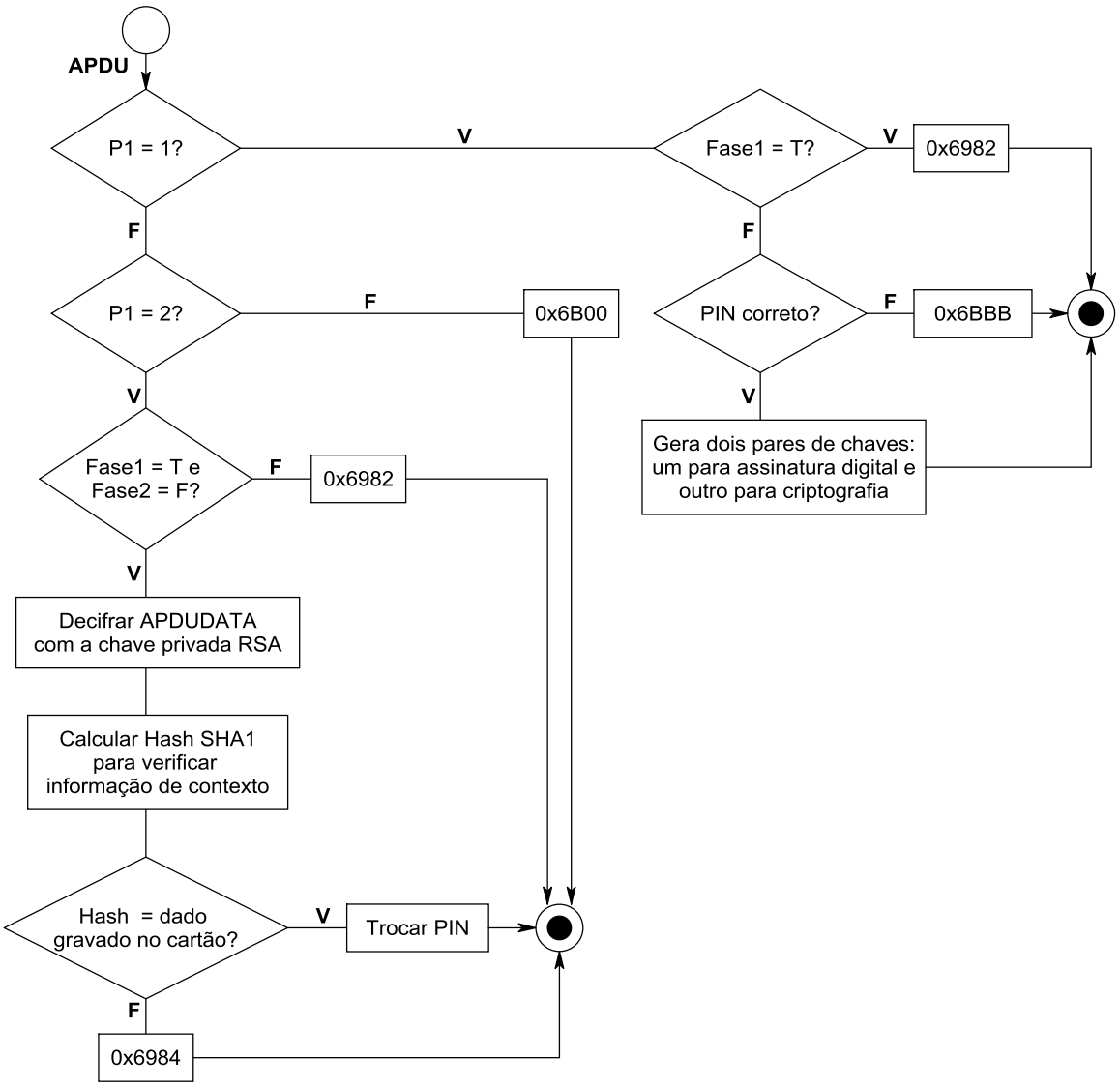

Gráfico 6-6 – Inicializar Cartão

O Gráfico 6.7 ilustra o procedimento de verificação do PIN o qual é implementado através da função VerificarPIN(APDU apdu) constante do Apêndice B.

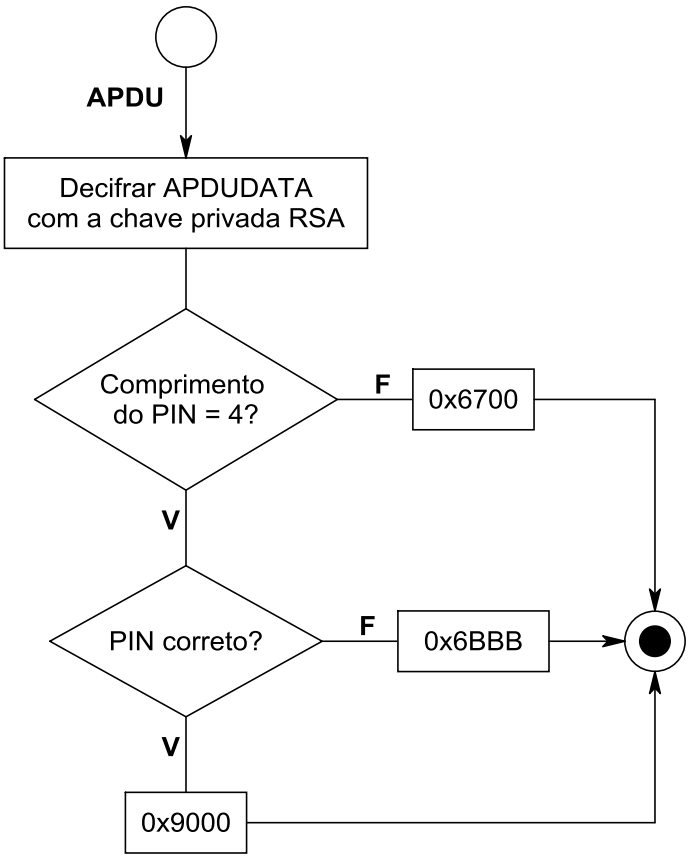

Gráfico 6-7 – Verificar PIN

Os Gráficos 6.8 e 6.9 ilustram os procedimentos para recuperação de chaves públicas para criptografar e assinar os quais são implementados, respectivamente, através das funções RecuperarPKCriptografia(APDU apdu) e RecuperarPKAssinatura(APDU apdu) constantes do Apêndice B. Como o criptosistema escolhido para uso no protocolo foi o RSA, então o método de exportação da chaves públicas no applet deve exportar o módulo e o expoente que constituem a chave pública.

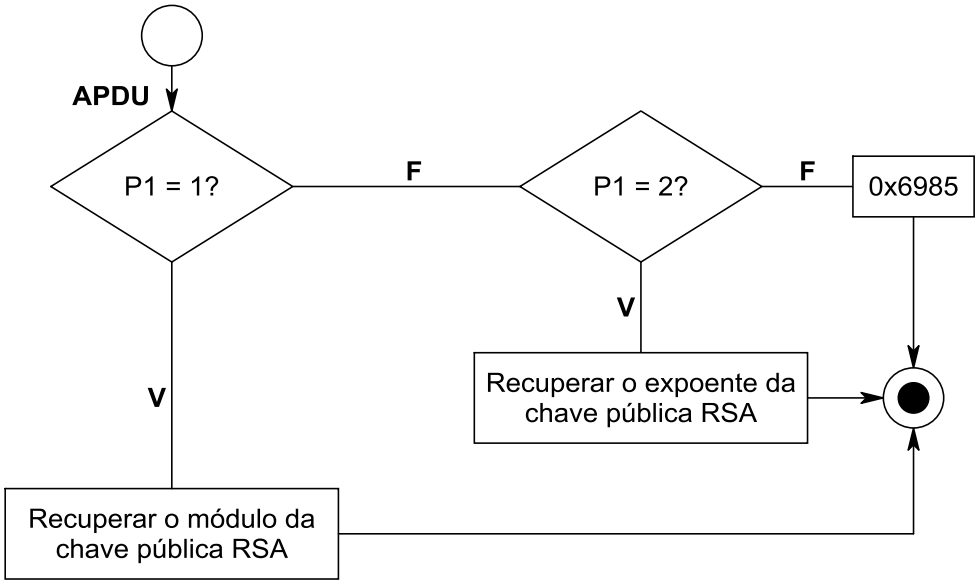

Gráfico 6-8 – Recuperar Chave Pública para Cifrar

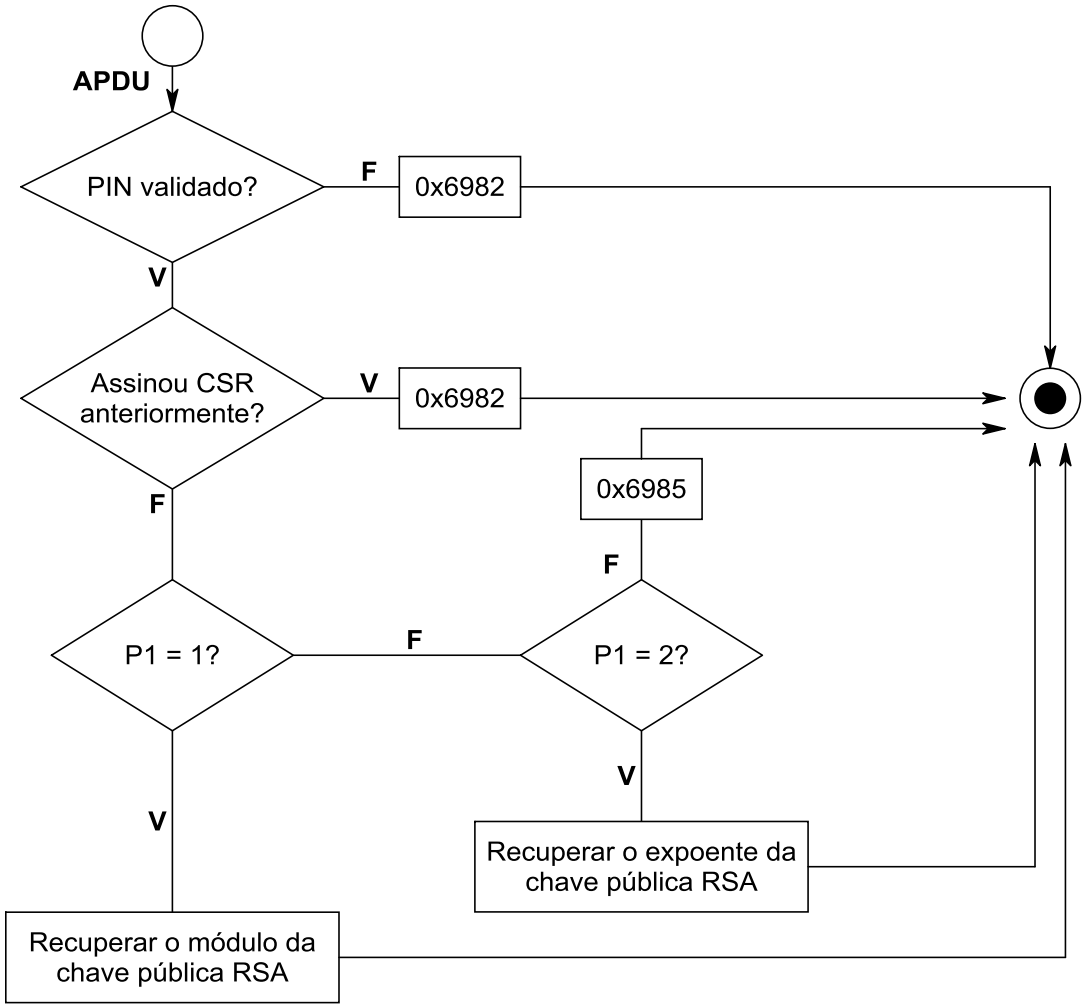

Gráfico 6-9 – Recuperar Chave Pública para Assinar

O Gráfico 6.10 ilustra o procedimento de *upload* do CSR assinado para o cartão o qual é implementado através da função UploadDadoAssinado(APDU apdu) constante do Apêndice B. Como este dado pode ser maior que 261 bytes e o cartão GX4 tem um *buffer* de APDU de 261 bytes, então, para se encaminhar mensagens maiores que 261 bytes é necessário dividi-la em blocos menores.

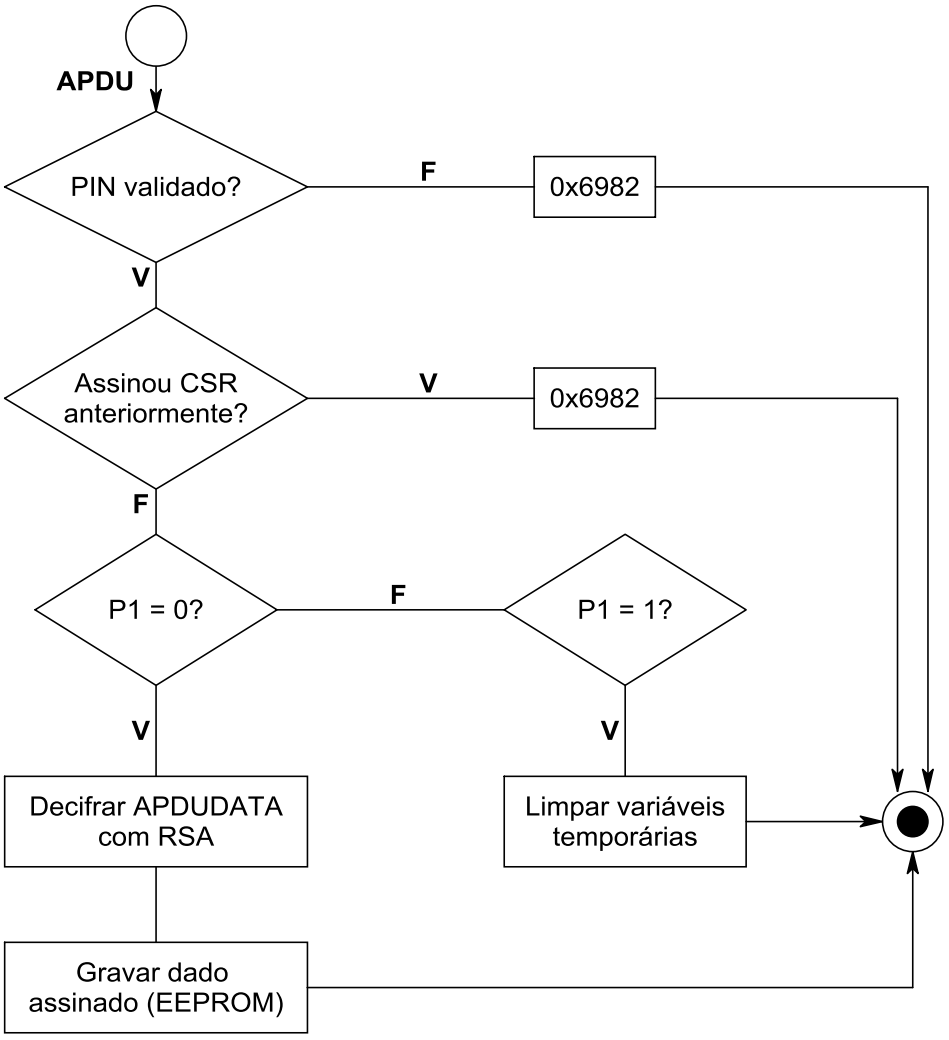

Gráfico 6-10 – *Upload* de CSR Assinado

O Gráfico 6.11 ilustra o procedimento de assinatura RSA o qual é implementado através da função AssinarRSA(APDU apdu) constante do Apêndice B. Para realizar a assinatura não é necessário, em nenhum momento, a exposição da chave privada do cartão.

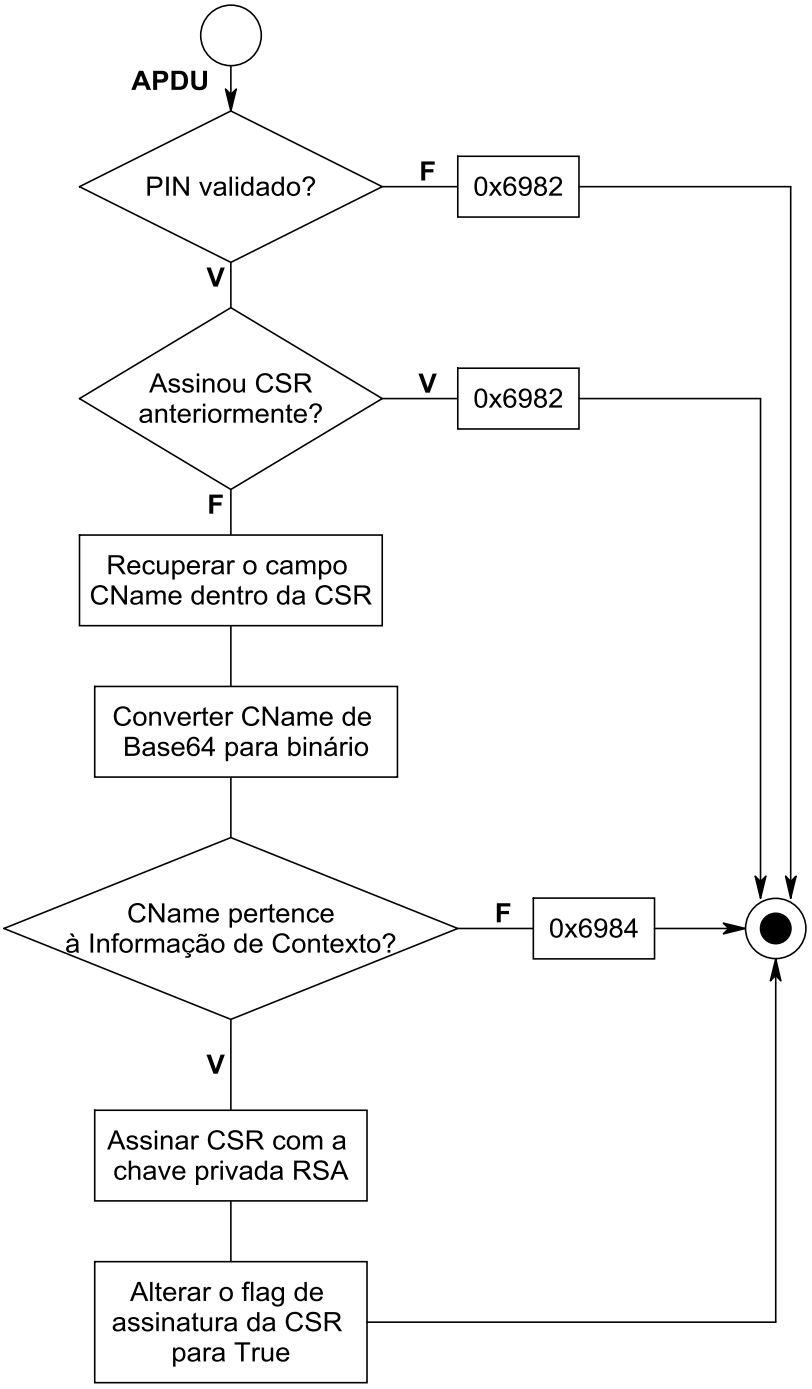

Gráfico 6-11 – Assinar RSA

O Gráfico 6.12 ilustra o procedimento de *upload* do Certificado Digital para o cartão o qual é implementado através da função UploadCertificadoDigital(APDU apdu) constante do Apêndice B. Como este dado pode ser maior que 261 bytes e o cartão GX4 tem um *buffer* de APDU de 261 bytes, então, para se encaminhar mensagens maiores que 261 bytes é necessário dividi-la em blocos menores.

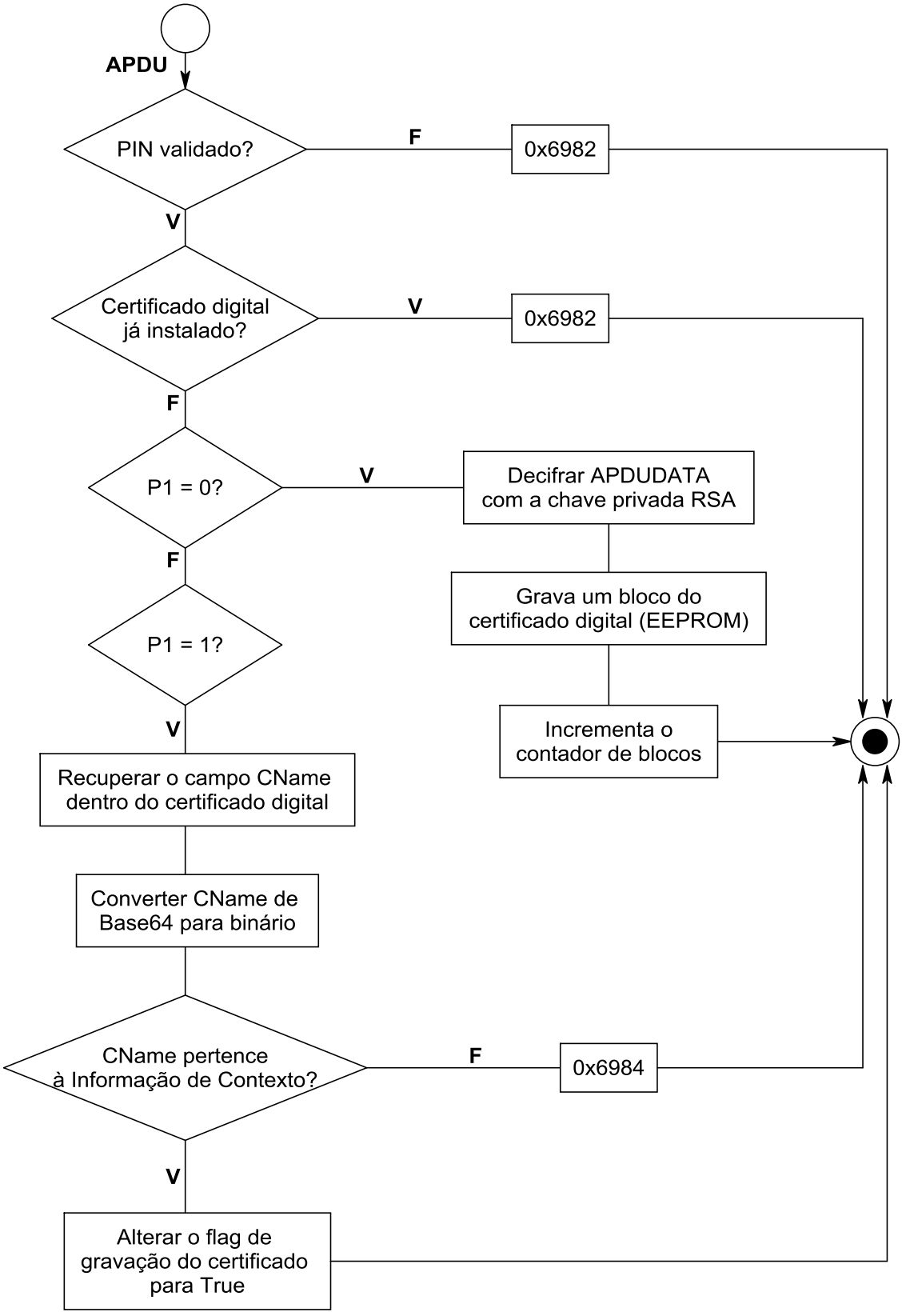

Gráfico 6-12 – *Upload* Certificado Digital

O Gráfico 6.13 ilustra o procedimento de leitura do Certificado Digital do cartão o qual é

implementado através da função RecuperarCertificadoDigital(APDU apdu) constante do Apêndice B. Como este dado pode ser maior que 261 bytes e o cartão GX4 tem um *buffer* de APDU de 261 bytes, então, para se receber mensagens maiores que 261 bytes é necessário dividi-la em blocos menores.

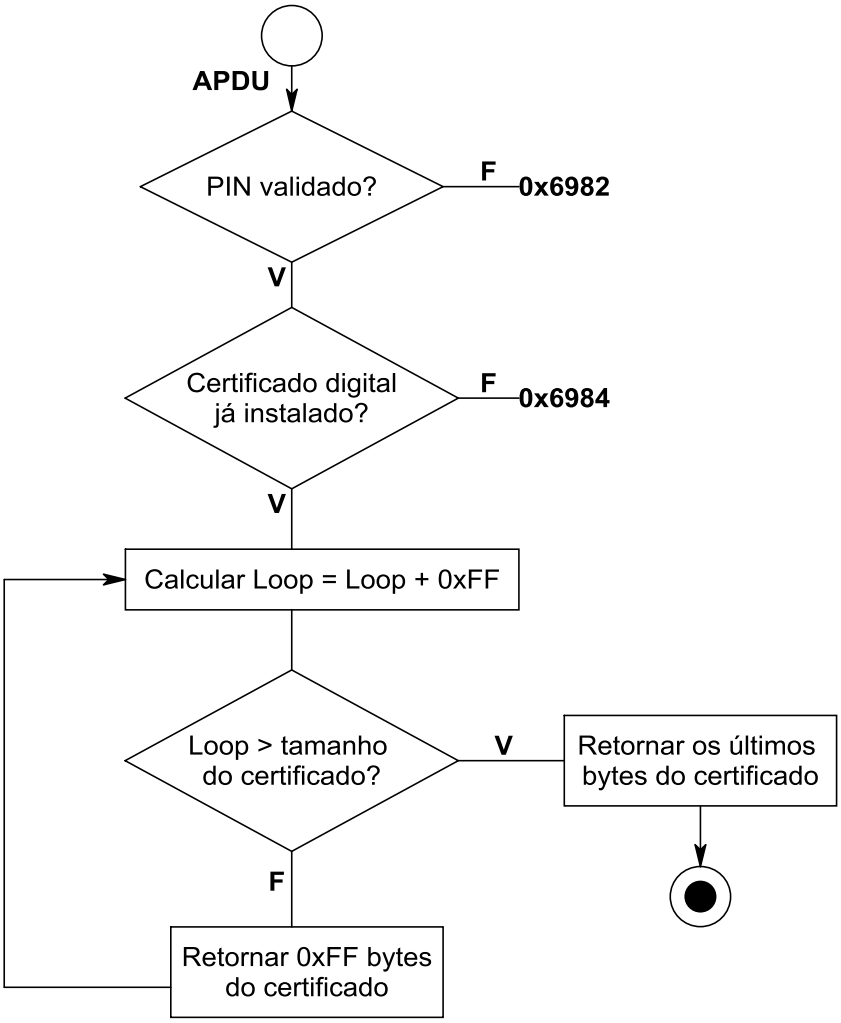

Gráfico 6-13 – Recuperar Certificado Digital

6.3.2 Cliente Java

O *software* cliente, escrito em Java, utiliza a biblioteca **javax.smartcardio** para enviar e receber APDU´s ao cartão. A seção do protocolo de registro que faz conversação com o cartão inteligente inicia-se no passo 9 do protocolo, conforme ilustrado no Gráfico 6.4, com as fases INIT Fase 1 e INIT Fase 2.

Todas as mensagens trocadas entre o applet e o cliente são criptografadas. Quando o cliente quer decifrar uma mensagem enviada pelo applet ele pode pedir a chave pública à Autoridade Certificadora ou, como implementado nesta dissertação, pedir a chave pública ao cartão. O código fonte completo do *software* cliente Java encontra-se no Apêndice C.

O Gráfico 6.14 ilustra o procedimento de inicialização do cartão o qual é implementado através da função InicializarCartao( ) constante do Apêndice C.

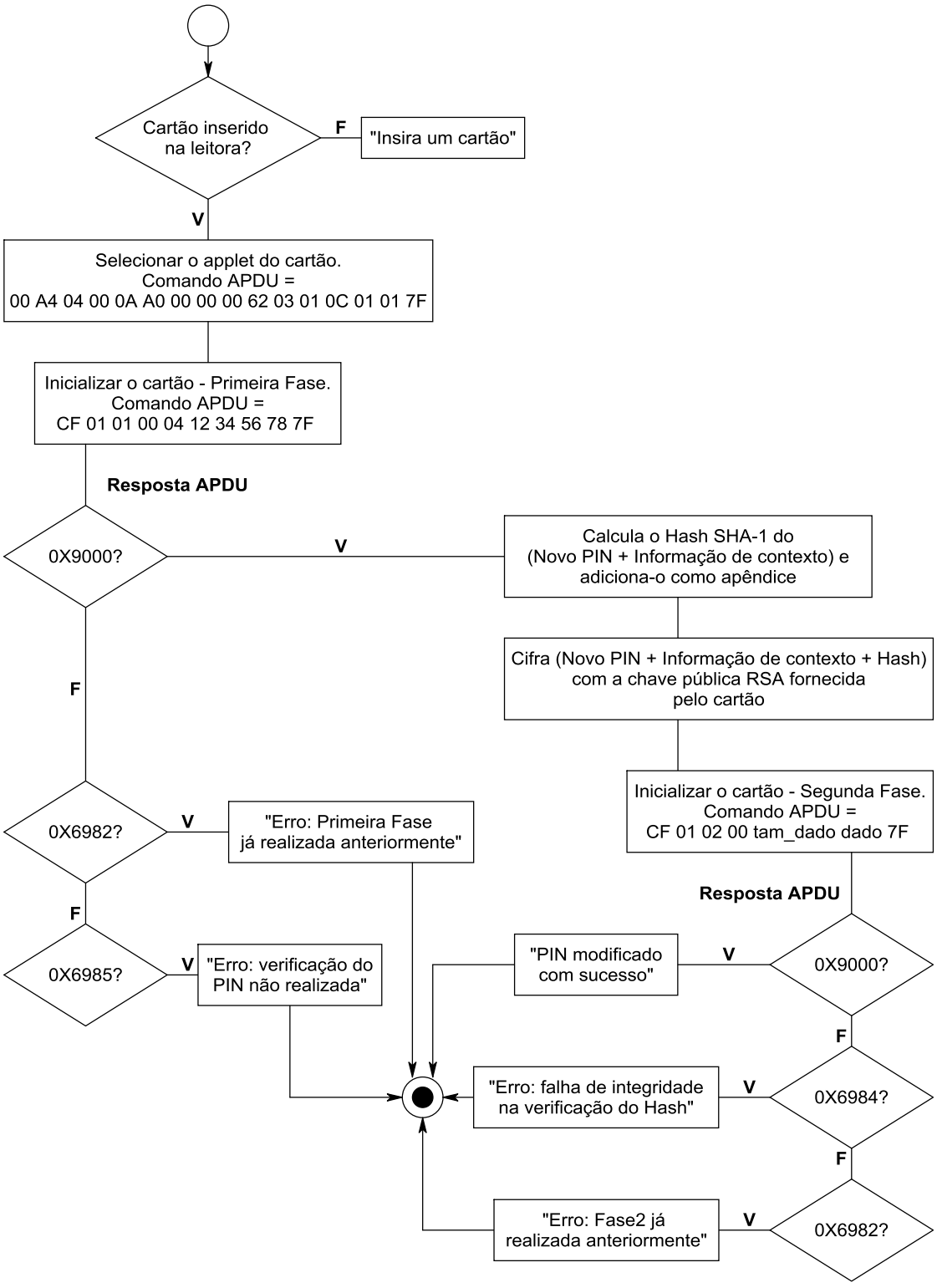

Gráfico 6-14 – Inicialização do cartão inteligente

O Gráfico 6.15 ilustra o procedimento de verificação do PIN o qual é implementado através da função VerificarPin(byte[ ] pin) constante do Apêndice C.

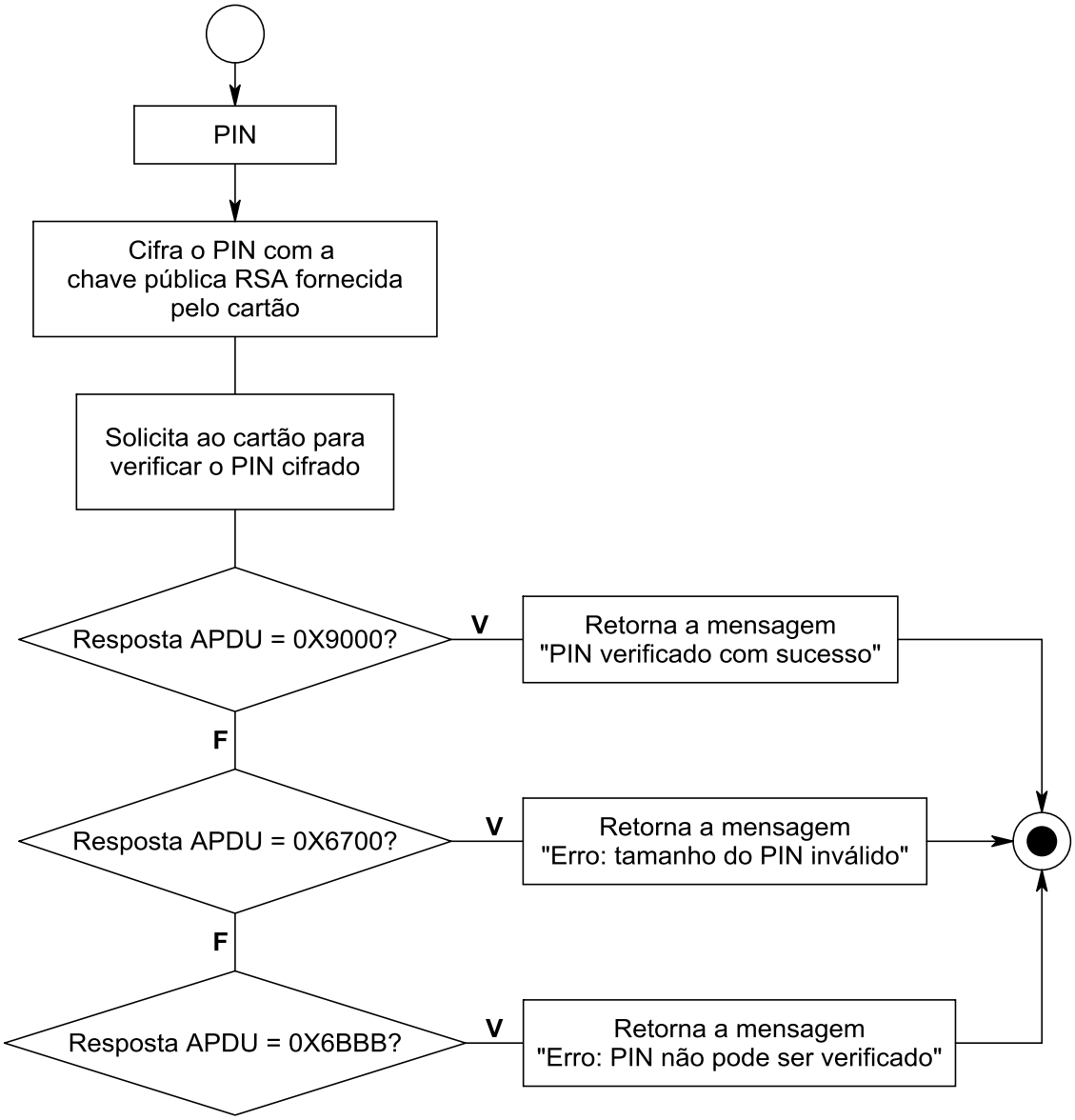

Gráfico 6-15 – Verificação do PIN

O Gráfico 6.16 ilustra o procedimento de gravação do Certificado Digital no cartão o qual é implementado através da função GravarCertificadoCartao( ) constante do Apêndice C. Como o cartão GX4 só aceita mensagens até 261 bytes, a função no cliente deve dividir as mensagens em blocos menores.

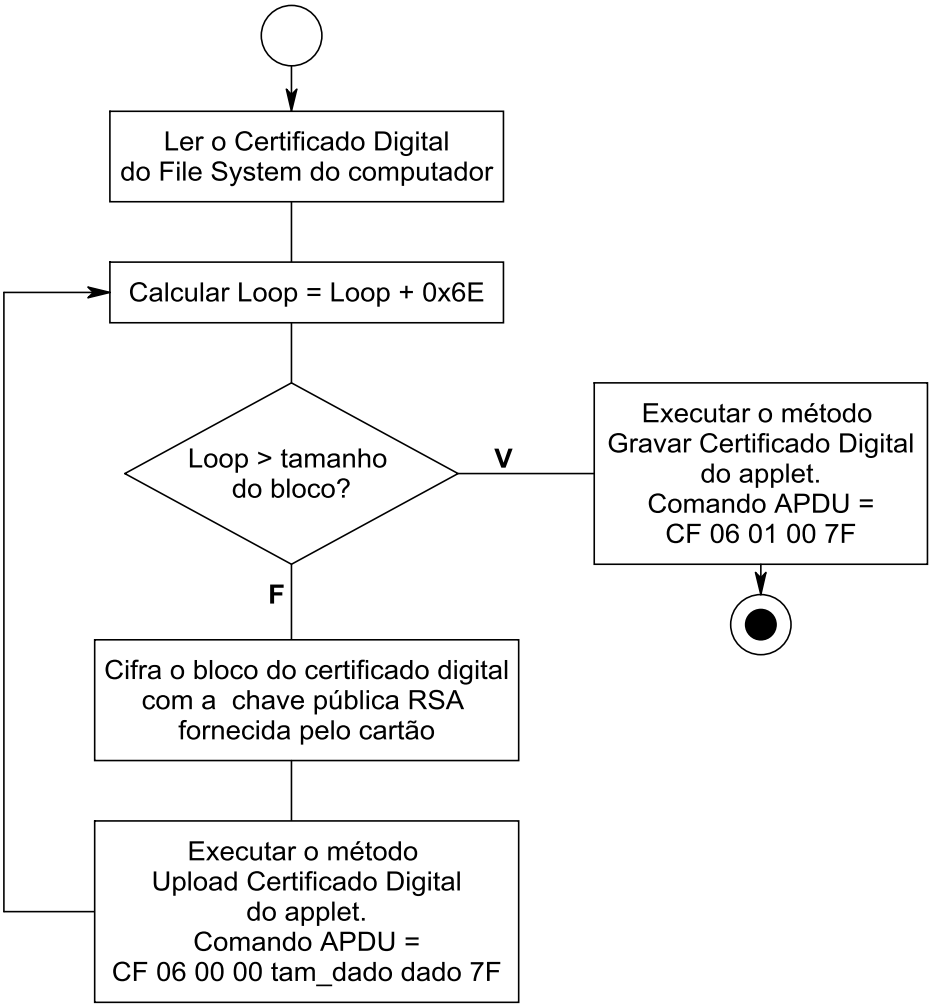

Gráfico 6-16 – Gravação do Certificado Digital no Cartão

O Gráfico 6.17 ilustra o procedimento de leitura do Certificado Digital no cartão o qual é implementado através da função RecuperarCertificado( ) constante do Apêndice C. Como o cartão GX4 só aceita mensagens até 261 bytes, a função no cliente deve receber as mensagens em blocos menores.

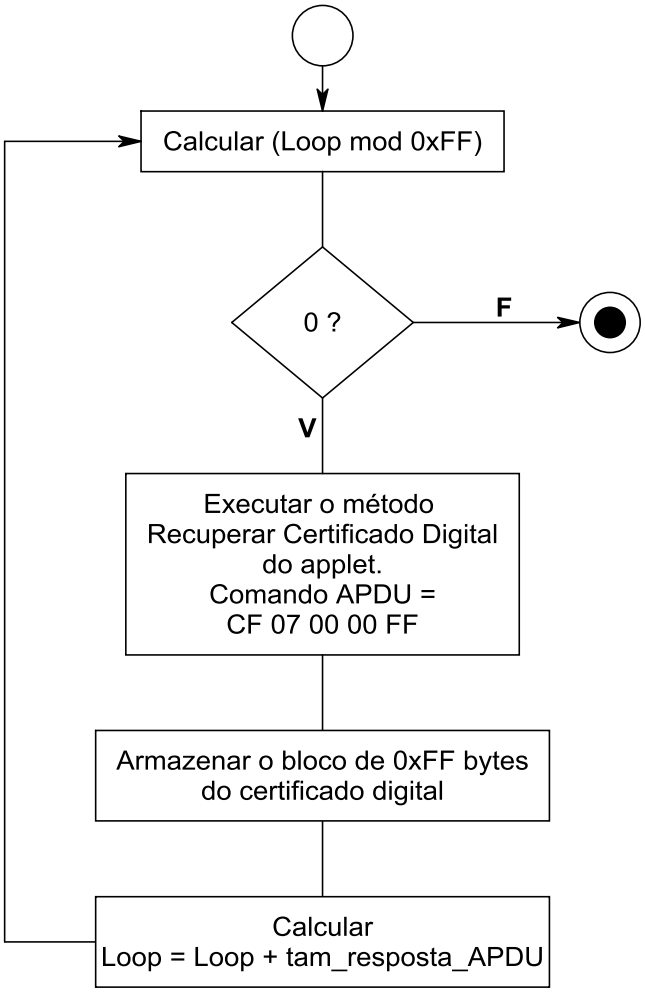

Gráfico 6-17 – Recuperação de Certificado Digital

O Gráfico 6.18 ilustra o procedimento de leitura das chaves públicas para criptografia e assinatura o qual é implementado através da função RecuperarPK(byte PKTipo) constante do Apêndice C.

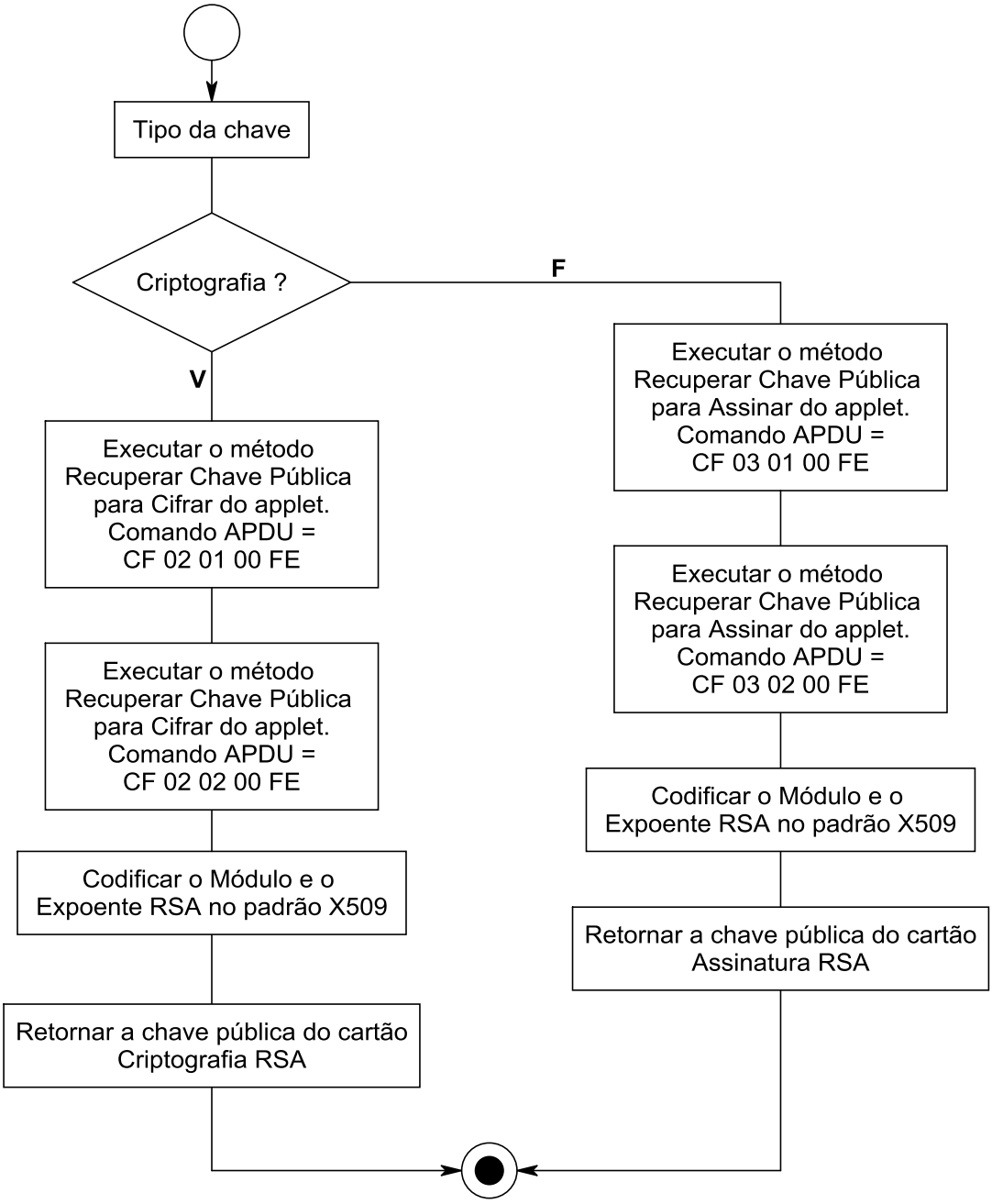

Gráfico 6-18 – Recuperação de Chaves Públicas

O Gráfico 6.19 ilustra o procedimento de assinatura no cartão o qual é implementado através da função AssinarNoCartao(byte[] DadoAssinar) constante do Apêndice C.

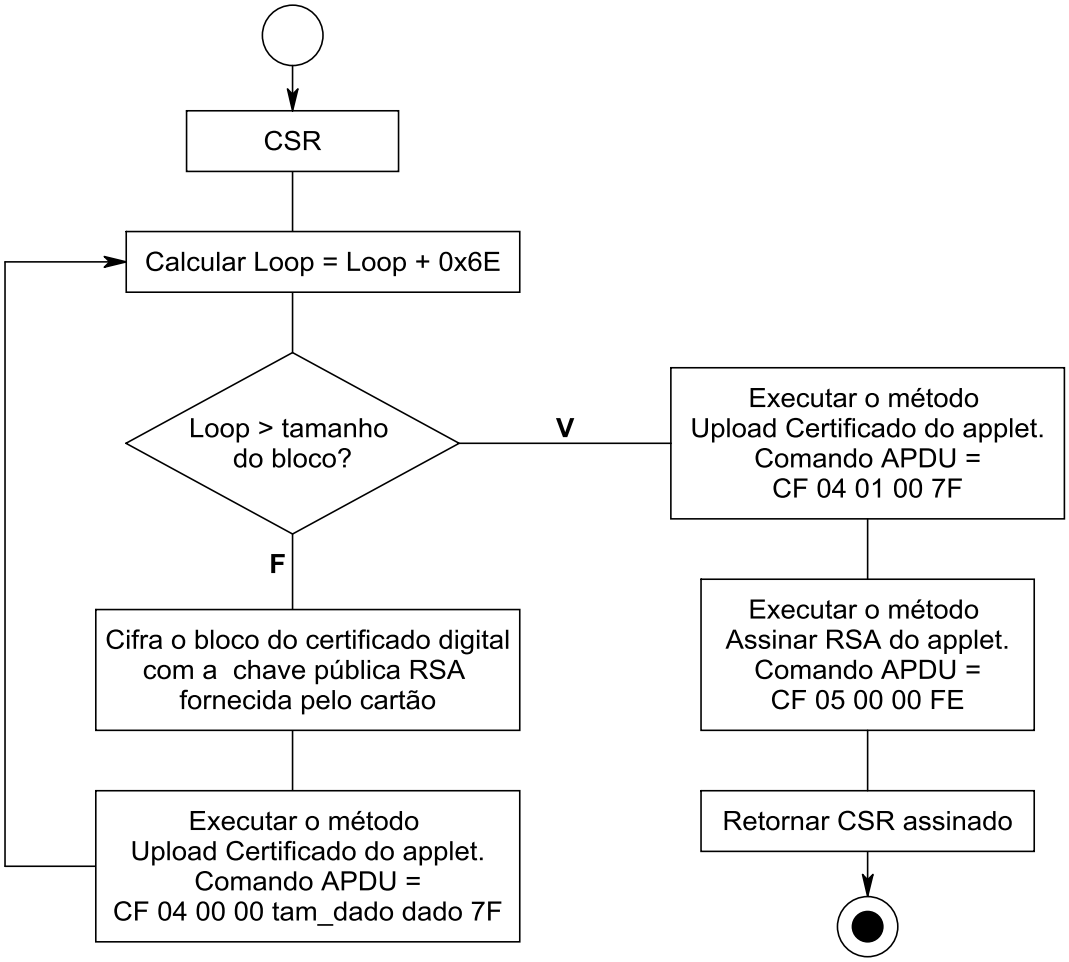

Gráfico 6-19 – Assinatura no Cartão

6.4. TESTES E SIMULAÇÃO

As ferramentas de testes e simulação do Java Card *Development Kit* 2.2.2 (JCWDE) não suportaram as funções criptográficas necessárias nesta implementação inviabilizando a possibilidade de simulação. Assim, os testes de implementação foram realizados a partir da instalação e remoção física do applet no cartão inteligente, fato este que torna os testes mais custosos por não se ter uma ferramenta automatizada.

6.5. EFICIÊNCIA

Como a alteração no protocolo de registro do c-SET exigiu a capacidade de criptografia e decriptografia no cartão, além da geração do par de chaves RSA, isto fez com que o mesmo fosse classificado como um cartão criptograficamente inteligente. Como estes cartões possuem unidades de processamento numéricas, também chamadas de

coprocessadores aritméticos, isto faz com que o valor monetário de um cartão para ser utilizado neste protocolo seja maior que de um cartão mais simples. Tipicamente, os chips mais simples existentes são etiquetas RFID que custam alguns centavos de Reais e os dispositivos mais complexos são chips para passaportes eletrônicos que custam algumas dezenas de Reais.

Portanto, esta adaptação do protocolo de registro do c-SET não deve ser utilizada em sistemas de pagamentos com cartões inteligentes mais simples que não sejam criptograficamente inteligentes. Este fato faz com que o custo financeiro dos cartões seja mais elevado.

7. CONCLUSÃO

Nesta dissertação foi proposta uma adaptação do protocolo de registro do c-SET, para pagamento eletrônico, que explora a potencialidade de cartões criptograficamente inteligentes e a possibilidade de uso de leitoras não confiáveis. Este protocolo proposto é inteiramente baseado em criptografia de chaves públicas, portanto, a segurança do protocolo é confiada à inviolabilidade de uma terceira parte chamada Autoridade Certificadora. Em muitos países, inclusive no Brasil, a entidade Autoridade Certificadora é regulamentada por lei trazendo legalidade ao protocolo proposto. O princípio da utilização de Certificação Digital em um esquema de pagamento eletrônico foi inicialmente publicado detalhadamente em Bellare et al (1995) no *framework* iKP e revisitado em Bellare et al (2000).

Relativamente à criptografia, nesta implementação procurou-se não utilizar esquemas criptográficos protegidos por patentes e que, além disto, fossem CCA seguros. A opção foi pela implementação, no applet Java Card, do criptosistema RSA OAEP. Esta implementação usou a criptografia e assinatura do próprio cartão inteligente, sem a necessidade de exportação de chaves para computadores pessoais ou servidores.

Cabe salientar ainda que o novo protocolo proposto é do tipo cartão de débito ou crédito na modalidade *online* e não-anônima. Estas características estão de acordo com o modelo de negócio das Instituições Bancárias, uma vez que são algumas das características necessárias para se inibir a lavagem de dinheiro, visto que sempre ocorrerá a identificação dos usuários.

Durante o desenvolvimento desta dissertação, notou-se que a plataforma Java Card é restrita quando comparada às potencialidades da linguagem Java devido principalmente ao *hardware* em questão, com pouco poder de processamento e memória, onde os applets Java Card são embarcados. A API Java Card é um subconjunto reduzido da linguagem Java. Os cartões inteligentes mais modernos, limitam a memória para aplicativos de usuários (EEPROM) a algo em torno de 72 kB e usualmente têm microcontroladores CISC ou RISC com freqüência interna de, no máximo, 30 MHz.

Um problema encontrado no decorrer do desenvolvimento do novo protocolo foi a

escassez de documentação a respeito do cartão TOP IM GX4, de forma a possibilitar o *download* do applet para a memória do cartão sem a utilização de ferramentas proprietárias do fabricante do cartão. Um dos objetivos da dissertação era utilizar o maior número possível de ferramentas de código aberto na implementação, o que foi conseguido com o programa *Global Plataform Shell* para *download* do applet e do Java *Card Development Kit* para desenvolvimento do applet.

Como trabalho futuro sugere-se a implementação completa, em Java Card, do protocolo c-SET fazendo-se as alterações necessárias para que o protocolo de pagamento eletrônico seja resistente a ataques iniciados com leitoras maliciosas. O caso de leitoras maliciosas é interessante pelo baixo custo financeiro do ataque, sendo relativamente fácil a troca de uma leitora confiável por uma não confiável. Desta forma, o tráfego do dado assinado digitalmente e posteriormente criptografado, em uma infraestrutura de chaves públicas, possibilita um pagamento mesmo em um ambiente com uma leitora não confiável.

Outra proposta de trabalho futuro pode ser a implementação de multi-applets em cartões inteligentes Java Card que compartilham o mesmo espaço de memória com a conseqüente análise de segurança dos mesmos. Por exemplo, quando se implementa um protocolo de pagamento eletrônico em um applet e um protocolo de identificação biométrica em outro applet, ambos em um mesmo cartão.

Como foi visto, a utilização do teclado virtual para entrada de dados é importantíssima no novo protocolo de registro proposto. Um outro trabalho futuro poderia ser a possibilidade de se utilizar o cartão inteligente para garantir a segurança do *software* do teclado virtual contra ataques do tipo *trojans*. Poder-se-ia explorar ainda a facilidade da assinatura digital do *software* de teclado virtual pelo cartão inteligente. Na situação em que o cartão estivesse conectado, seria possível a verificação da assinatura digital até mesmo em modo *offline*.

Finalmente, a implementação completa do protocolo poderá servir de base também para um protocolo de pagamento eletrônico a ser implantando, como um applet Java Card, em aparelhos celulares GSM ou 3G que tenham um SIM *card*. A idéia neste caso é explorar-se os recentes avanços criados nas áreas de *Near Field Communication* (NFC) e *Radio Frequency Identification* (RFID) em aparelhos celulares.

REFERÊNCIAS BIBLIOGRÁFICAS

ABADI, M; NEEDHAM, R. ***Prudent Engineering Practice for Cryptographic protocols***. IEEE Computer Society Symposium on Research Security and Privacy, 1994.

ABAD-PEIRO, J. L. et al. ***Designing a generic payment service***. *SEMPER Public Project Reports.* IBM Zurich Research Laboratory. 11/26/96.

ADAMS, C; LLOYD, S**.** ***Understanding PKI: Concepts, Standards, and Deployment Considerations.*** Second edition. Addison Wesley. 2002.

ANDERSON, R. ***Security Engineering. A Guide to Building Dependable Distributed Systems***. *Second Edition*. Wiley Publishing, Inc. 2008.

ANDERSON, R; NEEDHAM, R. ***Robustness principles for public key protocols***. 1995.

BELLARE, M; et al. ***iKP – A Family of Secure Electronic Payment Protocols***. *Proceedings of the First USENIX Workshop on Electronic Commerce*, USENIX, 1995.

BELLARE, M; et al. ***Design, implementation and deployment of the iKP Secure Electronic Payment System***. IEEE *Journal of Selected Areas in Communications*, Vol. 18, No. 4, *April* 2000.

CAMENISCH, J; et al. ***Compact E-Cash*****.** Cryptology ePrint Archive. Report 2005/060. *Received* 25 *Feb* 2005, *last revised* 27 *Mar* 2006.

CANARD, S; GOUGET, A. ***Divisible E-cash Systems can be Truly Anonymous***. EUROCRYPT 2007.

CHAN, A; et al. ***Easy come – easy go divisible cash***. EUROCRYPT 1998.

CHAUM, D; PEDERSEN, T. ***Transferred Cash Grows in Size***. EUROCRYPT 1992.

CHEN, Z. ***Java Card Technology for Cartões inteligentes***. *Architecture and Programmer's Guide*. Addison-Wesley. 2000.

COMPUTERWORLD1. **Mercado brasileiro de *cartões inteligentes* alcançará US$ 345 milhões em 2011**. Disponível em: http://computerworld.uol.com.br/seguranca/2006/03/31/idgnoticia.2006-03-31.1932573855/IDGNoticia_view. 31/03/2006. Acessado em 09/12/2008.

GEMALTO. ***Product information*: TOP IM GX4**. Disponível em http://www.gemalto.com/products/top_javacard/download/TOP_GX4_Nov08.pdf. Acessado em 12/12/2008.

GOLDREICH, O***. Foundations of Cryptography*** – *A Primer*. now Publishers Inc. 2005.

ICP1. **ICP-Brasil – Infraestrutura de chaves públicas brasileiras.** Disponível em https://www.icpbrasil.gov.br/apresentacao. Acessado em 10/12/2008.

ISO78164. ISO/IEC 7816 Part 4: ***Interindustry command for interchange***. 1998.

ITU. ***Security architecture for Open Systems Interconnection (OSI) for CCITT applications. Recommendation X.800***. Geneva. 1991.

JAIN, A; et al. ***Handbook of Biometrics***. Springer. 2008.

LAW, L; et al. ***How to Make a Mint: The Cryptography of Anonymous Electronic Cash***. *National Security Agency (NSA), Office of Information Security and Technology, Cryptology Division*, 18/06/1996.

MENEZES, A. et al. ***Handbook of Applied Cryptography***. CRC Press. 1996.

NAKANISHI, T; SUGIYAMA; Y. ***An Unlinkable Divisible Electronic Cash Using Secure Proxy Computation for DL One-way Function***. 2002.

NASCIMENTO, A. ***Overview of Public-Key Cryptography – Lecture 7***. *Department of Electrical Engineering*. Disponível em: http://www.ene.unb.br/~andclay/aulas/netsec/pkc.ppt. Acessado em 22/12/2008.

NSF. ***National Smartcard Framework***. Disponível em: http://www.finance.gov.au/e-government/security-and-authentication/smartcard-framework.html. Acessado em 10/12/2008.

OKAMOTO, T.; OHTA, K. ***Universal Electronic Cash***. *Advances in Cryptology* – CRYPTO 91*, Lecture Notes in Computer Science*, Vol. 576, pp. 324-337, Springer-Verlag, 1992.

RANKL, W; EFFING, W. ***Smart card Handbook.*** Third Edition. John Wiley & Sons. 2004.

RAUBER, F. **Desenho e implementação de um protocolo de pagamento eletrônico com *smart cards*.** UnB. 2008.

RIVEST, R; SHAMIR, A. ***PayWord and MicroMint - Two Simple Micropayment Schemes***. *Proceedings of 1996 International Workshop on Security Protocols*, (ed. Mark Lomas), (Springer, 1997*), Lecture Notes in Computer Sci*ence No. 1189, pages 69-87.

ROUSSEAU, L. ***Smart Card List* 2008**. Disponível em http://ludovic.rousseau.free.fr/softwares/pcsc-tools/smartcard_list.txt. Acessado em 20/12/2008.

SCA1. *Smart card* Alliance. ***Smart card applications***. Disponível em http://www.smartcardalliance.org/pages/smart-cards-applications. Acessado em 09/12/2008.

SHAMIR, A. ***Identity-based cryptosystems and signature schemes***. *Advances in Cryptology* – CRYPTO 1984.

SHANNON, C. ***A Mathematical Theory of Communication***, *Bell System Technical*

Journal, vol. 27, outubro/1948, pg. 379-423, 623-656.

SHANNON, C. ***Communication Theory of Secrecy Systems***, *Bell System Technical Journal*, vol. 28, outubro/1949, pg. 656-715.

SHERIF M. ***Protocols for Secure Electronic Commerce***. CRC Press, 2000. Boca Raton, Florida.

STALLINGS, W. ***Cryptography and Network Security. Principles and Practices***. *Fourth Edition*. Pearson Prentice Hall. 2006.

STINSON, D. ***Cryptography: Theory and Practice***. *Third Edition*. Chapman & Hall/CRC. 2006.

SUN1. ***Virtual Machine Specification***. *Java Card Platform*, Version 2.2.2. Sun Microsystems Inc. 15/03/2006.

SUN2. ***Java ME Technology***. Disponível em: http://java.sun.com/javame/technology/index.jsp. Acessado em 17/12/2008.

SUN3. ***Key Java Stats.*** *JavaOne 2008 Press Kit.* Disponível em: http://www.sun.com/aboutsun/media/presskits/javaone2008/index.jsp. Acessado em 22/12/2008.

SUN4. ***Java SE 6 Platform at a Glance***. Disponível em: http://java.sun.com/javase/6/docs. Acessado em 12/10/2008.

YAN, S. ***Number Theory for Computing***. Second Edition. Springer. 2002.

ZDNET. ***Hacker demos how to defeat Citibank's virtual keyboard***. Disponível em: http://blogs.zdnet.com/security/?p=195. Acessado em 17/12/2008.

APÊNDICES

A – FUNÇÕES PARA MANUSEIO DE APPLETS JAVA CARD

Verificação do funcionamento de um cartão inteligente

O primeiro passo quando se tem um aplicativo que utiliza cartões inteligentes é saber se o cartão não está danificado. Um teste normalmente utilizado baseia-se na obtenção do ATR e de um número pseudo-aleatório gerado pelo cartão. O comando APDU para retorno de um número psdeudo-aleatório no cartão GX4, que suporta o padrão GlobalPlatform, é a sequência hexadecimal: 80 50 00 00 08 48 6F 73 74 52 61 6E 64 1C conforme ilustrado no código fonte Java abaixo. O código fonte completo de um cliente em Java para retorno de um número pseudo-aleatório no padrão GlobalPlatform encontra-se no Apêndice A.

```
(...)

// Comando GlobalPlatform para obter número pseudo-aleatório no cartão TOP IM GX4
byte[] comando = { (byte) 0x80, (byte) 0x50, (byte) 0x00, (byte) 0x00,
                   (byte) 0x08, (byte) 0x48, (byte) 0x6F, (byte) 0x73,
                   (byte) 0x74, (byte) 0x52, (byte) 0x61, (byte) 0x6E,
                   (byte) 0x64, (byte) 0x1C };

(...)
```

A saída na console do Eclipse para a execução do código do Apêndice A, duas vezes, é mostrada abaixo. Observa-se que a "Resposta APDU" é diferente em ambas as execuções mostrando a pseudo-aleatoriedade e que, portanto, o cartão não está danificado nesta função. A resposta ATR 3b7d94000080318065b08301029083009000 indica que o modelo do cartão é o TOP IM GX4.

```
Leitora: OMNIKEY CardMan 5x21 0
Leitora: OMNIKEY CardMan 5x21-CL 0
Iniciando comunicação com cartão inteligente...
Conectado com T=0: PC/SC card in OMNIKEY CardMan 5x21 0, protocol T=0,
state OK
ATR: 3b7d94000080318065b08301029083009000
Comando APDU: 8050000008486f737452616e641c
Resposta
APDU:4d004082d513091126af0101fef369c21d98b475d2a1ce12d55892589000

Leitora: OMNIKEY CardMan 5x21 0
```

```
Leitora: OMNIKEY CardMan 5x21-CL 0
Iniciando comunicação com cartão inteligente...
Conectado com T=0: PC/SC card in OMNIKEY CardMan 5x21 0, protocol T=0,
state OK
ATR: 3b7d94000080318065b08301029083009000
Comando APDU: 8050000008486f737452616e641c
Resposta
APDU:4d004082d513091126af01019622e82530faea80d740b5f879d539929000
```

Listagem de applets carregados em um cartão

O segundo passo para se trabalhar com um cartão inteligente Java Card é identificar quais applets estão carregados no cartão. Para isto é essencial que se tenha um *software* de comunicação baseado na GlobalPlatform. Todos os fabricantes de cartões possuem *softwares* de manipulação de applets Java Card, porém estes *softwares* são proprietários. Uma alternativa gratuita é o software GPShell que é um aplicativo com licenciamento GPL v3 e tem o código fonte disponível para download no SourceForge além de implementar o padrão GlobalPlatform. O script abaixo é escrito para o GPShell e permite listar os applets de um cartão GX4, a chave ("- key 47454d5850524553534f53414d504c45") é específica para o cartão de desenvolvimento GX4 e significa, em ASCII, "gemxpressosample". Como ocorre com senhas e PIN´s, a tentativa de comunicação usando-se uma chave errada poderá bloquear o cartão, tipicamente é comum que o número de tentativas erradas seja limitado a dez tentativas. Um script XML para o Ant está ilustrado no Apêndice D e este permite que a saída seja na própria Console do Eclipse conforme Gráfico A-1.

```
mode_201
gemXpressoPro
enable_trace
establish_context
card_connect
select -AID A000000018434D00
open_sc -security 3 -keyind 0 -keyver 0 -key
47454d5850524553534f53414d504c45
get_status -element e0
card_disconnect
release_context
```

A saída da execução do script acima é ilustrada a seguir onde se pode identificar que há apenas um applet criado pelo usuário (state = F) cujo AID (Applet ID) é A0000000180a000001634200. Os outros applets (state = 1) são pré-carregados de fábrica.

```
mode_201
gemXpressoPro
enable_trace
establish_context
card_connect
* reader name OMNIKEY CardMan 3x21 0
select -AID A000000018434D00
--> 00A4040008A000000018434D00
<-- 6F198408A000000018434D00A50D9F6E061291334903029F6501FF9000
open_sc -security 3 -keyind 0 -keyver 0 -key
47454d5850524553534f53414d504c45 // Open secure channel
--> 80CA9F7F00
<--
9F7F2A4090662212913349030240823102 1A2726A8408200001293000003347018
000001010000000000000009000
--> 8050000008903D5422429083C600
<-- 4D00408231021A2726A80D01672CDF0D44B7952751AF8301D9305DDE9000
--> 8482030010DEA43FA1ACF4809EA02AD67F44AC4F05
<-- 9000
get_status -element e0
--> 80F2E000024F0000
<--
08A000000018434D000F9E10A000000018300602000000000000FF010010A000
000018300301000000000000FF010010A000000018300401000000000000FE
010010A000000018300602000000000000FE010010A000000018300601000000
00000000FF010010A000000018300301000000000000FE010008A00000001810
0106010008A000000018100101010007A000000003000010008A0000000181001
02010007A000000062020101 0007A000000062010201 0007A00000062010101 00
07A000000062000101 0008A000000018100108010 00CA0000000180A0000016342
000F009000
OP201_get_status() returned 17 items

List of applets (AID state privileges)
a000000018434d00  f     9e
a000000018300602000000000000ff    1     0
a000000018300301000000000000ff    1     0
```

```
a000000018300401000000000000000fe    1     0
a000000018300602000000000000000fe    1     0
a000000018300601000000000000000ff    1     0
a000000018300301000000000000000fe    1     0
a000000018100106  1     0
a000000018100101  1     0
a0000000030000    1     0
a000000018100102  1     0
a0000000620201    1     0
a0000000620102    1     0
a0000000620101    1     0
a0000000620001    1     0
a000000018100108  1     0
a0000000180a000001634200      f     0
card_disconnect
release_context
```

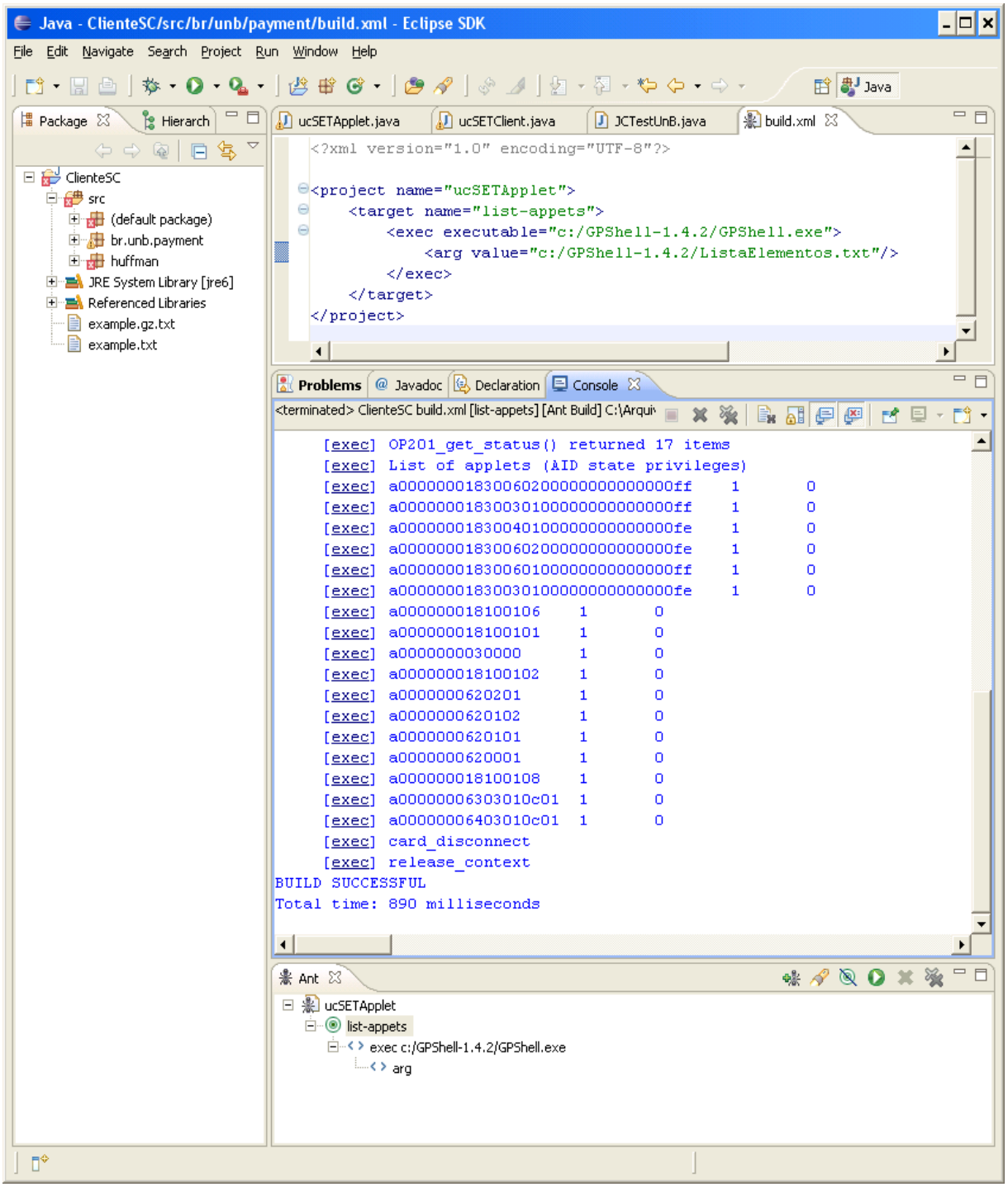

Gráfico A-1 – Saída na console do Eclipse para o script Ant de listagem

Remoção de applets de um cartão

Caso se necessite carregar um novo applet no cartão e o espaço disponível não seja suficiente é necessário apagar-se um ou mais applets do cartão. Usando-se um script para o GPShell é possível apagar-se um applet de um cartão, conforme ilustrado abaixo, o applet de AID a0000000180a000001634200 será apagado.

```
mode_201
```

```
gemXpressoPro
enable_trace
establish_context
card_connect
select -AID A000000018434D00
open_sc -security 0 -keyind 0 -keyver 0 -key
47454d5850524553534f53414d504c45
delete -AID a0000000180a000001634200
card_disconnect
release_context
```

Instalação de applets em um cartão

Para o caso dos cartões GX4, o applet do Apêndice B deve ser compilado com a versão 1.4.2 do JDK e pode usar as bibliotecas da versão 2.2.2 ou 2.2.1 do JCDK. A conversão da classe Java (.CLASS), resultante do passo anterior, para um arquivo do tipo *converted* applet (.CAP) deverá ser realizada apenas com a versão 2.1.1. do JCDK devido a uma limitação do cartão GX4. O comando para fazer esta conversão é:

```
C:\java_card_kit-2_1_1\bin\converter.bat -exportpath
"C:\java_card_kit-2_1_1\api21" br.unb.payment
0xa0:0x00:0x00:0x00:0x62:0x03:0x01:0x0c:0x01 1.0 -v -applet
0xa0:0x00:0x00:0x00:0x62:0x03:0x01:0x0c:0x01:0x01 payment
```

O comando acima diz ao aplicativo conversor do JCDK (converter.jar) para converter o pacote "br.unb.payment", com um AID de pacote "A0 00 00 00 62 03 01 0C 01", na sua versão 1.0, sendo que o AID do Applet será "A0 00 00 00 62 03 01 0C 01 01" e seu nome será "payment".

Finalmente, para se instalar um applet novo em um cartão pode-se usar o script do GPShell abaixo onde, por exemplo, um *converted* applet de nome "payment.cap" é instalado com AID A00000006203010C0101.

```
mode_201
gemXpressoPro
enable_trace
establish_context
card_connect -readerNumber 1
select -AID  A00000006203010C01
open_sc -security 3 -keyind 0 -keyver 0 -key
47454d5850524553534f53414d504c45
```

```
install -file payment.cap -sdAID A000000062030l0C0101 -nvDataLimit
4096 -instParam 00 -priv 2
card_disconnect
release_context
```

Exemplo de código para testes de comunicação com Java Cards

```
// Programa adaptado do NFC Secure Element Example External Reader
// Nokia Forum, acessado em 12/12/2008, disponível em:
//
http://wiki.forum.nokia.com/index.php/NFC_Secure_Element_Example_External_Reader
// Autores:
// Jose Maria Leocadio
// Departamento de Engenharia Elétrica
// Universidade de Brasília

package br.unb.payment;

import java.util.List;
import java.util.ListIterator;

import javax.smartcardio.Card;
import javax.smartcardio.CardChannel;
import javax.smartcardio.CardException;
import javax.smartcardio.CardTerminal;
import javax.smartcardio.CommandAPDU;
import javax.smartcardio.ResponseAPDU;
import javax.smartcardio.TerminalFactory;

public class JCTestUnB {

  public static void main(String[] args) throws CardException {
      List<CardTerminal> terminals = null;
      Card cartao = null;

      // Lista as leitoras configuradas e conecta o cartão inteligente na
primeira da lista
      try {
          terminals = getterminals();
          ListarLeitoras(terminals);
          cartao = ConectarCartao(terminals.get(0)); // 0 = primeiro da lista, ou
poder-se-ia utilizar a função
          // getTerminal com o nome do leitor, por ex. getTerminal("OMNIKEY
CardMan 5x21-CL 0")
```

```
        } catch (CardException e) {
            e.printStackTrace();
        }

        // Comando GlobalPlatform para obter número aleatório no cartão TOP IM GX4
        byte[] comando = { (byte) 0x80, (byte) 0x50, (byte) 0x00, (byte) 0x00,
                (byte) 0x08, (byte) 0x48, (byte) 0x6F, (byte) 0x73,
                (byte) 0x74, (byte) 0x52, (byte) 0x61, (byte) 0x6E,
                (byte) 0x64, (byte) 0x1C };

        // Obtem um canal de comunicação com o smart card
        CardChannel canal = cartao.getBasicChannel();

        // Envia o comando APDU e armazena a resposta
        ResponseAPDU resposta = canal.transmit(new CommandAPDU(comando));

        // Mostra o Comando APDU e a Resposta APDU
        System.out.println("Comando APDU: " + arrayToHex(comando));
        System.out.print("Resposta APDU:" + arrayToHex(resposta.getBytes()));

    }

    private static Card ConectarCartao(CardTerminal terminal)
            throws CardException {
        Card cartao = null;
        System.out.println("Iniciando comunicação com cartão inteligente...");

        // Verifica se cartão inteligente está inserido na leitora
        if (!terminal.isCardPresent())
            System.out.println("Insira um cartão inteligente");
        else
            cartao = terminal.connect("T=0");

        if (cartao != null) {
            System.out.println("Conectado com T=0: " + cartao);

            System.out.println("ATR: " + arrayToHex(cartao.getATR().getBytes()));
        }
        return cartao;
    }

    public static String arrayToHex(byte[] data) {
        StringBuffer sb = new StringBuffer();

        for (int i = 0; i < data.length; i++) {
            String bs = Integer.toHexString(data[i] & 0xFF);
```

```
            if (bs.length() == 1) {
                sb.append(0);
            }
            sb.append(bs);
        }
        return sb.toString();
    }

    private static void ListarLeitoras(List<CardTerminal> terminals) {
        ListIterator<CardTerminal> i = terminals.listIterator();
        while (i.hasNext()) {
            System.out.println("Leitora: " + ((CardTerminal) i.next()).getName());
        }
    }

    private static List<CardTerminal> getterminals() throws CardException {
        TerminalFactory factory = TerminalFactory.getDefault();
        // Retorna a lista de terminals configuradas no ambiente
        List<CardTerminal> terminals = factory.terminals().list();
        return terminals;
    }
}
```

Listagem de applets de um cartão TOP IM GX4 na Console do Eclipse

Script XML para o Ant

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="ucSETApplet">
      <target name="list-appets">
            <exec executable="c:/GPShell-1.4.2/GPShell.exe">
                  <arg value="c:/GPShell-
1.4.2/ListaElementos.txt"/>
            </exec>
      </target>
</project>
```

Script para o GPShell, “ListaElementos.txt”

```
mode_201
gemXpressoPro
enable_trace
establish_context
card_connect
select -AID A000000018434D00
```

```
open_sc -security 3 -keyind 0 -keyver 0 -key
47454d5850524553534f53414d504c45 // Open secure channel
get_status -element 20
card_disconnect
release_context
```

B – CÓDIGO FONTE DO CLIENTE JAVA E DO APPLET JAVA CARD

Encontram-se no CD-ROM, em anexo à dissertação.