Alexandre Nagano
Isabelle Louise Frete Miranda
José Eduardo Chiarelli Bueno Filho

Sistema de Autenticação por Biometria Baseado na Tecnologia Java Card: JBCARD

Trabalho Final apresentada à Escola Politécnica da Universidade de São Paulo para a disciplina PSI2594

Departamento de Engenharia Elétrica - Sistema Eletrônicos

Orientados: Professor Doutor Wang Jiang Chau

São Paulo
2013

Alexandre Nagano
Isabelle Louise Frete Miranda
José Eduardo Chiarelli Bueno Filho

Sistema de Autenticação por Biometria baseado na Tecnologia Java Card: JBCARD

Trabalho Final apresentada à Escola Politécnica da Universidade de São Paulo para a disciplina PSI2594

São Paulo
2013

AGRADECIMENTOS

Ao Prof. Dr. Wang Jiang Chau, nosso orientador, pela grande contribuição e constante apoio dados ao longo do desenvolvimento desse projeto.

Às nossas famílias por todo suporte no período da faculdade e, principalmente, durante o período em que nós empenhamos neste trabalho de conclusão de curso.

Aos nossos amigos de turma, companheiros nessa jornada iniciada há cinco anos, por todos os risos, tristezas e alegrias compartilhados entre nós.

RESUMO

O seguinte trabalho consiste da finalização do um Projeto de um sistema de autenticação por biometria baseado em smart cards com a abordagem Store-on-Card sustentada por um esquema de criptografia assimétrico.

No método Store-on-Card os dados biométricos são apenas guardados nos cartão, sendo que estes dados são puxados para um computador onde o algoritmo de comparação é executado. Este método foi o primeiro a ser desenvolvido, pois os smartcards ainda não tinham a capacidade de processamento necessária para se executar algoritmos complexos dentro do cartão. Quando a capacidade para se processar tais algoritmos dentro do cartão foi atingida o método Match-on-Card pôde ser desenvolvido. Neste método o algoritmo de comparação é executado dentro do cartão, não havendo, assim, a necessidade de se puxar os dados biométricos para um computador.

O ponto de partida foi a realização de análises de trabalhos relacionados que defendem a abordagem Match-on-Card, de onde algumas vulnerabilidades são explicitadas e utilizadas, posteriormente, para se realizar uma análise conceitual das alternativas para se solucionar um problema mais geral, o de se desenvolver um sistema de autenticação confiável. Concluindo o trabalho apresentaremos a discussão e um detalhamento dos passos e tecnologias necessárias para implementação da solução e as fases de implementação dos sistemas utilizados, suas verificações e validações e a qualidade e confiabilidade do produto apresentado.

Palavras-chave: Java card. Biometria. Linguagem de programação. Multi-aplicação.

ABSTRACT

The following project consists in completing a draft of a biometric authentication system, based on smartcard approach, with Store-on-Card supported by an asymmetric encryption design.

In the Store-on-Card method, biometric data is stored only in ~~on~~ the postcards, and the data is downloaded into a computer where the comparison algorithm is executed. This mechanism was first developed, because the smartcard does not have the required processing power to run complex algorithms within the design. The Match-on-Card method will only be able to be completely developed once the ability to process the algorithms inside the cards becomes viable. As such, there will be no need to download the biometric data to a computer, since the algorithm is executed within the card itself.

The starting point was the analysis of related works that defends the Match-on-Card approach, where some vulnerabilities are explained and used subsequently to develop a conceptual analysis of alternatives for solving a more general problem.

As we complete the presented paper, we discuss and specify the steps, required technologies and reliability, to fully implement the solution to the product displayed.

Keywords : Java card . Biometrics . Digital security . Multi application.

Lista de Figuras

LISTA DE TABELAS

SUMÁRIO

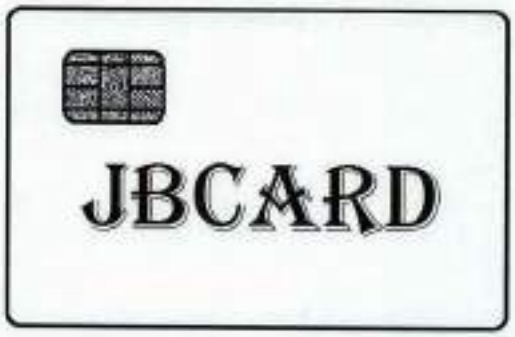
1 INTRODUÇÃO

1.1 TÍTULO, SIGLA E LOGOTIPO DO PROJETO

Título: Java Biometric Card

Sigla: JBCARD

Logotipo do Projeto:



1.2 MOTIVAÇÃO

A diminuição das dimensões dos transistores vem permitindo a integração de sistemas digitais cada vez maiores e mais complexos dentro de um único chip, caracterizando o fenômeno chamado VLSI (Very Large Scale Integration). Este fenômeno, por sua vez, permitiu o desenvolvimento de verdadeiros sistemas computacionais integrados, que configuram a base dos sistemas embarcados modernos. Dentre estes, o mais difundido comercialmente na atualidade é o smartcard, sendo utilizado em diversas áreas como banking, com os cartões de débito ou crédito, e no transporte público, com os cartões-passagem.

Com a progressão do VLSI, os smartcards começaram a se apresentar com cada vez mais memória e maior capacidade de processamento, chegando ao ponto de múltiplas aplicações poderem ser instaladas em um único cartão e de estas aplicações poderem ser escritas em linguagens de alto nível tais como Java e C, levando ao advento de plataformas completas de desenvolvimento de softwares de smartcards como, por exemplo, o Java Card.

A grande difusão dos smartcards promovida por estas evoluções tecnológicas fez surgir, porém, problemas de segurança como roubo e clonagem, fazendo com que empresas que fazem uso de serviços que envolvem smartcards não tenham garantia de que é o proprietário do cartão de fato que o está utilizando. Um método comumente adotado para se contornar este problema é a utilização de senhas para validar as operações com o cartão, no entanto, senhas podem ser adivinhadas,

roubadas ou fornecidas a terceiros, não havendo, deste modo, uma autenticação propriamente dita dos usuários.

Esta ausência de uma autenticação verdadeira configura uma vulnerabilidade nos serviços que utilizam smartcards. Este problema pode, entretanto, ser solucionado com o uso de processos biométricos, que geram dados únicos para cada indivíduo, como uma ferramenta para a autenticação.

Levando em conta a multiaplicabilidade que a evolução de hardware e das plataformas de desenvolvimento de software trouxeram aos smartcards, este trabalho propõe o desenvolvimento de um projeto de um sistema de autenticação por biometria de impressões digitais baseado na plataforma de desenvolvimento Java Card, que terá como objetivo oferecer às empresas, através da utilização de um smartcard, uma garantia de que quem está tentando acessar uma de suas funcionalidades é um usuário permitido.

1.3 Estrutura do Texto

A estrutura do texto está organizada da seguinte forma: a parte 2 mostra a contextualização formal do problema e as análises que foram necessárias para se chegar a uma arquitetura geral da solução; na parte 3 há o detalhamento da metodologia utilizada no trabalho juntamente com uma descrição das tecnologias que serão utilizadas pela solução; a parte 4 descreve como está o desenvolvimento do trabalho e as etapas que foram divididas para melhor implementação; a parte 5 exibe o cronograma do projeto; a parte 6 mostra o orçamento final ; a parte 7 mostra as relações de execução de tarefas e gerenciamento da equipe do projeto; a parte 8 apresenta os resultados obtidos; a parte 9 mostra a discussão final desenvolvida pelo grupo e finalmente a parte 10 apresenta as conclusões do projeto realizada pelo Grupo. Após essa estrutura também identificamos o Anexo e os Apêndices.

2 CONTEXTUALIZAÇÃO

2.1 CONTEXTO DO PROBLEMA

A cada 15 segundos, uma tentativa de roubo de dados pessoais acontece no Brasil, segundo pesquisa da Serasa Experian [Referência 33]. Por isso, bancos e operadoras de cartões de crédito estão investindo pesado em segurança. Como todo cuidado é pouco, os clientes também devem ficar atentos ao fazer compras ou acessar os bancos pela internet.

Novas pesquisas realizadas pelo Instituto de Ensino e Pesquisa (Insper) revelam que a preocupação pelo roubo de dados pessoais e financeiros supera o medo ao terrorismo, às epidemias e ao desemprego.

Os resultados de uma enquete mundial realizada pela Empresa Visa Internacional [Referência 34]., revelam que o roubo ou a perda de informações pessoais e financeira é a principal preocupação dos consumidores em todo o mundo, superando a preocupação pela degradação ao meio ambiente (que está em 2° lugar), pelo terrorismo (3° lugar) e outros gran des problemas como o desemprego, as doenças, as epidemias e os desastres naturais.

Cada vez mais os consumidores concordam que os benefícios de um maior empenho para proteger a informação pessoal e financeira compensam os inconvenientes ou os custos acarretados, sugerindo um amplo apoio para novos enfoques de segurança de dados.

Os avanços na tecnologia de autenticação do portador de cartão aumentaria a confiança dos consumidores na proteção de sua informação pessoal, fazendo com que se sentissem mais seguros se contassem com mais informações sobre como proteger-se contra o roubo de seus dados pessoais.

Outras possíveis melhorias incluem desenvolver novas leis e risco zero no uso fraudulento dos cartões, uma vez que os consumidores que se preocupam com a segurança da informação realizam menos compras on-line e compram menos por telefone.

Em um momento em que o roubo de dados digitais e os delitos similares são temas constantes de jornais impressos e da televisão, soluções como o desenvolvimendo de um cartão que além da segurança através de senhas possuirá

a segurança da biometria digital, pode ser considerado um produto que resolverá a insegurança dos consumidores em relação a esse Contexto.

2.2 Definição do problema

Os roubos e fraudes de cartões bancários são motivados pelo fato de que as senhas vinculadas a estes podem ser obtidas tanto com ou sem o consentimento de seus proprietários, fazendo com que os bancos não possam ter total certeza de que quem está utilizando o cartão é, de fato, seu dono. Este problema pode ser estendido a outras empresas que desejem saber se quem está utilizando seus serviços é um usuário permitido.

Sumarizando, observa-se uma carência de um sistema de autenticação verdadeiro nas empresas que desejam assegurar que os seus serviços só estejam acessíveis para os seus clientes, não havendo possibilidade de uso dos serviços por não clientes.

O único modo de se realizar um sistema de autenticação verdadeiro é através da biometria e existem dois métodos de integrá-la a smartcards, sendo eles o Store-on-Card (onde os dados biométricos são apenas guardados no cartão) e o Match-on-Card (os dados biométricos além de serem guardados são também comparados dentro do próprio cartão).

Porém, devido as limitadas capacidades de memória e processamento dos smartcards, a precisão dos dados biométricos e dos algoritmos de comparação suportadas por sistemas Match-on-Card são, também, limitadas, tornando-os susceptíveis a ataques que geram falsas entradas como o Hill-Climbing, eliminando a garantia de que quem está utilizando o cartão é o cliente.

Desse modo o método Store-on-Card torna-se uma boa alternativa, pois ele permitiria que algoritmos mais sofisticados fossem utilizados, pois todo o processamento da comparação seria realizado em um computador e dados biométricos mais detalhados poderiam ser utilizados, pois haveria mais espaço dentro do smartcard. Porém, para tornar o método Store-on-Card mais seguro ele deve ser acompanhado de um ou mais sistemas de criptografia.

2.3 Objetivos do projeto

Como objetivos que motivaram o grupo no desenvolvimento do projeto podemos citar:

- Desenvolver um sistema de cartão java card multiaplicativo cuja segurança do usuário seja garantido não só pela senha como através da biometria digital;
- Garantir que o cartão poderá ser utilizado por diversas empresas com aplicações diferenciadas e que haja a possibilidade de utilização de senha com biometria digital ou apenas senha;
- Desenvolver um sistema que seja confortável para o usuário;
- Buscar um sistema que não seja de alto custo;
- Desenvolver técnicas de criptografia para aprimorar a segurança do sistema a fim de que seja o mais seguro possível.

Para a caracterização adequada dos objetivos do projeto definiu-se claramente o problema a ser solucionado e, a partir disso, foi feita uma análise de viabilidade das alternativas disponíveis para a solução do problema. Cumpridas estas etapas a alternativa escolhida será detalhada para a definição de uma proposta para a solução do problema.

2.4 Definição da solução

A solução para o problema descrito acima pode, então, ser definida como um sistema de autenticação por biometria do tipo Store-on-Card acompanhada por um sistema de criptografia para o aumento da segurança. A solução apresenta os requisitos descritos abaixo. Para maiores detalhes de como foram definidos esses requisitos consultar APÊNDICE A.

2.4.1 Requisitos da solução

- → O sistema de biometria utilizado deve ser de baixo custo e deve ser confortável para o usuário
- → O smartcard utilizado deve suportar múltiplas aplicações, para que um único cartão possa ser utilizado por diversas empresas
- → O sistema de criptografia deve possuir uma entidade certificadora, para que não haja o risco dos dados serem descriptografados por um terceiro mal intencionado.

2.4.2 Elementos da solução

A próxima etapa para a definição completa da solução é a escolha, através de uma análise de viabilidade, das ferramentas que serão utilizadas para se construir esta solução. Estas ferramentas serão escolhidas dentro dos seguintes conceitos:

- ➔ Tipo de biometria
- ➔ Sistema operacional do cartão
- ➔ Estrutura do sistema de criptografia

2.4.3 Tipo de biometria

As opções para o tipo de biometria são:

- ➔ Íris
- ➔ Voz
- ➔ Assinatura
- ➔ Impressão digital

Para que a escolha de um desses métodos de autenticação seja feita de maneira sistemática, um processo chamado AHP (Analytical Hierarchy Process) será utilizado. Este método consiste na atribuição de pesos a determinados critérios de escolha, através de uma comparação dois a dois. Os pesos são determinados com a montagem de uma matriz onde esta comparação é feita.

Primeiramente, definem-se os critérios:

- ➔ **C1:** conforto para o usuário
- ➔ **C2:** baixo custo
- ➔ **C3:** alta precisão dos parâmetros (templates)
- ➔ **C4:** dados gerados de tamanho razoável

Com os critérios definidos as comparações dois a dois podem ser feitas, resultando em uma matriz que possibilitará o cálculo dos pesos:

Tabela 1 - Critérios de avaliação

	C1	C2	C3	C4	Média geométrica da linha	Peso(%)
C1	1	1/3	5	3	1,50	30,63%
C2	3	1	3	3	2,28	46,69%
C3	1/5	1/3	1	3	0,67	13,70%
C4	1/3	1/3	1/3	1	0,44	8,99%
				Σ	4,88	100,00%

Analisando quais critérios se encaixam nas opções, a que resultar na maior soma de pesos será a escolhida:

- → **Íris**: atende aos critérios C3 e C4, resultando em um peso de **22,69%**
- → **Voz**: atende aos critérios C2 e C3, resultando em um peso de **60,39%**
- → **Assinatura**: atende aos critérios C1 e C3, resultando em um peso de **44,33%**
- → **Impressão digital**: atende aos critério C1, C2 e C4, resultando em um peso de **86,31%**

Apesar de os templates gerados pelas impressões digitais não serem os mais precisos, este tipo de biometria atendeu a todos os demais critérios resultando, assim, na maior soma de pesos levando a conclusão de que as impressões digitais são o melhor tipo de biometria para o projeto.

2.4.4 Sistema operacional do cartão

As plataformas de desenvolvimento de software de smartcards são baseadas em sistemas operacionais de smartcards ou COS (Card Operating System). Estes sistemas operacionais começaram a surgir quando se percebeu que a fabricação de cartões com microcontroladores programados em hardware (mask-programmed microcontrollers) era cara e demorada demais. Assim a demanda por cartões que possuem rotinas pré-programadas em seus kernels aumentou, pois desse modo aplicações gravadas na EEPROM seriam utilizá-las conforme a necessidade. Estas rotinas eventualmente evoluíram e se tornaram verdadeiros sistemas operacionais.

Os dois sistemas operacionais de smartcards mais utilizados atualmente são os sistemas MULTOS e o Java Card. Estes sistemas permitem que códigos escritos por terceiros sejam carregados no cartão através de um mecanismo chamado API (Application Programming Interface) que elimina a necessidade de reprogramação de rotinas já presentes no sistema operacional.

O critério de escolha neste caso será a facilidade com que se podem manipular dados biométricos no COS. Levando em conta este critério o sistema operacional pode ser escolhido diretamente como sendo o Java Card, pois este sistema apresenta diversas API's relacionadas com biometria além de ser baseado na linguagem Java, o que facilita a integração de programas escritos nesta linguagem com o cartão. Estes propriedades do Java Card o levaram a ser o sistema operacional escolhido para o projeto.

2.4.5 Estrutura do sistema de criptografia

A estrutura do sistema diz respeito a como os dados biométricos serão criptografados no cartão. Isso pode ser feito de dois modos: criptografia simétrica ou assimétrica.

Se a criptografia simétrica for escolhida, os dados seriam criptografados com uma chave privada, fazendo com que todos os cartões possuíssem uma chave diferente. Esta alternativa mostra-se logo inviável, pois, com ela, todos os pontos onde o cartão fosse ser utilizado teriam que ter uma lista de chaves privadas, podendo comprometer a segurança do sistema.

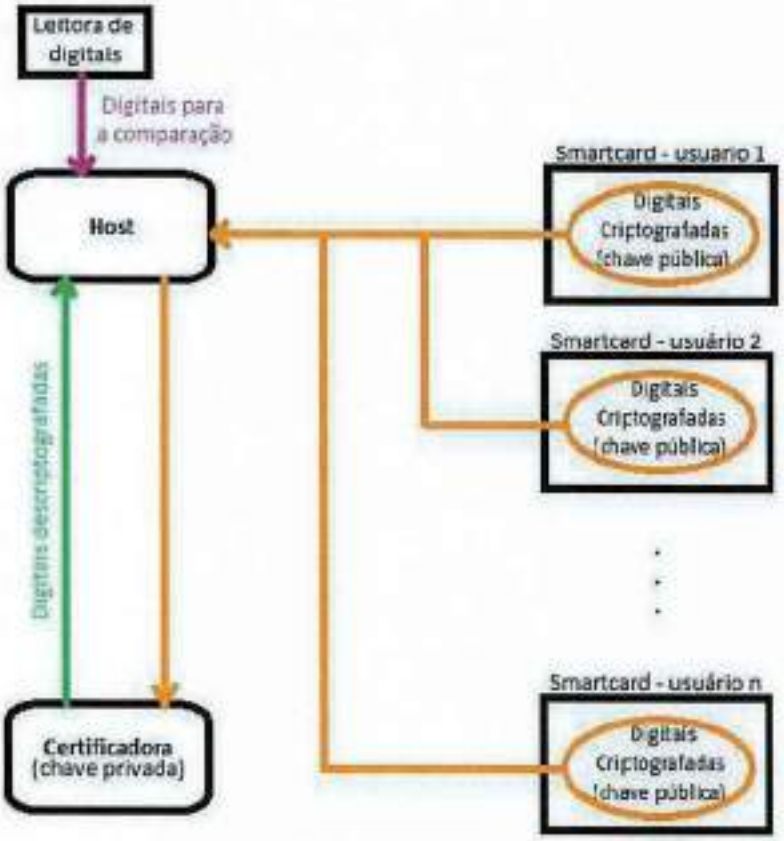
Assim, a segunda alternativa (criptografia assimétrica) será a escolhida, pois, com ela, os dados poderão ser criptografados nos cartões com uma única chave pública, sendo que a chave privada seria detida por uma empresa certificadora.

2.5 Arquitetura geral da solução

Escolhidas as ferramentas, a solução pode ser delineada:

Os dados biométricos da impressão digital do cliente serão guardados criptografados com uma chave pública em um smartcard com o sistema operacional Java Card, caracterizando um sistema Store-on-Card. Ao utilizar o cartão, os dados biométricos criptografados com a chave pública serão passados para o host (ponto de utilização), e serão enviados à empresa certificadora onde se encontra a chave privada que irá descriptografar estes dados. Após este processo os dados descriptografados serão mandados de volta para o host para que possam onde serão comparados com os dados obtidos por uma leitora de impressões digitais.

Figura 1 - Arquitetura geral da solução



2.6 Referencial Teórico

2.6.1 Criptografia

Sempre que há necessidade de se proteger informações importantes, necessitamos de um modo seguro de armazenar estas informações e de transmiti-las, para isso utilizamos a criptografia. Esta técnica consiste em tornar a informação a ser protegida ilegível, de forma que apenas o receptor desta informação consiga traduzi-la.

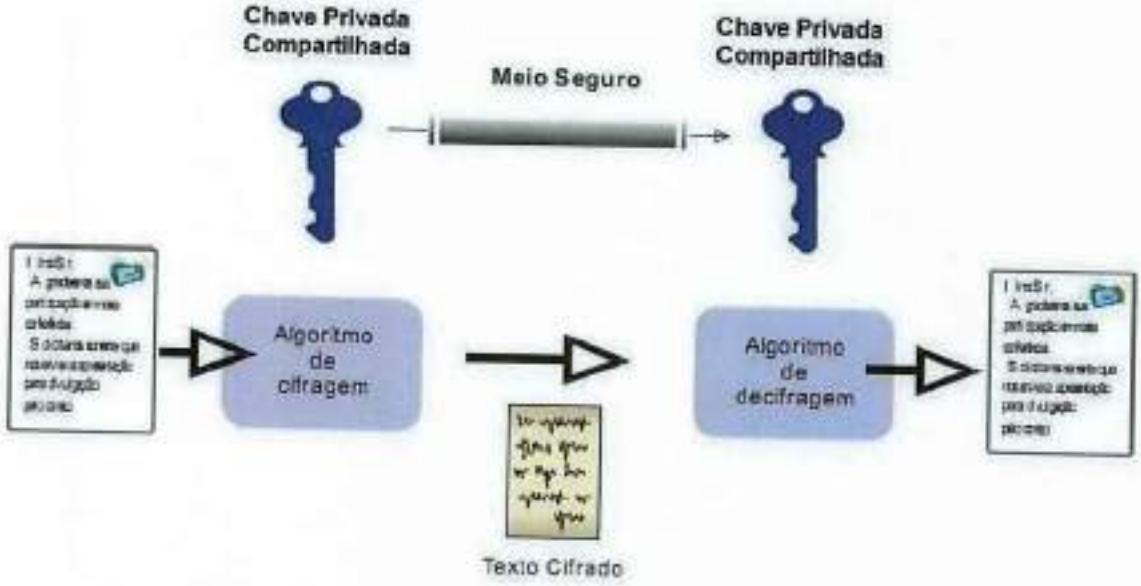
No processo de criptografia, temos dois itens principais, o algoritmo e a chave. O algoritmo consiste em fórmulas e transformações matemáticas que tornam a informação ilegível, porém a maioria desses algoritmos são publicados. A chave consiste em um vetor de bits, onde este modifica e controla o algoritmo de encriptação, fazendo com que para um mesmo algoritmo, tenhamos encriptações diferentes. Temos dois tipos de chaves: pública e privada. A chave pública é distribuída livremente, enquanto a chave privada possui apenas um dono.

Também classificamos a criptografia em dois tipos: assimétrica e simétrica.

2.6.1.1 Criptografia simétrica

Na criptografia simétrica trabalhamos com troca de informação usando chaves privadas como na figura abaixo.

Figura 2- Criptografia simétrica - Retirada de [5]

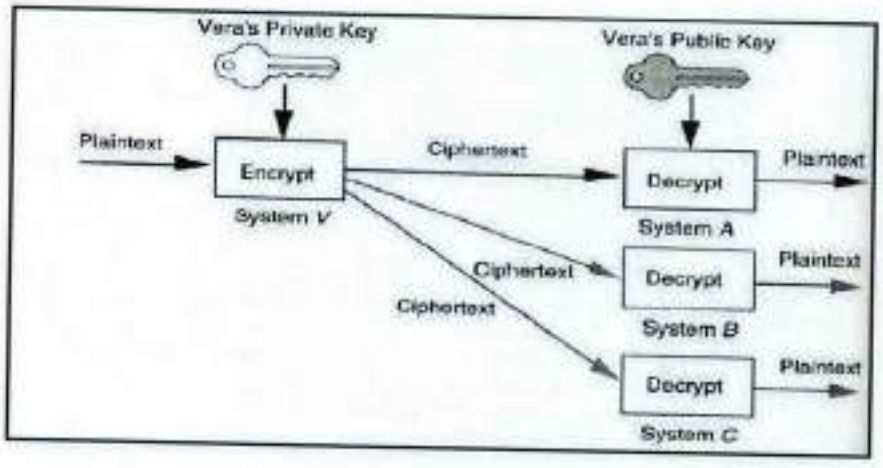


Neste tipo de criptografia, é mantido um canal confidencial de informação, onde apenas os detentores da chave privada conseguem decifrar o arquivo encriptado.

2.6.1.2 Criptografia assimétrica

Na criptografia assimétrica, teremos troca de informações usando chave pública e chave privada.

Figura 3- Criptografia assimétrica - Retirada de [5]



Neste caso, teremos duas vantagens:

- ➔ Confidencialidade: toda mensagem encriptada pela chave pública só poderá ser decriptada pelo detentor da chave privada.
- ➔ Autenticidade: toda mensagem decriptada pela chave pública possui a garantia de que a mensagem foi encriptada pelo dono da chave privada e não outra pessoa.

2.6.2 Biometria

A análise de características humanas já é utilizada vastamente nos dias de hoje. Podemos usá-la em identificações criminais e controle de acesso tanto de lugares como de gagdets, sendo um exemplo disso um celular recém lançado que controla o acesso do usuário por reconhecimento facial.

Diversos tipos de biometria foram desenvolvidos ao longo dos anos, como por exemplo reconhecimento de íris ou de retina, reconhecimento de voz e geometria da mão, porém neste projeto utilizaremos a impressão digital como parâmetro de acesso aos dados do cartão. A impressão digital humana é composta de linhas formadas pelas elevações da pele e é uma característica única entre seres humanos, mesmo entre irmão uni vitelinos.

A análise da impressão digital se baseia na identificação de formatos únicos da digital, onde teremos três níveis de visualização: o global, o local e o muito fino. No nível global, visualizaremos pontos de singularidade, chamados de núcleo e delta, que são muito importantes, mas não confere uma análise precisa da digital.

No nível local podem-se observar minúcias. As minúcias podem ser: terminação de uma crista, bifurcação, crista independente, ponto ou ilha, lago, espora e cruzamento. As duas minúcias mais importantes são terminação e bifurcação.

No nível muito fino, podemos observar os poros de suor, que necessitam de imagens de alta resolução para serem visualizados. Na figura abaixo, os poros são marcados como pontos brancos e as minúcias como pontos pretos. O processo de captação e análise eletrônica da impressão da impressão digital acontece nos seguintes passos.

Figura 4- Imagem dos pontos de singularidade – Retirada de [13]

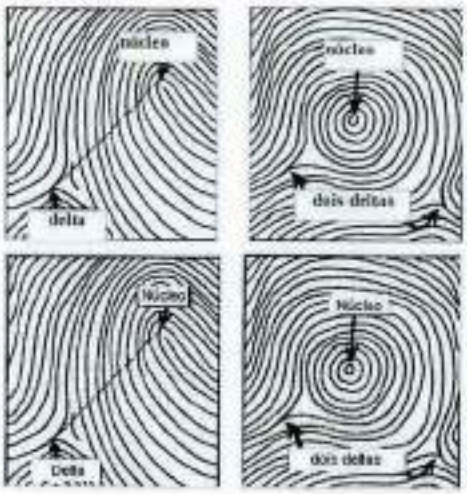
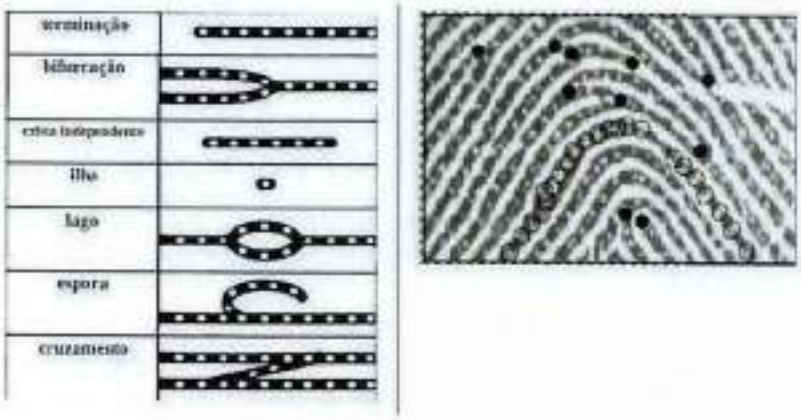


Figura 5- Imagem das possibilidades de minúcias encontradas – Retirada de [13]



1) O leitor capta sua digital e envia a imagem para seu computador similar à figura abaixo.

Figura 6- Imagem da digital original – Retirada de [13]

2º) Aplicando um filtro, a digital é reforçada em cor preta e o fundo muda para branco.

Figura 7- Imagem após aplicação de filtro - Retirada de [13]

3º) Após o filtro, a imagem é binarizada, com as linhas reduzidas à um único pixel de largura.

Figura 8- Imagem binarizada – retirada de [13]

4º) Com a imagem pixelizada, é possível identificarmos pontos de minúcias. Fazendo um exame de cada pixel na imagem, notamos que se houver um pixel branco sem vizinhos significa que encontramos um ponto terminal. Caso um ponto branco possua 3 pontos vizinhos, significa que encontramos uma bifurcação.

Figura 9- Imagem dos pixels e identificação das minúcias – Retirada de [13]

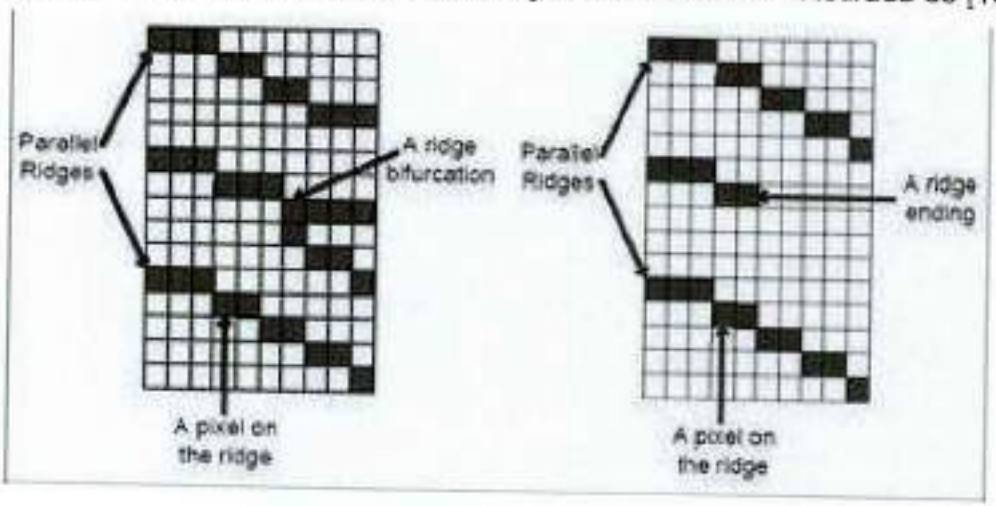


Figura 10- Minúcias com falsos positivos - Retirada de [13]

5º) Com os pontos de minúcia definidos, o próximo passo a ser implementado na biblioteca em termos de desenvolvimento é a comparação dos pontos de minúcias para obter o índice de similaridade entre duas impressões digitais: a cadastrada e a de acesso.

3 METODOLOGIA DO TRABALHO

3.1 DIAGRAMAS DE BLOCOS DOS PROCESSOS DO PROJETO

Esta seção tem como objetivo apresentar uma divisão dos procedimentos do projeto visando um melhor detalhamento funcional deste. São apresentados quatro diagramas de blocos, sendo que os três primeiros representam etapas fundamentais do sistema: cadastro, instalação da aplicação e comparação das impressões digitais. O último diagrama representa como o sistema funcionará como um todo.

Esta divisão é fundamental para a implementação, pois divide o trabalho em etapas e blocos interdependentes, o que facilita o gerenciamento do tempo e o trabalho em equipe.

3.1.1 Cadastro

- Primeiramente o usuário deverá buscar uma certificadora que possua contrato com a empresa desejada e a leitora de impressão digital para realizar o cadastro da digital
- Em seguida será captada a imagem da impressão digital que será criptografada com uma chave pública e armazenada no cartão do usuário
- A certificadora será responsável por definir uma senha de acesso a esse modelo armazenado
- A chave privada associada a chave pública com a qual o template da digital foi criptografado ficará em posse da empresa certificadora

Figura 11 - Diagrama Cadastro – Fonte Própria

3.1.2 Instalação da aplicação (host)

- ➔ Host deve fazer seu cadastro na empresa certificadora, para que esta seja certificada e receba todos os equipamentos e softwares necessários para utilizar a tecnologia do cartão.
- ➔ O host especifica as características funcionais do seu aplicativo.
- ➔ A certificadora desenvolve o aplicativo especificado
- ➔ Uma chave de sessão privada é utilizada para uma conexão segura entre host e certificadora.
- ➔ Aplicativo pronto é enviado de forma segura ao host, junto com executável que instala o aplicativo.
- ➔ Usuário faz cadastro na certificadora (detalhes em CADASTRO) e recebe o cartão com seus dados
- ➔ Usuário vai ao host cadastrado para utilizar o cartão
- ➔ Host instala o aplicativo com os arquivos enviados pela certificadora.

Figura 12 - Diagrama Host – Fonte Própria

3.1.3 Comparação das Impressões digitais

- ➔ Ao inserir o cartão na leitora do host, o template da digital será puxado para o computador (host), sendo que este template está criptografado com a chave pública.

- O template criptografado com a chave pública será enviado à empresa certificadora, onde se encontra a chave privada que irá realizar a descriptografia do template
- Após a descriptografia, o template será enviado de volta ao computador do host, onde será feita a comparação deste template com um outro retirado no momento da utilização do cartão por uma leitora de impressões digitais

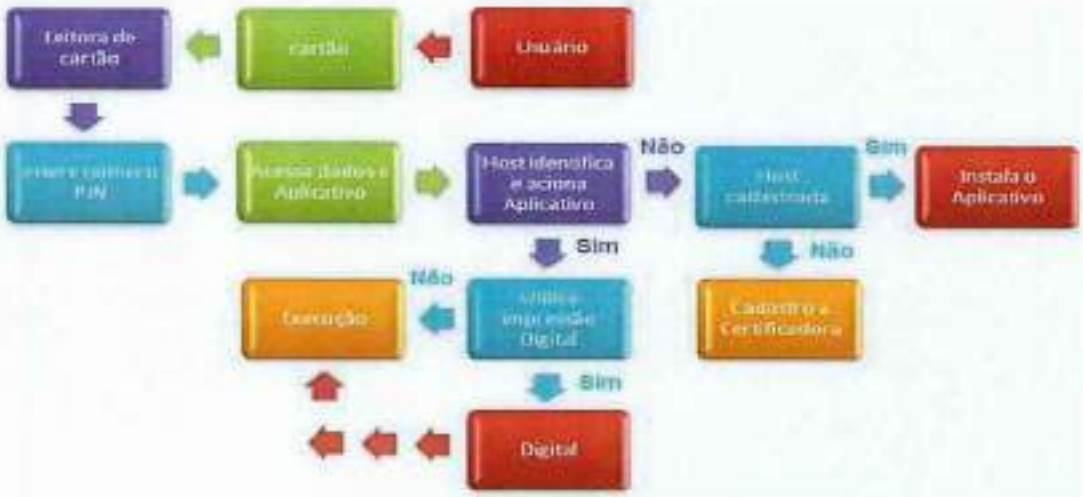
Figura 13 - Diagrama Comparação digital – Fonte Própria

3.1.4 Utilização do produto

A utilização do sistema se dará nos seguintes passos:

- O usuário insere seu cartão e sua senha (PIN) na leitora do host para que este possa visualizar as aplicações contidas no cartão
- O host verifica se a aplicação correspondente a sua empresa se encontra no cartão.
    - Se a aplicação não estiver presente é verificado se o host está cadastrado na empresa certificadora
        - Se não, um cadastro será realizado
        - Se sim, o host recebe a permissão para instalar sua aplicação no cartão
    - Se a aplicação do host já estiver presente, é feita uma verificação se esta aplicação utiliza o sistema de autenticação ou não
        - Se não, a aplicação é executada
        - Se sim, a comparação de digitais será feita para a validação da execução da aplicação

Figura 14 - Diagrama da utilização do produto – Fonte Própria



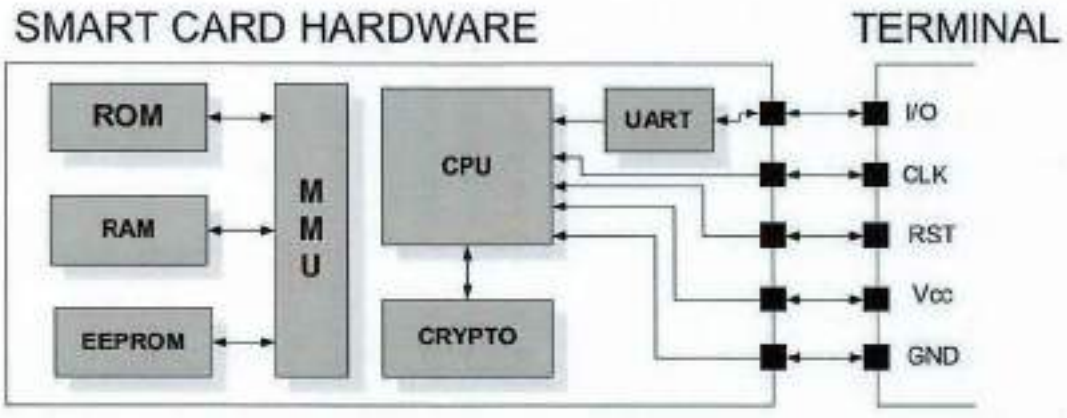
3.2 TECNOLOGIAS A SEREM UTILIZADAS

3.2.1 Smartcard

O smartcard consiste em um cartão que possui diversos módulos de hardware em seu interior, que conectam entre si e a terminais com funções especificas. Seu desenvolvimento foi impulsionado pela falta de segurança dos dados nos cartões com tarja magnética, onde os dados eram facilmente manipulados. A estrutura do smartcard será especificada abaixo:

3.2.1.1 Hardware

Figura 15- Smartcard Hardware - Retirado de [1]



- CPU: microprocessador (para grande quantidade de dados e alta velocidade) ou microcontrolador de 8 bits (para operações mais simples como operação entre bits, com acesso isolado)
- Memória ROM: para sistema operacional (8 - 96 kB)

- ➔ Memória RAM: para dados de entrada e saida
- ➔ Memória EEPROM: para dados do usuário
- ➔ CRYPTO: possui um hardware de criptografia, o que confere mais segurança, pois os dados criptografados por ele só podem ser descriptografados pelo mesmo
- ➔ UART (Universal Asynchronous Receiver/Transmitter): módulo de hardware que traduz dados serial para paralelo e vice-versa.

3.2.1.2 Terminais

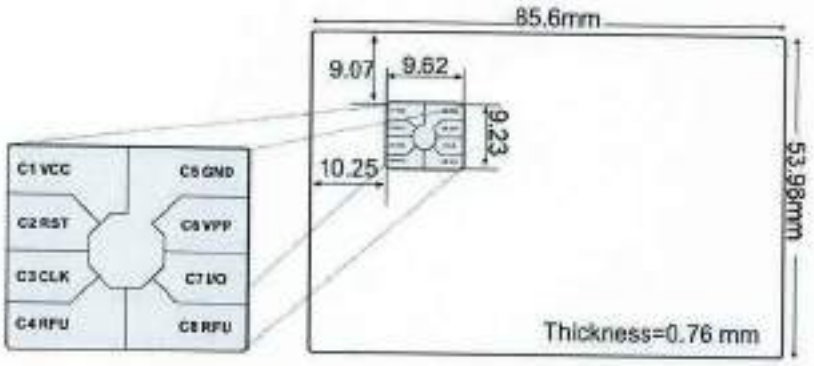
Figura 16- Terminais - Retirada de [1]

Tabela 2- Descrição dos terminais – Retirada de [1]

Position	Technical Abbreviation	Operation
C1	Vcc	Supply Voltage
C2	RST	Reset
C3	CLK	Clock Frequency
C4,C8	RFU	Reserved for future use
C5	GND	Ground
C6	Vpp	External Programming Voltage
C7	I/O	Serial Input/Output

3.2.1.3 Leiaute do cartão

Figura 17- Leiaute do cartão - Retirada de [1]



3.2.2 Leitor/Gravador (padrão PC/SC ISO7816)

Figura 18- Leitora/Gravadora padrão - Retirada de [17]

Possui um terminal USB 2.0 para conectar o leitor/gravador ao PC e assim poder inserir aplicativos e manipular os dados do cartão.

Importante ressaltar que a maior parte do processamento de dados e aplicativos é feita através do leitor e do computador da empresa, sendo que o cartão apenas armazena os dados biométricos e de aplicativos.

3.2.2.1 Software Development Kit (SDK)

Ambiente de desenvolvimento Integrado (IDE em inglês), que é um programa de computador que reúne características e ferramentas de apoio ao desenvolvimento de software com o objetivo de agilizar este processo. No nosso projeto utilizaremos um ambiente de desenvolvimento em Java disponibilizado gratuitamente pela Sun's chamado JCDK (Java Card Development Kit) e seu software de desenvolvimento chamado netBeans.

3.2.3 Leitor de impressões digitais

Figura 19- aparelho da DigitalPersona – Retirada de [14]

Devido aos fatores de descrição, aplicações, características, facilidade ao acesso do Software Development Kit (SDK), especificações e preço, o leitor biométrico escolhido para a realização deste trabalho foi o Digital Persona U.are.U 4000B Reader.

Abaixo serão citadas as aplicações, características e especificações do leitor.

Aplicações:

- Segurança do PC
- PCS Móveis
- Aplicações feitas sob encomenda
- Uso pessoal e Empresarial

Características:

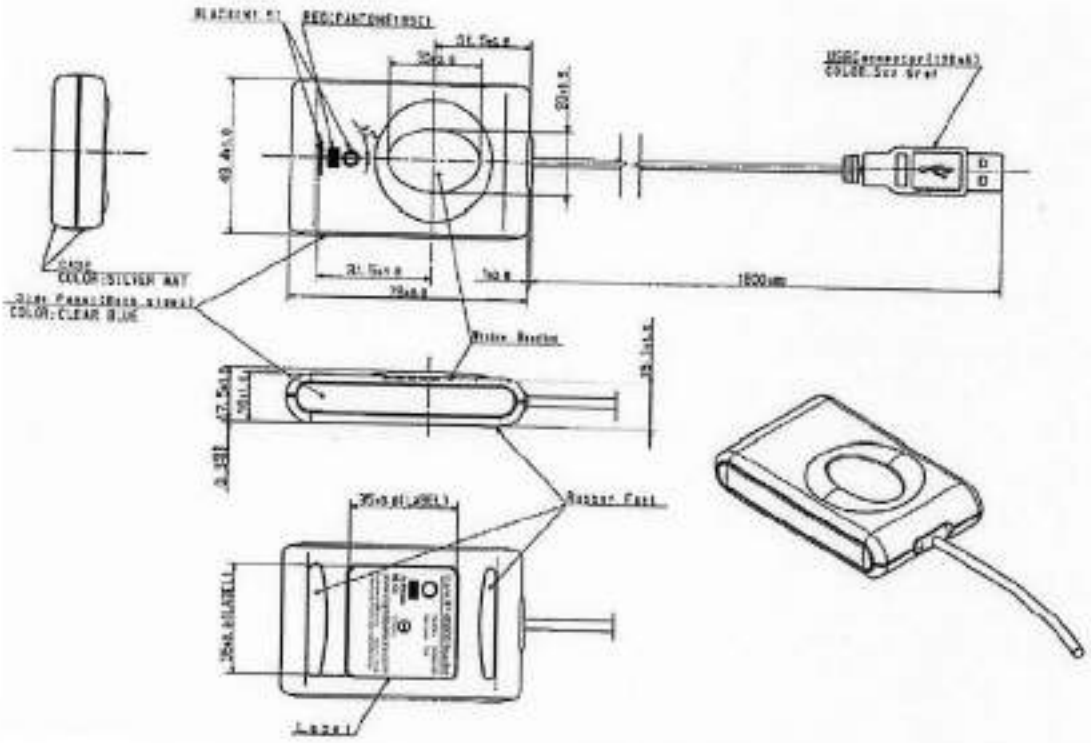
- Formato pequeno
- Funções internas de criptografia da imagem capturada - a comunicação de dados entre o sensor de leitura de impressão digital e o computador é criptografada
- Mecanismo de autocalibração: ajuste automático da luminosidade para otimizar a qualidade da imagem capturada
- Rejeição de impressões digitais latentes
- Sistema operacional: Windows 98, Me, NT 4.0, 2000, XP, Vista
- Vida útil superior a um milhão de toques

Especificações:

- Imagem de alta resolução: 512dpi
- Ampla área de captura: 14.6 x 18.1 mm - permite captura da imagem com maiores números de pontos de detalhe característicos (minúcias).
- Lente de leitura revestida com película de silicone: facilidade de leitura de dedos secos (com pouca oleosidade natural) ou dedos úmidos (suor)
- Compatível com todas as aplicações de U.are.U. SDK trial valido por 30 dias para testes de integração do leitor
- Comunicação padrão USB 2.0 full speed: rapidez de leitura da impressão digital e transferência para o computador
- Dimensões: 79 mm x 49 mm x 19 mm

Especificações Mecânicas:

Figura 20- Imagem das dimensões do leitor – Retirada de [14]



3.2.3.1 Software Development Kit (SDK)

SDK é a sigla de Kit de Desenvolvimento de Software ou Kit de Desenvolvimento de Aplicativos. Normalmente são disponibilizados por empresas para que os programadores se integrem melhor acerca do software proposto. O SDK inclui documentação, código e utilitários para o desenvolvimento de aplicações de acordo com o padrão do sistema em questão. Para o trabalho utilizaremos o Eclipse (IDE) que é de código aberto e oferece uma estrutura flexível pois utiliza linguagem JAVA, tornando mais fácil a criação, integração e utilização das ferramentas. Outro SDK utilizado é o do próprio leitor biométrico, o SDK da Digital Persona One Touch for Windows – Java Edition.

3.2.4 Eclipse

O Eclipse é um IDE (Integrated Developmente Environment) que foi desenvolvido pela empresa americana IBM (International Business Machines), em novembro de 2001. O Eclipse é bastante usado, por ser de fácil uso e também por

ser um software livre. Sua tecnologia baseada em plugins fornece um grande suporte aos programadores para a realização de diferentes projetos de diferentes formas.

O Eclipse é um IDE de código aberto utilizado para desenvolvimento da linguagem java, mas é também multilinguagem, ou seja, suporta outras linguagens de programação tais como C/C++ instalando-se os devidos plug-ins adicionais para cada linguagem. É um projeto livre de patentes por ser um software livre. E é portátil, ou seja, sua aplicação funciona em vários ambientes.

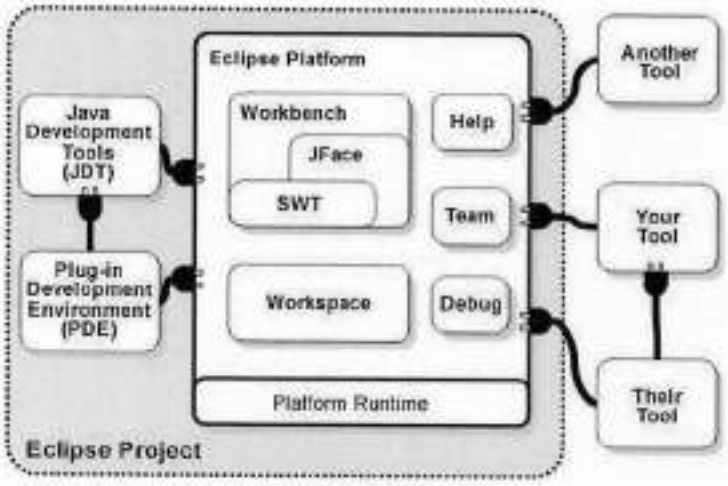
A plataforma do Eclipse fornece vários pacotes de desenvolvimento, tais como Eclipse JDT, que é a base para qualquer plug-in na linguagem java, o Eclipse SDK, que é o pacote de distribuição da IDE Java, o Eclipse WTP (Web Tools Platform), que é usado para desenvolvimento de linguagem para web, e o compilador do JDT, que é seu próprio compilador java, que é mais rápido e de código aberto.

O software permite também a refatoração do código, que é uma forma organizada de reestruturar o código para ser melhorado. Um aspecto importante de uma refatoração é que ela melhora o design sem mudar a semântica do design, não adicionando nem removendo sua funcionalidade.

Algumas desvantagens do Eclipse, é que ele não possui uma versão online, ou seja, totalmente web, não é um software tão leve, a instalação de plug-ins pode não ser uma opção muito eficiente e pode causar grandes dificuldades ao usuário no início da utilização do software, algumas implementações podem ser difíceis de ser feitas devida a má instalação dos plug-ins.

A arquitetura do Eclipse, que será mostrada na figura abaixo, é de pequena runtime, e consiste em Workbench (que é a interface do Eclipse), workspace (responsável por administrar os recursos do projeto), help (que é um sistema de documentação extensível) e team support (que facilita o uso do controle na versão).

Figura 21- diagrama de blocos do programa Eclipse – Retirada de [10]



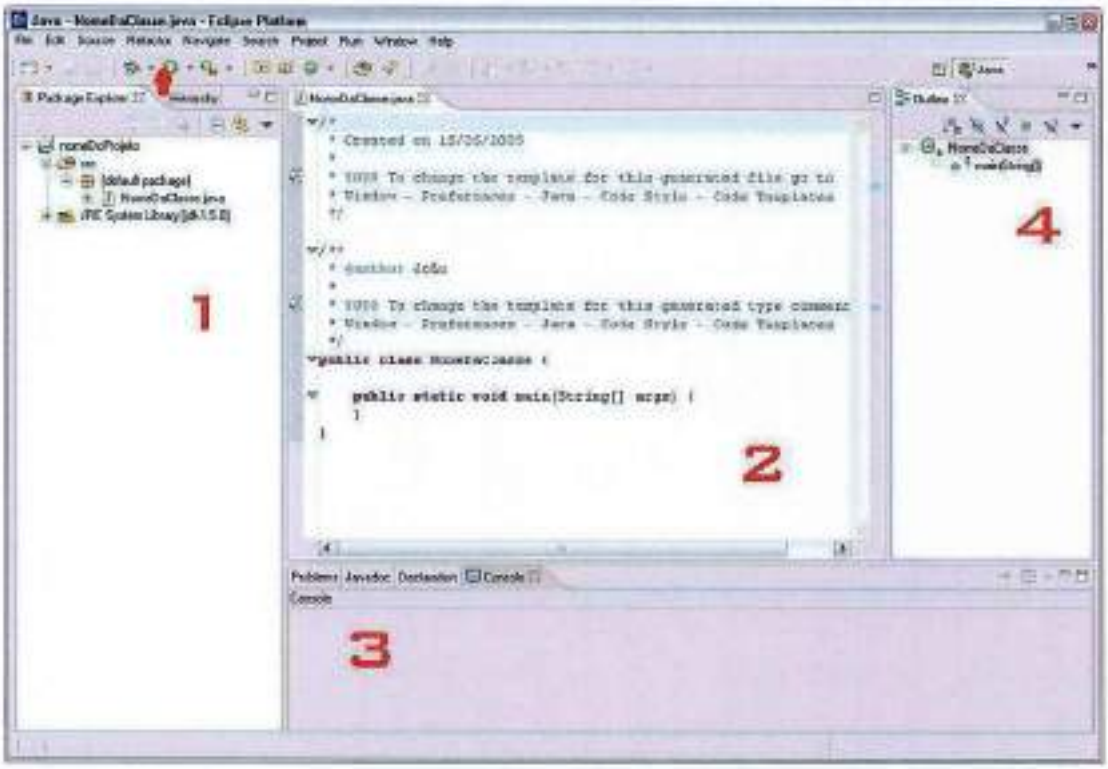
O Eclipse não é um software que se instala para executá-lo basta clicar no ícone com o nome de Eclipse após fazer o download e descompactar o arquivo. Após clicar neste ícone abrirá uma janela que pedirá para escolher o local no qual se deseja armazenar os arquivos do Eclipse, o chamado workspace.

Depois de instalado, os projetos são criados normalmente, com suas classes, sendo feitos os comentários para documentação e uma inovação do Eclipse que permite customizar o ambiente de trabalho.

O Eclipse encontra-se um terminal de console para a entrada de dados do teclado assim como no JDK e possui uma janela outline que funciona semelhante ao package explorer, mas voltada para a estrutura interna do arquivo. Podemos também adicionar ao projeto bibliotecas e importar e exportar arquivos do mesmo.

O programa é compilado normalmente e pode ser depurado para acompanhar todos os seus passos facilitando o encontro dos erros. O Eclipse é um IDE de fácil entendimento e manuseio, podendo ser usada por qualquer pessoa que deseja se dedicar à programação. Sua interface é mostrada abaixo, onde se destaca os principais componentes.

Figura 22- design do programa Eclipse – Retirada de [11]



1– Package Explorer: que permite visualizar toda a estrutura de arquivos e diretórios contidos no projeto;

2 – Editor de texto: onde se escreve o código com algumas palavras destacadas que são palavras-chave do pacote Java e possui auto completar para as palavras já conhecidas, indica com um "X" vermelho a linha que contém erros e coloca e desenho de uma lâmpada caso o desenvolvedor necessite de ajuda;

3 – Terminal de Console: para entrada de dados do teclado e saída de dados que o programa escreve e

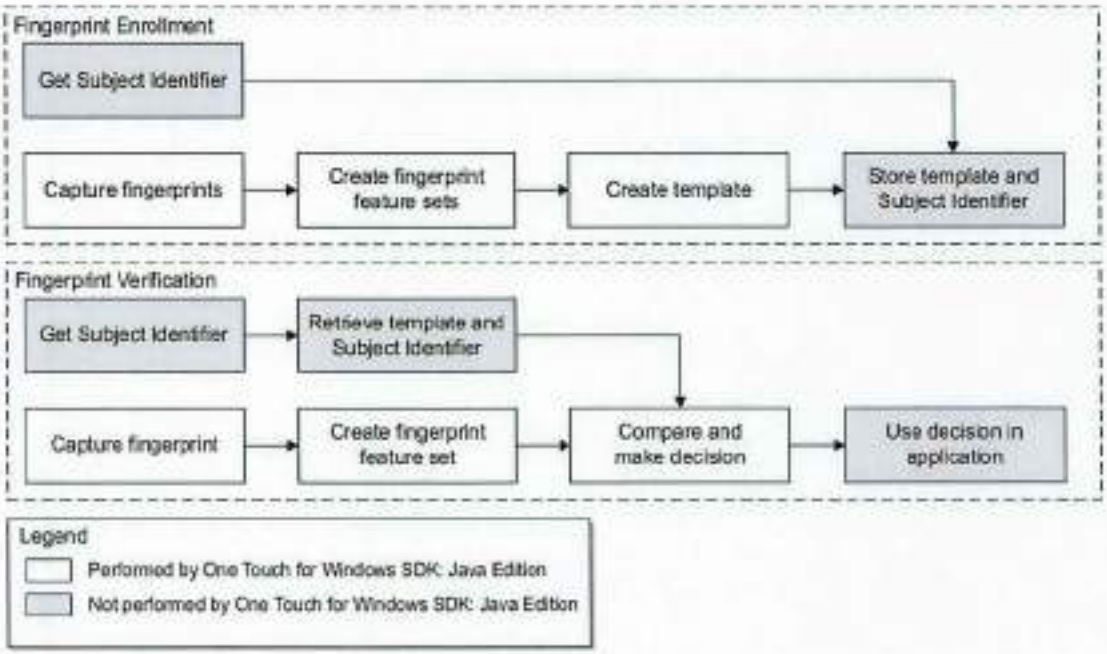
4 – Janela outline: visualizador das classes existente no projeto. A seta em vermelho indica onde se deve clicar para executar os códigos-fonte no Eclipse.

One Touch for Windows SDK: Java Edition

O One Touch for Windows SDK: Java Edition é uma ferramenta de desenvolvimento para integração biométrica da digital através de um conjunto de aplicações, serviços e produtos baseados em Java. Permite ao desenvolvedor executar operações básicas como captura de impressão digital a partir de um leitor

Digital Persona, extração das características da impressão (minúcias) e armazenamento dos dados num molde para comparação posterior.

Figura 23- diagrama de blocos da Captura e Verificação da impressão digital – Retirada de [9]



3.2.4.1 Arquivos e Pastas:

Tabela 3 - Arquivos e Pastas – Retirada de [9]

Pastas	Arquivos/Descrição
Docs	Guia para desenvolvedores
Samples	Subpasta para as amostras fornecidas como parte do SDK, contendo arquivos de códigos para executar os aplicativos
Ui Support	Contém o código fonde build.bat e run.bat para os aplicativos Samples que demonstram a funcionalidade gráfica e a interface do usuário
Enrollment	Contém os codigos fonte para realização do registro da impressão digital e sua verificação
Console	Código fonte para as funcionalidades adicionais tais como seleção de um leitor ou adicionar um usuário ao banco de dados.

3.3 Captura e registro da Impressão Digital

É o processo inicial de coleta de dados de impressões digitais de uma pessoa e armazenamento dos dados para posterior comparação. O procedimento seguinte descreve o processo de registro da impressão digital.

1 - Inscrever o usuário,

2 - Captura da impressão digital usando o leitor DigitalPersona,

3 - Extração dos parâmetros da amostra de impressão digital,

4 - Repetir o processo 2 e 3 até que se obtenha parâmetros suficientes para criar o modelo de impressão,

5 - Criação do modelo da impressão digital,

6 - Associar o modelo com um identificador do usuário: nome, e-mail, etc e

7 – Armazenar o modelo juntamente com o indentificador para posterior comparação.

Nas especificações do manual do Leitor podem-se encontrar as operações desta etapa de DPFPCapture:

- Tipo: interface

- Pacote: captura

- Descrição: Esta interface descreve a operação de captura de amostras a partir de um leitor de impressões digitais. Um leitor de impressões digitais pode ser monitorizada por várias operações de captura. Cada operação de captura tem uma prioridade especifica, que define como os eventos do leitor serão distribuídos entre várias operações simultâneas.

3.4 Verificação da Impressão Digital

É o processo de comparação entre a digital do modelo armazenado e a amostra fornecida no ato de utilização do leitor. O procedimento seguinte descreve o processo de verificação da impressão digital.

1 - Identificar a pessoa a ser verificada,

2 - Capturar uma amostra da impressão digital utilizando o leitor DigitalPersona,

3 - Recuperar o modelo do banco de dados associado ao identificador do usuário,

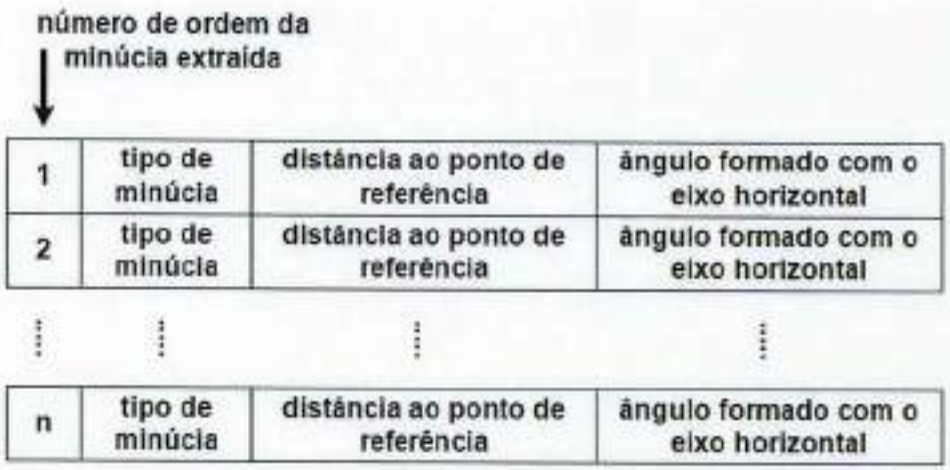
4 - Realizar a comparação um-para-um entre o conjunto de características do modelo com a amostra e decidir positiva ou negativa a comparação e

5 - Verificar a decisão para o acesso ou não do respectivo aplicativo ao qual o programa é associado.

A interface de verificação da impressão digital permite a comparação de um conjunto de características de impressões digitais (extraída de um leitor de captura de impressão digital), com um modelo de impressão digital (criado durante o registro da impressão digital). Ele usa um falso aceitar rate (FAR) como limite para decidir se quer ou não o conjunto de recursos e verificar o modelo que coincidir com o outro o suficiente para aceitar o conjunto de recursos.

As impressões digitais podem ser descritas por uma estrutura de dados que representa as minúcias extraídas. Esta consiste num vector de minúcias, em que cada entrada contém a informação relativa a uma minúcia, tipo e localização.

Figura 24- Informações relativas as minúcias – Retirada de [15]



O alinhamento das minúcias consiste em efetuar a correspondência entre as minúcias presentes na estrutura de dados de uma impressão digital e as presentes na estrutura de dados de outra. O alinhamento deve ser realizado minúcia a minúcia, garantindo que cada par de minúcias alinhadas são do mesmo tipo e se encontram à mesma distância do ponto de referência. Note-se que o ângulo depende da posição relativa do dedo no sensor durante a aquisição. O valor deste deve ser ajustado de modo a obter o máximo de minúcias emparelhadas.

4 DESENVOLVIMENTO DO TRABALHO

Como já dito anteriormente, os potencias problemas que podem surgir são referentes a compatibilidade de dados. A estratégia que será adotada para se combater este problema será a modificação direta dos códigos da biometria, da criptografia e/ou da gravação dos dados no cartão. As modificações serão referentes aos tipos de dados (variáveis char, string, float, etc.) utilizados nas saídas e entradas dos códigos.

4.1 ETAPA 1: DISCUSSÃO SOBRE OS MATERIAIS ESCOLHIDOS

4.1.1 Leitor Digital Persona U.areU 4000b

Um dos motivos da escolha do Leitor U.are.U 4000b pelo grupo deu-se por ele ser um sensor de impressão digital USB projetado para o uso com aplicações de U.are.U da DigitalPersona e Softwares desenvolvidos para sua aplicação.

Com funcionamento básico, rápido e simples, o usuário simplesmente coloca o dedo no sensor, e o dispositivo automaticamente captura a imagem da impressão digital. A eletrônica on-board calibra o dispositivo e cifra os dados da imagem antes de enviá-los pela porta USB.

Os produtos da DigitalPersona utilizam a tecnologia ótica da exploração da impressão digital para a qualidade da imagem e a confiabilidade superiores do produto. O software de reconhecimento do sensor U.are.U 4000 tem a capacidade de reconhecer até as impressões digitais mais difíceis.

4.1.2 Java card 2.2.1 128Kbytes

Abaixo segue um quadro com as especificações do cartão escolhido. Para a utilização ne gravação e leitura do arquivo contendo o template da digital e chave de criptografia e inclusão do aplicativo que será utilizado como exemplo, o cartão escolhido possui as configurações necessárias de EEPROM (18K) e plataforma Global 2.1.1. sendo o Java Card 2.2.1.

Figura 25 - Especificações Java Card 2.2.1 recebidas via e-mail do Fornecedor

Product Features		JCOP21
JavaCard 2.2.1		+
GlobalPlatform 2.1.1		+
Visa Card Implementation Requirements 2.1.1 (Visa Errata 2.1.1)		+
MasterCard Requirements		+
EMVCo EMV 4.1 Book1		+
Visa Contactless payment spec		
Common Criteria Security Certificates**		EAL 4+
VISA Type Approval		VISA
T=0 / T=1	ISO 7816-3	+
T=CL	ISO 14443-4	
USB 2.0 LS	ISO7816-12	
NFC Interface type	S2C	
Crypto support		
DES3, DES		+
RSA up-to 2432 bit		+
RSA Key Generation		+
ECC GF(2n)		+
ECC GF(p)		
SEED		+
SHA1		+
MD5		+
AES		+*
MIFARE 1KB / 4KB emulation (byte)**		
Available EEPROM options (byte)		18K / 36K / 72K
Applets for payment included in ROM**		Visa
Applets for eGovernment		
Custom Masking for Applets in ROM		+
Delivery Type Wafer**		+
Delivery Type Module**		PCM1.1
Supported Application Features		
Extended Access Control (EAC)		
Basic Access Control (BAC)		
Dynamic Data Authentication (DDA)		+
Static Data Authentication (SDA)		+
Multiple Security Domains		+
Data Authentication Pattern		+

4.1.3 Leitora de smartcard USB Gemalto

O grupo priorizou esse tipo de leitora / gravadora devido ao preço mais acessível e por possuir suporte para os cartões ISO 7816-3 que foi o Java card acima citado e comprado pelo grupo.

Pode-se citar também o fato da garantia de 100,00 ciclos de inserção e facilidade de compatibilidade com o sistema operacional usado nos computadores dos integrantes dos grupos conforme pode ser observado melhor nas especificações da Leitora abaixo.

Especificação Técnica Smart-Card interface:

- Possui suporte a todo os cartões ISO7816 Classe A, B e C (5V, 3V, 1.8V)

- Possui suporte a todos os parâmetros de cartões ISO7816 TA1 (até 344 Kbds)

- Lê e escreve em todos os cartões ISO 7816-1,2,3,4 com microprocessador, T=0 e T=1 (cartões de memória sob consulta)

- Possui detecção de curto Circuito

- Smart card connector é o de 8 contatos de fricção - ISO location,

- Garantia de 100,000 ciclos de inserção - EMV nível 1 mecanicamente compliant.

- Cartões que possuam embossamento em alto relavo são suportados.

Host InterfaceGemPC Twin [em modo USB]:

- USB full speed (12 Mbps)

- Hubless

- Cabo de 1,5M

- Conector USB tipo A.

- power supply através da porta USB.

- Voltagem de operação [4.4 --> 5.5V]

Standards/Certifications:

- ISO/IEC 7816-1,2,3,4: IC cartões com contato

- EMV level 1, EMV96 version 3.1.1

- Microsoft Windows Hardware Quality Labs (WHQL)

- Windows Logo Program WLP 2.0

- USB 2.0 full speed (GemPC Twin em modo USB)

- CCID - Chip card Interface device 1.0 (gemPC Twin em modo USB)

- Mondex® level 1 Purse Approved (version: Purse 2, ChipSafe and ChipSafe+)

Sistemas Operacionais Suportados:

- Windows Vista 32bits e 64bits - somente entradas USB

- Windows 7 - somente entradas USB

4.2 Etapa 2: Obtenção dos Parâmetros da Digital e Criptografia dos dados com algoritmo assimétrico

O sistema de software do projeto consiste, no momento, de quatro programas escritos na linguagem Java com a adição do SDK (Software Development Kit) Digital Persona, que implementa métodos relacionados a leitora de impressões digitais. Os

programas são GeraChavesRSA, GeraChaveSessao, Host e Certificadora. A seguir é feita uma explicação do que cada um destes programas faz.

4.2.1 GeraChavesRSA

Este programa gera as chaves pública e privada através do algoritmo RSA e grava estas chaves nos arquivos "public.key" e "private.key" respectivamente. Foi feito um programa em separado para esta operação para que a geração das chaves se tornasse isolada dos demais programas, evitando-se assim, que haja incompatibilidades.

4.2.2 GeraChaveSessao

Este programa gera uma chave simétrica através do algoritmo AES para funcionar como chave de sessão na comunicação (via uma rede teoricamente insegura) entre o Host e a Certificadora. Esta chave é exportada para o arquivo "session.key" para que esta chave seja a mesma nas máquinas do Host e da Certificadora.

4.2.3 Host

Este é programa que se encontrará nos Hosts do sistema. Ele oferece ao usuário duas opções, a de se cadastrar no sistema ou a de autenticar (comparar) a impressão digital cadastrada, fazendo com que haja duas execuções diferentes para o programa.

Mesmo com estas duas possíveis execuções, tomando-se como base a teoria de orientação a objetos na linguagem Java, foi possível desenvolver este código utilizando-se as mesmas classes, que são as seguintes:

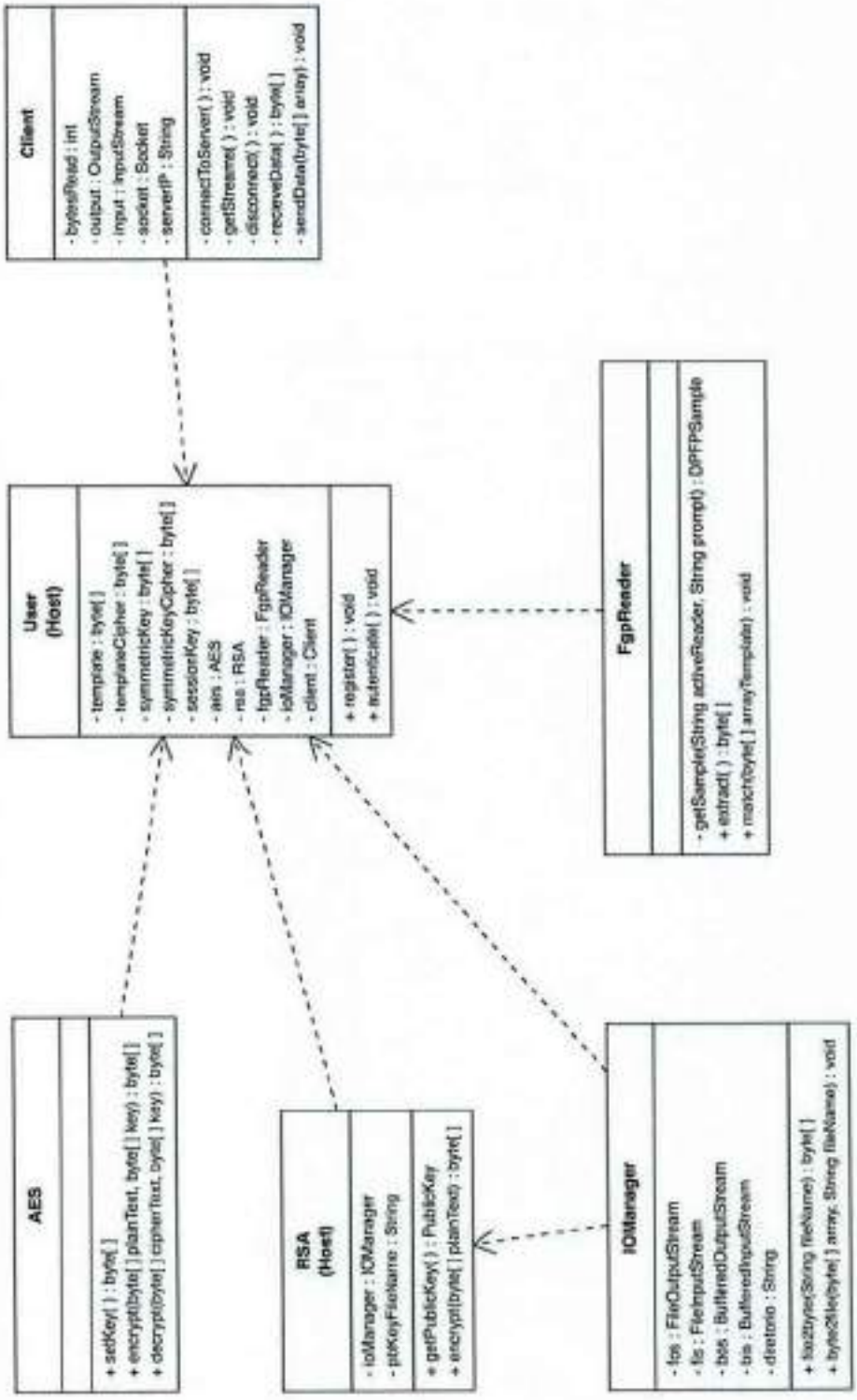
- FgpReader: classe na qual as operações disponibilizadas pelo SDK Digital Persona são utilizadas. Esta classe permite a extração e a comparação de impressões digitais
- AES: classe que implementa o algoritmo AES. Permite tanto a encriptação quanto a decriptação de dados, sempre recebendo a chave simétrica como parâmetro.
- RSA: classe que implementa o algoritmo RSA. Permite somente a encriptação de dados (pois a decriptação assimétrica só pode ser realizada na Certificadora). Esta classe carrega o arquivo "public.key" e criptografa a

chave simétrica gerada pela classe AES e exporta o resultado desta operação para o arquivo "symmetricKey.cipher", que será gravado no cartão.

- IOManager: gerencia as operações de entrada e saída de arquivos do programa, permitindo que arquivos sejam transformados em vetores (arrays) de bytes ou vice-versa.
- Client: classe que gerencia a comunicação com a Certificadora, enviando e recebendo dados. Funciona em conjunto com a classe Server da Certificadora.
- User: esta classe organiza as operações das outras classes de modo a permitir que as operações de cadastro e autenticação sejam isoladas.

A seguir é apresentado um diagrama de classes (UML) que representa o código como um todo.

Figura 26- Diagrama Geral dos códigos – Fonte Própria



O funcionamento do cadastro (user.register()) e da autenticação (user.autenticate()) estão sumarizados nos seguintes diagramas:

Figura 27- Diagrama funcionamento user.register() – Fonte Própria

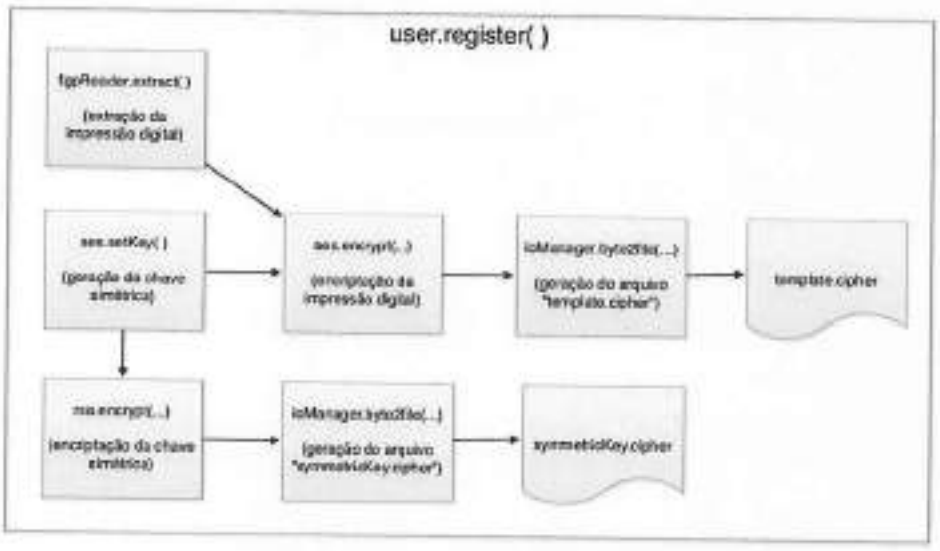
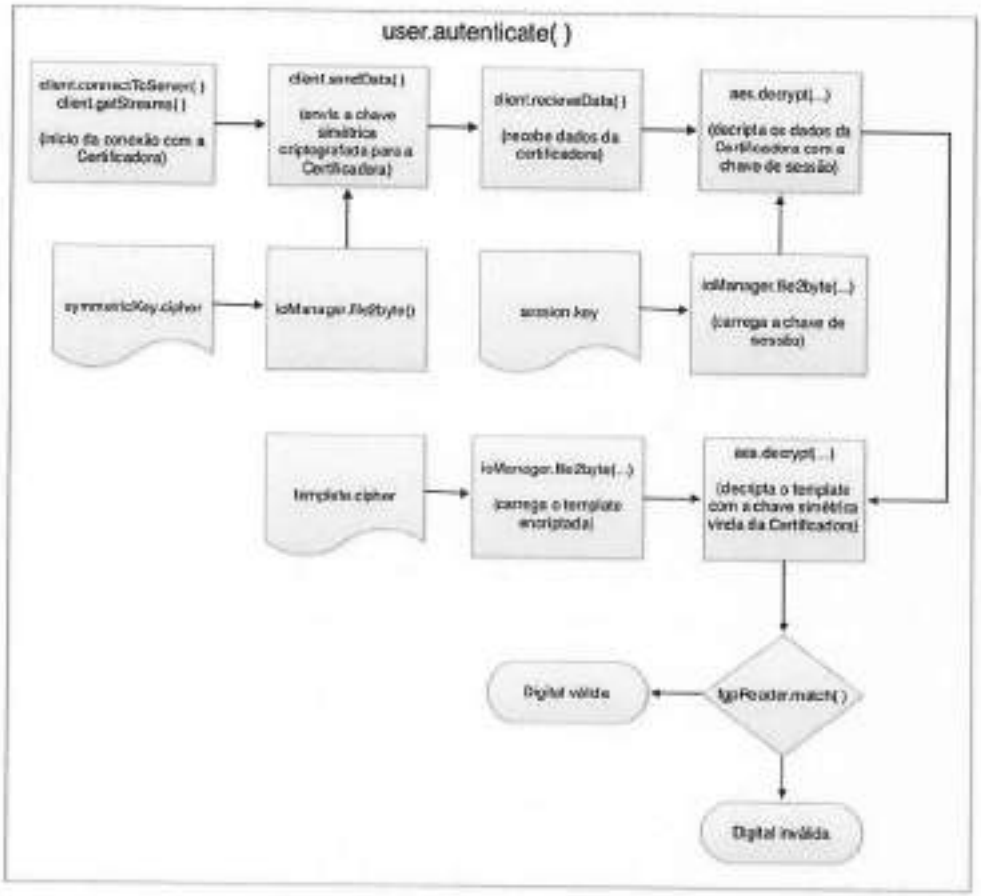


Figura 28 - Diagrama de funcionamento user.autenticate() – Fonte Própria



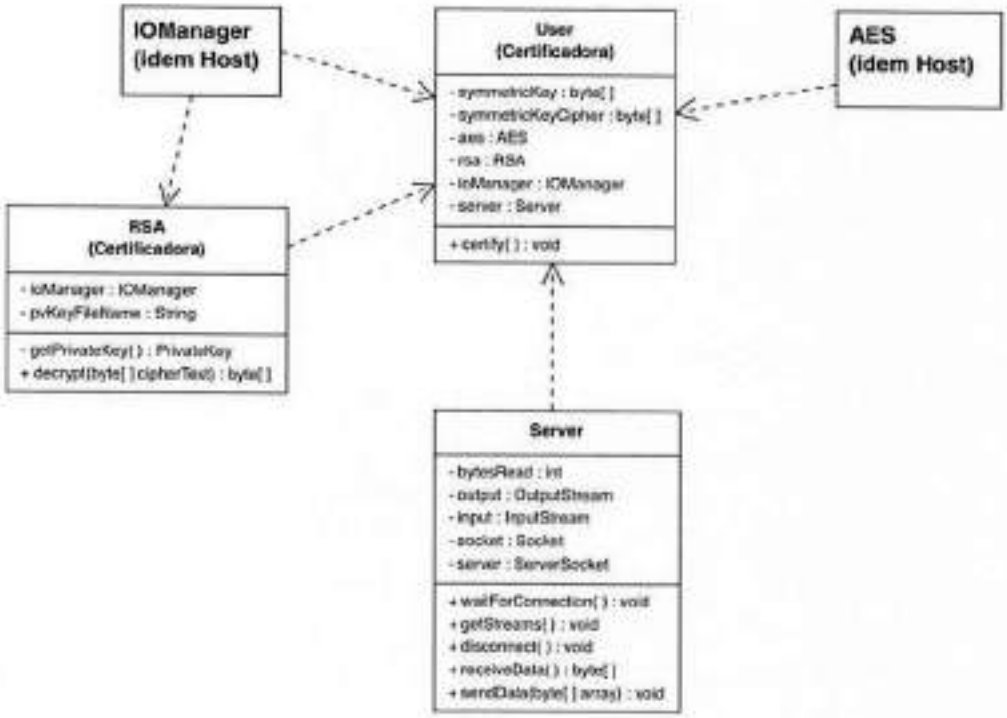
4.2.4 Certificadora

Este é programa que se encontra na Certificadora. Para sua construção também foram adotados princípios de orientação a objetos, tendo sido programadas as seguintes classes:

- AES: funcionamento idêntico ao AES do Host
- RSA: esta classe implementa a parte de decriptação do algoritmo RSA, para isso o arquivo "private.key" é carregado
- IOManager: funcionamento idêntico ao IOManager do Host
- Server: classe que gerencia a comunicação com o Host, recebendo e enviando dados. Funciona em conjunto com a classe Client do Host
- User: esta classe organiza as operações das outras classes de modo a permitir a certificação do usuário

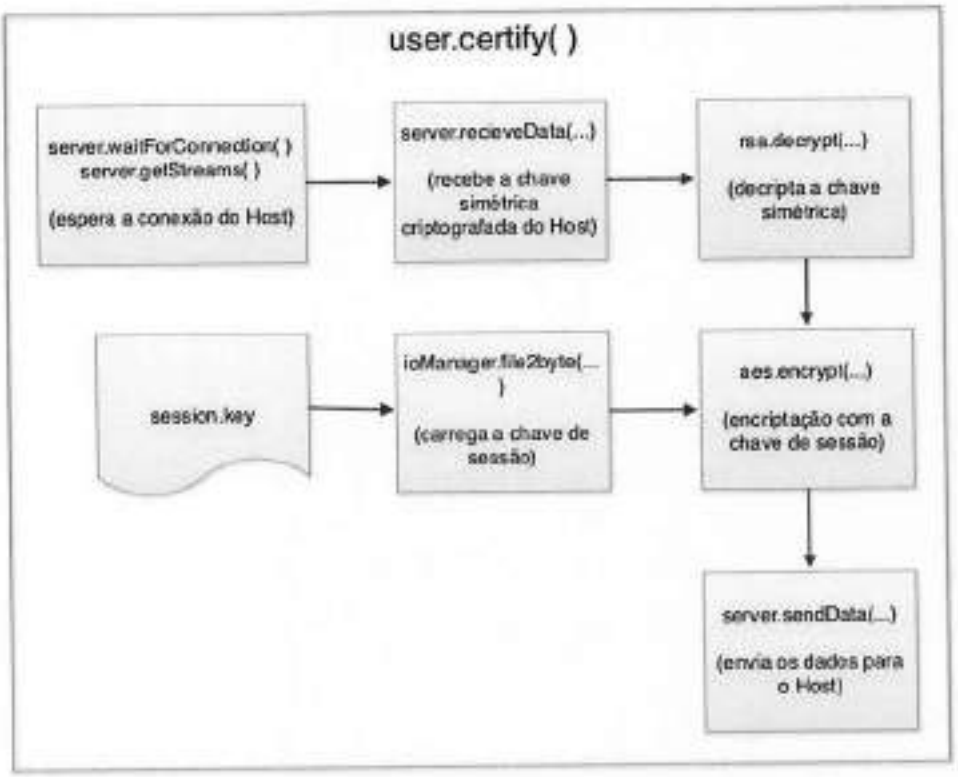
O diagrama de classes a seguir representa o programa da Certificadora:

Figura 29 - Diagrama do programa da Certificadora – Fonte Própria



O funcionamento da certificação (user.certify()) está sumarizado no seguinte diagrama:

Figura 30 - Diagrama de funcionamento do user.certify() – Fonte Própria



4.3 ETAPA 3: GRAVAÇÃO E ACESSO DE APLICATIVOS E DADOS EM JAVA CARD

Antes de detalhar de como será gravado e acessado um aplicativo no Java Card, alguns conceitos deverão ser mostrados.

4.3.1 APDU

Protocolo de comunicação host/cartão.

4.3.1.1 Command APDU – envio de dados para o cartão

Figura 31- Estrutura do pacote de dados Command APDU - Retirado de [17]

Mandatory header				Optional body		
CLA	INS	P1	P2	Lc	Data field	Le

- Mandatory Header - dados que devem obrigatóriamente estar presentes no pacote.
- "Optional Body - dados opcionais no pacote.
- **CLA** - Tamanho 1 byte - Class Byte, identifica a categoria do APDU de comando e resposta.
- **INS** - Tamanho 1 byte - Instruction Byte, especifica a Instrução do Comando.
- **P1** - Tamanho 1 byte - Parameter Byte, paramentro do comando.
- **P2** - Tamanho 1 byte - Parameter Byte, paramentro do comando.
- **Lc** - Tamanho 1 byte - Length Command, Quantidade em bytes de dados contidos no "Data Field".
- **Data Field** - Tamanho de 1 à 256 bytes - Data Field, dados a ser enviado para o Cartão.
- **Le** - Tamanho 1 byte - Length Expected, Quantidade de dados esperado em resposta do Cartão.

4.3.1.2 Response APDU

Figura 32- Pacote de dados Response APDU - Retirado de [17]

Optional body	**Mandatory Trailer**	
Data field	SW1	SW2

- Mandatory Trailer - dados que devem obrigatoriamente estar presentes no pacote.
- Optional Body - dados opcionais no pacote.
- **Data Field** - Tamanho de 1 a 256 bytes - Data Field, dados a ser enviado para o Host, geralmente possui a quantidade especificada no campo **"Le"** que o Host previamente envia no Command APDU porém isso não é regra.
- **SW1** - Tamanho 1 byte - Status Word 1 - Status do Processamento 1.
- **SW2** - Tamanho 1 byte - Status Word 2 - Status do Processamento 2.

Juntas SW1 e SW2 formam uma Word (Palavra), geralmente conhecida pelos programadores Assembler, uma Word tem o tamanho de 2 bytes ou short.

Um exemplo de Status Word é 0x9000 que segundo a ISO 7816 significa **OK.**

Na referência, item 12.32, há uma lista das Status Words e as suas mensagens correspondentes. Para que seja possível o envio de arquivos maiores que 128 bytes (Restrição do cartão Java Card 2.2.1), é necessário utilizar o conceito de chained APDU, que consiste em enviar APDUs sequencialmente até que todos os dados do arquivo sejam enviados/transmitidos.

Para versões de cartão acima, já é possível enviar APDUs com dados acima de 128 bytes, pois o buffer criado pela biblioteca do JCDK de versões acima suporta uma quantidade de dados maiores.

4.3.2 AID – Application Identifier

Basicamente ele é utilizado para se selecionar a aplicação dentro do cartão analogamente podemos comparar o AID com o nome de um Arquivo no HD do seu PC. Ele segue os Padrões **ISO 7816** e tem o seguinte formato **[RID + PIX]**.

- → **RID (Registered Provider Identifier)** consiste de obrigatoriamente 5 bytes e identifica o fabricante/desenvolvedor da aplicação e é publicada por uma autoridade **ISO 7816-5**.
- → **PIX (Proprietary Application Identifier Extension)** consiste no identificador de Aplicativo e é definido pelo fabricante/desenvolvedor do aplicativo.

4.3.3 Global Plataform

Organização sem fins lucrativos, que procura desenvolver especificações que visam melhorar a segurança e facilitar a interoperabilidade em tecnologia de smartcards.

4.3.4 Card Manager

O Card Manager é Motor do GP (Global Platform) e todos os cartões que implementam a especificação GP tem um, o card manager implementa o Security Domain, que tem a função básica de gerenciar a autenticação entre Host e Cartão através de uma conexão segura ou mutual authentication, o ciclo de vida do cartão e aplicativos, carga/instalação/exclusão de aplicativos (Applets no caso do JC), gerenciamento de chaves do algoritmo criptográfico usado para operações em ambiente seguro (DES/TripleDES/AES).

O Card Manager pode ser acessado utilizando-se o comando select (ISO 7816-4) mais o AID (Application Identifier), gravem bem esse conceito pois será muito utilizado lá na frente.

4.3.5 ATR – Answer to Reset

Após alimentar ou resetar o cartão, este envia ao host o ATR, que é uma mensagem de 33 bytes contendo informação de protocolo de comunicação, velocidade de comunicação e identifica tipo de cartão e fabricante.

4.3.6 Instalando um aplicativo no Java card

Nesta etapa do projeto utilizaremos o software ECLIPSE versão 3.1.2 e um plug-in do ECLIPSE chamado JCOP Tools, onde este plug-in facilita a comunicação em pacotes APDU, inicializando o cartão e canais de comunicação em poucos comandos e criando command APDUs automaticamente para algumas funções como instalar e deletar aplicativos. Além disto, é possível realizar simulações sem que seja necessário o cartão fisicamente.

A instalação do plug-in é facilmente encontrada na internet e não será colocada neste relatório. Para auxiliar o desenvolvedor na utilização do plug-in é possível encontrar após a instalação no Help Contents do ECLIPSE um guia chamado "JCOP Tools User Guide".

Para ilustrar a facilidade de uso, abaixo será inserido o conteúdo que aparece no console do JCOP Tools (JCShell) para a instalação de um aplicativo no Java Card. Os comandos estão em preto e negrito e a resposta a eles em azul.

Em resumo a instalação segue as seguintes etapas:

- Abertura do canal de comunicação
- Identificação do cartão inserido no leitor (Requisição do ATR)
- Atribui chaves de autenticação do JCShell
- Inicia autenticação
- Completa autenticação
- Deleta versão antiga do aplicativo (Se existir)
- Faz upload do arquivo.cap do aplicativo a ser instalado
- Instala o aplicativo ao cartão
- Ao final mostra as informações do cartão, listando os AID dos aplicativos já instalados no cartão

No APÊNDICE B encontram-se os comandos a serem digitados para a instalação do aplicativo. Para mais informações consultar o CD no anexo.

4.3.7 Acessando o aplicativo utilizando JAVA SE

Para que o Host possa acessar os aplicativos e dados do cartão, devemos criar programas utilizando o Java SE, onde estes são programados para selecionar o aplicativo correto e realizar a troca de informações entre o host e o leitora do cartão através de APDUs.

Para acessar o cartão, serão utlizado dois pacotes de classes que vêm juntas com o kit de desenvolvimento do Java Card.

- Pacote **Java.util.list:** pacote com classes que permite criar variável tipo lista. Possui funções para manipular estas listas.
- Pacote **Javax.smartcardio:** pacote que possui classes para manipulação de dados de I/O do smartcard. Possui as classes:

Tabela 4 - Índice de Classes – Retirada de [17]

Índice de Classes	
ATR	A Smart Card's answer-to-reset bytes.
Card	A Smart Card with which a connection has been established.
CardChannel	A logical channel connection to a Smart Card.
CardPermission	A permission for Smart Card operations.
CardTerminal	A Smart Card terminal, sometimes refered to as a Smart Card Reader.
CardTerminals	The set of terminals supported by a TerminalFactory.
CommandAPDU	A command APDU following the structure defined in ISO/IEC 7816-4.
ResponseAPDU	A response APDU as defined in ISO/IEC 7816-4.
TerminalFactory	A factory for CardTerminal objects.
TerminalFactorySpi	The TerminalFactorySpi class defines the service provider interface.

Para a confecção do código a ser compilado e executado para acessar o cartão, seguimos as seguintes etapas:

- ➔ Criamos uma lista de leitores, que armazena um ou mais terminais de leitores ligados ao host
- ➔ Criamos uma variável terminal, que será definido como qual leitor da lista acima estaremos utilizando.
- ➔ Instanciamos um objeto Card, que será o cartão inserido neste leitor definido por terminal.
- ➔ Instanciamos um CardATR, que representa o ATR retornado do cartão.
- ➔ Instanciamos um objeto cardChannel, que representa o canal de comunicação host/cartão.
- ➔ Instanciamos os pacotes de mensagens APDU, command APDU e responde APDU

Após as instanciações, utilizando objetos dos pacotes de classes acima:

- ➔ Escolhemos o leitor a ser utilizado
- ➔ Estabelecemos conexão do leitor com o cartão
- ➔ Requisitamos o ATR após a conexão
- ➔ Abrimos um canal de comunicação entre host/cartão
- ➔ Cria-se APDU a ser enviado para o cartão (commandAPDU)
- ➔ Transmite-se o commandAPDU e espera-se o Response APDU do cartão, contendo os dados requisitados e as Status Words (SW1 e SW2) que identificam se ocorreu problema ou se ocorreu tudo conforme esperado.

No Apendice B, há exemplo de códigos utilizados pelo host para gravação e leitura de dados no cartão.

É importante ressaltar que os aplicativos instalados são separados por firewall, para evitar que uma aplicação hostil prejudique o sistema, porém estes podem se comunicar entre si através do RMI (remote method invocation).

Podemos instalar aplicativos e dados da digital e estes podem se comunicar através do RMI.

4.3.8 Compatibilizações

As compatibilizações que terão de ser realizadas deverão levar em consideração as seguintes restrições e normas:

4.3.8.1 Caracteristicas Java Suportadas (Java Card 2.2.1)

- → Tipos Primitivos: boolean, byte e short.
- → Arrays de Uma Dimensão.
- → Java packages, classes, interfaces e exceptions.
- → Orientação a Objetos, heranças, métodos virtuais, sobrecarga e escopo de acessos.
- → O Tipo int de 32 bit integer é opcional.

4.3.8.2 Caracteristicas Java Não Suportadas (Java Card 2.2.1)

- → Tipos Primitivos: long, double e float.
- → char e String.
- → Arrays Multi-Dimensionais.
- → Carga de Classe Dinamica.
- → Gerenciador de Segurança.
- → Serialização de Objetos.
- → Clonagem de Objetos.

4.3.8.3 ISO 7616 – Regras internacionais que definem as estruturas dos smartcards

- → ISO 7816-1: Caracteristicas Fisicas.
- → ISO 7816-2: Cartões com Contato - Dimensões e Locais dos Contatos.
- → ISO 7816-3: Cartões com Contato - Interface Elétrica e Protocolos de Transmissão.
- → ISO 7816-4: Organização, Segurança e Comandos para Intercâmbio.
- → ISO 7816-5: Registro dos Provedores de Aplicações.
- → ISO 7816-6: Elementos de Dados para Intercâmbio Interindustriais.
- → ISO 7816-7: Comandos Interindustriais SCQL (Structured Card Query Language).
- → ISO 7816-8: Comandos para Operações de Segurança.
- → ISO 7816-9: Comandos para o Gerenciamento do Cartão.

Na referência 12.30, temos uma lista de cabeçalhos de command APDU que seguem as regras da ISO7816-4

4.4 Etapa 4: Gravação de dados na memória e criação de aplicativos

Antes de falarmos dos aplicativos, será introduzido um conceito sobre o tipo de objeto criado nos aplicativos e como este é gravado na memória.

Anteriormente foi citado que o Java Card possui três tipos de memória:

- ROM: Para sistema operacional, bibliotecas java pré-inseridas, JCDK pré-instalado.
- EEPROM: Para dados a serem gravados
- RAM: Para dados de entrada e saida

A memória ROM é inalterável e já vem pré-programada de fábrica. Para gravar dados no cartão que são permanentes utilizamos a memória EEPROM. Para gravar dados na EEPROM , no construtor do aplicativo a ser gravado devemos criar um objeto permanente no formato byte array.

Para gravar dados no cartão que são transitórios, que só serão utilizados quando o cartão está alimentado, utilizamos a memória RAM.Para gravar dados na RAM, no construtor do aplicativo criamos um objeto transiente, utilizando a função JCSystem.makeTransientByteArray

Na figura abaixo, há um exemplo de como instanciar um objeto permanente (EEPROM) e um objeto transitório (RAM).

Figura 33- Exemplo de como instanciar um objeto permanente e um objeto transitório – Retirada de [17]

```
private MemoryTest() {

 //Persistent, Objeto e Dados gravados na EEPROM
 this.eeprom_ba = new byte[ARRAY_SIZE];

 //Transient, Objeto na EEPROM e Dados na RAM
 this.ram_ba = JCSystem.makeTransientByteArray(
                ARRAY_SIZE, JCSystem.CLEAR_ON_DESELECT);
}
```

Visto como são criados arquivos nas memórias, falaremos sobre os aplicativos a serem criados.

Neste projeto há dois tipos de aplicativos a serem instalados no JBCard:

- Gravação/leitura dos dados no cartão
- Aplicativos específicos de cada host

Para o primeiro caso, cada arquivo a ser inserido no cartão deve possuir um aplicativo instalado para criar o objeto que receberá os dados a serem gravados na memória do cartão e realizar o envio/recepção destes dados em APDU.

Para este caso há dois aplicativos a serem feitos:

- ➔ Leitura/Gravação dos dados da digital criptografada
- ➔ Leitura/Gravação dos dados da chave simétrica

Estes possuem uma AID pré-definida, uma rotina para gravação dos dados da digital quando o cartão recebe os dados e uma rotina de envio dos dados da digital criptografada guardadas no cartão.O que define se o cartão receberá ou enviará dados é o campo INS do cabeçalho do pacote de dados APDU, caso INS seja 0xDA os dados enviados ao cartão serão gravados na memória do cartão e se INS for 0xCA os dados serão enviados ao host.

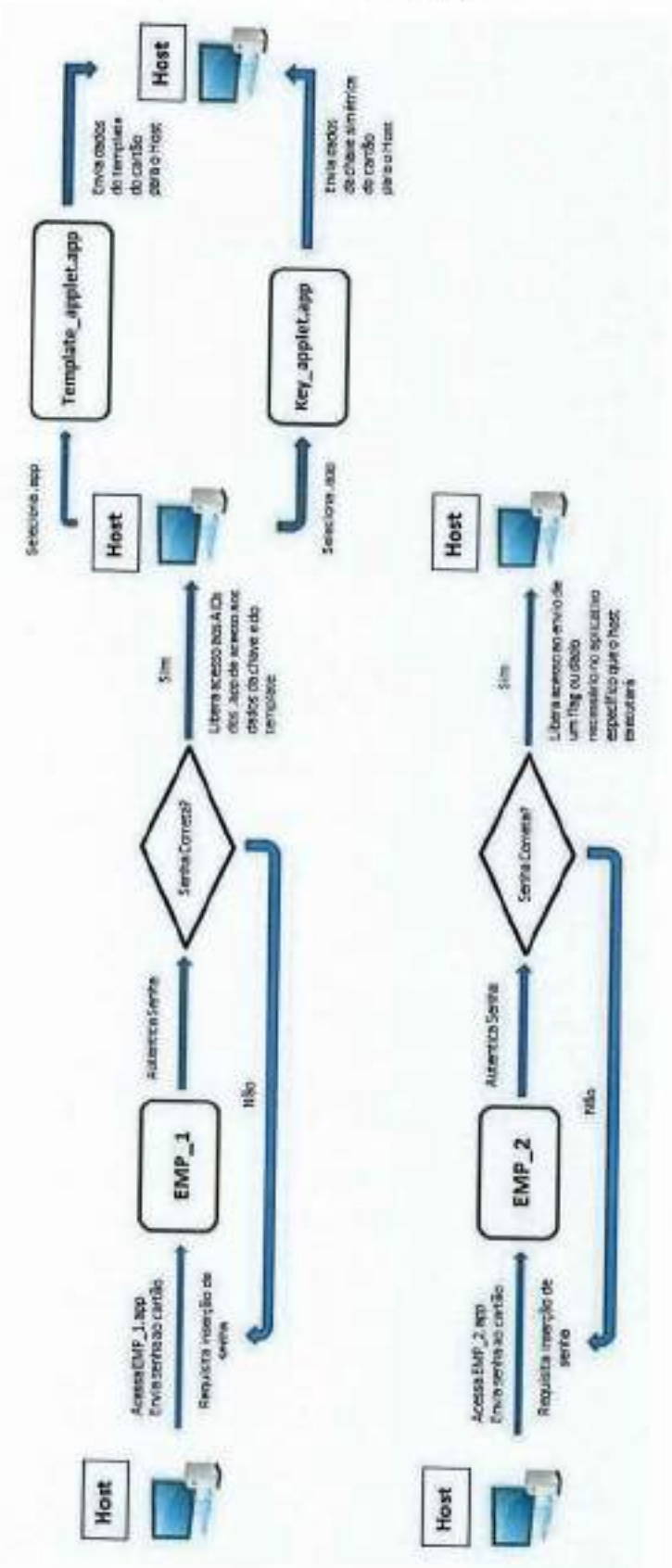
Visto que os arquivos a serem gravados são maiores que 128 bytes, utilizaremos o conceito de chained APDU apresentado no item 1.1.1.1 APDU.

No APÊNDICE B há os códigos dos aplicativos para a leitura/gravação de dados ao cartão.

Para o segundo caso criamos aplicativos especificos de cada host, onde este autentica e verifica uma senha especifica deste aplicativo e caso a senha esteja correta, é liberado os AIDs dos aplicativos da chave simétrica e da digital. Para hosts que não utilizam a biometria, este aplicativo apenas autentica e verfica a senha e envia um flag ou dado necessário para a aplicação off-card do host.

Para um melhor entendimento da dinâmica da comunicação do host com os aplcativos inseridos no cartão, abaixo há um fluxograma do processo.

Figura 34- Fluxograma do processo de gravação e acesso aos aplicativos - Fonte Própria



Para a verificação do funcionamento correto destes aplicativos utilizamos o JCardSim – Java Card Runtime Enviroment Simulator, que consiste em uma biblioteca java que possui pacotes e classes que permitem a simulação dos aplicativos a serem inseridos no cartão e também utilizamos o plug-in JCOPTools que também permite simular tais aplcativos.

Como já dito anteriormente, os potencias problemas que podem surgir são referentes à compatibilidade de dados. A estratégia que será adotada para se combater este problema será a modificação direta dos códigos da biometria, da criptografia e/ou da gravação dos dados no cartão. As modificações serão referentes aos tipos de dados (variáveis char, string, float, etc.) utilizados nas saídas e entradas dos códigos.

		Nome	Duração	Início	Fim	Predecessores	Recursos
1		Adquirir materiais especificados no Relatório de TF I	11d?	01/07/2013	15/07/2013		Alexandre,Isabelle,José
2		Aprimorar códigos de software para obtenção da Digital	11d?	01/07/2013	15/07/2013	1	Isabelle
3		Aprimorar códigos para criptografar a Digital	11d?	01/07/2013	15/07/2013	1,2	José
4		Aprimorar cógidos para leitura e gravação no cartão JavaCard	11d?	01/07/2013	15/07/2013	1,2,3	Alexandre
5		Testes com diferentes digitais e código parcial	7d?	16/07/2013	24/07/2013	2	Isabelle
6		Testes de codificação e criptografia dos dados	7d?	16/07/2013	24/07/2013	2,3	Alexandre,José
7		Testes com gravação dos dados no cartão	7d?	16/07/2013	24/07/2013	2,3,4	Alexandre
8		Finalização dos códigos de Biometria Digital	7d?	25/07/2013	02/08/2013	2,5	Isabelle
9		Finalização da leitura e gravação da digital já criptografada no cart	7d?	25/07/2013	02/08/2013	3,6	Alexandre,Isabelle,José
10		Desenvolvimento de um possível aplicativo	7d?	02/08/2013	12/08/2013		Alexandre,Isabelle,José
11		Aprimorar código do aplicativo	10d?	13/08/2013	26/08/2013	10	Alexandre,Isabelle,José
12		Testes com o aplicativo pensado e desenvolvido	10d?	28/10/2013	11/11/2013	10,11	Alexandre,Isabelle,José
13		Desenvolvimento da apresentação Oral Final	32d?	11/10/2013	25/11/2013	1,2,3,4,5,6,7,8,9,	Alexandre,Isabelle,José
14		Desenvolvimento do Poster Final	5d?	20/11/2013	26/11/2013	13,15	Alexandre,Isabelle,José
15		Desenvolvimento do Relatório Final	46d?	18/09/2013	20/11/2013	1,2,3,4,5,6,7,8,9,	Alexandre,Isabelle,José
16		Última Apresentação Oral a Banca Examinadora	1d?	27/11/2013	27/11/2013	13,17,18	Alexandre,Isabelle,José
17		Entrega do Trabalho Impresso	1d?	22/11/2013	22/11/2013	13,14,15	Alexandre,Isabelle,José
18		Apresentação Pública	1d?	02/12/2013	02/12/2013	17	Alexandre,Isabelle,José
19		Entrega do Trabalho Encadernado	1d?	06/12/2013	06/12/2013	13,14,15,16,17,1	

6 Orçamento Final

Abaixo é apresentada uma lista com o valor final dos principais itens adquiridos para o JBCARD, tanto para a etapa 1 quanto para as demais etapas 2 e 3. Com base nessa tabela de gastos apresentada, verifica-se que o custo de componentes do projeto foi de R$396,01. Esse orçamento está baseado em quantidades suficientes para a utilização de 2 cartões, ou seja, para 2 usuários do produto. É importante ressaltar que esses valores se referem ao custo de componentes comprados em quantidades pequenas para realização dos protótipos do projeto. O custo de produção do JBCARD certamente seria inferior se os componentes fossem comprados em grande volume.

Além dos custos dos componentes, deve-se considerar o custo referente ao tempo de trabalho no projeto. Considerando o salário mínimo vigente de R$755,00 para o Estado de São Paulo, e o piso salarial de 8,5 salários mínimos (para 8horas/dia de trabalho), chega-se a um custo de aproximadamente R$ 17.000,00 considerando os 3 integrantes do projeto. Sendo assim, estima-se que o custo total do projeto girará em torno de R$17.396,01. Considerando uma produção em larga escala, sabe-se que esse custo seria reduzido drasticamente, o que atinge um dos objetivos iniciais do projeto que é o desenvolvimento de um projeto de baixo custo de produção.

Tabela 5 - Orçamento Final – Fonte Própria

Descrição do Material	Quantidade	Unidade	Total
Smart Card Criptográfico Jcop21 Java card 18 K	2	R$ 42,48	R$ 84,96
Leitor Smartcard USB Gemalto	1	R$ 47,39	R$ 47,39
Leitor de impressão Digital Persona U 4000B	1	R$ 263,66	R$ 263,66
IDE Netbeans / Eclipse	1	gratuito	gratuito
SDK Digital Persona	1	gratuito	gratuito
Notebook para configurar leitora/gravadora	1	grupo	grupo
TOTAL	-	-	**R$396,01**

7 GERENCIAMENTO

- **Professor Wang Jiang Chau:** orientador do trabalho de conclusão de curso.
- **Alexandre Nagano:** entendimento e desenvolvimento de programação em Java e Java Card, pesquisa e desenvolvimento do software e hardware do smartcard, background e concepção e alternativas para o projeto. Leitura e Gravação do arquivo no cartão.
- **Isabelle Miranda(gerente):** pesquisa sobre as técnicas de biometria existentes, implementação da biometria digital no projeto, desenvolvimento do cronograma do semestre, estimar o orçamento preliminar, busca por empresas que se interessem pelo projeto e possam apoiar o desenvolvimento do mesmo, códigos para captura da digital.
- **José Eduardo Chiarelli:** infraestrutura e sistema do Smart Card, pesquisa do problema e da solução do projeto, matriz de decisão, requisitos de engenharia, identificação das necessidades do usuário, esclarecimento do problema, proposta de solução, identificação dos riscos, gerenciamento da integração do sistema e desenvolvimento dos códigos de criptografia do template da digital e comunicação da Certificadora com o Host.

8 RESULTADOS

Esta seção apresenta os principais resultados obtidos na realização do projeto. Os resultados são apresentados em uma sequência que facilita a compreensão do texto e permite acompanhar a evolução do projeto.

8.1 TESTES

Os testes utilizados na pesquisa foram aplicados no ambiente da própria Universidade. Foram aplicados vários testes para garantir a validação do produto, como o teste com uso apenas do Leitor de Digital; um teste com uso da Criptografia e Descriptografia já aplicadas juntamente com a biometria digital; teste com acesso ao cartão java card; gravação de diferentes arquivos no cartão; leitura desses arquivos para comparação com o original e os testes mais importantes que envolvem o englobamento digital, criptografia, gravação no cartão, leitura do cartão e conexão Certificadora Host.

8.1.1 Verificação da Impressão Digital

Os seguintes testes de verificação do aparelho Digital Persona U.are.U 4000B foram realizados com os três integrantes do Grupo. Para a obtenção do Template base foram feitas 4 extrações da digital para cada dedo do integrante e adicionado ao banco de dados. Após todos os 3 templates estarem devidamente armazenados fizemos um teste com cada uma das pessoas, e para todos os testes foram encontradas as digitais compatíveis conforme o esperado. Esses testes basicamente nos demonstraram que o aparelho funciona corretamente, uma vez que o programa teste foi fornecido pelo SDK do Digital Persona.

8.1.2 Verificação da Impressão Digital com a utilização da Criptografia e Descriptografia do Template

- Cadastro de um usuário

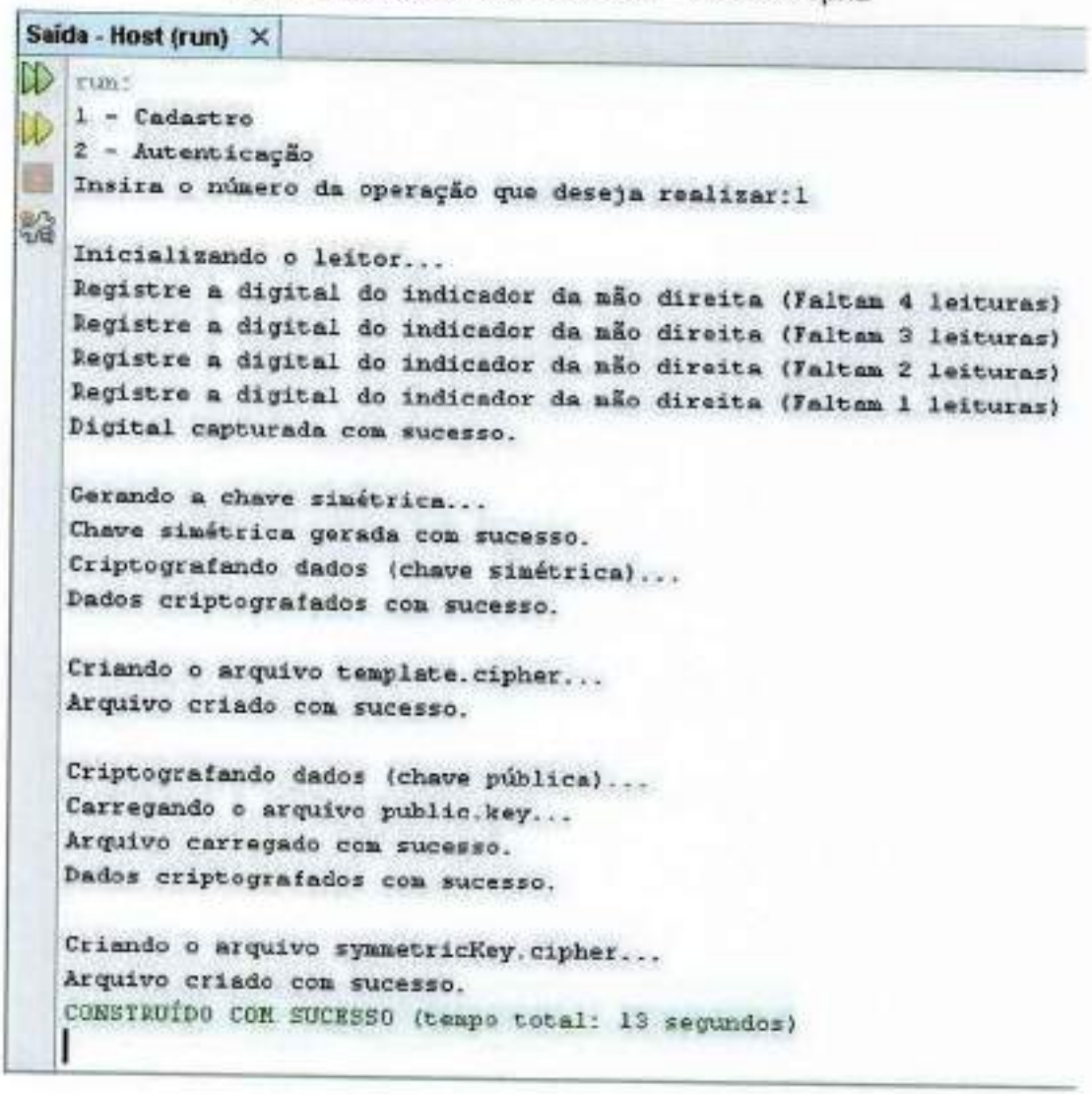
Primeiramente o programa GeraChavesRSA é executado, fornecendo as chaves pública e privada para a criptografia da chave simétrica e do template da digital. Os arquivos de saida serão: public.key e private.key.

Em seguida após a captura da digital, o template será criptografado com a chave simétrica e criará o arquivo template.cipher. Em seguida a Chave simétrica

que foi utilizada para criptografar o template, será criptografada com a chave pública gerando o arquivo symmetricKey.cipher. Para que finalmente o cadastro seja concluído.

Esse processo descrito acima pode ser observado na Figura 29 abaixo.

Figura 35- Cadastro do Usuário – Fonte Própria



- Autenticação de um usuário

Primeiramente o programa do Host é executado com a opção 2 (autenticação). Esta execução carrega o arquivo "symmetricKey.cipher" e envia os dados deste arquivo para a Certificadora, onde ele será descriptografado com a chave privada (proveniente do arquivo "private.key" contido na Certificadora). Após a finalização desta decriptação, o resultado é criptografado novamente com uma chave de sessão (proveniente do arquivo "session.key" também contido na

Certificadora) e enviado de volta ao Host. Ao receber os dados de certificadora, o Host descriptografa estes dados com a mesma chave de sessão (que ele também possui) e, a partir disso, obtém a chave simétrica que descriptografará o template.

Finalmente este template obtido é comparado com uma nova extração de impressão digital do usuário para a autenticação, podendo haver um resultado positivo ou negativo. Seguem as capturas de tela da execução da autenticação (atentar que estas execuções são realizadas simultaneamente):

Figura 36 - Autenticação de um Usuário – Fonte Própria

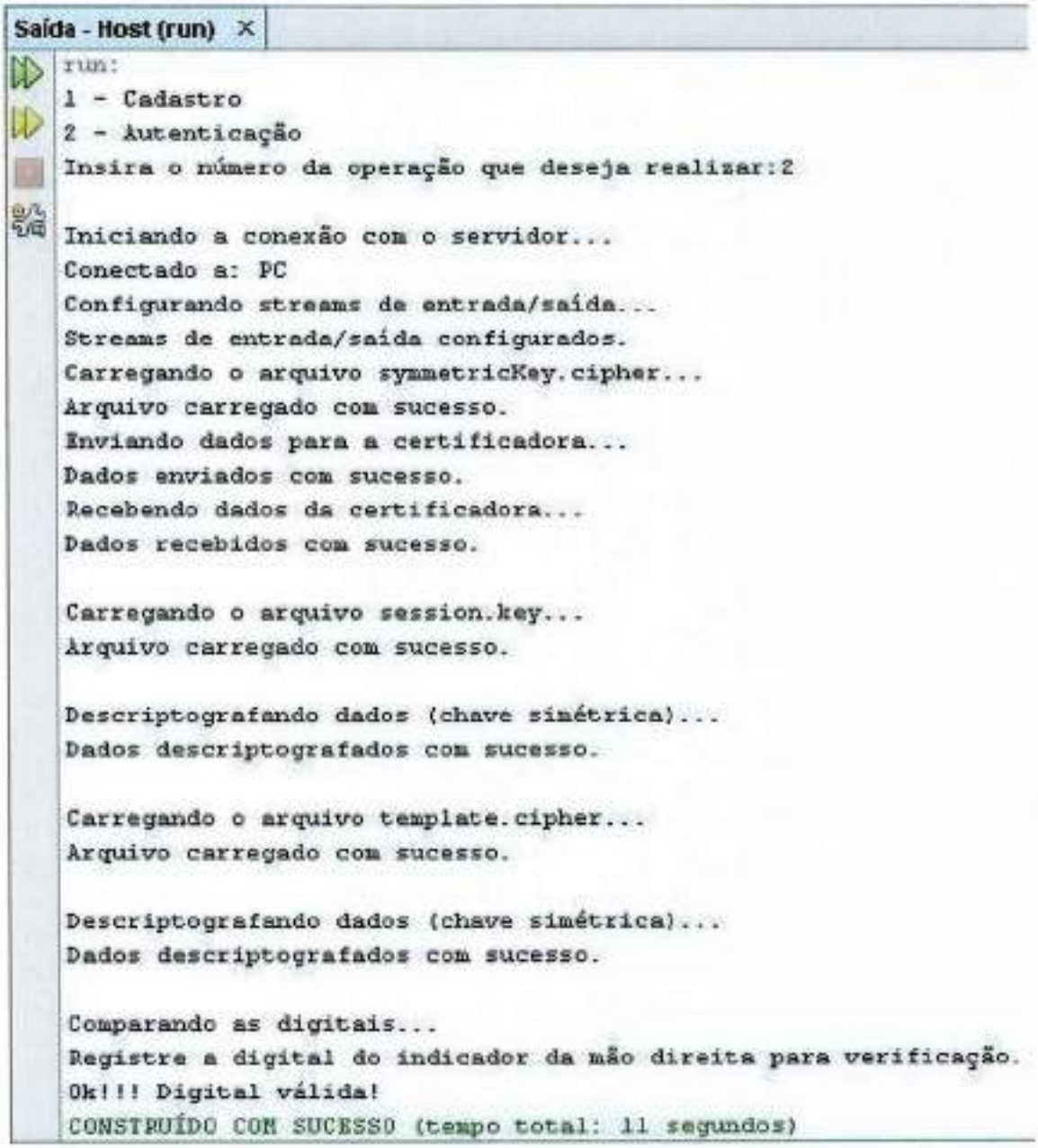
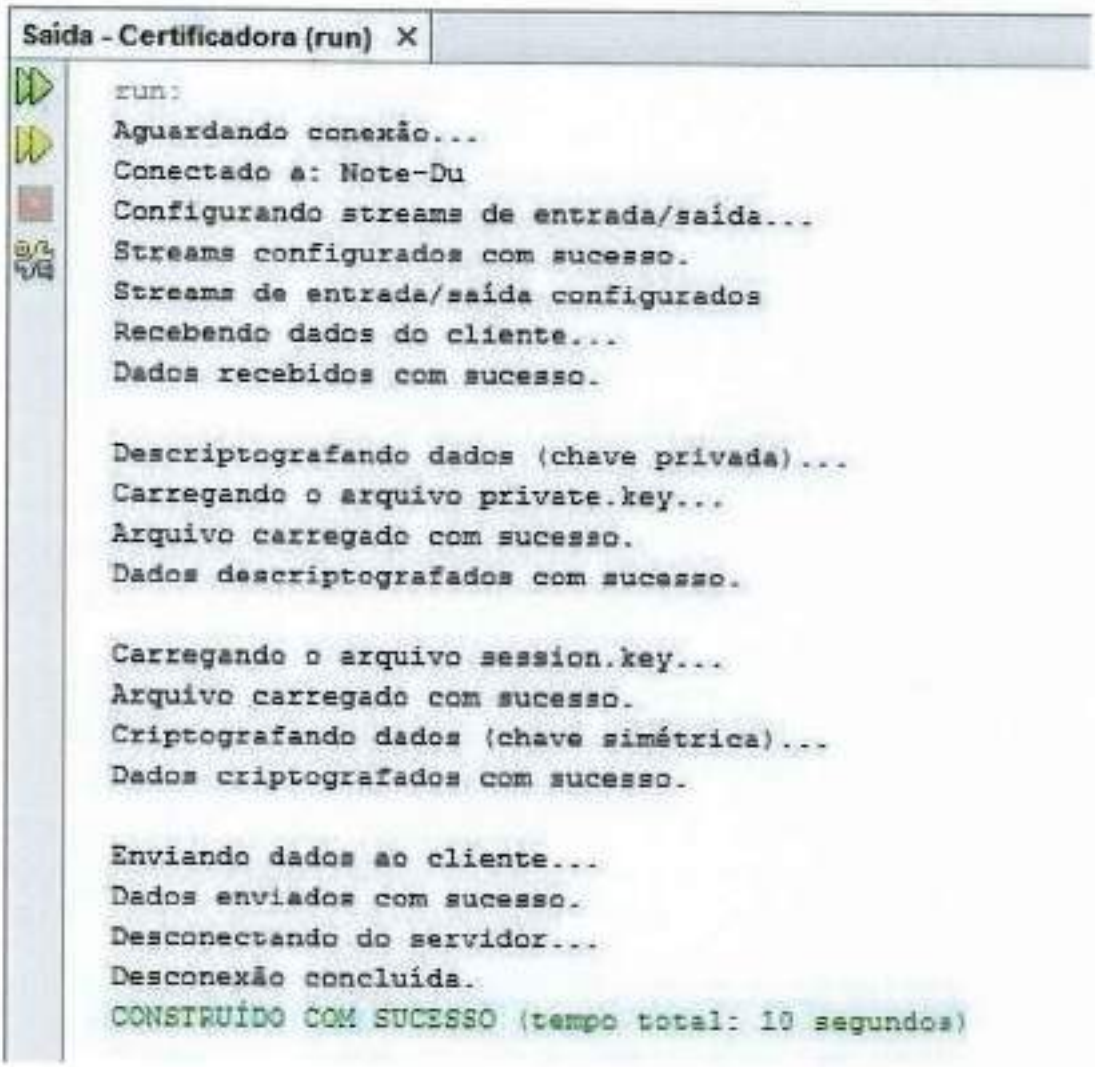


Figura 37 - Envio de dados – Fonte Própria



8.1.3 Teste de gravação da Chave Simétrica no cartão

Abaixo podemos ver o passo a passo a ser desenvolvido para acessar a gravação no cartão, e a utilização dessa função para testar a chave simétrica que deve ser armazenada no cartão.

Como código mais importante temos o 9000, que garante que sempre que retornar (<= 9000) a Gravação/leitura foi realizada corretamente.

Figura 38- Gravação da Chave simétrica no cartão – Fonte Própria

Seguindo essa mesma metodologia realizamos testes com a gravação do Template da Digital no cartão e criou-se uma rotina de requisição do PIN para o acesso e leitura do template e da chave simétrica que estão armazenados no cartão.

Todos os exemplos acima citados podem ser encontrados no APÊNDICE B, que será entregue no CD a parte do trabalho, para que esses exemplos possam ser reproduzidas corretamente.

Abaixo pode-se observar uma foto do protótipo final, onde identifica-se o computador da Certificadora, do Host e os aparelhos da digital e leitora/gravadora do cartão.

Figura 39 - Protótipo final do produto e do Serviço

8.2 Métodos de Depuração

O Eclipse Platform apresenta um depurador de linguagem Java integrado com a funcionalidade de depuração padrão, incluindo a capacidade de realizar execução por etapas, configurar pontos de interrupção e valores, inspecionar variáveis e valores, e suspender e retomar encadeamentos. É usado também para depurar aplicativos em execução em uma máquina remota. A plataforma Eclipse é principalmente um ambiente de desenvolvimento Java, mas a mesma visualização Eclipse Debug também está disponível para C/C++, PHP e diversas outras linguagens de programação.

No Anexo 1 pode-se encontrar um passo a passo de um método de depuração facilmente encontrado na internet [Referência 35].

Para a parte dos aplicativos no cartão a depuração é feita através do envio da status words correspondente ao erro encontrado na execução do aplicativo. Para isto utilizo o **ISOException.throwIt** (CTE DA BIBLIOTECA JAVACARD.FRAMEWORK.ISO7816). As constantes desta biblioteca seguem a lista de Status words específicas para cada tipo de erro definidas pela ISO-7816.

Para erros na parte offcard, em que o host acessa o cartão para leitura/gravação, em caso de recebermos alguma status word (SW) que não seja 0x9000 (Sem erro), o processo é interrompido e é mostrado o SW enviado. Caso ocorra algum outro tipo de erro, o netBeans acusa qual erro foi encontrado na execução e onde ocorreu este erro.

Além destas medidas, foram feitas separadamente classes para cada etapa do projeto, como:

- Captação da digital
- Criptografia de dados
- Gravação da chave simétrica
- Gravação do template da digital
- Leitura da chave simétrica
- Leitura do template da digital
- Execução do aplicativo da empresa 1
- Execução do aplicativo da empresa 2

Desta forma fica mais fácil de identificar em que etapa do projeto ocorreu o erro.

9 Discussão

9.1 Comparação e contraste com outras tecnologias

Existem muitos trabalhos a respeito da integração smartcards-biometria. Este é um assunto que tem recebido grande atenção por oferecer a possibilidade de se criar sistemas de autenticação distribuídos, ou seja, sistemas em que as informações biométricas dos usuários não precisem ficar armazenadas em bancos de dados remotos, pois há possibilidade de estas informações serem armazenadas em um ambiente seguro, neste caso o smartcard.

Pode se afirmar que o smartcard é um ambiente seguro pois os dados guardados em sua memória só se tornam "acessíveis pelo mundo" quando o cartão é inserido em uma leitora, diferentemente dos dados em bancos de dados remotos, que estarão sempre acessíveis, legal ou ilegalmente.

Partindo deste conceito de que os smartcards são um ambiente seguro para o armazenamento das informações biométricas de seu proprietário, surge a questão sobre de que modo os dados obtidos por uma leitora biométrica no momento da utilização do cartão serão comparadas com os dados contidos no cartão. Existem dois tipos de abordagem para esta questão: o método Store-on-Card ("guarda no cartão) e o método Match-on-Card ("compara no cartão).

No método Store-on-Card os dados biométricos são apenas guardados nos cartão, sendo que estes dados são puxados para um computador onde o algoritmo de comparação é executado. Este método foi o primeiro a ser desenvolvido, pois os smartcards ainda não tinham a capacidade de processamento necessária para se executar algoritmos complexos dentro do cartão. Quando a capacidade para se processar tais algoritmos dentro do cartão foi atingida o método Match-on-Card pôde ser desenvolvido. Neste método o algoritmo de comparação é executado dentro do cartão, não havendo, assim, a necessidade de se puxar os dados biométricos para um computador.

À primeira vista o método Match-on-Card parece ser mais seguro que o Store-on-Card, pois os dados biométricos não precisam ser "expostos ao mundo" e desse modo correrem o risco de serem alvo de terceiros mal intencionados. Porém uma questão pode ser levantada: como garantir se o resultado da comparação é verdadeiro? Um trabalho publicado pela Universidad Autonoma de Madrid afirma que sistemas Match-on-Card são mais susceptíveis a ataques do tipo Hill-Climbing

(envio de diversos parâmetros biométricos pseudoaleatórios adaptados iterativamente pelas saidas do algoritmo de comparação) do que sistemas utilizam um computador para fazer a comparação.

Estes fatos levam à conclusão de que sistemas Store-on-Card podem vir ser mais seguros do que os Match-on-Card trazendo, ainda, uma segunda vantagem de ocuparem menos memória e de possuírem requisitos de processamento menores, deixando assim mais espaço e mais capacidade de processamento para outros recursos.

9.2 Limitações do Produto

Após análises entre o professor Orientador e os integrantes do grupo, podem-se identificar duas frentes diferentes que se identificam possíveis limitações do JBCARD e prováveis situações onde o produto possa ser utilizado indevidamente.

Temos o caso em que um profissional, especializado em linguagem java, que saiba trabalhar com java card teria por objetivo a fraude do cartão. Para esse caso ele poderia substituir o template da digital armazenada no cartão para um que seja de seu desejo, porém isso só seria viável se ele tivesse o conhecimento do PIN que utilizamos para acesso dos dados da digital e chave criptografados.

Outro fato que pesou no momento de desenvolvimento do JBCARD foi o fato de cada aplicativo possuir um ID de identificação, e caso o indivíduo mal intencionado queira inutilizar o cartão, ele poderia acessá-lo e através do ID apagaria os dados de cada aplicativo, não permitindo a utilização do cartão, ocasionando grandes problemas ao dono do java card. Uma possível solução que só imaginamos na etapa de finalização do projeto foi a possibilidade de utilizarmos nomes embaralhados nos ID's dos aplicativos, dificultando que qualquer pessoa não permitida saiba exatamente qual ID ativa cada aplicativo.

Podemos citar esses como alguns dos casos em que uma pessoa com experiência na área poderia tentar utilizar o cartão de forma indevida. Já no caso de pessoas que não tem nenhum conhecimento sobre as técnicas utilizadas, o acesso indevido do cartão seria praticamente remoto. Podemos citar apenas o fato de uma abordagem criminosa onde o dono do cartão seria obrigado a utilizá-lo em algum dos Host's permitidos sem vontade própria. No demais casos analisados, o produto atende as necessidades de segurança objetivadas desde o início do projeto.

9.3 Implementações a ser consideradas para o futuro

Considerando as limitações do tempo e atividades desenvolvidas no decorrer do desenvolvimento do projeto, o grupo identificou algumas implementações que poderiam ser mais bem exploradas futuramente. Para o atual momento, preferiu-se priorizar um projeto básico, mas funcional, que cumprisse os objetivos iniciais e estivesse finalizado em tempo da apresentação ao Público.

Como ideias de implementações podemos citar:

- Criar uma interface mais fácil para o usuário, Host e Certificadora (executáveis diretos, sem utilização de digitação dos códigos);
- Utilização de uma digital extra para situação de perigo. Nesse caso o dono do cartão cadastraria um dedo extra e ao colocá-lo no aparelho da digital ele teria a opção de utilizar o dedo normal ou o dedo "de perigo". Com essa identificação diferenciada, por exemplo, o limite do cartão poderá ser diminuído, ou algum aviso para a certificadora poderia ser enviado, seria evitado seu acesso ao local, algo que demonstre que o usuário está tentando "pedir ajuda" e com isso limitações sejam aplicadas imediatamente.
- E finalmente, ampliar o sistema para cartões de contato apenas e não com o chip que estamos utilizando nesse java card atual. Podendo ser ampliado para utilização no Campus da USP onde integraria o bilhete Único estudantil, a carteirinha USP que dá acesso ao bandejão, o acesso as bibliotecas no campus, dentre outras oportunidades de implementações.

10 Conclusões

Sobre o projeto pode-se dizer que ele representou mais do que uma oportunidade para demonstrar o que aprendemos durante a vida acadêmica como também um desafio intelectual para desenvolver um sistema numa área que vem se desenvolvendo cada vez mais.

Seu caráter multidisciplinar foi algo incentivador. Essa união duas áreas do conhecimento em elétrica, sistemas eletrônicos e computação, agregou novas informações a formação dos alunos envolvidos e garantiu aplicação das matérias estudadas no decorrer do curso.

Ao término do projeto o grupo identificou como objetivos alcançados o fato de o desenvolvimento do JBCARD ser de acordo com a descrição de sistema pretentido, possuir um custo de menor que R$400,00 reais de confecção, o que pode ser considerado baixo para os padrões de tecnologia nessa área, a confortabilidade do sistema foi atingida, uma vez que o usuário apenas tem de escanear o dedo durante o processo e também o conhecimento adquirido sobre técnicas de criptografia, smartcards e biometrias.

Não podemos deixar de discutir também as pretenções futuras do grupo em ampliar o sistema e implementar algumas das idéias já citadas no texto, e aprimorar o JBCARD a fim de diminuir as limitações apresentadas nessa tese.

Para finalizar, é indiscutível a experiência trazida pelo projeto de conclusão de curso na formação de um engenheiro. Trabalhar com um projeto de tamanha complexidade prepara o estudante para as dificuldades enfrentadas durante sua vida profissional. Além disso, o projeto abre espaço para o uso da criatividade e da responsabilidade de criar algo que não possui uma forma pré-definida, como é o caso da maior parte dos trabalhos da universidade. O desempenho do projeto depende muito dos membros, pois as informações, motivações e inovações, embora auxiliadas até certo ponto pelo orientador, devem partir dos integrantes do grupo.

11 Referências

1. EFFING, W; RANKEL, W. Smart Card Handbook. 4th Edition. Wiley, 2010. 1088p.
2. HENDRY, M. Smart Card Security And Applications. 2nd Edition. Artech House, 2001. 305p.
3. HENDRY, M. Multi-application Smart Cards: Technology And Appli- cations. 1st Edition. Cambridge University Press, 2007. 266p.
4. BIOMETRIC CONSORTIUM INTEROPERABILITY, ASSURANCE AND PERFORMANCE WORKING GROUP. Java CardTMBiometric API White Paper. Version 1.1. 2002. 17p.
5. OSBORNE, M; RATHA, R.K. A JC-BioAPI Compliant Smart Card with Biometrics for Secure Access Control. IBM Zurich Research Lab and IBM T. J. Watson Research Lab Yorktown Heights, NY. 2003. 8p.
6. KIM, D.S; LEE, S.Y; et al. On the Design of an Embedded Biometric Smart Card Reader. 5p.
7. SANCHEZ-REILLO, R. Securing Informations and Operations in a Smartcard Through Biometrics. E.T.S.I. Telecomunicacion - Dpt. Tecnologia Fotonica Ciudad Universitaria, s/n, E-28040 – Madrid, SPAIN. 4p.
8. MARTINEZ-DIAZ, M; FIERREZ-AGUILAR, J; et al. Hill-Climbing and Brute-Force Attacks on Biometric Systems: A Case Study in Match-on-Card Fingerprint Verification. ATVS/Biometrics Research Lab, Escuela Politecnica Superior - Universidad Autonoma de Madrid. 9p.
9. One Touch for Windows SDK Java Developer Guide. Guia do desenvolvedor. DigitalPersona, Inc. V 1.6.
10. CAMPOS, S., Leal, F., Henrique, J., Borba, P. "Introdução ao Eclipse". Disponível em <http://www.cin.ufpe.br/~phmb/ip/MaterialDeEnsino/IntroducaoAoEclipse/Introduca oAoEclipse.htm>, 2008. Acessado em 12 de Junho de 2013.
11. OLIVEIRA, E. C. M. "Projeto Eclipse for Java". Disponível em <http://www.linhadecodigo.com.br/Artigo.aspx?id=677>, Abril, 2005. Acessado em 12 de Junho de 2013.

12. VASCONCELOS, R. C. Sistemas Automáticos de Identificação de Impressões Digitais. FATEC. 10p.6. Biometria. Wikipedia. Disponível em <http://pt.wikipedia.org/wiki/Biometria>. Acessado em 2 de Maio de 2013.
13. FARIA, A. Biometria: Processamento de imagens capturadas em leitores de impressão digital. Disponível em <http://www.linhadecodigo.com.br/artigo/1162/biometriaprocessamento-de-imagens-capturadas-em-leitores-de-impressao-digital.aspx>. Acessado em 3 de Maio de 2013.
14. Digital Persona U.are.U® 4000B Reader USB Fingerprint. Disponível em <http://www.signtechbiometric.com/pdf/digitalpersona/uareu4000breader.pdf>. Acessado em 12 de Junho de 2013.
15. REIS C. M. S. Autenticação com Impressão Digital. Instituto superior de Engenharia de Lisboa. Dezembro 2003.
16. Planet Smart Cards – Global Platform. Disponível em <http://planetsmartcards.blogspot.com.br/2010/12/o-global-platform.html>. Acessado em 10 de Junho de 2013.
17. Planet Smart Cards – Acessando Smart Cards Usando Java. Disponível em <http://planetsmartcards.blogspot.com.br/2010/12/acessando-smart-cards-usando-java.html>. Acessado em 12 de Junho de 2013.
18. Wikipedia - Global Platform. Disponível em <http://en.wikipedia.org/wiki/GlobalPlatform>. Acessado em 12 de Junho de 2013.
19. Wikipedia – ISSO/IEC 7816. Disponível em <http://en.wikipedia.org/wiki/ISO/IEC_7816>. Acessado em 12 de Junho 2013.
20. Planeta Smart Cards – ISSO/IEC 7816. Disponível em <http://planetsmartcards.blogspot.com.br/2010/12/isoiec-7816.html>. Acessado em 12 de Junho de 2013.
21. Definição do Tamanho dos arquivos gerados. Disponível em <http://stackoverflow.com/questions/10007147/getting-a-illegalblocksizeexception-data-must-not-be-longer-than-256-bytes-when> Acessado em 20 de Setembro de 2013.
22. Transferência de Arquivos entre 2 computadores. Disponível em <http://mrbool.com/file-transfer-between-2-computers-with-java/24516>. Acessado em 20 de Setembro de 2013.

23. Manipulação de arquivos .txt. Disponível em <http://www.mballem.com/post/manipulando-arquivo-txt-com-java>. Acessado em 20 de Setembro.
24. Chaves de Criptografia Java. Disponível em <http://www.etpenguin.com/cgi-bin/index2file.cgi?/pub/Encryption/Java-com.brettlee.cryptosamples/EncryptWithAESfromKeystore.java>. Acessado em 20 de Setembro de 2013.
25. Chained APDU. Disponível em <https://forums.oracle.com/thread/1751775>. Acessado em 21 de Setembro de 2013.
26. Memórias no JavaCard. Disponível em <http://planetsmartcards.blogspot.com.br/2010/12/o-modelo-de-memoria-java-card.html>. Acessado em 21 de Setembro de 2013.
27. Guia de desenvolvimento Java Card. Disponível em <http://www.informatik.uni-augsburg.de/lehrstuehle/swt/se/teaching/fruehere_semester/ws0708/javacard/Dokumentation/JCADG.pdf> Acessado em 28 de Setembro de 2013.
28. Importando dados de .txt. Disponível em < http://www.mkyong.com/java/how-to-read-file-from-java-bufferedreader-example/> Acessado em 29 de Setembro de 2013.
29. Enviando APDUs apenas uma vez. Disponível em <https://forums.oracle.com/thread/1751775> Acessado em 01 de Outubro de 2013.
30. Command APDUs – ISO7816. Disponível em <http://www.cardwerk.com/smartcards/smartcard_standard_ISO7816-4.aspx> Acessado em 12 de Outubro de 2013.
31. Command APDUs. Disponível em <http://web.archive.org/web/20090630004017/http://cheef.ru/docs/HowTo/APDU.info> Acessado em 20 de Setembro de 2013.
32. Response APDUs. Disponível em <http://web.archive.org/web/20090623030155/http://cheef.ru/docs/HowTo/SW1SW2.info> Acessado em 20 de Setembro de 2013.
33. Pesquisa Serasa. Disponível em <http://www.serasaexperian.com.br/pesquisaaplicada/> Acessado em 10 de Outubro de 2013.

34. Pesquisa Visa Internacional. Disponível em <http://www.caixa.gov.br/voce/Cartoes/Visa/Internacional/index.asp> Acessado em 10 de Outubro de 2013.
35. Depuração da plataforma do Eclipse. Disponível em <http://imasters.com.br/artigo/24205/desenvolvimento/depurando-com-a-plataforma-eclipse/> Acessado em 15 de Novembro de 2013.

12 Anexo 1 – Descrição de Depuração do Eclipse

O Eclipse SDK – em especial, o projeto Java Development Tools (JDT) – apresenta um depurador Java integrado que fornece toda a funcionalidade de depuração, incluindo a capacidade de realizar execução por etapas, configurar pontos de interrupção e valores, inspecionar variáveis e valores, e suspender e retomar encadeamentos. Adicionalmente, é possível depurar aplicativos executando em uma máquina remota. A plataforma Eclipse é robusta de tal modo que outras linguagens de programação podem usar os recursos de depuração dos seus respectivos tempos de execução de linguagem. Como você verá abaixo, a mesma visualização Debug do Eclipse também está disponível para as linguagens de programação C/C++.

O ambiente de trabalho da plataforma Eclipse e suas ferramentas são desenvolvidos em torno dos componentes JDT, que fornecem os seguintes recursos para o Eclipse:

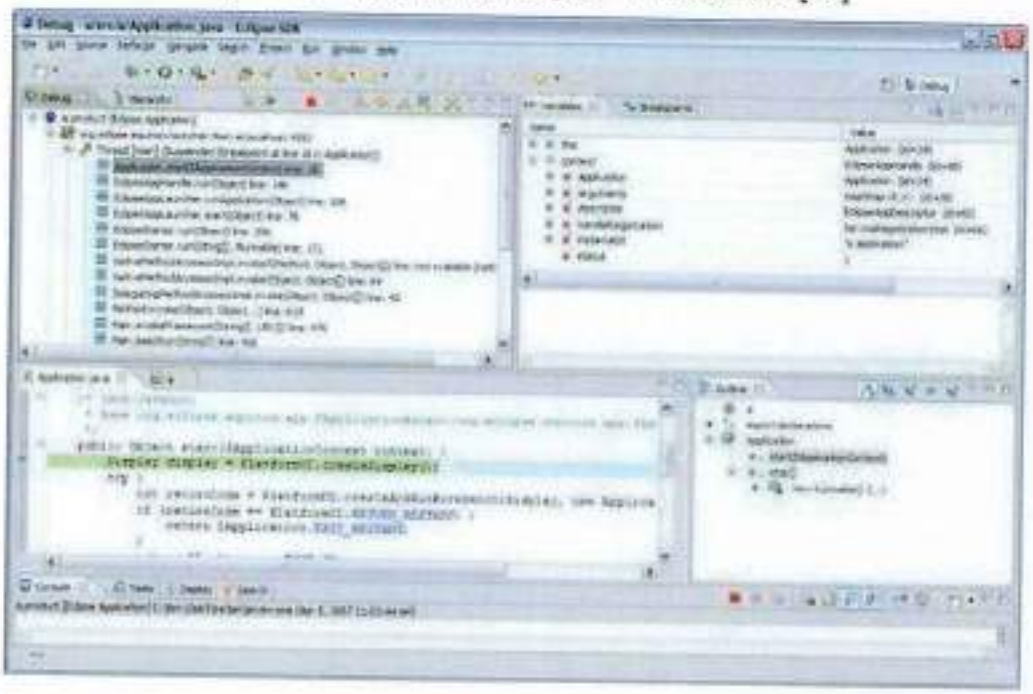
- Ferramentas de gerenciamento de projeto;
- Perspectivas e visualizações;
- Funções Builder, editor, procura e desenvolvimento;
- O depurador.

O depurador do Eclipse existe como um conjunto padrão de plug-ins incluído no Eclipse. O Eclipse também tem uma visualização de depuração especial que permite o gerenciamento da depuração ou execução de um programa no ambiente de trabalho. Ele exibe a estrutura da pilha dos encadeamentos suspensos para cada destino que você está depurando. Cada encadeamento no programa aparece como um nó na árvore e a visualização Debug exibe o processo de cada destino que você está executando. Se o encadeamento estiver suspenso, suas estruturas de pilha são mostradas como elementos filho.

Antes de começar a usar o depurador do Eclipse, presume-se que você tenha o Java SDK/JRE apropriado (recomendo usar o Java VM V1.4) e o Eclipse Platform SDK V3.3 instalado, e que ambos estejam executando sem problemas. No geral, é uma boa ideia testar as opções de depuração usando as amostras do Eclipse primeiro. Se você deseja desenvolver e depurar projetos C/C++, terá adicionalmente de obter e instalar as C/C++ Development Tools (CDT). Sobre links para o Java

SDK/JRE, a plataforma Eclipse e amostras, e as CDT, consulte "Recursos". A Figura 1 mostra a visualização geral da perspectiva Debug.

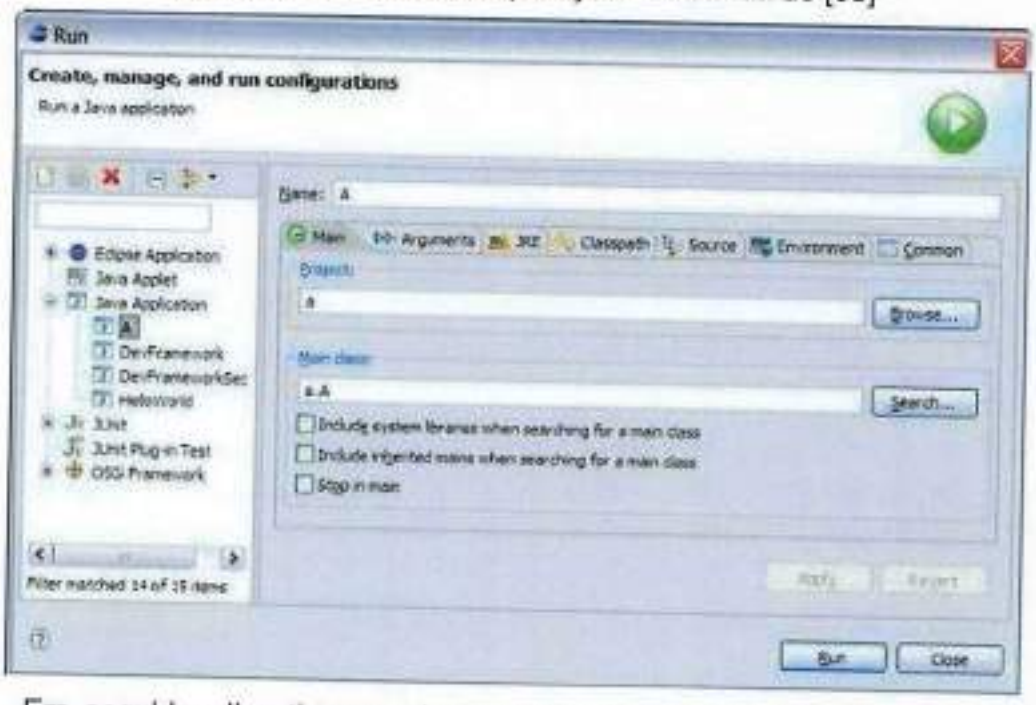
Figura 40 - Plataforma Eclipse - Retirada de [35]



12.1 Depurando programas da linguagem Java

Antes que seja possível depurar seu projeto, o código precisa compilar e executar de modo nítido. É necessário criar uma configuração de execução para o seu aplicativo e assegurar que ele inicie devidamente. Em seguida, é necessário definir a configuração de depuração em um modo similar usando o menu Run > Debug. Também é necessário selecionar a classe a ser usada como a classe Java principal pelo depurador (veja a Figura 2). É possível ter tantas configurações de depuração para um único projeto quanto desejado. Quando o depurador é iniciado (em Run > Debug), ele é aberto em uma nova janela, e você está pronto a iniciar a depuração.

Figura 41 - Janela de Depuração - Retirada de [35]

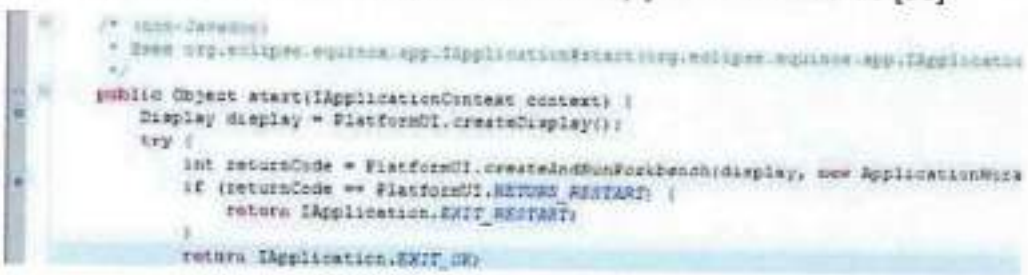


Em seguida, discutiremos algumas das práticas de depuração comuns no Eclipse.

12.2 Configurando pontos de interrupção

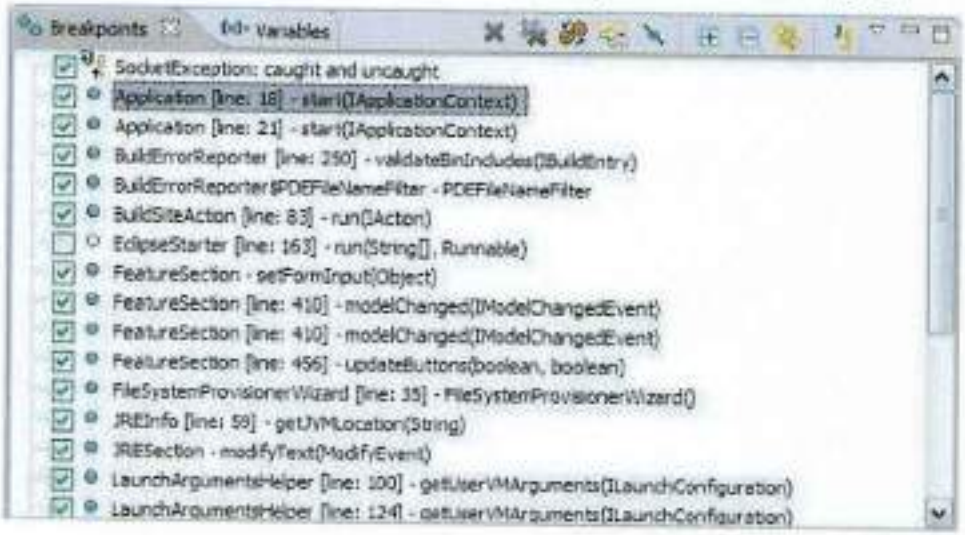
Ao iniciar o aplicativo para depuração, o Eclipse alterna para a perspectiva Debug automaticamente. Sem dúvidas que o procedimento de depuração mais comum é configurar pontos de interrupção que permitirão a inspeção de variáveis e valores dentro de instruções condicionais ou loops. Para configurar pontos de interrupção na visualização Package Explorer da perspectiva Java, clique duas vezes no arquivo do código de origem selecionado para abri-lo no editor. Percorra o código e posicione o cursor na barra de marcação (ao longo da borda esquerda da área do editor) na linha com o código suspeito. Clique duas vezes para definir o ponto de interrupção.

Figura 42- Definição do ponto de interrupção - Retirada de [35]



Agora, inicie a sessão de depuração no menu Run > Debug. É importante não colocar várias instruções em uma única linha porque não é possível depurar parcialmente ou definir pontos de interrupção de linhas em mais de uma instrução na mesma linha.

Figura 43- Início da sessão de depuração - Retirada de [35]



Também há uma visualização Breakpoints conveniente para gerenciar todos os pontos de interrupção.

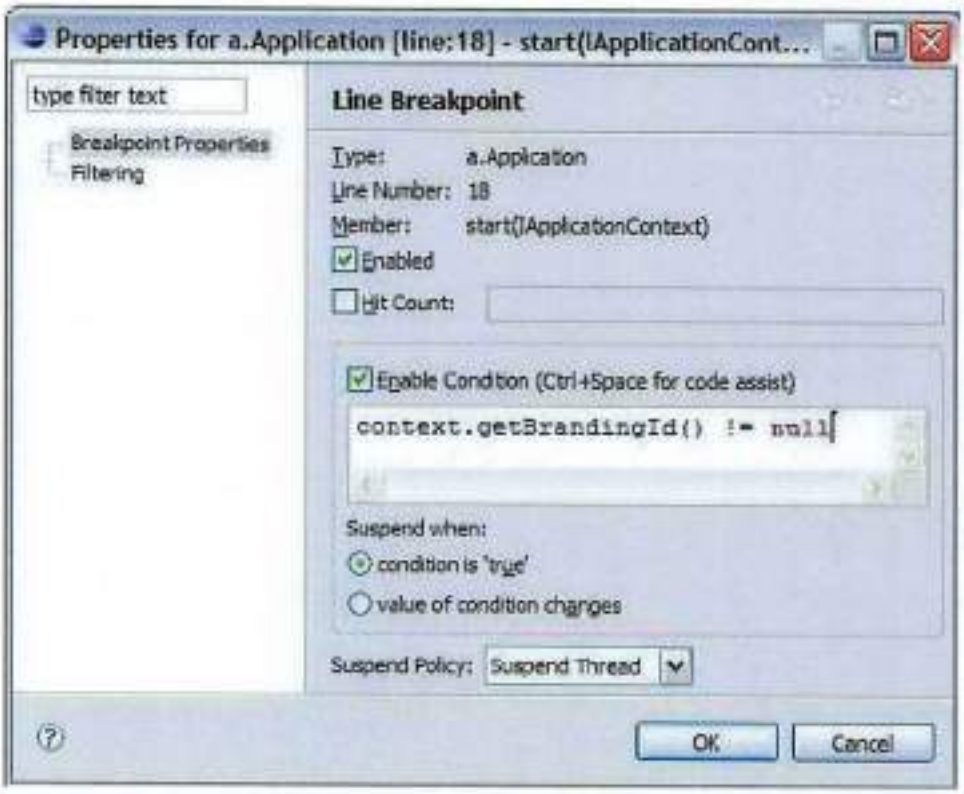
Figura 44- Visualização de Breakpoints - Retirada de [35]

12.3 Pontos de interrupção condicionais

Assim que identificar o local onde o erro está, você desejará saber o que o programa está fazendo certo antes de travar. Um modo de fazer isso é percorrer cada instrução no programa, um por vez, até atingir o ponto de preocupação.

Algumas vezes, é melhor apenas executar uma seção de código e interromper a execução nesse ponto de modo que seja possível examinar os dados nesse local. É possível declarar pontos de interrupção condicionais acionados todas as vezes que o valor de uma expressão muda (veja a Figura 6). Em adição, o assistente de código está disponível quando você digita na expressão condicional.

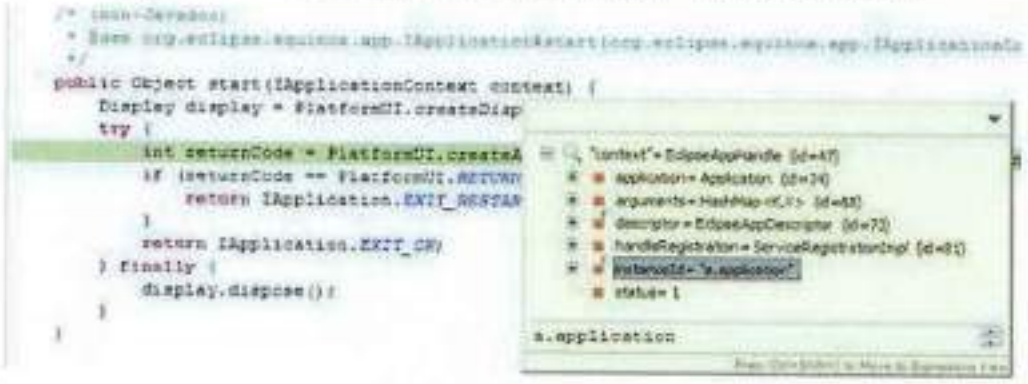
Figura 45- Propriedades para as aplicações - Retirada de [35]



12.4 Avaliando expressões

Para avaliar expressões no editor, na perspectiva Debug, selecione a linha inteira onde o ponto de interrupção está definido e, no menu de contexto, selecione a opção Inspect via Ctrl+Shift+I ou clicando com o botão direito do mouse na variável em que você está interessado (veja a Figura 7). A expressão é avaliada no contexto da estrutura de pilha atual, e os resultados são exibidos na visualização Expressions da janela Display.

Figura 46 - Visualização das Expressions - Retirada de [35]



12.5 HOTSWAP BUG FIXING: CORREÇÃO IMEDIATA DE CÓDIGO

Se estiver executando o Java Virtual Machine (JVM) V1.4 ou posterior, o Eclipse suporta um recurso denominado Hotswap Bug Fixing (não disponível na JVM V1.3 ou inferior). Ele permite a alteração do código de origem durante uma sessão do depurador, que é melhor que sair do aplicativo, alterar o código, recompilar e iniciar outra sessão de depuração. Para usar essa função, simplesmente mude o código no editor e retome a depuração. Esse recurso tornou-se disponível porque o JVM V1.4 é compatível com a Java Platform Debugger Architecture (JPDA). A JPDA implementa a capacidade de substituir código modificado em um aplicativo em execução.

Evidentemente, isso é especialmente útil quando leva um longo tempo para iniciar o aplicativo ou para iniciar o aplicativo ou acessar o ponto onde ele falha.

Se o programa não tiver executado completamente quando a depuração estiver concluída, selecione a opção Terminate no menu de contexto, na visualização Debug. Um erro comum é usar Debug ou Run em vez de Resume enquanto estiver em uma sessão do depurador. Isso ativará outra sessão do depurador em vez de dar continuidade na sessão atual.

13 APÊNDICE A – PROCESSO DE ANÁLISE DE HIERARQUIA

Seguem abaixo os passos a serem seguidos para a definição dos critérios e pesos considerados para a Matriz de Decisão.

Passo 1: Definir o objetivo

Aumentar a segurança em Java Cards.

Passo 2: Estruturação dos elementos em critérios, subcritérios e alternativas

Passo 3: Comparar os elementos dois a dois

Passo 4: Calcular os pesos e a razão de consistência (CR)

Passo 5: Avaliar as alternativas de acordo com os pesos

Passo 6: Fazer um rambing

C1 – Praticidade

C2 – Facilidade de integração do sistema

C3 – Preços dos dispositivos necessários para a implementação

C4 – Dificuldade de reprodução/ roubo dos parâmetros

Tabela 6 – Matriz Final de Decisão

	C1	C2	C3	C4	Média geométrica da linha	Peso(%)
C1	1	1/3	5	3	1,50	30,63%
C2	3	1	3	3	2,28	46,69%
C3	1/5	1/3	1	3	0,67	13,70%
C4	1/3	1/3	1/3	1	0,44	8,99%
				Σ	4,88	100,00%

14 APÊNDICE B – Códigos

Os códigos não foram diretamente disponibilizados nessa tese devido à quantidade de páginas que ocupavam, uma vez que são muitos códigos para a execução do projeto haveria pelo menos 50 páginas adicionais nesse trabalho.

Para maiores informações sobre os códigos utilizados, está sendo disponibilizado um CD juntamente com o trabalho final possuindo todos os códigos necessários para garantir a aplicabilidade do projeto e que ele possa ser reproduzido conforme o esperado.

Qualquer dúvida deverá ser tirada diretamente com os componentes do grupo para maiores informações e esclarecimentos.

15 APÊNDICE C – BIOMETRIA DIGITAL

15.1 FALSOS POSITIVOS E FALSOS NEGATIVOS

Quando um sistema biométrico é colocado em uso, ele irá encontrar ou não um par para o dado biométrico extraído. Uma nota é dada para a comparação entre o template e o novo exemplo. Se a nota for maior que a base para uma aplicação, o par é encontrado. Esta técnica dá à biometria muito mais flexibilidade que o "sim ou não" aproximado utilizado pelas tecnologias baseadas em PIN ou senhas.

Por alguns anos a indústria biométrica tem utilizado duas medidas de desempenho para graduar o nível de precisão da comparação. Elas estão focadas na habilidade do sistema de permitir o acesso a usuários autorizados e negar o acesso àqueles que não são autorizados. São conhecidas como a Taxa de Falsa Rejeição (FRR) e a Taxa de Falsa Aceitação (FAR). Para mistificar ainda mais o processo, algumas vezes o FRR é denominado como taxa de erro do Tipo I, enquanto o FAR é denominado como taxa de erro do Tipo II.

A taxa de falsa rejeição refere-se à quantidade de vezes que um indivíduo autorizado é falsamente rejeitado pelo sistema. A taxa de falsa aceitação refere-se à quantidade de vezes que um indivíduo não-autorizado é falsamente aceito pelo sistema. Ambas as taxa são expressas na forma de porcentagem, utilizando-se os cálculos que seguem abaixo:

FRR:

$$\frac{\textbf{Número de falsas rejeições}}{\textbf{Número de reconhecimentos autorizados ou tentativas de verificação}} * 100$$

FAR:

$$\frac{\textbf{Número de falsas aceitações}}{\textbf{Número de reconhecimentos impostores ou tentativas de verificação}} * 100$$

As fórmulas acima são surpreendentemente simples, dado o grande número de variáveis que pode ser apresentado ao sistema biométrico.

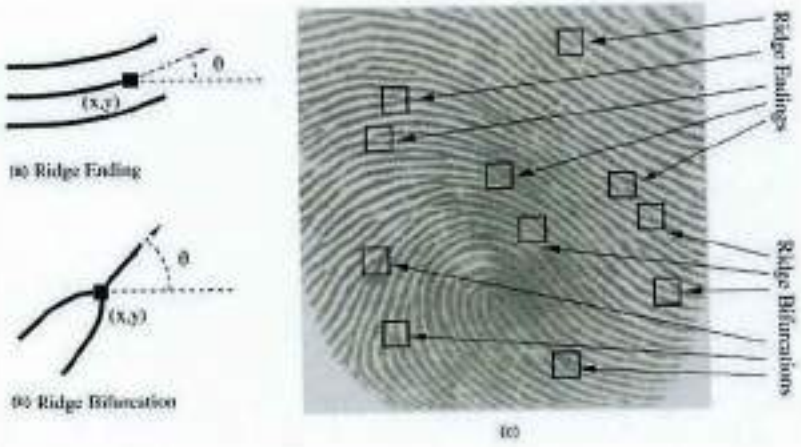
A FRR e a FAR são um jeito simplificado de avaliar o desempenho. São tanto uma vantagem quanto uma desvantagem. Verificando a taxa de erros e

considerando a FRR e a FAR em concordância é fácil determinar o desempenho básico do sistema, sempre lembrando que as circunstâncias de uma aplicação não devem ser esquecidas, especialmente se um sistema biométrico pretende operar dentro dos limites de uma certa aplicação.

15.2 Tratamento da Imagem e Extração Digital

As minúcias são caracterizadas por três parâmetros: tipo, posição e orientação. O tipo da minúcia define se estamos na presença de uma terminação ou de uma bifurcação. A posição indica o local onde se verifica a existência da minúcia e a orientação nos dá a direção da linha que originou a minúcia.

Figura 47- Imagem da orientação das minúcias - Retirada de [15]



Em um sistema de identificação ou autenticação automática, a máquina tem que calcular uma medida de semelhança entre a impressão que serviu como registo e a impressão que se pretende autenticar. Caso a semelhança entre ambas seja elevada, como no caso de gêmeos idênticos, então pode acontecer que o processo de comparação não seja suficientemente robusto para garantir a correta identificação. Neste caso pode-se ajustar o processo de comparação para resolver esta questão, sacrificando o desempenho global. Deve-se realçar que nem todos os casos de gêmeos idênticos trazem problemas na identificação, apenas existirá dificuldade naqueles em que as suas impressões digitais são muito semelhantes.

A separação das impressões digitais em classes reduz o tempo de processamento durante a identificação pois só é necessário efetuar comparações com as impressões da mesma classe. Abaixo vemos as principais classes identificadas na população.

Figura 48- Imagem das classes de uma impressão digital – Retirada de [15]

O tratamento da imagem é a etapa mais importante deste sistema. Ela visa melhorar a qualidade da imagem fornecida pelo leitor biométrico para que o programa identifique as minúcias com maior precisão. Uma imagem de má qualidade acaba sendo prejudicial ao sistema, gerando muitos erros, como distorções na imagem e minúcias espúrias.

A imagem agora é submetida a um processo de binarização, que consiste em transformar os tons de cinza da imagem em preto e branco, ou seja, transformar os tons de cinza em uma matriz de zeros e uns. Esta é uma etapa muito importante do tratamento da imagem, pois uma binarização mal executada acaba gerando muita perda de informação.

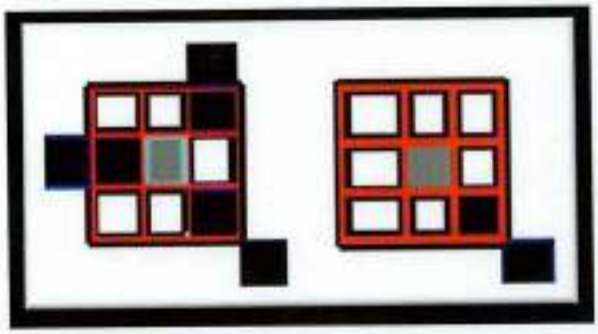
Os pixels da imagem possuem tons de cinza variando entre 0 e 256. As linhas da imagem possuem valores elevados em relação ao restante da imagem, a parte vazia. Para binarizar uma imagem, a linha cinza escura deveria ser transformada em preto e o espaço vazio da imagem em branco. Para isso, toda a imagem foi analisada em quadros de 8 x 8 pixels e obtido um threshold para cada uma das regiões através da média geométrica dos pixels daquela região. Assim, todos os pixels que estivesse acima desse threshold seriam transformados no valor 1 e o que estivesse abaixo em 0.

O próximo passo é o afinamento das linhas. O algoritmo de afinamento analisa a imagem da ID e remove os pixels redundantes das linhas que formam as cristas. Esse processo é repetido até que não se tenha mais pixels redundantes. A varredura da imagem é feita linha a linha, examinando a vizinhança e verificando se o pixel pode ou não ser apagado. Quando um pixel é apagado, seu valor muda de 1 para 0, e a imagem é dita afinada.

15.3 Extração das Minúcias

Nesta etapa, um processo chamado de crossing number analisa os pixels da imagem em uma matriz 3 x 3 à procura de alguns pontos característicos. Quando o pixel central contiver um único vizinho, ele é dito terminação, e quando ele contiver três vizinhos, ele é dito bifurcação.

Figura 49- Bifurcação à esquerda e terminação à direita – Retirada de [15]



Porém são obtidas várias minúcias espúrias, pois no processo de afinamento da imagem, a linha pode apresentar algumas interrupções repentinas e logo em seguida a sua continuação, gerando para o algoritmo duas terminações. Neste caso, a qualidade da imagem e o tratamento da mesma naquela região não foram suficientes para gerar um bom afinamento das linhas.

Isso dependerá muito da qualidade da imagem e do seu tratamento utilizando um algoritmo capaz de eliminar essas falsas minúcias através do cálculo da distância euclidiana entre elas. Assim, quaisquer minúcias que distam em até nove pixels entre si podem ser eliminadas. A figura abaixo compara as minúcias encontradas inicialmente e logo após o processo de eliminação. Os pontos vermelhos representam as terminações e os pontos azuis representam as bifurcações.

Figura 50- Terminações e bifurcações - Retirada de [15]

16 APÊNDICE D – SITE/BLOG DO PROJETO