Universidade Federal de Santa Catarina

Ciências da Computação

Luca Fachini Campelli

DESENVOLVIMENTO DE UMA APLICAÇÃO PARA COLETA DE DADOS E EMISSÃO DE PASSAPORTES ELETRÔNICOS NA PLATAFORMA JAVACARD

Florianópolis/SC
2018

Luca Fachini Campelli

DESENVOLVIMENTO DE UMA APLICAÇÃO PARA COLETA DE DADOS E EMISSÃO DE PASSAPORTES ELETRÔNICOS NA PLATAFORMA JAVACARD

Trabalho de Conclusão de Curso para a graduação no curso de Ciências da Computação UFSC

Florianópolis, 2018

Resumo (Melhorar)

Foi desenvolvido um software para computador capaz de coletar as informações do usuário, e emitir um passaporte eletrônico em um Javacard que segue o padrão ICAO 9303. Este cartão é munido de todas as informações necessárias para a verificação da identidade do usuário, incluindo informações biométricas como identificação facial, e digitais, juntamente com todos os mecanismos de segurança descritos pelo documento ICAO 9303.

Palavras-chave: Javacard, Passaporte,JMRTD, Segurança, Passaporte-Eletrônico, e-Passport, SmartCard, JCOP

Abstract

A computer software was developed capable of collecting user information, and emiting an electronic passport on a Javacard that follows the ICAO 9303 standard. This card posesses all required information to verify the user's identity, including biometrics such as face recognition and fingerprints, together with all security mecanisms described in the ICAO 9303 document.

Key-Words: Javacard, Passaport,JMRTD, Security, Electronic-Passport, e-Passport, SmartCard, JCOP

Sumário

Figuras

1 Introdução

Quando começaram a ser utilizados, os passaportes funcionavam apenas por meio físico, na forma de uma caderneta ou documento escrito, com todas as informações sendo verificadas manualmente, comparando as informações com o outros papéis, o que abria brechas para a falsificação. Conforme o documento foi evoluindo, até se tornar a caderneta que conhecemos hoje, várias formas de prevenir falsificações como marcas d'água, padrões de impressão e manufatura do papel foram utilizadas, porém a inovação mais impactante e discutida foi a inserção de um chip eletrônico dentro do passaporte (*Passport*). Embora várias informações possam ainda ser verificadas visualmente mais ainda podem ser verificadas por meio deste chip interno do passaporte, e mais rapidamente, permitindo até que a aferição do passaporte seja automatizada. Este chip pode guardar as informações biométricas do dono, junto com as informações visuais, e ainda mais importante, possuindo diversos protocolos de segurança para garantir a autenticidade dos dados conferidos (HomeOffice: 2013).

Diversos softwares existem para a leitura de passportes e, daqueles escritos em Java, em sua maioria se utilizam da biblioteca JMRTD(Oostdijk: 2010) como base para seu funcionamento, mas não foi possível encontrar o software utilizado por aeroportos para a verificação de passaportes. Sendo assim, seria possível emitir um passaporte na plataforma javacard, utilizando bibliotecas open-source, que seja equivalente ao passaporte utilizado hoje em dia?

1.1 Objetivos

O objetivo principal é a criação do sistema de coleta de dados e emissão de um passaporte eletrônico em um cartão Javacard, onde os dados sejam armazenados depois possam ser extraídos e validados em qualquer tipo de máquina que possua este sistema.

Desta forma, alguns requisitos são propostos para que o objetivo se dê por alcançado:

1.O sistema deve coletr os dados básicos do usuário

2.O sistema deve tirar a foto do usuário e extrair dela os dados para o reconhecimento facial.

3.O sistema deve coletar os dados biométricos como digital e íris.

4.Ter o cartão preenchido e que cumpra as especificações do padrão ICAO9303.

5.O cartão deve ser equivalente a um passaporte oficial, na questão de sua estrutura interna e dados armazenados

Ao final do projeto espera-se ter concluído a criação de um sistema que colete os dados do usuário, armazenando-os no cartão, conforme as especificações ditadas no padrão ICAO9303

1.2 Trabalhos Correlatos e Revisão Bibliográfica

Parametrizar o método de busca para deixar mais forte esta seção. Com qual query de busca que se chegou o resultado, qual foi a extensão da busca.
Várias bibliotecas e aplicativos funcionais existem, que fazem possível a leitura de passaportes eletrônicos, ou por via de chips de contato ou RFID, escritos em várias linguagens de programação. Muitos poucos permitem a emissão e personalização destes passaportes. Para a linguagem Java, vários se utilizam também da biblioteca JMRTD.

A pesquisa foi efetuada utilizando-se a plataforma de busca Google, com as seguintes pesquisas: "e-passport reader python", "e-passport reader java", "e-passport reader c++", "e-passport creation", "passport creation python", "passport creation java", sendo analisada toda a primeira e segunda páginas da busca, e os que mais se destacaram foram:

1. pypassport(Houzard and Roger: 2009) - PYTHON - provê a emissão e leitura de passaportes RFID

2. innoValor(InnoValor: 2013) - JAVA - provê apenas a leitura de passaportes RFID

3. e-passport NFC reader(Tananaev: 2016) - JAVA - provê apenas a leitura de passaportes RFID, utiliza a biblioteca JMRTD

4. epassport reader(Bozhanov: 2016) - JAVA - provê apenas a leitura de passaportes, utiliza a biblioteca JMRTD

5. RFIDIOt(Laurie: 2011) - PYTHON - Ferramentas para trabalhar com cartões RFID, podendo ler e escrever nos cartões

6. wzmrtd(waazaa.org and ariadNEXT: 2011) - C++ - provê apenas a leitura de passaportes RFID.

2 Fundamentações teóricas e práticas

2.1 Criptografia Simétrica e Assimétrica

É possível cifrar a informação para que apenas quem possua a chave da cifra possa acessa-la. O ato de embaralhar o significado da mensagem se chama cifrar, e o inverso, decifrar. Para tal, utiliza-se algum tipo de algoritmo que, em conjunto com uma chave, seja capaz de mascarar a informação original e que seja reversível, para que com o uso da chave, se possa desfazer a criptografia e acessar o conteúdo original da mensagem. Nesta área existem dois tipos de protocolos para cifrar mensagens: a criptografia Simétrica e a Assimétrica.

Na criptografia simétrica, uma única chave é utilizada por ambas as partes, para cifrar e decifrar a mensagem. Na criptografia assimétrica, são utilizadas duas chaves distintas, uma pública e uma privada. Ambas podem ser usadas para cifrar, porém o que uma cifrar, somente poderá ser decifrado pela outra. Devido às dificuldades de se utilizar este protocolo ele é mais utilizado para certificações digitais.

2.2 Função Hash - SHA

Esta função é amplamente utilizada para garantia de consistência tanto em criptografia quanto em várias outras áreas. Ela funciona embaralhando os bits de uma mensagem à um ponto que seja impossível inverter o processo. Ela aceita uma mensagem de qualquer tamanho, porém sempre retornará uma cadeia de bytes de tamanho fixo, não importando qual seja a entrada. Porém a maior característica desta função está no fato de que para duas entradas distintas x e y quaisquer, o resultado da função sempre será diferente para as duas. Desta forma, é possível se confirmar a integridade de uma mensagem, aplicando a função hash sobre ela antes de ser enviada, e depois de recebida, comparando a hash da mensagem com a hash recebida, pois se a mensagem foi alterada, por menor que seja a alteração, os dois resultados da função serão completamente diferentes. O algoritmo mais utilizado para esta função é o SHA 2 - Secure Hash Algorithm 2(Algoritmo de Hash Seguro 2), projetado pela NSA, e possui seis versões diferentes, onde a mudança principal está no número de bits que eles retornam: SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256.

2.3 Assinaturas Digitais

A confiança entre duas partes de uma troca de mensagens pode não ser suficiente para que informação sensível seja transferida. Uma parte “A” pode forjar uma mensagem alegando ter sido enviada pela parte “B”, ou “B” pode simplesmente negar ter enviado qualquer mensagem, mesmo que o tenha. Desta forma, um meio de proteger âmbas as partes contra fraude mútua são as assinaturas digitais.

Uma assinatura digital consiste normalmente da própria mensagem ”M”, passada pela função hash e cifrada com a chave privada do remetente, concatenada ao fim da mensagem. Ao ser recebida esta assinatura pode ser verificada, removendo-a da mensagem, decifrando-a com a chave pública do remetente e comparando-a com a hash da mensagem recebida. Sendo assim duas afirmações podem ser concluídas: apenas o remetente pode ter enviado a mensagem, pois apenas a sua chave privada pode ser decifrada com sua chave pública, e a mensagem não foi alterada durante seu trajeto, pois as hash’s coicidem.

2.4 Certificados Digitais

Certificados digitais são documentos eletrônicos que garantem a autenticidade de um certo elemento. Este que pode ser uma entidade, um site da internet ou um terminal de cartões. Um certificado é composto de algumas informações da entidade, a chave pública da entidade, e uma assinatura digital sobre o certificado. Dependendo do tipo de certificado, esta assinatura adicionada ao final do documento é cifrada ou com a chave privada de quem o criou, denominado de certificado auto-assinado, ou com a chave privada de uma autoridade certificadora (CA - Certificate Authority).

2.5 O Documento ICAO 9303

O documento ICAO 9303 é o documento emitido que regulamenta passaportes eletrônicos, e é o padrão utilizado para toda formatação de passaportes eletrônicos. ICAO é uma sigla para “International Civil Aviation Organization”, que é uma agência especializada das Nações Unidas. Foi estabelecida

em 1944 para gerenciar a admnistração e governância da Convenção de Aviação Civil Internacional (Convenção de Chicago).

O documento é escrito em 12 partes, cada uma descreve um aspecto sobre como um passaporte eletrônico deve ser desenvolvido, a primeira parte dando uma introdução sobre o as características de um passaporte eletrônico, e definindo siglas e acrônimos para os documentos seguintes.

A segunda parte da especificações sobre a segurança física do cartão, desde o design interno, quanto a produção, transporte e criação do cartão.

A terceira parte especifica as características sobre a apresentação de todos os tipos de passaporte, como fonte, linguagem, campos de preenchimento e representação das informações.

A quarta parte específica como os dados para passaportes legíveis por máquina, que possuem forma de livreto como são hoje, mostrando dimensões e formatos dos campos, também chamado de tipo TD3.

A quinta parte especifica como deve ser o layout dos dados para passaportes do tipo cartão plástico impresso, chamado TD1. Este cartão possui uma zona legível por máquina (MRZ), que possui os dados básicos de seu portador, formatados para mais fácil leitura humana e por máquina, impresso em seu exterior junto com todas as outras informações no verso, e uma imagem do portador.

A sexta parte se assemelha à quinta, pois descreve as características de cartões do tipo TD2, que possuem de diferente do tipo TD1 apenas suas dimensões, tendo todas as informações necessárias em um lado do cartão, permitindo informações opcionais serem adicionadas ao verso.

A sétima parte se refere à vistos legíveis por máquina. Estes se assemelham aos tipos de passaportes TD1 e TD2, porém possuem as informações referentes aos vistos.

A oitava parte não foi preenchida ainda, e é mantida reservada para uso futuro.

A nona parte fala sobre como se deve dispor das informações biométricas do portador do passaporte, formatos e dados, tomando como principal a identificação facial da pessoa, e como identificações secundárias as digitais e íris.

A décima parte especifica a estrutura logica de dados (LDS) do cartão, e como se deve armazenar os dados biométricos dentro do cartão.

A décima primeira parte especifica como funcionam os protocolos de segurança necessários à comunicação do cartão e obtenção dos dados biométricos para verificação da identidade da pessoa.

Por último a décima segunda parte especifica o funcionamento de todas as assinaturas digitais necessárias à certificação do cartão via chaves públicas, e sua infra-estrutura.

2.6 Javacard

O Javacard é um cartão eletrônico que possui dois componentes principais, Um chip de contato e um processador interno. Este chip de contato se faz presente do lado externo do cartão, da mesma forma que cartões de crédito ou débito modernos, e cuida da comunicação do cartão com a leitora. O processador interno funciona como um computador, ele roda um sistema operacional Java, o que torna este cartão um Javacard. Ele também possui uma memória interna capaz de armazenar dados. O que diferencia este cartão de cartões de crédito por exemplo, é o fato de o cartão possuir um máquina virtual Java (JVM) instalada. A JVM faz possível o cartão possuir aplicativos em sua memória, conhecidos como applets, e dependendo do programa que os acessar, pode escolher qual deles executar. Estes applets são diferentes dos aplicativos Java desenvolvidos para computador, pois devem levar em conta as limitações de processamento e memória dos cartões. Para tal, existe uma biblioteca de desenvolvimento independente conhecida como Java Card Development Kit oferecida pela empresa Oracle, que também disponibiliza o kit de desenvolvimento java convencional. Dentro do cartão, os applets conversam com o terminal por meio de mensagens, chamadas APDU's, que são em sua maioria padronizadas em formatos específicos. O Chip de contato fornece energia e faz a comunicação entre a leitora ou terminal e o processador. Estes cartões possuem diversos mecanismos de segurança embutidos na própria JVM, por exemplo, os aplicativos rodam isolados uns dos outros, portanto os dados de um aplicativo nunca poderão serem lidos por outro. Os cartões também suportam funções de criptografia, que dependem dos aplicativos para serem utilizadas.

2.7 O applet e sua estrutura lógica de dados

A biblioteca JMRTD possui consigo um applet a ser instalado e executado no cartão para o funcionamento da biblioteca. Este applet foi feito juntamente com a biblioteca no padrão ICAO9303, portanto ele pode ser lido inclusive por outras bibliotecas que implementem o padrão. Este applet tem

como função armazenar todas as informações referentes a um passaporte, e portanto possui uma estrutura de dados lógica interna descrita pelo padrão ICAO9303:

DG 1 — Informação da Zona Legível por Máquina
DG 2 — Características Faciais (Foto)
DG 3 — Informações de identificação adicionais (Digitais)*
DG 4 — Informações de identificação adicionais (Iris)*
DG 5 — Fotografia impressa na frente do cartão*
DG 6 — Reservado para uso futuro*
DG 7 — Assinatura escrita digitalizada*
DG 8 — Características de dados* **
DG 9 — Características estruturais* **
DG 10 — Características substanciais*
DG 11 — Detalhes pessoais adicionais*
DG 12 — Detalhes do documento adicionais*
DG 13 — Detalhes opcionais*
DG 14 — Informações da chave pública para Autenticação Passiva***
DG 15 — Informações da chave pública para Autenticação Ativa***
DG 16 — Pessoas para contato*

* - Opcional
** - Ainda não definido, estrutura geral que acomoda qualquer tipo de dados
*** - Condicional, apenas se suportado

Além destes existem mais dois arquivos que não armazenam dados do usuário mas sim dados do cartão, estes são: COM e SOD.

O arquivo COM, ou Arquivo de Cabeçalho e de Presença de Grupos de Dados, possui a função de armazenar a presença dos arquivos de dados, ou seja, quais arquivos estão presentes no cartão, e informações de versionamento do sistema de arquivos do cartão.

O arquivo SOD, ou Document Security Object - Objeto de Segurança do Documento, possui a função de armazenar o resultado da função hash de cada arquivo, ou seja, no processo de emissão do cartão, cada arquivo é passado também pela função hash, e seu resultado é armazenado dentro do arquivo SOD. Ele também armazena todas as informações de segurança do cartão como os algoritmos utilizados para a função hash e criptografia.

O cartão possui também uma zona que é ilegível por meios externos, informações sensíveis a criptografia como chaves privadas são armazenadas ali na sua criação, e depois só podem ser lidas e utilizadas pelo próprio cartão.

2.8 As bibliotecas utilizadas

O sistema principal será desenvolvido na linguagem de programação Java, porém algumas das bibliotecas escolhidas foram escritas na linguagem de programação C ou C++ exigindo a utilização da Interface Nativa Java (JNI) para sua utilização.

JMRTD

Java Machine Readable Travel Documents: é uma biblioteca para a criação, edição, e aferição de passaportes eletônicos escrita na linguagem de programação Java. Ela foi desenvolvida juntamente com a biblioteca Scuba, que a complementa. Ela possui a capacidade de criar, e editar novos passaportes, além que resgatar as informações de um passaporte pronto.

SCUBA

Smart Card Utilities for Better Access: é uma biblioteca para a comunicação com cartões eletrônicos da plataforma javacard, escrita na linguagem de programação Java. Ela faz a ponte entre o JMRTD e o javacard possuindo a capacidade de transmitir mensagens e prover a comunicação com os cartões.

STASM

Active Shape Models with STASM: é uma biblioteca de reconhecimento facial escrita na

linguagem de progrmação C++. Ela se utiliza de modelos de formas para reconhecer um rosto em uma foto e retirar os pontos de referência do rosto para que depois seja feito o reconhecimento da pessoa, fazendo uso da biblioteca Open-CV. Possui a maioria dos pontos de reconhecimento parecida com o padrão ISO(ISO/IEC: 2011), e foi escolhida por ser de fácil utilização e possuir uma extensa documentação com exemplos mínimos disponíveis, facilitando a sua compreensão, além de ser complementar à biblioteca Open-CV.

LIBFPRINT

É uma biblioteca de coleção e verificação de digitais biométricas escrita na linguagem de programação C. Ela provê funções para se coletar uma digital e guardá-la como uma imagem, e depois extrair as minutas desta imagem para que se compare com uma digital recém tirada. Foi escolhida por ter uma documentação extensa, exemplos mínimos e ter sido recomendada pelo orientador deste projeto.

OPEN-CV

Open Source Computer Vision Library: é uma biblioteca para manipulação de imagens e visão computacional escrita na linguagem de programação C++ e Java. Ela possui funcionalidades para conversão de imagens, e acessar a câmera do dispositivo para tirar fotos ou videos. Ela também provê visão computacional, permitindo identificar formas e rostos na imagem. A biblioteca Stasm amplia a funcionalidade para rostos, detectando mais pontos de interesse no rosto da pessoa, e foi escolhida por ser a mais bem recomendada biblioteca de manipulação de imagens e visão computacional, com relação a sua utilização.

EJBCA

Open Source PKI Certificate Authority: é uma biblioteca para criação e manejo de certificados digitais auto-assinados escrita na linguagem de programação Java. É necessária para o funcionamento da biblioteca JMRTD, para criação dos certificados verificáveis por cartão (CVC), necessários para a execução dos protocolos de segurança do cartão.

BOUNCYCASTLE

Bouncy Castle Security Provider: é um provedor de segurança, uma biblioteca que possui funções criptograficas e de geração e acordo de chaves para criptografia escrita na linguagem de programação Java. Toda a segurança da biblioteca do JMRTD é feita com ele, portanto por motivos de compatibilidade foi-se utilizado o bouncycastle como provedor para o sistema final.

2.9 Os protocolos de segurança

Para o acesso as informações do cartão diversos protocolos de segurança devem ser efetuados para garantir a troca segura de mensagens entre o terminal e o cartão, e ter a certeza de que o terminal e o cartão não foram adulterados de alguma maneira. Estes protocolos devem ser corretamente configurados durante a criação do cartão para que garantam a segurança das informações que nele estão guardadas. Todos os protocolos estão completamente descritos no documento ICAO 9303 parte 11

BAC - Basic Access Control - Controle de Acesso Básico

Antes de fazer qualquer tipo de operação no cartão, deve-se efetuar este protocolo. Ele utiliza as informações do número do documento, data de nascimento e de validade do cartão para criar chaves simétricas seguras para a troca de mensagens entre o terminal e o cartão criando assim um canal seguro de comunicação. Esta informação estará impressa na frente do cartão, e portanto a execução com sucesso deste protocolo não só garante um canal de comunicação seguro, mas também confirma que as informações impressas batem com as armazenadas no cartão.

PA - Passive Authentication - Autenticação Passiva

Este protocolo faz uso dos arquivos COM e SOD para seu funcionamento. O arquivo SOD armazena a hash de cada Grupo de Dados presente no cartão, e o arquivo COM indica sua presença. Desta forma o protocolo se inicia verificando se para cada arquivo cuja presença esteja indicada em COM, incluindo o próprio arquivo COM, se sua hash coincide com a hash armazenada em SOD. Se todas coincidirem então nenhum arquivo foi alterado desde a fabricação do cartão.

O segundo passo consiste em utilizar o certificado de assinatura encontrado no SOD, con-

struindo uma corrente de certificação até um certificado assinado por uma CA reconhecida, e garantindo que cada certificado da corrente seja válido. Passados por estes dois passos então se pode confirmar que os dados do cartão são válidos e não foram alterados.

AA - Active Authentication - Autenticação Ativa

Este protocolo deve ser feito após o PA, pois ele confirma que o SOD foi lido de um cartão com um chip válido. Ele consiste em uma troca de mensagens de desafio-resposta onde a aplicação envia uma mensagem ao cartão e este responde com esta mensagem cifrada pela chave privada do cartão. A aplicação então decifra utilizando a chave pública armazenada no arquivo DG15 e, caso a resposta coincida com a mensagem enviada, então o chip não foi adulterado, já que a chave privada interna do cartão, inacessível externamente, é o par da chave pública gravada no arquivo DG15.

EAC - Extended Access Control - Controle de Acesso Extendido

Este protocolo deve ser feito após o BAC e é necessário para se obter acesso as biometrias adicionais do cartão. Ele consiste em autenticar o terminal para o cartão, e o cartão para o terminal em duas etapas, utilizando-se de um par de chaves assimétricas. Ao término do protocolo um canal de comunicação de maior segurança entre o cartão e o terminal é gerado, e o acesso às biometrias adicionais é liberado.

3 Desenvolvimento - Reescrever?

O projeto foi desenvolvido tendo como base o aplicativo JMRTD(Oostdijk: 2010), que engloba uma Biblioteca Java e um Applet, para que seja criado um aplicativo que emita os passaportes, e que possa ser expandido caso haja a necessidade, para englobar mais documentos e não só o passaporte.

Foi feita uma extensa pesquisa sobre os documentos ICAO 9303(Organization: 2015), e a documentação da biblioteca JMRTD para se iniciar o projeto.

O programa foi desenvolvido com a linguagem de programação Java, utilizando o sistema JNI para a integração de bibliotecas que existem somente em C e C++, e as bibliotecas JMRTD, como a biblioteca principal para a construção do cartão, BouncyCastle(*Bouncy Castle*) e EJBCA(*EJBCA Open Source PKI Certificate Authority*) como provedores de segurança e certificação, libfprint(*LibFPrint*) para a coleta das digitais biométricas, Stasm(Milborrow and F.Nicolls: 2014) para o reconhecimento facial, e OpenCV(*OpenCV Library*) para processamento de imagem.

Utilizando uma parte do código da versão 0.4.9 da biblioteca JMRTD, é possível enviar informações para o cartão, desta forma pode-se fácilmente coletar e enviar as informações à respeito do MRZ. A biblioteca OpenCV foi usada para o processamento da foto da pessoa, e possui funções para tirar a foto com a camera da máquina. Com a biblioteca Stasm, se pode retirar os pontos faciais da pessoa, e depois tratá-los para se adequarem ao padrão ICAO. Foi necessário utilizar a API nativa do java para integrar as bibliotecas Stasm e libfprint ao projeto, pois elas existem apenas em C. Com a biblioteca libfprint foi possível a extração da digital para sua inserção.

Durante o desenvolvimento, alguns pontos chave tiveram uma quantidade de esforço maior para serem desenvolvidos. Para a captura de digitais, e leitura facial, as bibliotecas libfprint e Stasm, que são escritas em C e C++ respectivamente, tiveram de ser integradas ao aplicativo. Para isso, foi utilizada a API nativa da linguagem Java, chamada de JNI. Ela exige que seja construída uma biblioteca dinamica com as funções a serem utilizadas das bibliotecas em C/C++, para serem acessadas por uma classe dedicada a fazer a ligação do aplicativo em Java com o JNI.

Como não é possível a troca de mensagens entre a interface nativa (JNI) e a aplicação, quando é necessária a execução de uma função nativa não é possível, por exemplo, enviar um sinal a uma Thread Java avisando do início da captura da digital. Aparentemente, também, alguns tipos de Threads, como as de repintura de janelas, parecem parar de funcionar durante a execução da biblioteca em C.

Existem classes dedicadas à captura da foto e das digitais. Para a foto, utliza-se a biblioteca OpenCV para acessar a camera, em uma thread, que mostra o vídeo na tela. Quando se é tirada a foto, ela é primeiro codificada em jpeg com 100% de qualidade, e é passada para a biblioteca Stasm para a retirada dos pontos de reconhecimento facial. Depois, se codifica novamente a foto original em jpeg mas desta vez com 50% da qualidade, após ter sido escalada em 50% nos dois eixos cartesianos para que esta seja enviada ao cartão. Isto efetivamente reduz o tamanho da imagem enviada a pelo menos um quarto do seu tamanho, o que garante que sobre espaço de armazenamento para as outras imagens.

Para a coleta das digitais o processo é semelhante porém desta vez é utilizada a biblioteca libfprint, uma biblioteca escrita em C. Esta teve de ser um pouco modificada, para que duas funções

internas da API ficassem visíveis externamente antes que a biblioteca fosse montada. Assim, quando a biblioteca é chamada para escanear uma digital, ela acessa as portas seriais a procura de algum hardware de coleta, caso encontre, o abre e emite uma mensagem para que a digital seja escaneada. Após sua aquisição, a biblioteca gera uma imagem em formato .pgm, que é então escalada em 50% nos dois eixos cartesianos e codificada em formato jpeg com também 50% da qualidade. Com estes valores, ainda se torna possível a validação das digitais atravéz da própria biblioteca libfprint, e se reduz o tmanho das imagens obtidas a pelo menos um quarto do tamanho original, garantindo que todas as digitais de uma mão possam ser enviadas para dentro do cartão, sem ocorrerem problemas de espaço em memória.

Com estes parametros, e utilizando uma webcam "LG-Webpro2" com resolução de 640x420 pixels, e o aparelho de coleta de digitais integrado a um notebook "Dell Vostro 4600" com resolução média de 160x500 pixels[1], sem se modificar as imagens, o tamanho da foto facial original, em média era de 20Kb, e de cada digital, 13Kb. Com 10 digitais, e a foto do portador, o espaço necessário para o armazenamento no cartão é de 150Kb, desconsiderando o resto das informações necessárias para as outras funções, o que é inviável, já que o modelo com mais memória interna de javacard disponível no mercado, NXP J3A081 80K, possui 80Kb de memória. Com a manipulação das imagens, o tamanho da foto caiu para 5.5Kb, e cada digital em média caiu para 4.5kb, deixando assim um tamanho total de imagens armazenadas de aproximadamente 51Kb, deixando ainda 29Kb de espaço para todas as outras informações necessárias[2]

Para a implementação do protocolo de segurança de Autenticação Ativa, a classe de personalização de cartões da versão 4.9 do JMRTD já possui um método que envia as informações necessárias para o cartão, porém se estava encontrando dificuldades para enviar a chave privada do cartão, utilizada neste protocolo. Depois de analisar tanto o método da biblioteca quanto as funções efetuadas pelo cartão ao receber tal instrução, percebeu-se uma discrepância entre os dados enviados e os dados lidos. A biblioteca enviava um buffer contendo dois identificadores, identificando os dados e o seu tipo, seguidos pela informação, porém o cartão fazia uma série de leituras a mais, lendo uma tag, depois um valor de tamanho, pulando estes bytes lidos, lendo uma nova tag, e após lendo o conteúdo do buffer, o que fazia o cartão ler a primeira tag, tomar a segunda como tamanho de dados a serem pulados, pular um número de bytes, ler uma nova tag que agora faz parte da informação, e tomar um valor errôneo como tamanho da informação, o que causava uma excessão pois quase sempre o tamanho da informação restante era menor que o tamanho lido. Como no aplicativo do cartão, estas tags não eram conferidas, retirar as primeiras três ações, ler tag, ler tamanho e pular, resultou no funcionamento correto do envio da chave, e na correta configuração do protocolo.

3.1 Funcionamento

O aplicativo desenvolvido possui duas funções, coletar as informações do usuário e com ela gerar um novo passaporte válido, e verificar se as informações coletadas estão devidamente inseridas no cartão. A parte da verificação de identidade do proprietário não faz jús ao escopo deste trabalho. O aplicativo possui as funções de retirar uma foto da pessoa, e de coletar todas as suas digitais. O aplicativo também necessita de certificados digitais e pares de chaves para que os protocolos de segurança possam ser executados. *Expandir*

4

5 Referências

Bouncy Castle, The Legion of the. *Bouncy Castle*. URL: `http://www.bouncycastle.org/java.html`.

Bozhanov, Bozhidar (2016). *epassport Reader*. URL: `https://github.com/Glamdring/epassport-reader`.

HomeOffice (2013). *How your passport is made – exclusive behind-the-scenes footage*. Youtube. URL: `https://www.youtube.com/watch?v=Ha5VPXZ3ILs`.

[1] Sendo este um leitor de corrida, ou seja, para a captura é necessário correr o dedo pela faixa leitora, a resolução no eixo y se torna variável, de acordo com o quanto do dedo se correu pela leitora. Diversas passagens em diferentes dedos obtiveram assim uma média de 500 pixels de resolução no eixo y

[2] Foram-se escolhidos todos os valores para manipulação de imagem de forma a manter a possibilidade de análise a olho nu da foto da pessoa, e manter a verificabilidade de cada digital.

Houzard, Jean-Francois and Olivier Roger (2009). *PyPassport, Python library to read the biometric ePassport*. URL: https://code.google.com/archive/p/pypassport/.
InnoValor (2013). *ReadID*. URL: https://www.readid.com/.
ISO/IEC (2011). *ISO/IEC 19794-5 Biometrics*. Internation Organization of Standardization. URL: https://www.iso.org/standard/50867.html.
Laurie, Adam (2011). *RFIDIOt.py - RFID IO tools for python*. URL: https://github.com/AdamLaurie/RFIDIOt/,http://www.rfidiot.org/.
Milborrow, S. and F.Nicolls (2014). "Active Shape Models with SIFT Descriptors and MARS". In: *VISAPP*. URL: http://www.milbo.users.sonic.net/stasm/.
Oostdijk, Martin (2010). *JMRTD*. URL: https://www.jmrtd.org.
Organization, International Civil Aviation (2015). *ICAO 9303: Machine Readable Travel Documents*. ICAO. URL: https://www.icao.int/publications/pages/publication.aspx?docnum=9303.
PrimeKey. *EJBCA Open Source PKI Certificate Authority*. URL: https://www.ejbca.org.
Tananaev, Anton (2016). *e-Passport NFC Reader*. URL: https://github.com/tananaev/passport-reader.
Team, LibFPrint. *LibFPrint*. URL: https://www.freedesktop.org/wiki/Software/fprint/libfprint/.
Team, OpenCV. *OpenCV Library*. URL: https://opencv.org/.
waazaa.org and ariadNEXT (2011). *wzMRTD*. "Site original http://waazaa.org/wzmrtd não existe mais na data em que este documento foi escrito". URL: https://github.com/ariadnext/wzmrtd.
Wikipedia. *Passport*. Wikipedia. URL: https://en.wikipedia.org/wiki/Passport.