UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

DESENVOLVIMENTO DE UM SISTEMA DE TRANSMISSÃO DE IMAGENS MÉDICAS DIGITAIS CIFRADAS UTILIZANDO O PROTOCOLO DICOM

FRANCISCO MARCELO MARQUES LIMA

ORIENTADORA: JULIANA FERNANDES CAMAPUM
DISSERTAÇÃO DE MESTRADO EM ENGENHARIA ELÉTRICA

PUBLICAÇÃO: PPGENE.DM-370/09
BRASÍLIA/DF: 02 – 2009

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

DESENVOLVIMENTO DE UM SISTEMA DE TRANSMISSÃO DE IMAGENS MÉDICAS DIGITAIS CIFRADAS UTILIZANDO O PROTOCOLO DICOM

FRANCISCO MARCELO MARQUES LIMA

DISSERTAÇÃO DE MESTRADO ACADÊMICO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA, COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE.

APROVADA POR:

_____
JULIANA FERNANDES CAMAPUM, DRª., ENE/UNB
(ORIENTADORA)

_____
JANAÍNA GONÇALVES GUIMARÃES, DRª., ENE/UNB
(EXAMINADORA INTERNA)

_____
PEDRO DE AZEVEDO BERGER, DR., CIC/UNB
(EXAMINADOR EXTERNO)

BRASÍLIA, 10 DE FEVEREIRO DE 2009

FICHA CATALOGRÁFICA

LIMA, FRANCISCO MARCELO MARQUES	
Desenvolvimento de um Sistema de Transmissão de Imagens Médicas Digitais Cifradas Utilizando o Protocolo DICOM [Distrito Federal] 2009	
xvii, 156p., 210 x 297mm (ENE/FT/UNB, Mestre, Dissertação de Mestrado – Universidade de Brasília. Faculdade de Tecnologia.	
Departamento de Engenharia Elétrica	
1. Dicom	2. PACS
3. TLS	4. Segurança
5. Imagens Médicas	
I. ENE/FT/UnB	II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

LIMA, FRANCISCO M. M. (2009). Desenvolvimento de um Sistema de Transmissão de Imagens Médicas Digitais Cifradas Utilizando o Protocolo DICOM. Dissertação de Mestrado em Engenharia Elétrica, Publicação PPGENE.DM-370/09, Departamento de Engenharia Elétrica, Universidade de Brasília, DF, 156p.

CESSÃO DE DIREITOS

AUTOR: Francisco Marcelo Marques Lima

TÍTULO: Desenvolvimento de um Sistema de Transmissão de Imagens Médicas Digitais Cifradas Utilizando o Protocolo DICOM.

GRAU: Mestre ANO: 2009

_____

Francisco Marcelo Marques Lima
QI 05 Bloco G Apartamento 202
71.020-074 – Guará I – Distrito Federal - Brasil
marcelo.marques@incra.gov.br / marques.marcelo@gmail.com

AGRADECIMENTOS

Aos(as) professores(as) Fábio Caldas, Luciana Cavalcante, Irna Rocha e Sena Siqueira que muito contribuíram na fase de revisão deste texto.

A professora Dra. Juliana Fernandes Camapum, orientadora e incentivadora deste trabalho acadêmico.

De modo especial, não posso deixar de registrar minha gratidão a minha esposa Catia e filha Danielle, de cujo convívio furtei-me em tantos momentos especiais, sempre com a compreensão e apoio, que ajudaram a tornar este sonho possível.

RESUMO

DESENVOLVIMENTO DE UM SISTEMA DE TRANSMISSÃO DE IMAGENS MÉDICAS DIGITAIS CIFRADAS UTILIZANDO O PROTOCOLO DICOM.
Autor: Francisco Marcelo Marques Lima.
Orientadora: Juliana Fernandes Camapum.
Programa de Pós-graduação em Engenharia Elétrica da Universidade de Brasília
Brasília, Fevereiro de 2009.

Palavras-chave: Dicom, PACS, TLS, Segurança, Imagens Médicas.

Interessada nas facilidades oferecidas por um sistema de arquivamento de imagens, a direção do Hospital Universitário de Brasília (HUB) decidiu implementar um servidor PACS que permitisse o recebimento e armazenamento de imagens por meio do protocolo DICOM de forma a tornar possível a visualização de imagens e confecção de laudos médicos através de uma interface Web. Como solução para este problema, foi utilizado o servidor PACS proposto pelo INCOR-SP conhecido como miniWebPACS. Em agosto de 2007, o autor dessa dissertação foi convidado pela orientadora para avaliar a segurança da aplicação miniWebPACS em uso no Hospital Universitário de Brasília. Após avaliar o código fonte da aplicação, foram encontradas duas falhas de segurança. A primeira falha permite que um invasor mal intencionado possa, através de ferramentas de captura de tráfego de rede, identificar os parâmetros de autenticação utilizados pelo protocolo DICOM e transmitir imagens médicas de um paciente, se passando pelo equipamento médico. A segunda refere-se à possibilidade de remontar a imagem médica capturando todos os pacotes de rede transmitidos entre o equipamento médico e o servidor, destruindo, desta forma, o sigilo das informações do paciente. As falhas de segurança eram extremamente graves e, se exploradas, poderiam denegrir a imagem do HUB e de toda sua equipe médica. A simples troca do servidor miniWebPACS por uma outra versão era inviável. O HUB já havia desenvolvido várias soluções para manipulação de dados médicos utilizando o atual servidor. O projeto hub2pacs nasceu com o objetivo de desenvolver um novo servidor PACS para a transmissão de imagens médicas de forma segura, utilizando como referência a biblioteca livre dcm4che e respeitando o modelo de dados em uso no Hospital Universitário de Brasília. O hub2pacs resolveu, definitivamente, o problema de compatibilidade das aplicações antigas já desenvolvidas além de oferecer garantia do sigilo do canal de comunicação e a autenticação das partes, porém, a análise do ambiente demonstrou a necessidade de permitir que equipamentos médicos sem suporte ao DICOM/TLS pudessem transferir imagens de forma segura. Para resolver este problema foi desenvolvido um novo projeto chamado hub2conversor.

ABSTRACT

DEVELOPMENT OF A SYSTEM FOR CRYPTOGRAPHY TRANSMISSION OF MEDICAL IMAGES USING DICOM PROTOCOL.
Author: Francisco Marques Marcelo Lima.
Advisor: Juliana Fernandes Camapum.
Postgraduate Program in Electrical Engineering of University of Brasília (UNB).
Brasília, February 2009.

With the facilities that an image archiving system offers, the HUB – Hospital Universitário de Brasília (Brasilia University Hospital) decided to implement a PACS server that allowed it to receive and store images using the DICOM protocol. This, in turn, allowed it to view images and make medical diagnoses using a Web interface. As a solution to this problem, a PACS server suggested by INCOR-SP named "miniWebPACS" was used. In August 2007, the author of this dissertation was invited by his Thesis Orientation Professor to evaluate the security of the "miniWebPACS" application at use at the HUB. After evaluating the source code of this application, security failures were found. The first failure permitted an ill-meaning invader to, through network traffic capture tools, identify the parameters through which the DICOM protocol authenticates, thus permitting illicit medical images to be sent in the name of a licit networked piece of medical equipment. The second failure is the possibility of reassembling the medical image by capturing all the network packets transmitted between the medical equipment and the server, therefore destroying the patients' information's confidentiality. The security failures were extremely serious, and, if exploited, could damage the HUB's and its doctors' images. The simple substitution of the "miniWebPACS" server for another, more secure, server was not viable, given that the HUB had already developed several solutions for manipulating medical data using the current server. Thus, the "hub2pacs" project was born with the objective of developing a PACS server for the transmission of medical images in a more secure manner, using as a reference the "dcm4che" free development software library, and preserving the data model at use at the HUB. The "Hub2pacs" solved, for once and for all, not only the compatibility issues with the old applications that had already been developed, but also offered a guarantee of confidentiality in the communications channel and the authentication of all the parts involved. However, an environmental analysis pointed out the necessity of allowing medical equipment without support for DICOM/TLS to transfer images securely too. To solve this problem a new project was started called "hub2conversor".

Key-words: Dicom, PACS, TLS, Security, Medical Images.

SUMÁRIO

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE ABREVIAÇÕES E SIGLAS

AC – Autoridade Certificadora

AE – Application Entity

AES – Advanced Encryption Standard

AL – Fluxos Alternativos

API – Application Programming Interface

AR – Autoridade De Registro

ARP – Address Resolution Protocol

ASCII - American Standard Code for Information Interchange

BIOS - Basic Input/Output System

BPCS – Bit-Plane Complexity Segmentation

CBC – Cipher Block Chaining

CFB – Cipher Feedback

CLI – Commons Language Interface

CORBA – Common Object Request Broker Architecture

CT - Computed Tomography

DAC – Discretionary Access Control

DES – Data Encryption Standard

DHCP - Dynamic Host Configuration Protocol

DICOM – Digital Imagin And Communications Association

DIMSE - DICOM Message Service Element

DNS – Domain Name System

DOS – Denied Of Service

EDE – Encrypt-Decrypt-Encrypt

EPR – Eletronic Patients Records

ERP – Enterprise Resource Planning

EX – Fluxos De Exceções

FB – Fluxo Básico

GUT – Gravidade, Urgencia e Tendência

HCO – Health Care Organization

HIPAA – Health Insurane Portability and Accountability Act

HIS – Hospital Information System

HUB – Hospital Universitário De Brasília

IEC – International Electrotechnical Commission

INCOR – Instituto do Coração

IOD - Information Object Definition

IP – Internet Protocol

ISO – Internacional Organization For Standardization

J2EE – Java2 Platform Enterprise Edition

JASS – Java Authentication and Authorization Service

JNI – Java Native Interface

JPEG - Joint Photographic Experts Group

JSSE – Java Secure Socket Extension

LDAP - Lightweight Directory Access Protocol

LSB – Last Significant Bit

MAC – Message Authentication Code

MD5 - Message-Digest Algorithm 5

MR - Magnetic Resonance

NEMA – National Electrical Manufactures Association

OID – Object Identifiers

OPENSSL – Open Secure Sochet Layer

PACS – Picture Archiving and Communication System

PDU – Protocol Data Unit

PDV – Protocol Data Value

PKI – Public Key Infrastructure

PSNR – Peak Signal to Noise Ratio

RBAC – Role Based Controls

RIS – Radiology Information System

RMI – Remote Method Invocation

RMI-IIOP – Java Remote Method Invocation about Internet Inter-Org Protocol

RSA - Rivest Shamir e Adleman

RUP – Rational Unified Process

SCP - Service Class Provider

SCU – Service Class User

SDK - Standard Development Kit

SGBD – Sistema Gerenciamento de Banco De Dados

SOAP – Simple Object Access

SOP – Service Object Pair

SQL – Structured Query Language

SSL – Secure Sochet Layer

TCP/IP - Transmission Control Protocol/Internet Protocol

TLS – Transport Layer Security

UID – Unique Identifier

UML – Unified Modeling Language

URL - Uniform Resource Locator

WAN - Wide Area Network

WINS – Windows Name System

XML – Extensible Markup Language

1 INTRODUÇÃO

O constante avanço da tecnologia da informação tem permitido à sociedade a implementação e o desenvolvimento de novos serviços, buscando sempre a facilidade e qualidade dos resultados auferidos. A Internet, como ferramenta de comunicação, coloca à disposição das pessoas inúmeras facilidades, possibilitando a comunicação, a troca de informações e o acesso a uma quantidade de dados a uma velocidade nunca antes imaginada.

As últimas décadas foram marcadas pela integração dos computadores no mundo e na vida das pessoas. Para isto, foram criados serviços baseados em nosso modelo social, o que permitiu a conexão entre redes financeiras, supermercados, lojas, clínicas, postos de saúde e até mesmo hospitais. Hoje, é possível comprar qualquer objeto pela rede de computadores, manipular informações bancárias, acessar diversos bancos de dados que contêm grande parte do conhecimento humano e, se necessário, anunciar um produto pessoal em um site de trocas. Entre as últimas novidades deste mundo virtual, aparecem os laboratórios médicos que disponibilizam acesso a resultados de exames de seus pacientes pela Internet, diminuindo, de forma significativa, o tempo de espera e removendo a necessidade de retorno para buscar um "simples papel".

Embora a entrega de resultados traga benefícios diretos aos pacientes e laboratórios, questões como a transmissão, manipulação e armazenamento de resultados, imagens e laudos são levantadas como possíveis vulnerabilidades que, se exploradas, podem provocar grandes perdas financeiras às organizações que fazem uso deste tipo de tecnologia. Muitos pesquisadores e empresas têm investido tempo e dinheiro em soluções que permitam manipular exames e resultados médicos de forma segura, garantindo ao paciente o direito ao sigilo de suas informações e histórico médico.

Em 1983, com o objetivo de unificar o processo de transmissão de imagens médicas entre equipamentos de diferentes fabricantes, a Faculdade Americana de Radiologia (*American College of Radiology*) em conjunto com a Associação Nacional de Fabricantes Elétricos (*National Electrical Manufacturers Association*) criaram o protocolo DICOM (*Digital Imaging and Communications in Medicine*). O principal objetivo desse protocolo é propor um formato único para transmissão e armazenamento dos arquivos de imagens médicas e

permitir, assim, a interoperabilidade entre equipamentos de diferentes fabricantes facilitando o desenvolvimento de aplicações para clínicas e hospitais [1].

Não demorou muito para aparecer os primeiros sistemas de comunicação e arquivamento de imagens médicas integrando diferentes dispositivos de aquisição e apresentação de imagens, além de outros sistemas de informações médicas tais como: Sistema de Informação Radiológica (*Radiology Information System*) e Sistema de Informação Hospitalar (*Hospital Information System*). O desenvolvimento desses softwares, conhecidos como PACS (*Picture Archiving and Communication System*), permite, por exemplo, que um médico, localizado em qualquer parte do mundo avalie, "on-line", o exame que acabou de ser realizado ou, se necessário, compare exames antigos realizados pelo paciente ou por outras pessoas. Esse recurso aumenta, de forma significativa, a qualidade do atendimento e a certeza do diagnóstico [2].

Com o objetivo de exemplificar o funcionamento básico desse tipo de sistema, a figura 1.1 apresenta, graficamente, o ambiente DICOM-PACS. De forma simplificada, um servidor permite o recebimento, armazenamento e visualização de diversos tipos de imagens geradas por equipamentos específicos, tais como: CT, MR, US e X-ray. Cada função (Modalidades de Imagem, Arquivamento PACS e Aplicativo de Visualização) deve possuir um aplicativo com capacidade de manipular o protocolo DICOM e fornecer interfaces intuitivas para médicos e pacientes [2].

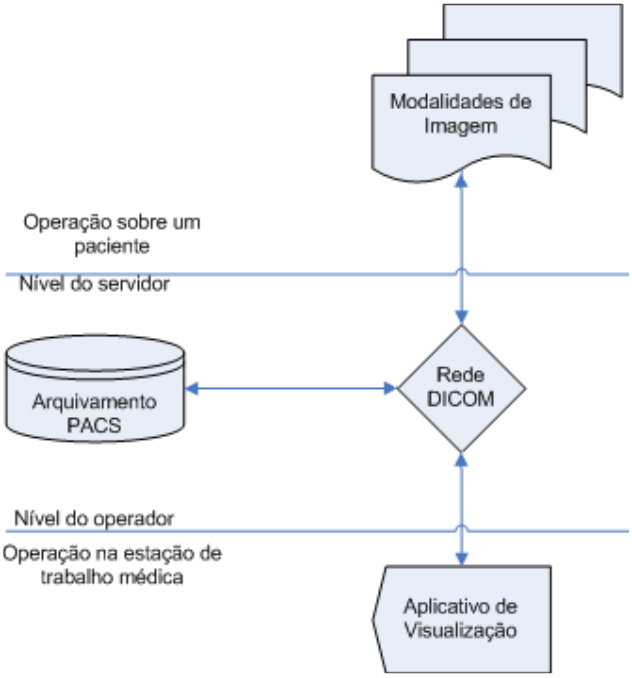

Figura 1.1 - Ambiente DICOM-PACS [2].

Interessada nas facilidades oferecidas por um sistema de arquivamento de imagens, a direção do Hospital Universitário de Brasília (HUB) decidiu iniciar estudos para a implementação de um servidor PACS que possibilitasse o recebimento e armazenamento de imagens por meio do protocolo DICOM de forma a permitir a visualização de imagens e confecção de laudos médicos através de uma interface Web. Por se tratar de um problema complexo, este projeto foi dividido e construído em dois produtos diferentes:

O primeiro é referente ao recebimento e armazenamento de imagens. Para esse fim, foi adotado o uso de um software gratuito desenvolvido pelo INCOR - SP, chamado MiniWebPACS. Esse aplicativo foi desenvolvido com a utização da linguagem de programação JAVA, a biblioteca pública dcm4che e o banco de dados Postgres [3].

O segundo projeto foi a construção de um módulo de visualização de imagens, permissão de acesso e confecção de laudos médicos por meio de uma interface web. Esse software é conhecido como GsWeb e foi desenvolvido por alunos de mestrado da UnB. Ele é um produto que, atualmente, permite acesso aos dados textuais e às imagens médicas, dando condições para a manipulação dessas informações para geração de tabelas e gráficos, bem como a geração de laudos automáticos provisórios, os quais são apresentados aos profissionais de saúde e auxiliam na emissão do laudo definitivo.

1.1 OBJETIVO GERAL

O objetivo desta pesquisa é desenvolver um servidor PACS para a transmissão de imagens médicas de forma segura, utilizando como referência a biblioteca livre DCM4CHE e respeitando o modelo de dados em uso no Hospital Universitário de Brasília. Este servidor deve permitir o uso de certificados digitais e cifrar todo o processo de transmissão de imagens entre equipamento médico e servidor.

1.2 OBJETIVOS ESPECÍFICOS

Os objetivos específicos da pesquisa visam:

- Estudar os principais conceitos sobre segurança da informação;
- Estudar as principais características para o desenvolvimento de sistemas seguros;
- Entender e fazer uso de mecanismos de criptografia simétrica, assimétrica e infra-estrutura de chaves públicas;
- Entender o funcionamento do protocolo DICOM e sua convergência com o uso de certificados digitais;
- Desenvolver um novo servidor PACS para o Hospital Universitário de Brasília que permita o recimento de objetos utilizando certificados digitais e o protocolo TLS (*Transport Layer Security*); e
- Desenvolver um conversor DICOM que possibilite equipamentos médicos sem suporte a TLS transmitir imagens de forma cifrada.

1.3 MOTIVAÇÃO

A motivação deste trabalho surgiu quando o autor foi convidado para avaliar o nível de segurança do servidor miniWebPACS em uso no HUB. Após a avaliação do código fonte da aplicação miniWebPACS e estudos sobre o funcionamento do protocolo DICOM, foram descobertas duas vulnerabilidades na forma como o MiniWebPACS foi implementado.

A primeira falha de segurança permite que um invasor mal intencionado possa, através de ferramentas de captura de tráfego de rede, identificar os parâmetros de autenticação utilizados pelo protocolo e transmitir imagens médicas de um paciente, se passando pelo equipamento médico. A segunda refere-se à possibilidade de remontar a imagem médica transmitida, capturando todos os pacotes de rede transmitidos entre o equipamento médico e o servidor destruindo, desta forma, o sigilo das informações do paciente.

Durante a fase de pesquisa foi encontrado um servidor PACS gratuito chamado dcm4chee. O dcm4chee implementa o uso de certificados digitais, mas não armazena as informações sobre a imagem médica em um servidor de banco de dados no formato suportado pelo aplicativo GSWeb. Tal limitação inviabilizava seu uso, pois obrigava a adaptação do GSWeb e de todas as aplicações já desenvolvidas pelo HUB ao novo formato de dados proposto pelo dcm4chee.

Para solucionar estas vulnerabilidades e resolver o problema de compatibilidade do dcm4chee, esta pesquisa desenvolveu dois projetos:

- Um novo servidor PACS que permite a autenticação do equipamento médico e do servidor, utilizando certificados digitais, a cifração de todos os dados trafegados na rede e a compatibilidade com o miniWebPACS facilitando sua substituição; e

- Um conversor DICOM que permita ao equipamento médico em uso no HUB transmitir imagens, utilizando o protocolo DICOM-TLS.

1.4 METODOLOGIA UTILIZADA PARA AVALIAR A SEGURANÇA DA APLICAÇÃO MINIWEBPACS

A avaliação de segurança de uma determinada aplicação pode ser dividida em quatro grandes atividades:

- a seleção do alvo;
- a busca de informações;
- o levantamento de ferramentas e técnicas; e
- a realização do ataque.

1.4.1 Seleção do alvo

A seleção do alvo aconteceu em agosto de 2007 quando o autor dessa dissertação foi convidado pela orientadora para avaliar a segurança da aplicação miniWebPACS em uso Hospital Universitário de Brasília. Na primeira visita para o reconhecimento do ambiente, foram identificados os seguintes ativos na rede do HUB (figura 1.2): um servidor Windows 2003 rodando o banco de dados postgres versão 8.1, o Java SE Development Kit (JDK) versão 1.5, o servidor miniWebPACS 1.0, um switch e um equipamento médico de ultrason fabricado pela Philips, modelo Ultrasound - HDI 3500 System.

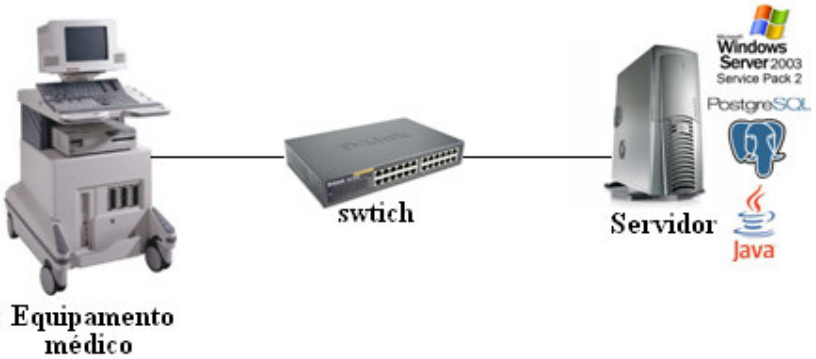

Figura 1.2- Ambiente atual do Hospital Universitário de Brasília.

1.4.2 Busca de informações

Buscar informações sobre um aplicativo é entender seu funcionamento de forma detalhada. A pesquisa bibliográfica inicial tinha como objetivo entender o funcionamento do padrão DICOM. Por meio dela foi possível identificar que o protocolo pode ser implementado utilizando o TLS (*Transport Layer Security*), ou seja, o desenvolvedor que criou o servidor PACS pode optar por ativar o uso de certificados digitais e de criptografia no canal de comunicação. Quando esta restrição de segurança não está ativada, toda a informação trocada entre cliente e servidor passa em texto claro, podendo ser capturada, remontada e interpretada por um invasor.

O servidor miniWebPACS é uma aplicação de código aberta e disponível para download pela Internet. Tal situação permitiu estudar, em laboratório, como a aplicação implementava o protocolo DICOM e avaliar as possíveis falhas de segurança. A análise do código fonte permitiu identificar três vulnerabilidades: as imagens são armazenadas no servidor DICOM sem ser cifradas, não existe nenhum tipo de criptografia durante a transmissão de dados e a autenticação do equipamento médico é baseada em dois campos *ae-called-title e ae-calling-title* que representam, respectivamente, o nome de identificação do servidor e o nome de identificação do cliente. As principais tabelas desse sistema são: *img_patientlevel* (contém as informações do paciente) e *img_imagelevel* (contém a referência para as imagens recebidas pelo servidor).

A pesquisa sobre o equipamento médico (figura 1.3) revelou que ele não oferece suporte ao protocolo DICOM-TLS. Desta forma, a única maneira de transmitir informação é se as funcionalidades do TLS forem desativadas. Assim, a solução a ser adotada precisa operar com essa limitação.

Figura 1.3 – Philips Ultrasound - HDI 3500 System.

Todos os equipamento são conectados através de um switch LAN (local area network) que implementa o uso da comutação de pacotes. Tal característica permite que a comunicação entre dois hosts em uma rede LAN possa ser realizada de forma independente do meio, ou seja, todos os pacotes trocados entre cliente e servidor permanecem em um circuito virtual não sendo possível, a partir de uma porta qualquer, capturar pacotes e remontar as informações. Na prática apenas os pacotes de broadcast são visualizados.

Como o objetivo do estudo era avaliar a segurança da aplicação miniWebPACS, não foram consideradas falhas de segurança referentes ao sistema operacional Windows 2003 e do servidor de banco de dados Postgres. Para garantir a segurança destas duas aplicações é necessário, resumidamente, realizar as seguintes tarefas: atualizações diárias, inserir configurações específicas recomendadas pelo fabricante e a instalação de um filtro de pacotes (*firewall*) de estação permitindo conexões apenas na porta do serviço PACS.

Também não foram avalidadas questões referentes à segurança física do ambiente. Nesse caso, a recomendação é evitar que pessoas estranhas tenham acesso ao servidor e/ou equipamento médico.

1.4.3 Levantamento de ferramentas e técnicas de ataque

Após estudar o ambiente, ficou claro que poderia ser realizada uma captura de tráfego de rede com o objetivo de identificar os campos de autenticação (*ae-called-title e ae-calling-title*) para transmitir imagens falsas, remontar a imagem a partir dos dados capturados e tentar substituir uma imagem válida por uma imagem inválida no servidor PACS.

Este tipo de captura possui dois grandes limitadores: exige que o invasor esteja conectado fisicamente no mesmo segmento de rede do servidor e equipamento médico; e exige que o switch não crie circuitos virtuais, pois esta característica não permite a captura de todos os pacotes trocados na rede.

A limitação física é facilmente vencida apenas utilizando portas do switch que não estavam em uso espalhadas pelo hospital. O problema de comutação de circuitos virtuais pode ser burlado com a utilização de duas técnicas de ataque ao switch: o *arp poisoning* e o *mac flooding*. É importante destacar que qualquer forma de ataque *man in the middle* (homem no meio) também poderia capturar todos os pacotes trafegados.

A técnica de *arp poisoning* explora o recurso de divulgação de endereços físicos (*mac addres*s) disponível no protocolo TCP/IP. Um *host* pode propagar seu endereço físico e endereço IP para outros *hosts,* utilizando um pacote ARP *broadcast*. Ao receber este pacote, todos os hosts conectados à rede irão armazenar o endereço físico e IP divulgado em sua tabela ARP evitando, desta forma, a realização de consultas ARP sempre que for iniciar uma nova transmissão. Um atacante pode enviar um pacote ARP *broadcast* filiando o seu endereço físico ao endereço IP do servidor válido. Sempre que um host desejar transmitir informações para o servidor, ele irá consultar sua tabela ARP e encontrar o endereço físico do atacante. Na prática, o atacante irá receber todas as informações enviadas pelo Cliente e ele poderá reencaminhá-las para o servidor válido, evitando a detecção do ataque.

Para que um *switch* possa realizar a criação de circuitos virtuais, é necessário que ele armazene todos os endereços físicos (*mac address*) de todos os hosts conectados à rede. Como esta tabela possui uma quantidade máxima de posições para armazenar endereços físicos, um atacante pode enviar vários registros de *broadcast* para um *switch* e provocar o

"estouro" da tabela ARP. Essa técnica é conhecida como *mac flooding* e, quando aplicada o *switch* irá propagar todos os pacotes da rede em todas as portas e o atacante poderá receber todas as informações trocadas entre equipamento médico e servidor.

Para realização do experimento, foram utilizados os programas abaixo relacionados. Ao lado de cada software foi inserida uma breve descrição cujo objetivo é demonstrar a utilidade de cada uma das ferramentas no teste de segurança.

- Servidor MiniWebPACS versão 1.0 – Embora possua as funcionalidades de servidor DICOM e cliente Web para visualização e digitação de Laudos, neste experimento foi utilizada apenas a função de servidor de imagens [3];

- Biblioteca DCM4CHE versão 1.4 – foi utilizado o sistema de envio de imagens DICOM "dcmsnd.jar" [4];

- Ethereal versão 0.99.0 – ferramenta utilizada para captura de tráfego de rede com o objetivo identificar o "ae-called-title" e "ae-calling-title" da imagem DICOM [5];

- Frhed versão 1.1.0 – editor hexadecimal utilizado para modificar o cabeçalho da imagem DICOM com o objetivo de modificar o "sopinsuid" [6];

- Photoshop 7.0 – ferramenta gráfica para edição de imagens. Foi utilizada para inserir, em uma imagem DICOM válida, características inexistentes [7];

- DICOMAccess versão 1.5.5 – *Plugin* que fornece ao Photoshop a capacidade de manipular imagens DICOM [8];

- Debian Sarge 3.12r Stable – Sistema operacional Linux utilizado para servir de plataforma operacional para o Servidor DICOM MiniWebPACS [9];

- Microsoft Windows XP Professional – Sistema operacional utilizado para servir de suporte à biblioteca DCM4CHE para envio de imagens [10];

- Java Sun SDK 1.5.6 – máquina virtual JAVA utilizada para executar o servidor MiniWebPACS e o DCM4CHE [11];

- PostgreSQL 7.4 Debian Sarge – serviço de banco de dados relacional utilizado para armazenar as características da imagem armazenada pelo servidor MiniWebPACS [12]; e

- EMS PostgreSQL – aplicação utilizada para gerenciar servidores de banco de dados relacionais utilizando a plataforma PostgreSQL [13].

1.4.4 Realizando o ataque - Inserindo Imagens Falsas

Foram montados dois ambientes distintos. Um servidor Debian que fornecerá suporte às aplicações do servidor DICOM MiniWebPACS e uma máquina Windows XP que será utilizada pelo atacante para promover a inserção de imagens falsas. A instalação inicial desses dois ambientes foi realizada de forma padrão.

No Servidor Debian foi instalado o postgreSQL através do repositório debian (ftp.br.debian.org), ou seja, foi executado na console o comando # apt-get install postgre.

Com o objetivo de permitir o acesso remoto à base de dados postgreSQL e modificar o esquema de autenticação local para trust, foi modificado no arquivo \etc\postgresql\pg_hba.conf a seguinte linha:

```
host all all 0.0.0.0 0.0.0.0 trust
```

Na máquina do atacante, foram instalados os softwares: EMS PostgreSQL, PhotoShop 7.0, DICOMAccess 1.5.5, Fhred, DCM4CHE 1.4 e ethereal 0.99.

O servidor MiniWebPACS foi instalado conforme manual disponibilizado no site do fabricante [3]. Nessa instalação são configurados dois parâmetros importantes para o funcionamento do servidor e do cliente DICOM, são os campos ae-called-title e ae-calling-

title. O primeiro parâmetro identifica o nome do servidor e o segundo, o nome atribuído ao equipamento de análise clínica.

Para que um atacante possa enviar uma imagem para o servidor ele deve, de alguma forma, descobrir esses dois parâmetros. Por definição, o MiniWebPACS não aceitará a transferência de imagens sem que estes parâmetros sejam enviados antes da transferência.

Uma forma simples de descobri-los é capturar o tráfego de comunicação entre o cliente e o servidor DICOM. Para isso, foi utilizado o software Ethereal e, sem muitas dificuldades, foram encontrados os campos ae-called-title e ae-calling-title, conforme mostra a figura 1.4, a seguir:

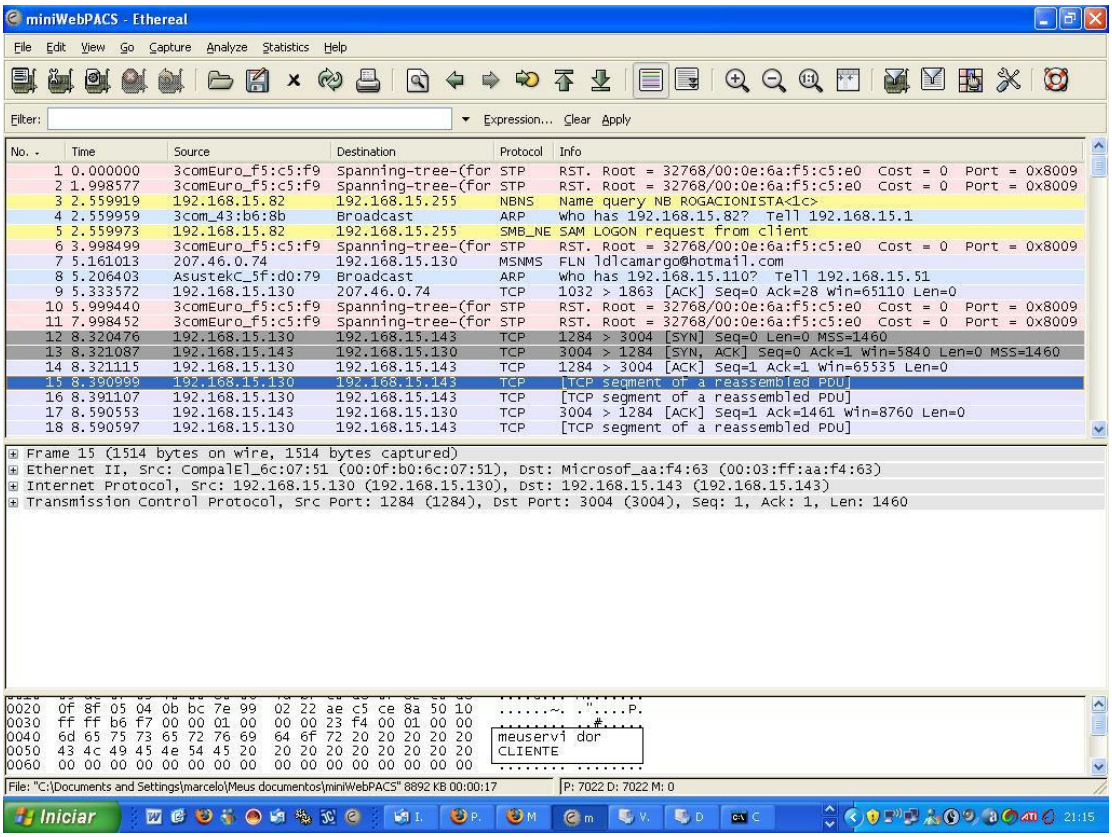

Figura 1.4 - Captura de tráfego de rede utilizando o ethereal.

De posse dessa informação, devemos então preparar uma imagem para transmiti-la para o servidor se passando pelo equipamento de análise clínica. Para cumprir esse objetivo foi utilizado o software de edição de imagens Photoshop e o conjunto de plugins DICOMAccess

para tratamento de imagens. Foi inserida, propositalmente, uma mancha preta no exame de ultra-som da paciente MARY GAMAGE (figura 1.5) remetendo ao médico responsável pelo laudo um problema inexistente. Neste momento, o importante é manter o atributo nome_paciente e ID sem modificações para que o servidor DICOM entenda isso como uma nova imagem do paciente (figura 1.6).

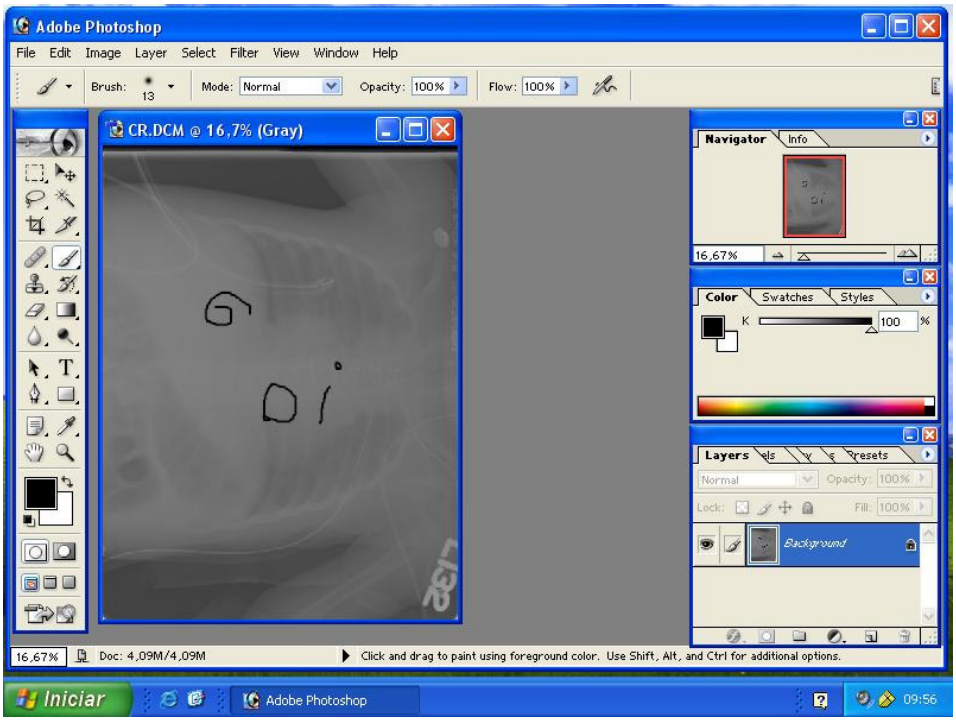

Figura 1.5 - Imagem DICOM adulterada.

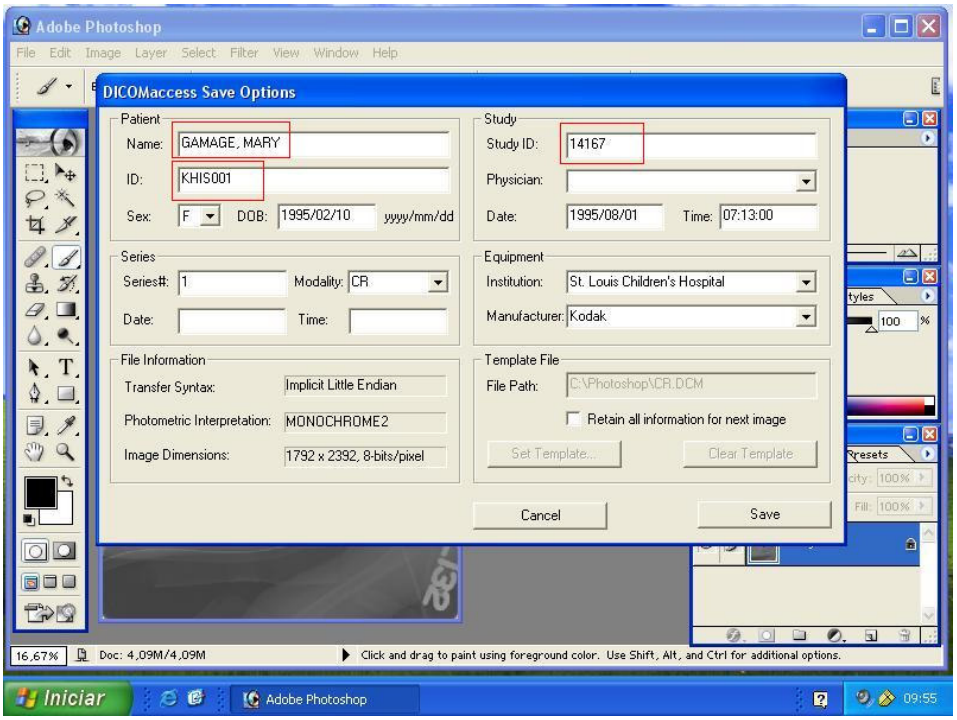

Figura 1.6 - Atributos de uma imagem DICOM.

Utilizando a biblioteca DCM4CHE e os campos ae-called-title e ae-calling-title foi montado o seguinte comando de envio:

```
java -jar dcmsnd.jar dicom://meuservidor:CLIENTE@10.133.184.5:3004
c:\imagens\CR
```

Onde:

- ae-called-title = meuservidor;
- ae-calling-title = CLIENTE;
- Endereço do servidor = 10.133.184.5;
- Porta do servidor = 3004
- Endereço da imagem = C:\imagens\CR

Ao consultar o banco de dados postgreSQL (figura 1.7), verificou-se que não foi cadastrado na tabela img_patientlevel um novo paciente, ou seja, a imagem foi atribuída a um paciente já cadastrado (figura 1.8).

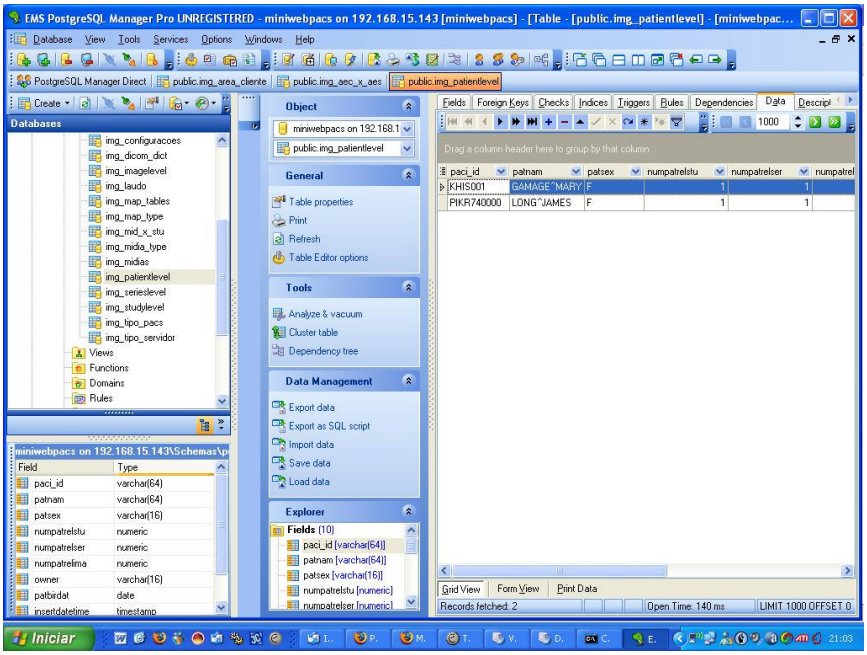

Figura 1.7- Tabela postgres que contém os atributos da tabela paciente.

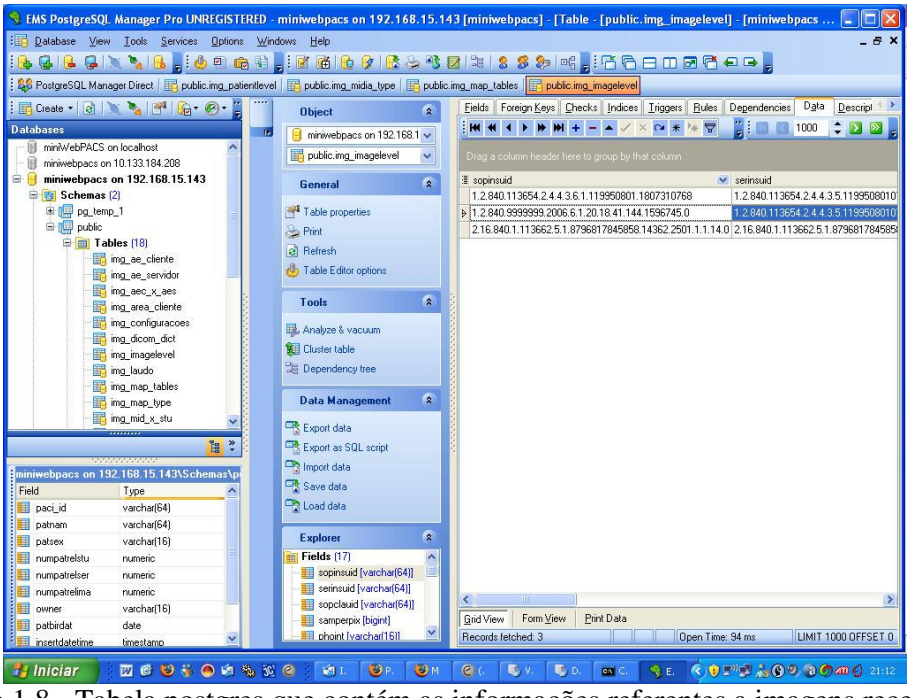

Figura 1.8 - Tabela postgres que contém as informações referentes a imagens recebidas.

Desta forma, qualquer cliente de visualização utilizado para avaliar imagens no servidor DICOM irá demonstrar ao médico uma imagem inválida atribuída ao paciente. Esse problema prejudicará o laudo do profissional de saúde, gerando possíveis consequências jurídicas para a organização.

1.4.5 Realizando o ataque: Outras Vulnerabilidades Encontradas

Ao capturar todos os pacotes transmitidos entre o servidor e cliente, um atacante poderia remontar uma imagem médica, comprometendo o sigilo do exame do paciente. Essa falha, se explorada, poderia comprometer a imagem da instituição perante a sociedade, provocando perdas irreparáveis.

Outra vulnerabilidade possível, embora mais complexa, seria a possibilidade de o atacante substituir uma imagem já armazenada pelo servidor por uma falsa. Nessa técnica, a única diferença seria a necessidade de modificar, no cabeçalho DICOM, o atributo sopinsuid, que define os atributos referentes à imagem enviada. Para essa função, o atacante pode fazer uso de um editor hexadecimal e modificar a sequência de identificação conforme figura 1.9, abaixo.

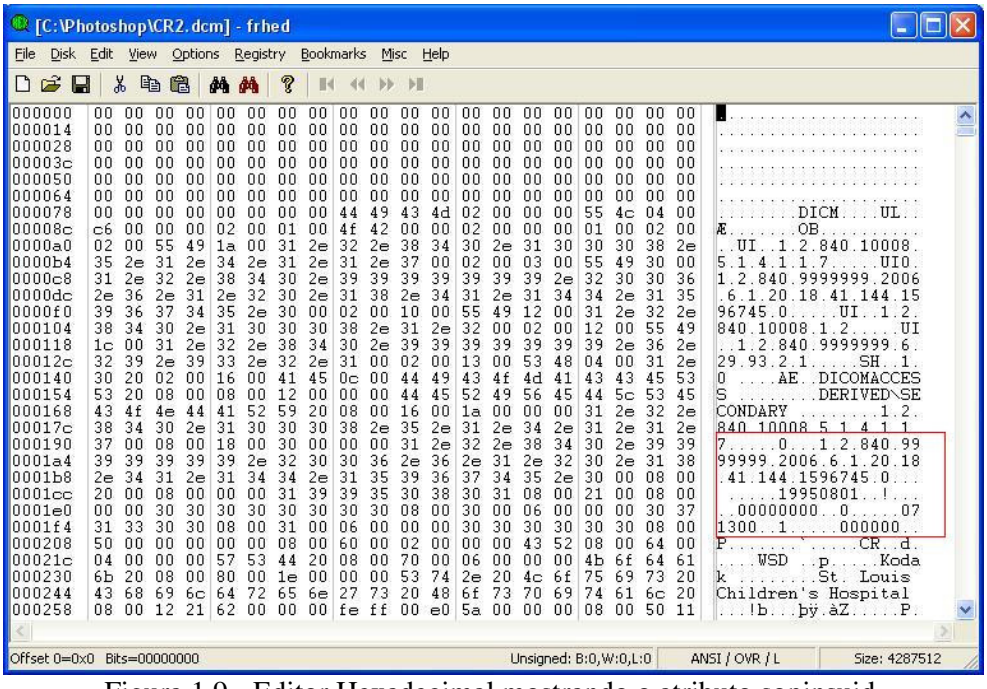

Figura 1.9 - Editor Hexadecimal mostrando o atributo sopinsuid.

Para que um atacante possa descobrir o valor do atributo sopinsuid é necessário que ele tenha capturado uma imagem válida já enviada ao servidor pelo equipamento médico. Ao modificar o sopinsuid de uma imagem e enviá-la novamente ao servidor MiniWebPACS, o aplicativo irá realizar uma atualização na tabela img_imagelevel e trocar a linha que identifica a localização da imagem válida por uma nova linha inválida.

1.4.6 Conclusões das Falhas Encontradas

O ataque proposto foi realizado com êxito. Não houve nenhum óbice ou dificuldade aparente. Porém, em um ambiente real, o atacante poderia encontrar algumas dificuldades como a implementação de redes segmentadas (*switches*). Neste caso, antes de iniciar a captura de informações, o atacante deve comprometer o *switch* para que sua captura de tráfego seja realizada com sucesso.

As vulnerabilidades encontradas e exploradas neste relatório são relativamente simples. Para solucioná-las é necessário utilizar mecanismos que permitam a criptografia das informações trocadas entre equipamento de exames e servidor DICOM. Uma forma de realizar isso é utilizar algoritmos criptográficos e certificados digitais que validem e autentiquem o cliente e o servidor.

1.5 ORGANIZAÇÃO DO TEXTO

O presente texto foi dividido em quatro capítulos. No primeiro, ou seja, neste capítulo, é apresentada a introdução, os objetivos, a motivação, a metodologia do experimento e a justificativa em forma de relatório demonstrando as vulnerabilidades encontradas no sistema atual em uso no HUB. No segundo, consta a revisão da literatura pertinente, que busca explicar o funcionamento e os principais conceitos sobre segurança da informação, DICOM, PACS, Criptografia e TLS. No terceiro capítulo, é apresentado o documento de visão, o diagrama de caso de uso, de classe e os diagramas de sequência dos principais métodos para os dois sistemas desenvolvidos nessa dissertação (hub2pacs e hub2conversor). No útimo capítulo, é apresentado o funcionamento do protótipo e são realizados alguns testes para verificar a eficiência do novo servidor PACS e do conversor proposto.

2 REVISÃO DA LITERATURA

O capítulo de revisão da literatura busca fornecer ao pesquisador e ao leitor o conhecimento teórico sobre o assunto tratado nesta dissertação. Neste momento são apresentados conceitos sobre segurança da informação, análise e gestão do risco, regras para desenvolver aplicações seguras, o funcionamento do protocolo DICOM e do TLS, o uso da criptografia simétrica, assimétrica, funções hash, certificados digitais, infra-estrutura de chaves públicas e a linguagem UML (*Unified Modeling Language*) para representação de produtos de software.

2.1 SEGURANÇA DA INFORMAÇÃO – PRINCIPAIS CONCEITOS

A segurança da informação pode ser definida como a área de conhecimento dedicada à proteção de ativos contra acessos não autorizados, alterações indevidas ou sua indisponibilidade [14].

Embora a definição acima seja conceitualmente clara, outros autores apresentam aspectos fundamentais que integram esse conceito. Dias [15] destaca três princípios básicos que são fundamentais para o entendimento do que vem a ser segurança da informação. São eles:

- Confidencialidade: proteger as informações contra acesso de pessoas não explicitamente autorizadas pelo dono da informação. Computacionalmente, para se garantir este princípio, pode-se fazer uso de criptografia ou simples regras de acesso;

- Integridade: evitar que os dados sejam apagados ou de alguma forma alterados, sem a permissão do proprietário da informação. Para cumprir este objetivo, pode-se fazer uso de algoritmos de *hash* que, ao processarem uma determinada informação, fornecem um número único que identifica aquela porção de bits. Qualquer alteração nos bits originais irá gerar um número totalmente diferente permitindo, assim, identificar claramente quando a informação foi alterada; e

- Disponibilidade: proteger os recursos de tal forma que não fiquem indisponíveis sem a devida autorização. Em termos gerais, este princípio pode incorporar a duplicação do ativo ou o uso de equipamentos de tolerância à falha.

É importante frisar que esses conceitos estão intimamente relacionados e são, por consequência, mutuamente dependentes. Por exemplo, se analisarmos confidencialidade e sua relação com disponibilidade é possível perceber que quanto mais sigilosa for uma informação, menos disponível ela deve estar. Para representar esse conceito, basta imaginar uma fortaleza com vários mecanismos de controle de acesso. Quanto maior for a quantidade de critérios utilizados para autenticar um usuário, maior será a probabilidade de um usuário válido ser rejeitado.

Ao avaliarmos a disponibilidade e sua relação com a integridade, surge a necessidade de avaliar estes conceitos frente às regras do negócio existente. Numa aplicação de e-commerce, se um endereço de um determinado cliente estiver errado, o máximo que acontecerá é ter de pedir desculpas e restituir-lhe o dinheiro pago pelo pedido. Porém, se o sistema ficar indisponível por algumas horas, as perdas serão significativas e poderão representar o encerramento das atividades do site. Já em uma aplicação bancária, errar o saldo de um cliente é tão ou mais danoso quanto deixar o sistema "off-line" por algumas horas.

Além desses princípios básicos, existem outros aspectos que, de uma forma ou de outra, contribuem significativamente para a definição de segurança da informação [16]:

- Autenticação: capacidade de garantir que um usuário, sistema ou informação é mesmo quem alega ser. Este princípio pode ser garantido utilizando os três pilares de identificação, ou seja, o que eu sei (exemplo: senhas), o que eu tenho (exemplo: cartão, *token*, etc) e o que eu sou (exemplo: biometria). Um sistema é definido como forte se combinar dois ou mais destes fatores no seu processo de autenticação;

- Não repúdio ou irretratabilidade: capacidade do sistema de provar que um usuário executou determinada ação no sistema. Este, talvez, seja o princípio mais complicado de implementar no mundo computacional. Pergunta-se: Como garantir que nenhuma das partes poderá negar a realização de uma determinada transação, já que os usuários

envolvidos podem estar a kilômetros de distância um do outro? A resposta deste questionamento vem da aplicação da criptografia assimétrica filiada ao uso de certificados digitais. Tais assuntos serão tratados mais adiante neste texto;

- Legalidade: aderência de um sistema à legislação. Neste sentido o sistema deve estar em conformidade com a política de segurança interna da organização bem como com as normas e legislações pertinentes da nação. Baseado no princípio da legalidade, um administrador não poderia, por exemplo, realizar o monitoramento do correio eletrônico de um usuário sem o conhecimento deste. Tal ação fere diretamente os direitos fundamentais (sigilo) do cidadão garantidos pela constituição federal;

- Privacidade: capacidade de um sistema de manter incógnito um usuário, impossibilitando a ligação direta da identidade do usuário com as ações por este realizadas; e

- Auditoria: capacidade do sistema de auditar tudo o que foi realizado pelos usuários, detectando fraudes ou tentativas de ataque.

Da mesma forma que os princípios básicos, esses novos fundamentos possuem forte ligação entre si. Ao avaliarmos a autenticidade com o não repúdio, observamos que o não repúdio só poderá ser garantido se existir um excelente mecanismo de autenticidade. Já ao relacionarmos privacidade à auditoria, vislumbramos que, se um sistema exigir um alto nível de privacidade para as operações realizadas por um usuário, não existirá forma de auditar suas ações. Existem várias formas de manter a privacidade e continuar tendo a possibilidade de auditar as ações realizadas em um sistema. Uma das mais simples de implementar é o uso de pseudônimos. Nessa técnica, filia-se o usuário a "apelidos" que serão relacionados a uma lista secreta. O sistema passa a registrar as operações realizadas de acordo com os apelidos utilizados. Um auditor poderá recuperar o nome do agente responsável pela ação, utilizando a lista secreta.

Para Beal [17], alguns aspectos de segurança emergem quando precisamos transmitir uma informação de um ponto a outro dentro de uma rede de computadores. Os aspectos por ela

citados são facilmente adaptados para o processo de transmissão e recepção de imagens médicas. Assim, para um sistema do tipo PACS é importante garantir:

- A integridade do conteúdo. A informação enviada pelo emissor deve ser recebida completa e exata pelo receptor;

- O não repúdio da comunicação. Nenhuma das partes envolvidas pode ter a capacidade de negar que a comunicação ocorreu de forma satisfatória;

- A autenticidade do emissor e do receptor. É necessário garantir computacionalmente que o emissor e o receptor sejam realmente quem dizem ser;

- A confidencialidade do conteúdo. Um atacante que consiga escutar o tráfego de rede gerado pelo emissor e receptor não pode conseguir remontar a imagem transmitida; e

- A capacidade de recuperação do conteúdo pelo receptor. O receptor deve ter a capacidade de remontar a informação original enviada pelo emissor.

O MiniWebPACS, utilizado atualmente pelo Hospital Universitário de Brasília, não realiza algumas das atividades acima citadas, desta forma, essa aplicação não garante a confidencialidade de conteúdo durante a transmissão, a autenticidade do emissor e do receptor e o não repúdio da comunicação. Tais aspectos devem ser adaptados em uma nova versão e integrados às funcionalidades já existentes.

Sêmola [14] demonstra que toda a informação manipulada por uma organização passa por quatro fases distintas (figura 2.1), sendo: o manuseio, armazenamento, transporte e descarte. Uma tecnologia que deseje prover segurança deve buscar o equilíbrio entre o ciclo de vida da informação, criando mecanismos de proteção para todas as fases deste processo, respeitando os conceitos básicos de segurança e possíveis aspectos complementares.

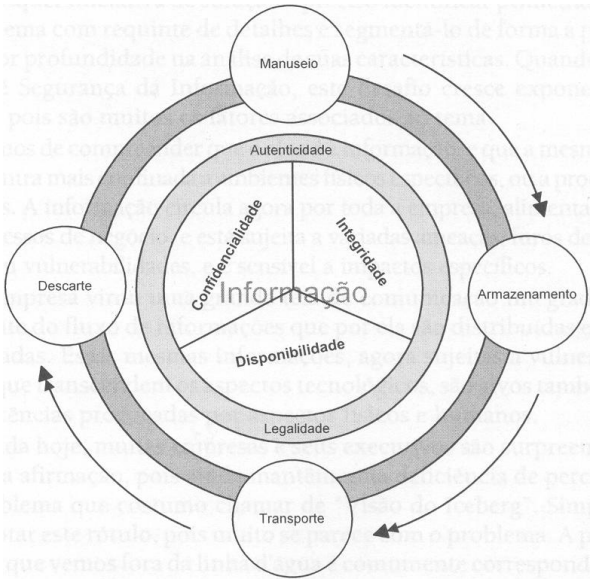

Figura 2.1 - Ciclo de vida da informação [14].

Para Bel [17], as etapas do ciclo de vida da informação são: a identificação das necessidades e dos requisitos para a classificação, a forma de obtenção, o tratamento dispensado, a distribuição entre as partes envolvidas, a forma de uso, o armazenamento e o descarte. Da mesma forma que Sêmola, essa autora mostra que cada uma das etapas possui regras e procedimentos próprios para gerir a segurança e que essas regras e procedimentos devem ser respeitados, caso se queira garantir o sigilo.

Ao avaliar os requisitos de segurança de uma nova tecnologia, faz-se necessário verificar todas as vulnerabilidades e ameaças existentes. A identificação dessas vulnerabilidades permite ao analista de segurança estudar os principais conceitos, o cenário de atuação e as consequências diretas e indiretas de sua utilização. Entre as inúmeras ferramentas utilizadas para este fim destaca-se a Análise de Risco e Vulnerabilidades.

2.1.1 Análise de Risco e Vulnerabilidades

Antes de se definir as métricas e formas de avaliar o risco em uma nova tecnologia, sistema ou ambiente, deve-se buscar um entendimento sobre este conceito e sua relação com as vulnerabilidades encontradas em um ambiente computacional. Desta forma, vulnerabilidade deve ser considerada como uma fraqueza ou deficiência que pode ser explorada por uma ameaça. Para Dias [15], o risco é definido como a "combinação de componentes, tais como

ameaças, vulnerabilidades e impactos". A vulnerabilidade é um componente elementar do risco, portanto não existiria o risco se não existisse uma vulnerabilidade.

É importante destacar que risco geralmente é utilizado como sinônimo para ameaça ou probabilidade. Tal interpretação pode trazer dificuldades ao refletir sobre esse termo, pois ameaça e probabilidade são apenas elementos que o compõem.

Antes de decidir a melhor estratégia de segurança a ser utilizada em uma tecnologia, é necessário saber contra o quê ela será protegida. Desta forma, a segurança poderia ser definida em termos de combate às ameaças identificadas. O processo de análise de risco e vulnerabilidade consiste, inicialmente, em "identificar as ameaças, determinar a probabilidade de uma ameaça se concretizar e entender os riscos potenciais, classificando-os por nível de importância e severidade das perdas, e os custos envolvidos na sua prevenção ou recuperação" [15].

Ao se deparar com um incidente de segurança, pode-se adotar uma abordagem proativa ou reativa. No modelo proativo, buscam-se formas de prever os incidentes, antecipando e executando ações que tenham como objetivo evitar o "mal maior". Embora seja o modelo "ideal", observamos que muitas ações são executadas de forma reativa, quando é esperado que o evento aconteça para providenciar soluções. As principais ações reativas que são desencadeadas frente a um incidente de segurança em andamento são (figura 2.2):

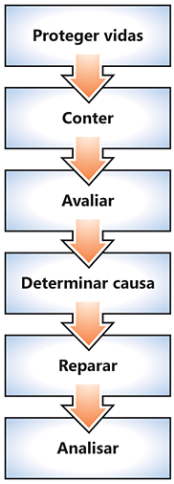

Figura 2.2 - Processo de resposta a incidentes [18].

- Proteger vidas: preservar todos os sistemas que, de alguma forma, ofereçam suporte à vida. Um exemplo: os equipamentos médico-hospitalares;

- Conter: diminuir os estragos causados pelo incidente, reduzindo o impacto sobre recursos computacionais;

- Avaliar: determinar a extensão dos danos causados pelo incidente, imediatamente após conter a situação;

- Determinar causa: determinar a origem do incidente, descobrindo quais recursos foram alvo e que vulnerabilidades foram exploradas;

- Corrigir: restaurar as operações de negócios normais e os dados que tenham sido perdidos durante o incidente; e

- Analisar: determinar, com a equipe, que etapas foram bem-sucedidas e quais foram os erros cometidos durante o processo de recuperação.

As medidas de proteção utilizadas para diminuir os riscos de segurança da informação podem ser classificadas, também, como preventivas reativas e detectivas. A figura 2.3, demonstra alguns exemplos de medidas de proteção e seu uso. O objetivo é representar algumas formas de reduzir o impacto de eventuais vulnerabilidades e ameaças sobre um determinado ativo.

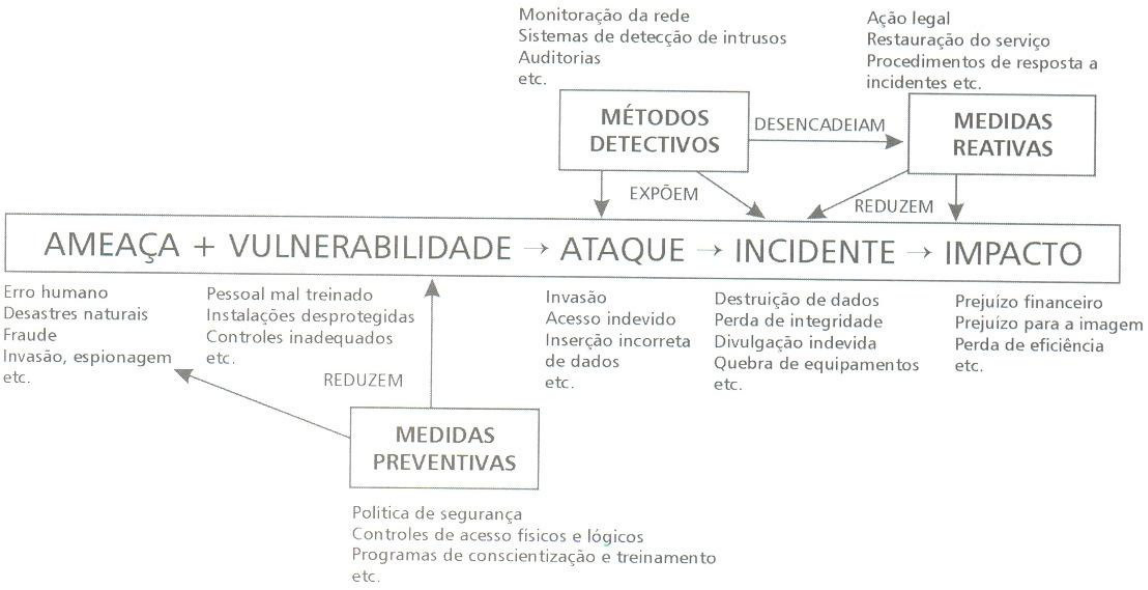

Figura 2.3 - Componentes do risco e medidas de proteção usadas para reduzi-los [17].

As vulnerabilidades encontradas em uma determinada tecnologia ou em ambientes devem ser classificadas de acordo com seu grau de criticidade. Em um projeto de análise de risco, podem ser utilizadas para este fim métricas qualitativas ou quantitativas que, basicamente, se diferem em termos de velocidade e relação com o negócio.

Os métodos quantitativos são vistos com cautela pelos estudiosos, devido à dificuldade de obtenção de resultados representativos. "É preciso dispor de um histórico confiável de incidentes de segurança passados e dos impactos financeiros a eles associados para garantir resultados representativos..." [17]. Um dos métodos mais conhecidos é o cálculo da Expectativa de Perda Anual (*Annual Loss Expectation*), por meio do qual cada vulnerabilidade encontrada será classificada de acordo com a multiplicação da perda prevista de um incidente pela frequência que ele possa ocorrer num período de 12 meses.

Os métodos qualitativos trabalham com a descrição literal dos riscos para avaliá-los. Os principais métodos de avaliação qualitativa do risco utilizam questionários e/ou matrizes de risco. Um exemplo retirado dos livros de administração é a matriz GUT (tabela 2.1).

Tabela 2.1 - Matriz GUT [19].

Processo de Planejamento e Implementação de Melhorias Fatores de avaliação da matriz GUT			
Pontos	**Gravidade**	**Urgência**	**Tendência**
5	Os prejuízos ou dificuldades são extremamente graves	É necessária uma ação imediata	Se nada for feito, o agravamento da situação será imediato
4	Muito Grave	Com alguma urgência	Vai piorar a curto prazo
3	Grave	O mais cedo possível	Vai piorar a médio prazo
2	Pouco Grave	Pode esperar um pouco	Vai piorar a longo prazo
1	Sem Gravidade	Não tem pressa	Não vai piorar ou pode até melhorar

Utilizando como referência esta matriz, podem-se classificar cada vulnerabilidade encontrada de acordo com seu nível de gravidade, urgência e tendência. Esta avaliação fornece importantes informações para que um administrador possa decidir qual será a ordem de correção dos problemas de segurança encontrados. Esse processo de decisão é chamado, comumente, de gerenciamento do risco.

Outra forma de análise qualitativa é a fórmula citada na figura 2.4:

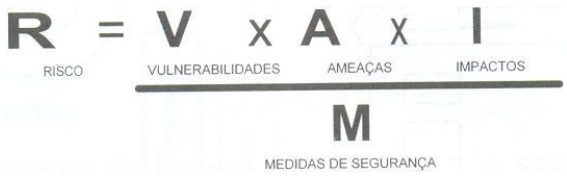

Figura 2.4 - Diagrama da equação do risco de segurança da informação [14].

A interpretação dessa equação é realizada definindo que o risco consiste na probabilidade de que as ameaças explorem vulnerabilidades, expondo os ativos a perdas dos princípios de segurança, causando, desta forma, impactos nos negócios. Esses impactos são limitados pelas medidas de segurança que protegem os ativos. Uma vez definidos os valores para cada uma das variáveis desta fórmula, as vulnerabilidades podem ser classificadas de forma decrescente a partir do risco calculado. A grande dificuldade está em como atribuir pontuações a cada um destes valores.

As técnicas de análise qualitativa de risco aqui citadas foram utilizadas para avaliar o nível de segurança do sistema de transmissão de imagens que está sendo utilizado no Hospital

Universitário de Brasília. Após a realização dessa atividade, ficou evidenciada a necessidade de implementar um novo sistema que diminuísse o risco associado às vulnerabilidades encontradas.

2.1.2 Gerenciamento do Risco

Para alcançar os objetivos de segurança e adotar um enfoque de gestão baseado nos riscos específicos para o negócio, é de extrema importância proteger os ativos de informação contra ameaças que, de alguma forma, possam prejudicar o seu nível de confidencialidade, integridade e disponibilidade [17].

Um processo de gerenciamento de riscos compreende quatro fases distintas: avaliando os riscos, oferecendo suporte às decisões, implementando controles e analisando a eficácia do programa (figura 2.5). Dentre estas fases, a que merece maior destaque neste texto é a avaliação do risco, que pode ser dividida em: planejamento, coleta de dados facilitada e priorização de risco [18].

Figura 2.5 - O processo de gerenciamento de riscos de segurança [18].

O processo de gerenciamento de risco inicia-se com o planejamento, sendo ele considerado o fator determinante para o sucesso ou fracasso de qualquer avaliação. Escopos mal definidos provocam erros que reduzem a eficácia do processo de segurança. A próxima etapa é a coleta de dados quando, nesta fase, são levantados os principais ativos, as ameaças, vulnerabilidades, os controles atuais em uso e os controles propostos. Para esse fim, devem ser utilizados textos que descrevam a nova tecnologia aliada a fóruns e *sites* especializados em descobrir novas falhas de segurança.

Cada vulnerabilidade encontrada deve estar classificada de acordo com o nível de confidencialidade, disponibilidade e integridade, permitindo que os problemas de segurança considerados graves sejam priorizados durante o procedimento de correção, sendo esta fase conhecida como priorização. Entretanto, a correção de falhas de segurança está, normalmente, associada a investimentos financeiros ou computacionais que não se encontram disponíveis a qualquer momento ou de forma infinita. Um exemplo seria utilizar algoritmos de criptografia com grandes chaves. Isto aumenta o nível de sigilo de um sistema, porém o seu uso em equipamentos com baixo poder de processamento representará perdas significativas de desempenho.

Considerando esses fatores, as ações de segurança devem ser avaliadas com cautela para que o custo da implementação do controle não seja superior ao valor do ativo que se deseja proteger.

2.2 PRINCIPAIS REGRAS PARA DESENVOLVER APLICAÇÕES SEGURAS

O uso intensivo de tecnologias como a Internet e dos sistemas ERP (*Enterprise Resource Planning*), pelas organizações, tem acrescentado inúmeras falhas de segurança em seu ambiente computacional. Tais falhas ou vulnerabilidades, se exploradas por um usuário malicioso, podem trazer perdas significativas de informação e, por consequência, prejuízos financeiros. Assim, segurança passa a ser um requisito elementar, tão importante quanto o desempenho ou as funcionalidades de um software. Ignorá-la poderá, num futuro próximo, determinar ou não a continuidade de uma organização ou negócio [20].

Embora o uso de segurança em aplicações seja amplamente divulgado e estudado, sua implementação em um projeto real denota dois grandes desafios: o desenvolvimento um sistema seguro e a garantia, para o cliente, de que a aplicação desenvolvida é segura. [16]. Em uma leitura rápida, pode-se imaginar que esses desafios representam a mesma coisa, mas não são. Um programador pode desenvolver o sistema mais seguro do mundo, fazendo uso dos melhores algoritmos de mercado e das técnicas mais refinadas de análise e projeto de software, porém, como garantirá, ao seu cliente, que os níveis de segurança da aplicação são adequados? Ou melhor, como garantir segurança sem permitir a visualização do código fonte da aplicação por um terceiro?

Assim, é importante entender que não se pode contar apenas com a "palavra" do construtor ou da equipe de desenvolvimento do software. Devem ser criados mecanismos que permitam avaliar o nível de segurança sem comprometer, por exemplo, o sigilo do código fonte da aplicação. Desta forma, a ISO 15.408 [4] aparece como ferramenta para padronizar todo o processo de análise, prototipagem, entrega e adequação do software. Nela, são descritos vários requisitos que, se utilizados, aumentam significativamente o nível de garantia da segurança de uma aplicação. Como premissas, são destacadas nesta norma, três preocupações básicas: segurança do ambiente de desenvolvimento, segurança da aplicação desenvolvida e a garantia de segurança da aplicação desenvolvida.

Por não se tratar do foco desta pesquisa, serão tratados neste texto requisitos necessários para garantir a segurança da aplicação desenvolvida e como demonstrar para o cliente que a aplicação possui o nível de segurança necessário para seu negócio.

2.2.1 Segurança da Aplicação Desenvolvida

Para garantir a segurança da aplicação desenvolvida é necessário seguir normas sobre boas práticas de programação. De forma geral, seguindo as normas, automaticamente é possível eliminar muitos dos erros e falhas de segurança. São exemplos de boas práticas [16]:

- Funções seguras: tratar todas as variáveis de entrada como não-confiáveis, verificando sua validade antes de usá-las. A boa avaliação dos dados de entrada poderá evitar eventuais erros de falhas gerais de proteção ou perda de controle da sequência de execução do sistema;

- Sempre verificar os códigos de erro retornados por uma função, principalmente nas chamadas API do sistema operacional. O tratamento adequado de cada erro poderá evitar que a aplicação execute operações ilegais que venham a comprometer a qualidade dos resultados fornecidos pelo software;

- Atentar sempre para o tamanho dos *buffers* e *array*s do sistema. Grande parte dos ataques a aplicações são baseados na simples implementação de *scripts* que estouram a capacidade de *buffers* e *arrays*. Este tipo de ataque fere diretamente o princípio da

disponibilidade, porém, em casos extremos, o aplicativo poderá fornecer um *prompt* de comando com permissões de administrador permitindo, desta forma, que um usuário malicioso possa executar qualquer comando a fim de danificar ou contornar os mecanismos de segurança utilizados;

- Verificar caracteres especiais. Um exemplo de ataque que faz uso de caracteres especiais é o *SQL Injection*. Seu uso é baseado na utilização de uma aspa simples ou dupla inserida em um campo do tipo "*textbox*". Imagine um sistema de autenticação que realize a seguinte consulta: SELECT * from Login Where login-'" &txtLogin &"'", onde TxtLogin é uma variável que possui o conteúdo de um caixa de texto informado por um usuário. Um atacante poderia digitar ' OR 0=0 # e obrigaria o sistema a executar a seguinte consulta: SELECT * FROM Login Where Login='' or 0=0 #'. Essa consulta possui uma verdade absoluta (0=0), desta forma, não interessa o login informado, o sistema irá liberar o acesso à aplicação;

- Utilizar componentes e bibliotecas confiáveis. Uma ação comumente realizada por grande parte dos construtores de software é o uso de bibliotecas de terceiros para resolver alguns problemas da aplicação como, por exemplo, impressão de relatório, comunicação com banco de dados, criptografia, transferência de arquivos, geração de boletos, etc. A segurança do software poderá ser comprometida em consequência de eventuais erros nestas bibliotecas. O desenvolvedor deve avaliar cuidadosamente o uso destas ferramentas, dando preferência a bibliotecas em que a equipe de desenvolvimento tenha condições de avaliar os requisitos de segurança por meio da análise de seu código fonte;

- Evitar informações sensíveis em arquivos temporários. Uma aplicação, eventualmente, poderá esquecer-se de remover os arquivos temporários que criou. Neste caso, o sigilo das informações armazenadas nestes arquivos estará automaticamente comprometido;

- Não colocar senhas e chaves criptográficas no código da aplicação. Uma senha ou chave criptográfica inserida no código da aplicação pode ser facilmente descoberta se o atacante fizer uso de ferramentas para decompilar o código executável da aplicação; e

- Documentar todo o código fonte. A documentação permite ao cliente entender a aplicação e seu funcionamento bem como facilita a evolução do produto, já que novas implementações podem ser realizadas por outra equipe de construtores.

A segurança de uma determinada aplicação deve ser divida em três grandes projetos: segurança do projeto, segurança por default e segurança na instalação [21].

A segurança do projeto tem como objetivo determinar um conjunto de passos a serem seguidos pela equipe de desenvolvimento visando garantir que o projeto do produto seja bom desde o começo. Esses passos consistem, por exemplo, no treinamento da equipe de desenvolvimento, na obediência de diretrizes de codificação e de projeto, na correção de bugs que divergem das diretrizes, no desenvolvimento de testes de penetração e na simplificação do código e dos modelos de segurança empregados.

A segurança por default garante que o produto seja suficientemente seguro quando saia da fábrica. Pode-se garantir esse princípio criando uma forma de instalação padrão que ative apenas os recursos mais usados e não exigir que o aplicativo precise de permissões administrativas para funcionar.

A segurança da instalação significa que o sistema é sustentável depois que seus usuários instalem o produto. O aplicativo deve possuir uma interface simples para administrar as funcionalidades de segurança.

O novo servidor PACS, proposto nessa dissertação, implementará algumas das características acima descritas. Entre elas destacam-se a não inserção de chaves criptográficas no código da aplicação, o uso de componentes confiáveis como o DCM4CHE, o tratamento de erros em cada chamada e a definição de tamanhos fixos para *arrays* (vetores).

2.2.2 Garantia de Segurança

Garantia de Segurança é um item que está diretamente relacionado ao "sentimento" que o cliente tem sobre o nível de segurança da aplicação. Em muitos casos, esta percepção pode ser empírica, demonstrando receios e dúvidas sobre sua confiabilidade. A norma ISO/IEC 15.408

introduz um importante conceito por meio do qual, por definição, um software é dito seguro quando [22]:

- A especificação de segurança foi realizada de forma clara e objetiva. O ideal é que todos os indivíduos envolvidos no processo de desenvolvimento tenham amplo conhecimento sobre os níveis desejados de segurança e sua relação com o negócio da organização;

- A aplicação foi construída de acordo com esta especificação. Afinal, a especificação determina o desejo de cada um dos envolvidos na construção do software; e

- O produto foi exaustivamente testado, a fim de verificar se a especificação inicial foi atendida.

"A melhor garantia de segurança para o desenvolvimento do software é através dos testes comprovado pelo cliente" [15]. A fim de facilitar o processo, o cliente pode transferir a responsabilidade dos testes para um laboratório independente, sistematizando a forma de auferir o nível desejado.

De forma geral, o software a ser avaliado será sempre encaminhado a um especialista em segurança que, para a realização dos testes, pode utilizar-se de estratégias como a decomposição do aplicativo em componentes fundamentais, identificando as interfaces dos componentes e classificando-as de acordo com as vulnerabilidades encontradas. Esta informação é extremante útil para determinar as estruturas de dados utilizadas em cada interface e localizar o problema de segurança injetando uma alteração nos dados de entrada e modificando, desta forma, o comportamento normal do software [21].

O modelo de testes descrito acima foi utilizado para determinar o nível de segurança da aplicação MiniWebPACS. A primeira atividade realizada teve como objetivo identificar as possíveis entradas e como explorá-las.

2.2.3 Controle de Acesso

O objetivo das funções de controle de acesso de um sistema é garantir que um determinado usuário tenha o direito ou não de visualizar, alterar, inserir ou excluir determinada informação. No final das contas, sempre se trata de um usuário, de um lado, e de uma informação, do outro, como o esquema da figura 2.6.

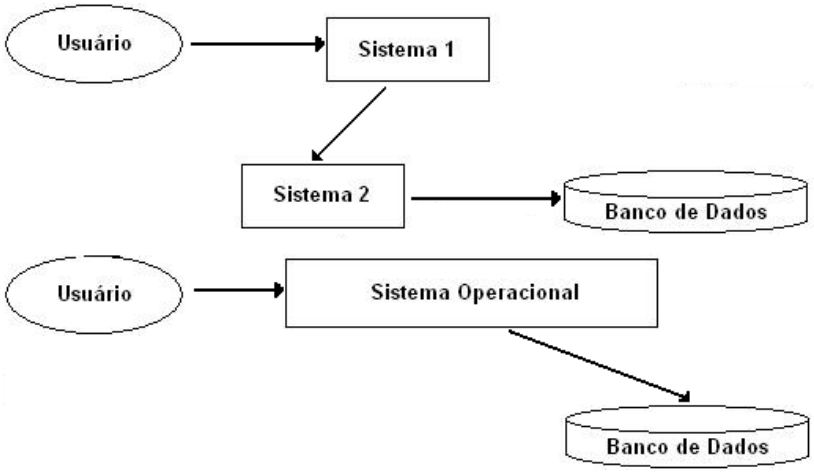

Figura 2.6 - Acesso do usuário à informação [16].

O controle de acesso é baseado em três fases distintas: a autenticação, que é responsável por validar a identidade do usuário; a autorização, que tem como objetivo identificar e limitar as atividades realizadas; e a auditoria, que serve como mecanismo para acompanhar os eventos gerados no sistema.

O processo de autenticação do usuário pode ser divido em três grandes grupos: o que você sabe (senhas e informações pessoais), o que você tem (*tokens, smart cards*, cartões magnéticos) e o que você é (biometria).

O método de autenticação, baseado no que você sabe, utiliza, normalmente, senhas de acesso. Um exemplo prático de seu uso seria o servidor miniWebPACAS, que faz uso de uma senha individual para identificar e autenticar o equipamento médico (*ae-title*). Embora grande parte dos sistemas de rede utilize este método, o uso de senhas oferece um grau de proteção menor do que se costuma atribuir-lhes. A principal técnica que explora esse tipo de autenticação é baseada no uso de ferramentas computacionais para realizar a busca exaustiva da senha por

meio de tentativa e erro. Uma forma de limitar a ação desse tipo de aplicativo é obrigar o uso de senhas fortes, contendo letras, números e caracteres especiais e estimular a troca periódica da senha [20].

A autenticação, baseada em algo que o usuário tenha, é feita, normalmente, com o uso de *tokens*, cartões magnéticos e *smart cards*. O problema desse mecanismo está no fato de que se o cartão for roubado por um invasor, ele conseguirá fácil acesso ao sistema. Para minimizar tal impacto, os cartões são normalmente associados a uma senha ou código de identificação pessoal.

O que você é baseia-se nas características físicas e comportamentais que identificam os seres humanos de forma única. São exemplos: o reconhecimento facial, digital, voz, padrões de íris, retina, assinatura, modo de digitar etc.

O método de autenticação não pode ser ambíguo, pois precisa associar uma identidade única para os usuários. As técnicas até aqui citadas podem ser combinadas para prover mecanismos mais seguros. Um exemplo é a utilização da *Public Key Infrastructure* (PKI) que realiza a autenticação, combinando o que você tem (certificado digital assinado) com o que você sabe (chave privada do certificado).

As políticas de controle de acesso (autorização) são baseadas em dois conceitos: o sujeito e a permissão (objeto + operação). São encontradas na literatura, basicamente, três tipos de política de controle de acesso, sendo [23]:

- *Mandatory Access Control* (MAC): os usuários não são os donos do objeto, desta forma não podem definir as permissões de acesso e são obrigados a obedecer à política implementada pelo administrador do ambiente. Baseia-se na classificação de objetos e usuários de acordo com um nível de segurança pré-estabelecido (exemplo: secreto, restrito e público). O acesso a um determinado objeto só é garantido se o nível do usuário requisitante for igual ou superior ao nível do objeto requisitado;

- *Discretionary Access Control* (DAC): os usuários são os donos dos recursos e, por consequência, detém o controle sobre a forma de acesso e de quem pode ou não

acessá-lo. Este modelo é muito utilizado em sistemas operacionais Linux/Unix onde o proprietário de uma informação pode definir permissões de um objeto para determinados grupos ou usuário; e

- *Role Based Access Controls* (RBAC): utiliza-se do conceito de papel, associando os usuários a determinadas permissões. Do ponto de vista de administração, essa associação torna-se interessante já que as organizações funcionam em cima de papéis / cargos. O acesso a um determinado objeto só é permitido se o usuário estiver associado a um papel que possua essa permissão.

Para saber se um determinado usuário tem direito a um determinado acesso, o sistema precisa verificar a permissão através de uma chamada a outro sistema ou esperar o código de erro que será gerado. Em aplicações com alto desempenho, esse tempo de espera pode provocar quedas sensíveis na resposta fornecida pelo software.

2.2.4 Proteção Contra Fluxo Ilícito

Normalmente, em grande parte das aplicações existentes no mercado, o controle de acesso às informações é realizado pelo próprio aplicativo. Desta forma, qualquer requisição deve ser avaliada pelo software desenvolvido que definirá, por meio de uma política própria, cadastrada anteriormente, qual o nível de acesso de cada usuário. Por ser um controle interno da aplicação, um usuário malicioso poderia "burlar" todas as regras de acesso, tentando acessar as informações diretamente no servidor de banco de dados. Essa tentativa e possibilidade de acesso sem o uso da aplicação é conhecida como fluxo ilícito.

Não seria necessário dizer que esse tipo de acesso deve ser evitado a todo custo. De forma geral, uma boa análise de risco pode identificar todos os possíveis canais de acesso a uma informação. Tal conhecimento permitirá ao desenvolvedor estabelecer mecanismos para bloquear, limitar ou monitorar os canais. Algumas formas de controlar o acesso ilícito são [16]:

- Diminuir o tempo de vida de uma informação;

- Utilizar mecanismos de autenticação ou bloquear a possibilidade de acesso remoto. Para esse fim são utilizados, normalmente, regras de *firewall*;

- Colocar senha de BIOS no sistema. Esse procedimento evitará, por exemplo, que um usuário possa inicializar o sistema por um dispositivo externo, fraudando qualquer regra de acesso configurada no sistema operacional;

- Remover placas de rede, *drivers* de disquete, interfaces USB etc. Realizando essa atividade, o administrador evita que qualquer informação seja copiada ou removida do equipamento;

- Cifrar todas as informações sigilosas; e

- Utilizar mecanismos de autodestruição em caso de tentativa de arrombamento. Esse recurso é muito utilizado para proteger a chave privada de uma infra-estrutura de chaves públicas. Caso alguém tente de alguma forma copiar a chave privada, o dispositivo a destrói, garantindo o sigilo.

2.3 PROTOCOLO DICOM – VISÃO GERAL

O padrão DICOM (*Digital Imaging and Communications in Medicine*) é utilizado para permitir a troca de imagens e informações médicas entre equipamentos de diferentes fabricantes. Para cumprir esse objetivo, o padrão fornece vários níveis de apoio, como, por exemplo, suporte para conexão a um servidor de banco de dados, gerenciamento e armazenamento de imagens, suporte a informações do paciente, padrões para a garantia da qualidade da imagem, segurança, etc.

O DICOM foi construído como um documento de várias partes para facilitar a extensão do padrão. Também foram estabelecidos componentes não só para as imagens, mas também para pacientes, relatórios e outros conjuntos de informações. Com os aperfeiçoamentos realizados no DICOM, tornou-se admissível o incremento e a expansão do arquivamento de imagens e sistemas de comunicação (PACS – *Picture Archiving and Communication Systems*),

viabilizando que sistemas de informações médicas tenham interfaces adequadas para a exibição do padrão.

O DICOM é constituído de 18 partes relacionadas, porém tratadas como documentos autônomos [24]:

- Parte 1 – Introdução e visão geral (*Introduction and Overview*): estabelece um aspecto global do padrão DICOM. Apresenta uma rápida exposição do conteúdo de cada componente;

- Parte 2 – Conformidade (*Conformance*): determina métodos de implementação que estejam em acordo com as condições constituídas;

- Parte 3 – Definições de objetos de informação (*Information Object Definitions*): descreve classes de objetos de informação que abastecem uma acepção abstrata de entidades do mundo real aplicável à comunicação de imagens médicas digitais e às informações catalogadas. Cada uma dessas classes contém uma definição de seu desígnio e as características que o determinam. Duas espécies de classes de objetos de informação são definidas:

    - Normalizadas: Incluem somente os atributos inerentes a entidades do mundo real representadas. Ex: Atributos de data e hora de um estudo de caso; e

    - Compostas: Podem adicionalmente conter atributos que estão concatenados aos pacientes, mas não próprios a entidades no mundo real.

- Parte 4 – Especificações de classes de serviços (*Service Class Specifications*): uma classe de serviço agrega um ou vários objetos de informação a um ou mais comandos executados sobre estes objetos;

- Parte 5 – Estrutura de dados e codificação (*Data Structure and Encoding*): aponta como aplicações DICOM edificam e compilam os conjuntos de dados (*Data Sets*)

originado do uso de Objetos de Informação e Classes de Serviços determinados nas partes 3 e 4 do padrão. Delibera as normas de codificação para conjuntos de caracteres internacionais empregados no DICOM;

- Parte 6 – Dicionário de dados (*Data Dictionary*): é um registro centralizado que determina o conjunto de todos os elementos de dados livres para reproduzir informações. Para cada elemento, esta parte do padrão especifica:

    - *Tag* exclusiva, que consiste em um grupo, e número de elemento;

    - Nome;

    - Sua representação de valores – VR (string, inteiro, etc);

    - Multiplicidade (quantos valores por atributo); e

    - Quando há exclusão.

- Parte 7 – Protocolo de troca de mensagens (*Message Exchange Protocol*): determina tanto o serviço quanto o protocolo empregado por uma aplicação para troca de dados;

- Parte 8 – Suporte à comunicação em rede para troca de mensagens (*Network Communication Support for Message Exchange*): define os serviços de comunicação e protocolos de camada superior necessários ao suporte, em ambientes de rede, à comunicação entre aplicações DICOM. Esses serviços de comunicação e protocolos afiançam que esta comunicação ocorra de forma concatenada e hábil através da rede;

- Parte 9 – Especificação de interferência ponto a ponto (*Point to Point Interference Specifications*): removida na versão 3.0 do padrão;

- Parte 10 - Armazenamento em mídia e formato de arquivo para a troca de dados (*Media Storage and File Format for Media Interchange*): determina o arquétipo geral

de armazenamento de imagens em mídia removível. O objetivo desta parte é fornecer um *framework* que permita a troca de vários tipos de imagens médicas e as informações agregadas em uma diversidade de tipos de mídias de armazenamento removível;

- Parte 11 – Perfis da aplicação de armazenamento em mídia (*Media Storage Application Profiles*): define subconjuntos específicos da aplicação que devem estar de acordo com o padrão. Tais subconjuntos são conhecidos como Perfis da Aplicação;

- Parte 12 – Formato de mídia e mídia física para troca de dados (*Media Formats and Physical Media for Media Interchange*): viabiliza o intercâmbio de informações entre aplicações de sistemas médicos, definindo tanto a estrutura de descrição de relacionamento entre o modelo de armazenamento em mídia, quanto uma mídia física específica, formato e característica de mídia;

- Parte 13 – Gerenciamento de impressão ponto a ponto (*Print Management Point-to-Point*): Removida na versão 3.0 do padrão;

- Parte 14 – Função de exibição de escala de cinza (*Grayscale Standard Display Function*): uniformiza as funções de exibição escala de cinza para imagens apresentadas em mídias diversas como, por exemplo, monitores e impressoras;

- Parte 15 – Perfis de segurança e sistema de gerenciamento de perfis (*Security and System Management Profiles*): determina o uso de protocolos externos (DHCP, etc.) e públicos para cada um dos perfis de segurança sugeridos. Estes protocolos também devem incluir o uso de criptografia de dados, chave pública e "*smart cards*";

- Parte 16 – Recurso de mapeamento de conteúdo (*Content Mapping Resource*): determina *templates* de elaboração de documentos, grupo de termos codificados, glossário de termos e traduções;

- Parte 17 – Informação redundante (*Explanatory Information*): determina anexos normativos e informativos abrangendo informações explicativas; e

- Parte 18 – Acesso web a objetos DICOM persistentes (*Web Access to DICOM Persistent Objects*): efetivado por meio de requisições http. A requisição contém um ponteiro para o objeto no formato de UID (*unique identifier*) de sua instância. Este padrão esclarece como esta requisição deve ser iniciada.

Conforme pode ser observado, na figura 2.7, as partes que subdividem o padrão podem ser resumidas e organizadas em três grandes grupos, sendo: gerais, comunicação de rede e mídia de troca e armazenamento de imagens.

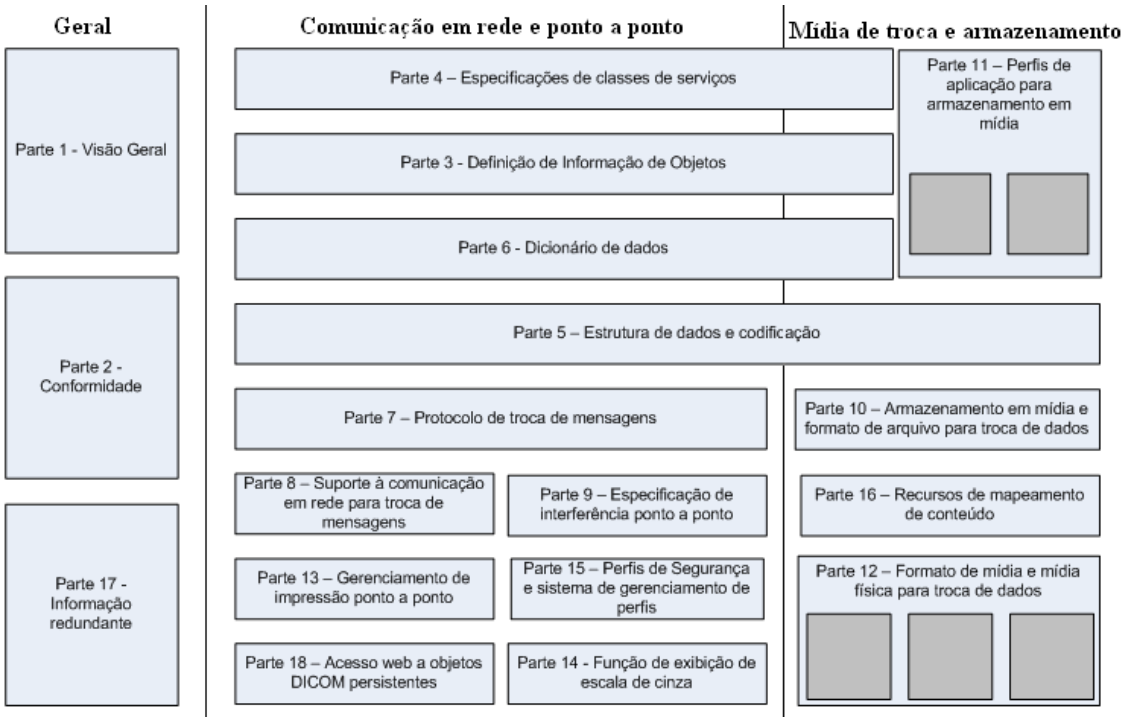

Figura 2.7 – Arquitetura das partes padrão DICOM [1].

2.3.1 Definição de Objetos de Informação (IOD)

Um dos principais elementos utilizados pelo protocolo são as definições de objetos de informações (IOD) que estabelecem, basicamente, o formato para as informações que serão trocadas entre SCU e SCP. Sua divisão em módulos permite que cada imagem possua um conjunto independente de informações com características específicas. Por exemplo, uma

imagem CT requer descritores diferentes no cabeçalho da imagem do que uma imagem ultra-som ou uma imagem oftalmologia. Esses modelos são controlados por identificadores únicos, que são registrados pelo *National Electrical Manufacturers Association* (NEMA).

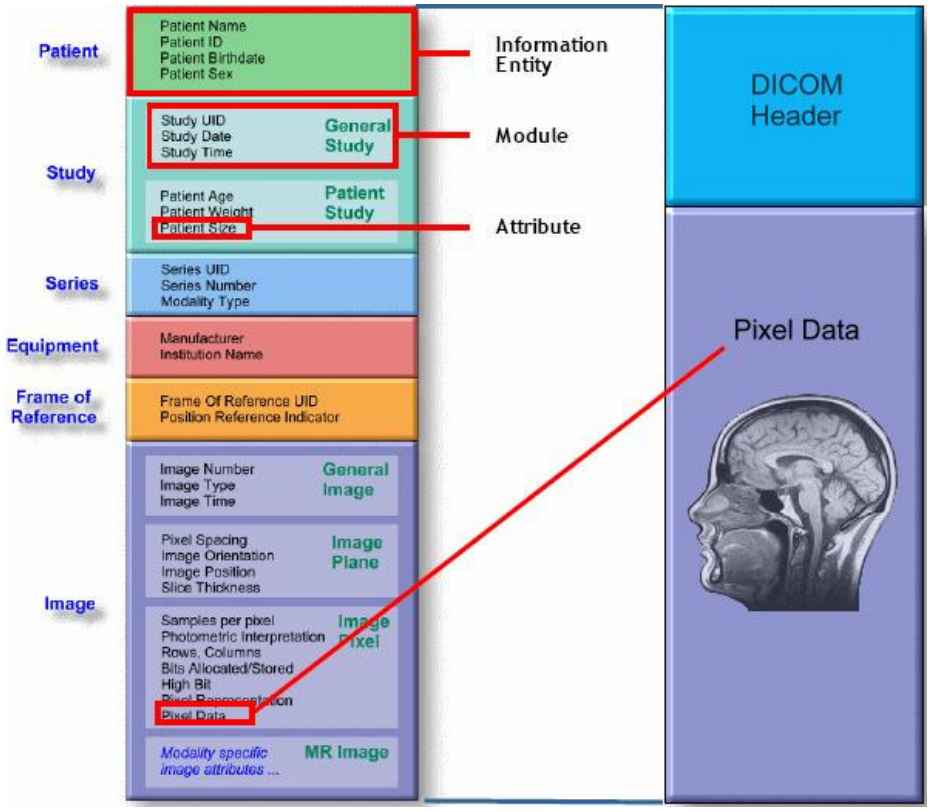

Figura 2.8 – Exemplo de objeto de informação [25].

Na figura 2.8, podem-se observar alguns atributos (*data fields*) referentes à imagem. É possível verificar que existem conjuntos de atributos que identificam o paciente, o tipo do estudo (exame), o equipamento, a imagem etc. Cada atributo possui um identificador único, conhecido como *Tag*. Para exemplificar quais os elementos que compõem uma *Tag* (0010, 0010), abaixo pode ser observado um exemplo de uma *Tag* utilizada para identificar o nome do paciente em uma determinada imagem:

```
(0010, 0010)      PN  17   Francisco^Marcelo
Group Element     VR  VL   Value
```

Os principais componentes de atributos definidos pelo padrão são: *unique Attribute Tag*, *Value Representation* VR, *Value Length* VL, *Attribute Value*, *Unique Attribute Name*,

Attribute Description, *Value Multiplicity* VM (1, 2, 1-n,…), *Attribute Type* (1, 1c, 2, 2c, 3) etc.

O diagrama entidade-relacionamento ilustrado abaixo (figura 2.9), demonstra alguns objetos de informação ("*Information Objects*"), como *Patient, Image, Report*, representados pelos retângulos.

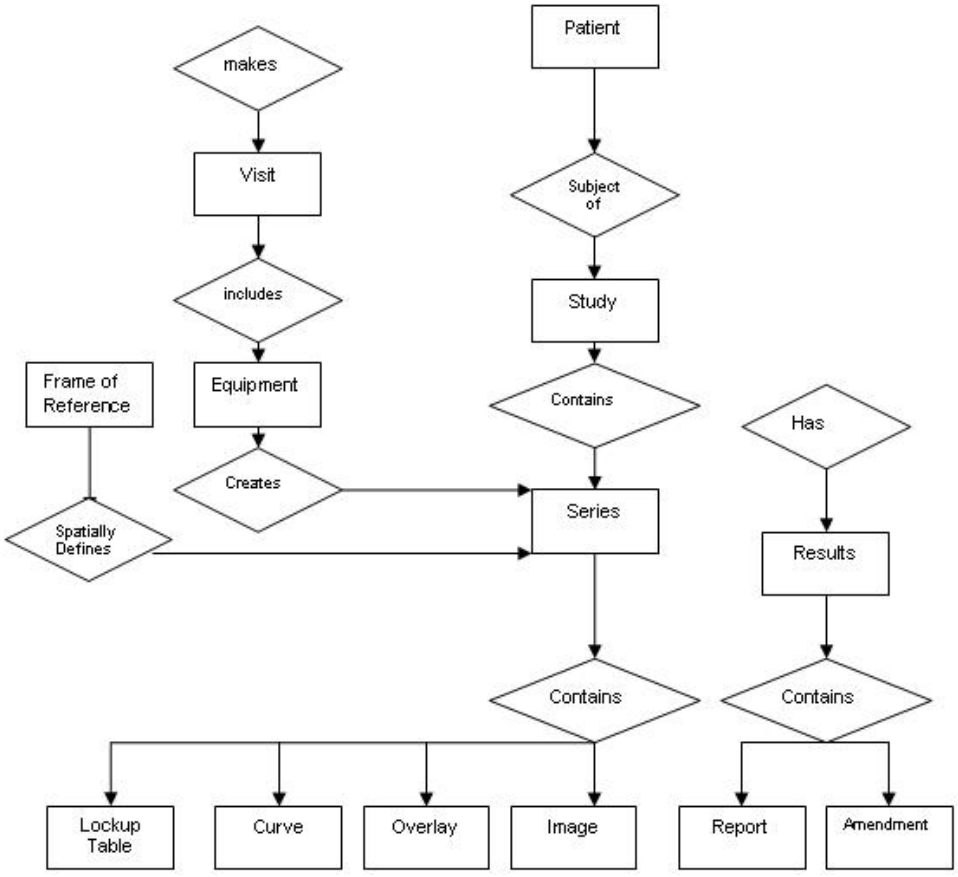

Figura 2.9 – Modelo Entidade Relacionamento DICOM [25].

Esse modelo exibe como as entidades estão interligadas. Os objetos possuem atributos, que são os elementos de dados do padrão DICOM. Para cada atributo, é determinada uma representação do valor (*Value Representation*) ou VR. Uma representação do valor descreve como um atributo é codificado em um elemento de dados. O conhecimento da representação do valor é dividido pelos parceiros na troca de informação. O processo de codificação/decodificação necessita de cuidado na seleção do VR (*Value Representation*) adequado para um determinado atributo, que é identificado pela sua etiqueta ou rótulo (*Tag*).

Esta informação pode ser compartilhada através de um dicionário de dados (*Data Dictionary*), o qual contém todos os atributos possíveis, trocados durante a comunicação ou simplesmente acrescentando a Representação do Valor como parte do Elemento de Dados. Apesar de incrementar a quantidade de informação a ser trocada entre os parceiros durante a comunicação, é bem mais flexível do que simplesmente compartilhar através de um dicionário de dados por se tornar difícil sua sincronização.

A mensagem é codificada com VR explícito (*Explict* VR) quando a representação do valor é incluída (tabela 2.2), e, no caso de não ocorrer essa inclusão, a codificação será feita com VR implícito *(Implicit* VR).

Tabela 2.2 – Representação VR explícito DICOM [25].

VR	NOME	VR	NOME
AE	*Application Entity*	**OW**	*Other Word String*
AS	*Age string*	**PN**	*Person Name*
AT	*Attribute Tag*	**SH**	*Short String*
CS	*Code String*	**SL**	*Signed Long*
DA	*Date*	**SQ**	*Sequence of Items*
DS	*Decimal String*	**SS**	*Signed Short*
DT	*Date Time*	**ST**	*Short Text*
FL	*Floating Point Single*	**TM**	*Time*
FD	*Floating Point Double*	**UI**	*Unique Identifier*
IS	*Image String*	**UL**	*Usigned Long*
LO	*Long String*	**UM**	*Unknown*
LT	*Long Text*	**US**	*Unsigned Short*
OB	*Other Byte String*		

Uma unidade de informação compõe um elemento de dados, que é determinado como uma entrada única no dicionário de dados. Esse elemento de dados é um atributo de um IOD (*Information Object Definition*) codificado, composto de um campo contendo uma etiqueta ou rótulo (*Tag*); de outro campo com o tamanho do valor (*Value Length*); e outro com o conteúdo do valor (*Value*).

Para algumas sintaxes de transferência específicas, um elemento de dado também contém um campo para armazenar o VR, onde a representação do valor (*Value Representation*) é explicitamente especificada.

2.3.2 Modelo de Comunicação

O processo de comunicação utilizado pelo protocolo é extremamente simples e robusto. Antes de iniciar a transmissão, cada comando DICOM é identificado e o emissor (conhecido como *Service Class User* – SCU) verifica se o receptor (conhecido como *Service Class Provider* - SCP) realmente compreende o tipo de serviço e o tipo de objeto que o SCU deseja utilizar. Quando um SCP não compreende o tipo de objeto, ele pode informar ao SCU, fornecendo, desta forma, um controle implícito de todo o processo de comunicação.

Tal negociação é conhecida como *handshake* DICOM e constrói uma Associação entre os pares SCU/SCP, estabelecendo o tipo do serviço (*Type Service*) e a sintaxe que será utilizada na transferência (*Transfer Syntax*) (figura 2.10). O conceito de *Transfer Syntax* nada mais é do que a codificação que será utilizada para a troca de informações. Um dispositivo poderia suportar o formato de sintaxe de transferência padrão ou utilizar uma norma específica que permite a compactação JPEG2000 e, assim, transferir o arquivo mais rapidamente.

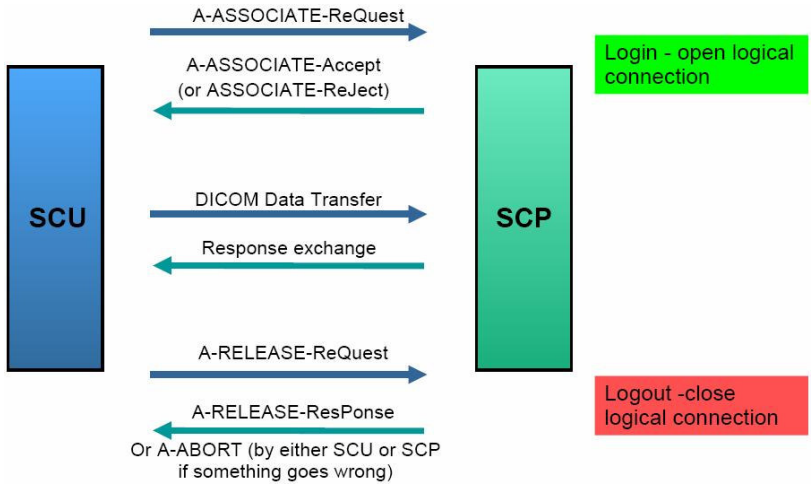

Figura 2.10 – Associação DICOM [25].

Uma mensagem DICOM é formada por dois elementos (figura 2.11) Commands Set e *DataSet*. O elemento de comando é sempre enviado com um campo "*Value Representation*" (VR) (*Implicit Little Endian*) e possui todos os elementos de comando (="atributos de serviço") necessários para definir o serviço. O *DataSet* consiste em todos os atributos do IOD da imagem que serão enviados.

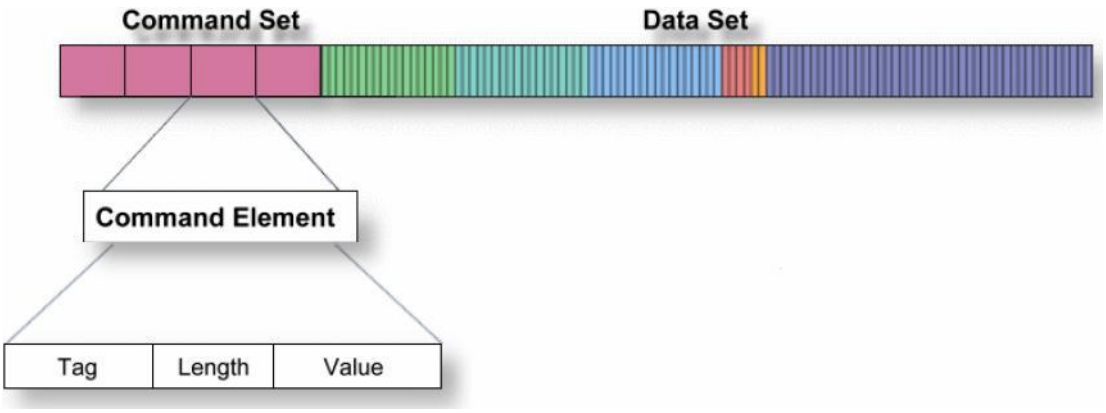

Figura 2.11 – Mensagem DICOM [25].

2.3.2.1 Comandos (*Commands* set)

O padrão DICOM define um conjunto de comandos "*Commands set*" para redes de comunicação que são mencionados como um "*DICOM Message Service Element*" (DIMSE). Cada serviço possui um identificador único (tabela 2.3) e emprega um subconjunto destes comandos para realizar um determinado serviço sobre a rede. Estes comandos são frequentemente ativados sobre uma instância de objeto. Os "*c-commands*" funcionam sobre instâncias de objetos compostas (*Composite Object Instance*) enquanto que "*n-commands*" atuam sobre instâncias de objeto normalizado (*Normalized Object Instance*). Por definição, um objeto é dito normalizado se ele representar apenas uma entidade de informação (IOD) e composto se representar várias entidades.

Tabela 2.3 – Comandos DICOM.

Comandos (*Commads*)	**Tipo (*Type*)**	**Classe de Serviço *(Service Class)***	**Descrição (*Description*)**
c-store	Composto	*Storage*	Transfere uma instância de objeto para uma AE (*application entities*) remota.
c-get	Composto	*Query/Retrieve*	Recupera uma instância de objeto de uma AE remota, cujos atributos coincidam com o conjunto de atributos especificados.
c-move	Composto	*Query/Retrieve*	Move uma instância de objeto de uma AE remota, cujos atributos sejam encontrados em um conjunto de atributos especificados para outra AE remota, ou até mesmo a própria.
c-find	Composto	*Worklist management*	Compara um conjunto de atributos com os atributos de um conjunto de instâncias de objetos de uma AE remota.

c-echo	Composto	*Verificação (verification)*	Verifica a comunicação ponto a ponto com uma AE remota.
n-event-report	Normalizado	*Storage Commitment*	Informa um evento para uma AE remota.
n-get	Normalizado	*Basic Print*	Recupera valores de atributos de uma AE remota.
n-set	Normalizado	*Modality Perf. Proc. Step*	Solicita a modificação de um atributo em uma AE remota.
n-action	Normalizado	*Storage Commitment*	Solicita uma ação por parte de uma AE remota.
n-create	Normalizado	*Basic Print*	Solicita que uma AE remota crie uma nova instância de objeto.
n-delete	Normalizado	*Basic Print*	Solicita que uma AE remota remova uma instância de objeto.

Para economizar mensagens e enviar a informação e o comando em um único pacote, o padrão criou o conceito de "*Service Object Pair*" (SOP) que se trata, apenas, da junção do identificador de um objeto DICOM (IOD) e o identificador de um Serviço (*Comands Set*). Dessa forma, o envio do número composto 1.2.840.10008.5.1.4.1.1.4, por exemplo, informa ao SCP que o IOD CT-Image está sendo enviado e que o serviço C-STORE deve ser executado.

Como no modelo orientado a objetos, uma Classe SOP define os campos (t*emplate*) enquanto a instância desta classe representa a imagem real do paciente. O principal atributo de uma instância é o SOP *Instance* UID que identifica a imagem de forma única para todo o sistema.

2.3.3 Considerações iniciais sobre segurança

Outra dimensão do DICOM está relacionada ao mecanismo de segurança. Observe que o padrão DICOM tem como objetivo facilitar a troca de informações e não requisitos para garantir a segurança das informações e da imagem do paciente. Para que a organização possa construir um ambiente seguro, é necessário observar vários aspectos que incluem, inclusive, necessidades de adequação e requisitos de segurança física.

Do ponto de vista lógico, no início deste texto foram mostradas as principais preocupações com a segurança da aplicação. De forma geral, deve ser estabelecido no software de visualização um controle de acesso com a utilização de senhas ou até mesmo de equipamentos

biométricos. Deve ser implementado, também, um mecanismo de auditoria e registro (*log*) que demonstre quem acessou o quê. Para finalizar, quando for utilizada uma linha de comunicação insegura entre cliente/servidor, deverão ser disponibilizados mecanismos de criptografia que garantam o sigilo e a autenticação das partes envolvidas [20].

2.3.4 Parte 15 – Perfis de segurança e sistema de gerenciamento de perfis

O padrão DICOM prevê apenas quais tecnologias poderiam ser utilizadas para implementar políticas de segurança no que diz respeito à troca de objetos DICOM entre aplicativos. A forma como estas tecnologias devem ser utilizadas, os critérios e permissões de acesso devem ser ditados pelo administrador da aplicação [26].

Resumidamente, questões como controle de acesso, registros de auditoria, proteção física, manutenção da confidencialidade e integridade de dados, mecanismos para identificar os usuários e os seus direitos de acesso aos dados são responsabilidade de cada entidade envolvida e depende, exclusivamente, do nível de confiança atribuído para o parceiro.

Um dos principais mecanismos citados pela norma é a opção de fornecer autenticação mútua para todas as entidades envolvidas e a garantia do sigilo do canal de comunicação utilizado. Para esse fim, é citado o uso do protocolo TLS junto com certificados digitais. Apenas como observação inicial, a biblioteca DCM4CHE atende esse requisito, ou seja, fornece um conjunto de métodos que permite o uso do TLS em um servidor PACS.

A norma divide o gerenciamento de segurança e sistema em perfis. Para cada perfil mostra um conjunto de tecnologias que o suporta. Os seguintes perfis são citados:

2.3.4.1 Perfil do usuário

Define, basicamente, os atributos e outras questões de segurança atribuídas aos usuários frente aos objetos DICOM. O armazenamento eletrônico *on-line* permite a aplicativos acompanhar e verificar o status das instâncias SOP nos casos em que o local de monitoramento das políticas de segurança exija o conjunto original de dataset e cópias subsequentes. A declaração de

conformidade deve indicar de que forma o sistema restringe o acesso remoto e controla os acessos a cada objeto DICOM.

2.3.4.2 Perfil para a conexão de transporte

Um perfil de segurança para a conexão de transporte descreve as seguintes informações [26]:

- Protocolo e framework utilizados pelos mecanismos de negociação;

- Modelo utilizado para a autenticação das entidades, incluindo: autenticação da identidade, o mecanismo pelo foram autenticadas e quaisquer considerações especiais para o *log* de auditoria e apoio;

- Modelo utilizado para a cifra dos dados transportados, incluindo: método de distribuição de chaves sessão e protocolo de cifra e parâmetros relevantes; e

- Descrição do mecanismo de integridade utilizado para a verificação dos dados.

Segundo o padrão, uma implementação básica do DICOM deve oferecer, no mínimo, suporte ao protocolo TLS v1.0 e aos algoritmos: autenticação de entidades (RSA baseado em certificados), troca de chaves secretas (RSA), verificação de integridade de arquivos (SHA) e privacidade (3DES, AES e CBC).

A aplicação deve abrir uma porta IP qualquer para aceitar apenas conexões TLS. É recomendável que os sistemas de suporte básico TLS abram conexões IP na porta número 2792. Esta recomendação não é uma obrigação, porém o número escolhido deve ser inserido na Declaração Conformidade.

A declaração de conformidade deverá, também, indicar quais os mecanismos foram utilizados para implementar a gerência de chaves. O perfil não deve especificar como uma conexão TLS é criada, ou o significado de quaisquer certificados trocados durante a autenticação das entidades. Estas questões são deixadas para a aplicação, que provavelmente estará seguindo

algum modelo de política pré-determinada. A identidade presente no certificado digital pode ser utilizada pela aplicação para log de auditoria ou para restringir o acesso a determinados objetos.

2.3.4.3 Perfil para a assinatura digital

Um perfil de assinatura digital consiste das seguintes informações [26]:

- O papel que desempenha a assinatura digital no sistema incluindo quem ou o que a entidade da assinatura digital representa, uma descrição da finalidade da assinatura digital e as condições sob as quais a assinatura digital está incluída no DataSet;

- Uma lista de atributos que devem ser incluídos na assinatura digital;

- Os mecanismos que serão usados para gerar ou verificar a assinatura digital incluindo o algoritmo utilizado para criar o MAC (*Message Authentication Code*), o algoritmo de cifra utilizado pelo sistema e o tipo de certificado ou o mecanismo de distribuição da chave que será utilizado;

- Quaisquer requisitos especiais para a identificação do signatário;

- O relacionamento com outras Assinaturas Digitais, se houver; e

- Quaisquer outros elementos necessários para criar, confirmar, ou interpretar a assinatura digital.

O algoritmo RSA é à base do perfil de assinatura digital utilizado pelo DICOM para cifrar, gerar o MAC e assinar. Esse perfil não especifica um determinado conjunto de elementos de dados utilizados para assinar.

Para criar uma assinatura digital, podem ser utilizadas as funções de *hash* RIPEMD-160, MD5 ou SHA-1 para gerar um MAC, que é, então, cifrado utilizando uma chave privada

RSA. Todos os validadores de assinaturas digitais devem ser capazes de usar um MAC gerado por qualquer uma das três funções *hash* citadas.

2.3.4.4 Perfil de segurança de armazenamento de mídia

Um perfil de segurança para mídia de armazenamento tem como objetivo permitir que arquivos DICOM sejam encapsulados de forma segura, garantindo aspectos de confidencialidade e integridade. As seguintes especificações devem estar contidas em um perfil [26]:

- Que aspectos de segurança são abordados pelo perfil;
- As restrições sobre os tipos de arquivos DICOM que pode ser garantido, se houver; e
- Como os arquivos DICOM são encapsulados e protegidos.

2.3.4.5 Perfis de gerência de endereços de rede

Os perfis de gerência de rede definem como são utilizados pelos equipamentos DICOM os protocolos que fornecem endereços de rede. Desta forma, os equipamentos recebem endereços e configurações de atributos da rede como endereços de servidores DNS, WINS, roteadores, etc. Um exemplo de protocolo dinâmico utilizado para esse fim é o DHCP [26].

2.3.4.6 Perfis de tempo de sincronização

Definem como são realizadas as operações de sincronismo de hora realizadas pelos equipamentos DICOM. Manter o sincronismo de hora é essencial para a auditoria, pois fornece condições de relacionar registros em outros equipamentos baseado no tempo [26].

2.3.4.7 Perfis de gerência de configuração da aplicação

Cada perfil descreve, de maneira geral, como os protocolos de rede não DICOM conseguirão obter endereços e capacidade de outros dispositivos de comunicação que estejam utilizando o protocolo DICOM. Esse perfil não foi detalhado, tendo em vista que sua implementação não faz parte deste projeto de pesquisa [26].

2.4 CRIPTOGRAFIA

O principal objetivo da criptografia é esconder informações de uma pessoa não autorizada. Seu uso permite implementar os principais conceitos de segurança computacional como integridade, confidencialidade, autenticidade e não repúdio [20].

Esconder seus segredos sempre foi um dos grandes desafios da humanidade. Os antigos generais precisavam transmitir informações para seus exércitos sem o perigo de ter suas mensagens interceptadas e traduzidas pelo inimigo. O uso da criptografia apareceu, possivelmente, nas primeiras guerras da antiguidade e seu primeiro relato de uso na história é atribuído ao imperador de Roma, Cesar [27].

Basicamente, um processo criptográfico envolve a aplicação de três conceitos elementares: a mensagem/texto, o algoritmo e a chave. A mensagem consiste, pura e simplesmente, na informação que se deseja transmitir de forma segura; o algoritmo é a forma que será utilizada para cifrar e decifrar uma mensagem; e a chave, em alguns modelos computacionais, pode ser entendida como o segredo compartilhado que deve ser conhecido apenas pelas duas partes envolvidas no processo de comunicação. A garantia da confidencialidade está em esconder do atacante o algoritmo ou a chave utilizada [28].

Um esquema de codificação criptográfica consiste em uma tupla (M, C, K, E e D) com as seguintes propriedades [28]:

- M é um conjunto conhecido como espaço de texto comum (*plaintext*);

- C é um conjunto conhecido como espaço de texto cifrado (*ciphertext*);

- K é um conjunto conhecido como espaço de chave;

- E é uma família de funções de codificação criptográficas tal que $E_k : M \rightarrow C$; e

- D é uma família de funções de decodificação criptográficas tal que $D_k : C \rightarrow M$;

2.4.1 Algoritmos criptográficos

O algoritmo criptográfico define a forma como a mensagem será cifrada e decifrada. A definição prévia do algoritmo pelas partes envolvidas (transmissor e receptor) é um dos fatores fundamentais no processo de comunicação seguro.

Os algoritmos criptográficos podem ser divididos em dois grandes grupos: algoritmos simétricos ou de chave secreta e algoritmos assimétricos ou de chave pública [28].

Os algoritmos simétricos utilizam uma única chave para cifrar e decifrar uma mensagem. O segredo do processo de comunicação consiste em manter essa chave em segurança. Embora possua um excelente nível de desempenho, o grande problema deste mecanismo é garantir que a chave privada seja transmitida de forma segura até o receptor. Este problema é conhecido como problema de distribuição da chave é se torna mais complicado à medida que se aumenta a quantidade de receptores da mensagem.

Exemplo de funcionamento: Alice e Bob compartilham uma chave secreta K e desejam se comunicar de maneira protegida. Alice cifra a mensagem P utilizando a chave K gerando uma mensagem cifrada C e envia para Bob que, por sua vez, aplica a chave K na mensagem cifrada C e recupera a mensagem P.

Os algoritmos de chave pública/assimétrico foram propostos por Diffie e Hellman no artigo *New Directions in cryptography* [29]. Nesse modelo, cada usuário possui um par individual de chaves (S, P) que devem respeitar duas condições fundamentais: a partir da chave "S" é

impossível calcular a chave "P"; e o que se faz com uma chave somente a outra desfaz, ou seja, quando uma mensagem for cifrada com a chave "S" ($E_S : M \rightarrow C$) somente a chave "P" poderá decifrá-la ($D_P : C \rightarrow M$).

Desta forma, como cada elemento envolvido na comunicação possui duas chaves, eles devem selecionar, individualmente, uma chave para ser sua chave pública(S) e outra para ser sua chave privada(P). A chave pública é divulgada para qualquer usuário enquanto a chave privada é mantida em segredo por seu proprietário.

Como exemplo, se Alice deseja enviar uma mensagem para Bob, Alice deve cifrar a mensagem utilizando a chave pública de Bob $E_{Bs} : M \rightarrow C$ que, ao receber a mensagem, decifra o texto utilizando sua chave privada $D_{Bp} : C \rightarrow M$. O algoritmo assimétrico resolve o problema de distribuição de chaves existente no algoritmo simétrico, porém, possui problemas de desempenho quando utilizado em mensagens grandes.

2.4.2 Assinatura digital

A idéia da assinatura digital é criar um mecanismo computacional que possa garantir, de forma similar a uma assinatura à mão, a autenticidade de uma mensagem. Os algoritmos de *hash* possuem características que vão ao encontro direto ao objetivo acima citado.

Neste tipo de algoritmo, a mensagem M, de um tamanho qualquer, é inserida em um algoritmo matemático, que gera um número, de tamanho fixo, que identifica o texto de forma única (*hash*). Bons algoritmos de *hash* possuem características que não permitem que duas mensagens diferentes (com sentido) produzam o mesmo *hash* e que pequenas mudanças na mensagem original produzam hash totalmente diferentes [27].

Assim, quando Alice desejar garantir para Bob que realmente fez a mensagem enviada, Alice deve calcular o *hash* da mensagem e cifrá-lo, utilizando sua chave privada (*hash-1*). Desta forma, somente a chave pública da Alice, que todos conhecem, será capaz de abrir a mensagem. Ao receber a mensagem e o *hash-1* cifrado por Alice, Bob calcula um novo *hash-2* da mensagem, decifra o *hash-1* enviado por Alice, utilizando a chave pública de Alice e

compara os dois *hash* (*hash-1 = hash-2*). Caso sejam iguais, a mensagem foi realmente feita por Alice e não foi alterada durante o envio para Bob.

Observe que o objetivo da assinatura digital não é garantir a confidencialidade da mensagem e sim sua autenticidade e o não repúdio. Na prática, nada impede que os algoritmos de criptografia possam ser utilizados, em conjunto, para garantir a confidencialidade da mensagem.

2.4.3 Certificados digitais

O problema da assinatura eletrônica está no fato de que se tem de filiar, de forma inequívoca, uma chave pública a seu proprietário. Um certificado digital consiste, resumidamente, em um documento que realiza esta associação entre chave e proprietário assinado por uma entidade em que tanto o receptor quanto o transmissor confiam plenamente.

A entidade que assina o certificado é conhecida como Autoridade Certificadora (AC) e é centro da confiança do sistema. A entidade responsável por filiar a chave pública ao seu proprietário e conhecida como Autoridade de Registro (AR) e pode exigir diversos documentos e, até, a presença física do proprietário para validar sua chave [20].

O certificado digital possui um formato padrão definido pela RFC 3280 (X.509 v3) e, resumidamente, possui os seguintes atributos (tabela 2.4):

Tabela 2.4 - Formato, resumido, padrão X.509 v3 [30].

Campo	OBS
signatureAlgorithm	Identifica o algoritmo de assinatura utilizado para a assinatura do certificado.
signatureValue	Contém o valor em BIT STRING do calculo da criptografia do certificado. Em outras palavras a assinatura.
Version	Contém a versão do certificado. A versão atual é a 3. O valor igual a 1 não permite a utilização dos campos de extensão do certificado. Estes campos existem somente na versão 3.
Serial number	Contém um número sequencial único utilizado para identificar o certificado na AC.
Signature	Este campo DEVE conter o mesmo identificador de algoritmo que o campo do signatureAlgorithm.

Issuer	O campo do issuer identifica a entidade que assinou e emitiu o certificado. DEVE conter um nome distinto não vazio (DN) e seguir o padrão X.501.
Validity	O período da validade do certificado é o intervalo de tempo durante o qual a AC manterá informações sobre o status do certificado. O campo é representado como uma SEQUÊNCIA de duas datas: a data em que o período da validade do certificado começa (notBefore) e a data em que o período da validade do certificado termina (notAfter). O notBefore e o notAfter podem ser codificados no formato UTCTime(YYMMDDHHMMSSZ) ou GeneralizedTime(YYYYMMDDHHMMSSZ). Certificados emitidos antes de 2050 devem utilizar o formato UTCTime.
Subject	Identifica a entidade associada ao certificado. DEVE possuir um nome distinto (DN) não vazio e seguir o padrão X.500.
Subject Public Key Info	Identifica a chave pública do certificado.
Unique Identifiers	Este campo é utilizado somente na versão 2 e 3. Os identificadores originais Subject e issuer atuais no certificado permitem reutilizar os nomes Subtect e/ou issuer. Este padrão não recomenda que nomes sejam reutilizados por diferentes entidades e que certificados da Internet para não empreguem Unique Identifiers. ACs em conformidade com este padrão NÃO DEVEM gerar certificados com Unique Identifiers. As aplicações em conformidade com este padrão DEVEM ser capazes de analisar gramaticalmente Unique Identifiers e fazer comparações.
Extensions	Este campo deve aparecer somente na versão 3. Se presente este campo é uma sequência de 1 ou mais extensões de certificado.

2.5 TLS – TRANSPOR LAYER SECURITY

O desenvolvimento do TLS se deu pela necessidade da concepção de um padrão aberto para conexões seguras, competente em afiançar integridade e privacidade aos dados que trafegam por redes TCP/IP.

O TLS foi baseado no protocolo *Secure Socket Layer* (SSL), desenvolvido pela empresa Netscape *Corporation*. Esse protocolo atua nas camadas de sessão e transporte, entre a camada de aplicação e o protocolo de transporte TCP, proporcionando dupla autenticidade, confidencialidade, garantia de integridade na troca de mensagens e não repúdio.

As propriedades fundamentais da conexão oferecida pelos registros TLS são [31]:

- Conexão privada. Criptografia simétrica é utilizada para cifrar os dados (DES, RC4, 3DES, etc). As chaves utilizadas na criptografia simétrica são sigilosas e exclusivas para cada conexão, formadas por negociação durante a etapa *handshake*.

- Conexão confiável. O deslocamento das mensagens inclui uma estrutura de averiguação de integridade empregando um MAC resguardado por chave. O MAC é gerado por funções de *hash* (SHA, MD5, etc).

O protocolo TLS possui duas camadas internas. O protocolo TLS *Record* é a camada mais baixa do protocolo e é utilizada para enviar todos os dados das camadas superiores. A camada superior deste protocolo consiste em múltiplos subprotocolos que são empregados no estabelecimento e na administração das conexões seguras entre as partes comunicantes. Estes protocolos são: TLS *Handshake*, TLS *Change Cipher Spec* e o TLS *Alert*.

A segurança de conexão fornecida pelo TLS *Handshake Protocol* tem três propriedades básicas [31]:

- A autenticação da identidade do "ponto/nó" que pode ser feita usando criptografia de chave pública (assimétrica);

- Negociação segura do *shared secret* (segredo compartilhado): o segredo negociado fica indisponível para qualquer elemento que esteja observando a conexão (*eavesdroppers*) e, nas conexões autenticadas, o segredo não pode ser obtido, mesmo que um invasor coloque-se no meio da conexão (*session hijacking*); e

- Negociação confiável: o invasor não pode modificar a comunicação de negociação sem ser detectado.

Certificados digitais podem ser utilizados pelo protocolo TLS, como forma de autenticar a identidade do cliente e/ou servidor e garantir a confidencialidade das informações trocadas. Cada certificado possui uma chave pública e o dono do certificado conserva uma chave

privada agregada a uma chave pública do certificado. O certificado é assinado por uma instituição fidedigna, conhecida por Autoridade Certificadora (AC).

Devido ao amplo volume de processamento matemático indispensável para processar as mensagens cifradas com qualquer algoritmo criptográfico de chave pública, o protocolo TLS emprega para as cifras das mensagens um algoritmo de criptografia de chave simétrica. A chave da sessão TLS (*master secret*) utilizada é fundamentada em valores casuais trocados entre as partes nas primeiras mensagens no protocolo TLS *handshake*. Com a utilização da chave pública, há garantia de que apenas o destinatário conseguirá desvendar o valor deste enigma. E esse número fortuito será utilizado na elaboração de sessão (*master secret*).

Inúmeros algoritmos simétricos estão previstos para uso neste protocolo. O TLS é um protocolo extensível, podendo, a qualquer momento, serem adicionados novos algoritmos para qualquer desses efeitos, desde que tanto o servidor quanto o cliente estejam conscientes dos novos acréscimos.

O TLS tem como uma de suas maiores vantagens a sua independência de protocolo de aplicação, o que permite que protocolos de camadas superiores possam ser colocados sobre ele de forma transparente.

As principais metas do protocolo TLS, em ordem de prioridade, são [31]:

- Segurança criptográfica: o TLS pode ser utilizado para estabelecer uma conexão segura entre duas partes;

- Interoperabilidade: programadores independentes podem desenvolver aplicações utilizando TLS que necessitem tocar parâmetros criptográficos sem precisar ter conhecimento dos códigos uns dos outros;

- Extensibilidade: o TLS procura fornecer um framework dentro do qual novos métodos de chave pública e criptografia possam ser incorporados quando necessário; e

- Eficiência relativa: como as operações criptográficas tendem a utilizar intensamente a CPU (principalmente operações de chave pública), o TLS incorporou um esquema opcional de *caching* para reduzir o número de conexões que precisam ser estabelecidas.

2.5.1 TLS *Handshake Protocol* – Versão 1.0 (RFC 2246)

O protocolo de troca (*handshake*) TLS define como os dados serão encapsulados e trocados entre entidades comunicantes. Cada pacote pode ser comprimido e adicionado a um MAC (*message authentication code*), ou cifrado, dependendo exclusivamente do estado em que se encontra a conexão. O pacote básico é composto por um campo *content type* que especifica o tipo do pacote, um campo de tamanho e um campo contendo a versão do TLS utilizada na comunicação.

Na solução proposta nesse trabalho, a autenticação do equipamento médico é um dos fatores de sucesso do projeto. Para cumprir a esse requisito, o TLS implementa um modelo de *handshake* específico que garante a autenticação mútua de ambas as partes envolvidas na comunicação. A seguir são detalhados os passos utilizados para este fim [31]:

- Um Cliente envia uma mensagem *ClientHello*, contendo a versão mais atual do protocolo que ele suporta, um número aleatório e uma lista de métodos de cifra e compactação;

- O Servidor responde com uma mensagem *ServerHello*, contendo a versão do protocolo escolhido, um número aleatório e o método de cifra e compactação escolhidos. O servidor pode, também, enviar um identificador para a sessão;

- O Servidor envia uma mensagem *ServerCertificate* contendo o seu certificado (dependendo do algoritmo de cifra escolhido);

- O Servidor envia uma mensagem *CertificateRequest,* solicitando um certificado do Cliente, de modo que a conexão possa ser autenticada mutuamente;

- O Servidor envia uma mensagem *ServerHelloDone*, indicando que está sendo realizada uma operação do tipo *handshake*;

- O Cliente responde com uma mensagem *Certificate* contendo o seu certificado;

- O Cliente envia uma mensagem *ClientKeyExchange*, que pode conter uma *PreMasterSecret,* a chave pública, ou nada (dependendo, se foi selecionado um algoritmo de cifra);

- O Cliente envia uma mensagem *CertificateVerify*, que é uma assinatura das últimas mensagens de *handshake* utilizando a chave privada do certificado do Cliente. Esta assinatura pode ser verificada usando a chave pública do certificado do Cliente. Isto permite que o servidor saiba que o cliente tem acesso à chave privada do certificado e, portanto, detém o certificado;

- O Cliente e o Servidor usam os números aleatórios e *PreMasterSecret* para calcular uma chave secreta comum, chamado de "*master secret"*. Todas as outras chaves utilizadas nesta conexão serão derivadas da *master secret* e dos números aleatórios gerados pelo Cliente e pelo Servidor;

- O cliente envia um pacote *ChangeCipherSpec,* informando ao servidor que tudo que for transmitido a partir deste momento deve ser cifrado;

- Finalmente, o Cliente envia uma mensagem *Finished* cifrada contendo o *hash* e o MAC usado em todas as mensagens *handshake;*

- O Servidor irá tentar decifrar a mensagem *Finished* enviada pelo cliente e verificar o *hash* e o MAC. Caso as verificações falhem, o *handshake* é considerado inválido e a conexão é reiniciada;

- Finalmente o Servidor envia a mensagem *ChangeCipherSpec* e a mensagem *Finished* cifrada. Caso as verificações falhem, o *handshake* é considerado inválido e a conexão é reiniciada; e

- Neste momento, o *handshake* foi concluído e o protocolo está ativo.

O formato do pacote TLS pode ser visualizado na tabela 2.5:

Tabela 2.5 - Pacote TLS v1.0 [31].

+	Bits 0–7	8-15	16-23	24–31
0	Tipo de conteúdo	Versão (MSB)	Versão (LSB)	Tamanho (MSB)
32	Tamanho (LSB)	Protocol Message(s)		
...	Protocol Message (cont.)			
...	MAC (opcional)			
...	Padding (somente para cifras de bloco)			

- *Content Type* (Tipo de conteúdo): identifica o tipo do conteúdo do pacote. Sã exemplos de valores válidos: *ChangeCipherSpec*, *Alert*, *Handshake* e *Application*;

- *Version* (Versão): identifica a maior e a menor versão suportada por quem gerou o pacote. Exemplo: SSLv3, TLS1.0, TLS1.1;

- *Length* (Tamanho): tamanho máximo do pacote. Por padrão, não pode exceder 2^14 bytes (16KiB);

- *Protocol Message*: uma ou mais mensagens. Observe-se que este campo pode ser cifrado, dependendo do estado da conexão;

- MAC: um código de autenticação de mensagem. Este campo pode ser cifrado, ou não incluído, inteiramente, dependendo do estado da conexão; e

- *Padding*: utilizado apenas para cifras de bloco.

2.6 DICOM UTILIZANDO TLS

O padrão DICOM assume que uma aplicação pode ser conectada a uma rede complexa e heterogênea e que dispositivos DICOM conectados a uma rede devem obedecer a certos perfis de segurança para que o sistema seja compatível e funcione corretamente. Além disso, define em sua norma de segurança que sistemas utilizarão este protocolo para comunicação, possuirão perfis definidos de acordo com as normas dos protocolos TLS, DHCP e LDAP para a troca de informações.

Um perfil de segurança nada mais é do que mecanismos de segurança (por exemplo, algoritmos de criptografia) que suportam funções de segurança.

Para uma aplicação DICOM suportar TLS, a parte 15 do padrão determina o uso do mecanismo de negociação especificado pelo protocolo TLS Versão 1.0. A porta a ser utilizada pelo TLS deve ser indicada na declaração de conformidade, bem como indicar qual mecanismo fundamental de gestão é suportado. Esta porta de comunicação não deverá ser utilizada por outros tipos de aplicações, sejam elas seguras ou inseguras. Sistemas DICOM que suportem este perfil usem a porta TCP 2762 registrada como "dicom-tls" para implementar o *upper layer* sobre TLS.

O uso do protocolo *Secure Transport Layer* (TLS) permite-lhe fazer uma comunicação segura DICOM. A Segurança reside na garantia dos aspectos de confidencialidade e integridade dos dados de todos os pontos da autenticação.

O perfil não especifica como uma conexão TLS é estabelecida, ou o significado de qualquer certificado trocado durante a autenticação. Uma vez que a Entidade de Aplicação tenha estabelecido uma conexão de transporte seguro, então uma associação em nível de *upper layer* poderá utilizar um canal seguro de dados.

2.7 DCM4CHE

A biblioteca dcm4che é uma coleção de utilitários e aplicações de código aberto desenvolvidas na linguagem de programação Java e que implementam o padrão DICOM. O

kit de desenvolvimento dcm4che-1.x é utilizado em muitas aplicações em todo o mundo, enquanto que o versão atual (2.x) da ferramenta tem sido refeita com o objetivo de aumentar o desempenho e flexibilidade de toda a solução. A versão 1.x possui os seguintes utilitários [5]:

- dcmdir: esta aplicação processa os file-sets dos arquivos de imagem de acordo com a parte 10 do padrão DICOM. Ela faz uso do serviço DICOMDIR para o armazenamento das imagens em um disco físico do servidor. A estrutura de armazenamento permite, até, oito subníveis com opção de seleção de diversas mídias de armazenamento;

- dcmrcv: esta aplicação implementa o servidor DICOM escutando em uma determinada porta TCP/IP. Esta é a implementação do serviço *Storage* SCP definido no padrão, ou seja, recebe a imagem e armazena em um local no sistema de arquivos ou em um file-set dependendo, exclusivamente, da opção de destino configurada;

- dcmsnd: esta aplicação envia objetos DICOM para um serviço de armazenamento Storage SCP. Está é a implementação do serviço *Storage* SCU definido no padrão, ou seja, os objetos são lidos em um sistema de arquivos local e enviados para o *Storage* SCP; e

- ImageJ: utilitário utilizado para visualizar objetos DICOM.

A figura 2.12 demonstra a relação entre cada utilitário. Observe a importância do serviço DICOMDIR como o centro do funcionamento de toda a arquitetura.

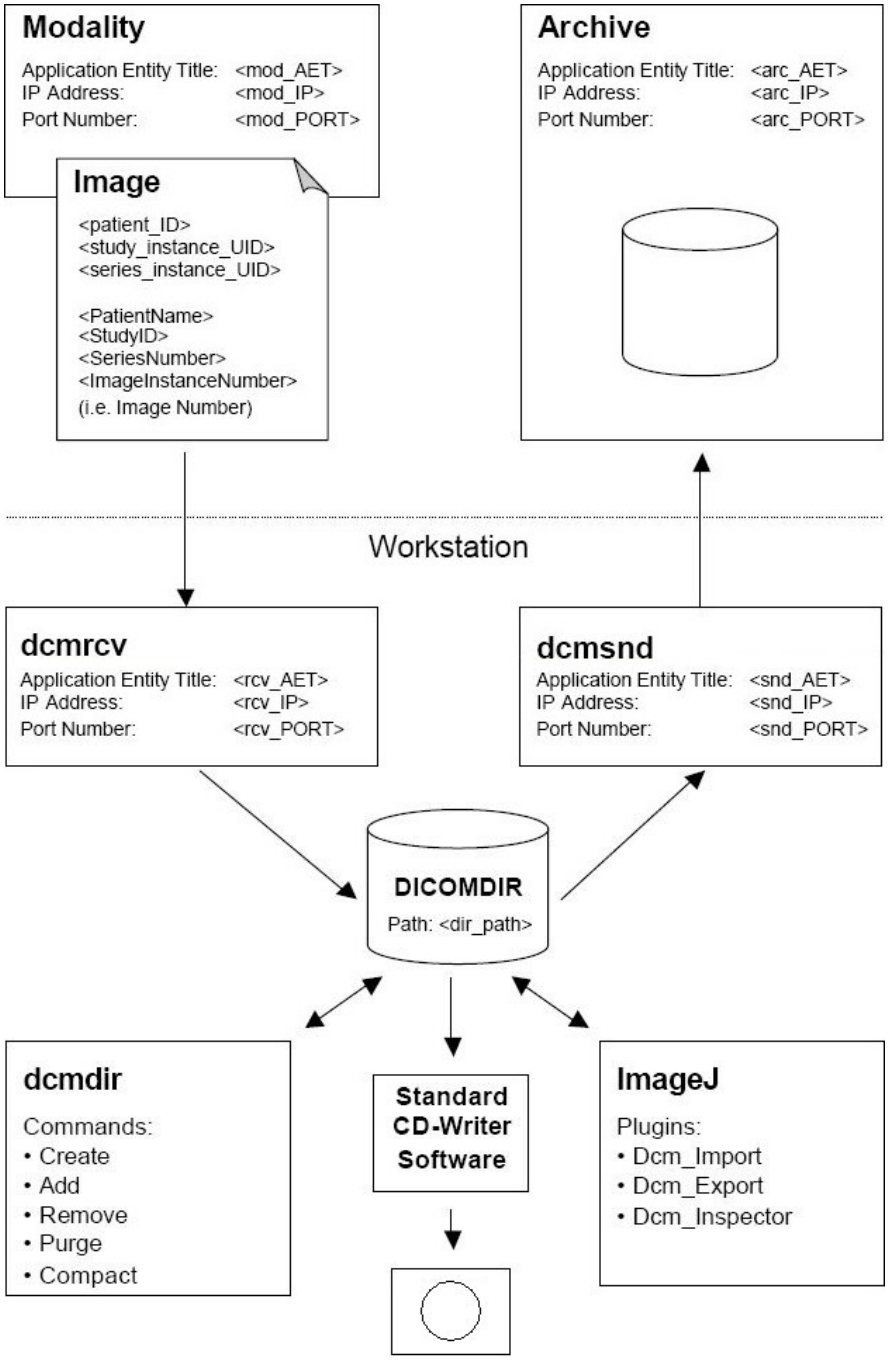

Figura 2.12: modelo de funcionamento básico do dcm4che.

2.7.1 Dcm4che – Aspectos de segurança

Obedecendo ao padrão DICOM, o dcm4che implementa dois mecanismos próprios de garantia de segurança. O primeiro é baseado na AE (*Association Entity*) e pode ser facilmente burlado caso um atacante consiga descobrir esses valores. O segundo é baseado no uso do

protocolo TLS como forma de garantir da confidencialidade e autenticidade sendo a forma de uso proposta nessa dissertação [5].

Como já descrito, o dcm4che foi desenvolvido, utilizando a linguagem de programação Java e, por consequência, a implementação do TLS utilizada faz uso dos frameworks de comunicação segura *Java Secure Socket Extension* (JSSE) e dos algoritmos implementados pela *Java Cryptographic Architecture*.

Como o TLS é baseado no SSL, é utilizado para autenticação das partes envolvidas na comunicação um algoritmo de chave assimétrico enquanto a garantia de sigilo e integridade é provida por um algoritmo de chave simétrico. Para resolver o problema de distribuição da chave simétrico já citado, o *handshake* do protocolo faz com que o emissor gere uma chave aleatória (chave simétrica) e envie para o receptor, usando o algoritmo assimétrico. Observe-se, apenas, que os certificados exigidos para autenticação devem estar disponíveis em cada uma das partes comunicantes [31].

O JSSE usa o conceito de *keystore* (deposito de chaves) definido pela JCA. Baseado no uso das chaves, um *keystore* pode ser de dois tipos [32]:

- keystore: contém chaves e certificados usados para autenticar a entidade junto da entidade remota (inclui a chave privada e o certificado emitido por uma AC);

- truststore: contém e certificados usados para autenticar entidades remotas (inclui certificados de autoridades de certificação);

O armazenamento seguro destas chaves é implementado através de arquivos cifrados, utilizando uma senha. Desta forma, para usar um certificado/chave, é necessário que cada uma das entidades possua informação das respectivas senhas utilizadas para cifrar os certificados digitais envolvidos [32].

No JSSE atual foi implementada a versão 1.0 do TLS (RFC 2246) [31], porém já se encontra disponível a versão 1.1 (RFC 4346) [33].

De forma geral, a implementação do TLS realizada pelo dcm4che necessita dos seguintes atributos para funcionar corretamente:

- EnabledProtocols: Lista as versões de protocolo habilitadas para funcionar na conexão. Exemplo: TLSv1, SSLv3;

- *SupportedProtocols*: versões de protocolos utilizados pelo provedor SSL. Exemplo: DES, 3DES, SHA-1, MD5;

- *KeyStoreURL*: URL que contém o endereço onde está localizado o certificado que será utilizado pelo cliente quando o servidor estiver utilizando o modo TLS;

- *KeyStorePassword*: senha utilizada para acessar o certificado especificado pelo atributo *KeyStoreURL*;

- *TrustStoreURL*: URL que contém o endereço onde está armazenado o certificado da AC (autoridade certificadora) quando o servidor estiver utilizando o modo TLS; e

- *TrustStorePassword*: senha utilizada para acessar o certificado especificado pelo atributo *TrustStoreURL.*

2.8 TRABALHOS RECENTES PUBLICADOS NA ÁREA

O objetivo deste tópico é fornecer uma visão sobre os principais trabalhos publicados na área de pesquisa desta dissertação. Os artigos abaixo citados buscam fornecer mecanismos que garantam a confidencialidade, disponibilidade e integridade das informações do paciente. Ao final deste tópico é realizada uma breve comparação entre o software desenvolvido e proposto neste trabalho com os artigos abaixo selecionados.

2.8.1 Combinação de dados heterogêneos (imagens e textos) em uma única imagem com o objetivo de esconder dados do paciente

A garantia de sigilo das informações de um paciente é um fator de extrema importância, principalmente quando falamos em transmissão de informações por um ambiente público como a Internet. Miaou-Shaou-Gang (et al) [34] apresenta uma técnica usada para cifrar e decifrar registros de pacientes (*Eletronic Patients Records* - EPR) para transmiti-los entre hospitais, através da Internet, visando confidencialidade, integridade e disponibilidade da informação.

Em seu artigo ele propõe a combinação de dados heterogêneos (imagens e textos) em uma única imagem, combinando valores de pixels e caracteres por meio de um sistema de codificação que utiliza múltiplas bases numéricas bipolares. Essas bases são geradas a partir de cálculos entre os valores dos pixels das imagens, que são convertidas para o formato JPEG e os respectivos valores em escala de cinza das imagens original e JPEG [34].

Na geração da imagem codificada a impressão digital dos médicos proprietários é incluída com o objetivo de prover um sistema de autenticação para o dado. Ela é codificada em conjunto com os demais registros de pacientes em uma única imagem estática. Exemplo: segmentos de eletrocardiogramas, relatórios textuais de sintomas/diagnósticos, assinaturas digitais, e etc [34].

A base gerada é bipolar (inclui números negativos) para aumentar o número de elementos e, consequentemente, a capacidade de codificação do algoritmo. A codificação é feita por intemédio de operações aritméticas e pseudo-randômicas que, com cálculos sobre os valores de pixels das imagens e dos elementos das bases bi-polares, embutem as imagens e os elementos textuais que compõem o EPR (*Electronic Patient Record*) a ser transmitido. Os documentos textuais recebem um tratamento similar após a conversão dos caracteres para uma representação numérica obtida a partir de operações aritméticas sobre os respectivos valores dos caracteres na tabela ASCII (*American Standard Code for Information Interchange*) [34].

A decodificação para restauração dos elementos originais implica no processo inverso e requer a imagem inicial utilizada no processo de codificação, o algoritmo e a imagem resultante do processo de codificação [34].

2.8.2 Técnica de criptografia de imagens médicas transmitidas utilizando o formato JPEG 2000

Brahimi-Zahia (et al) [35] propõe um esquema de criptografia de imagem médicas baseado no padrão JPEG 2000, onde apenas blocos da imagem considerados sigilosos são cifrados. A técnica combina o processo de criptografia seletiva com a permutação de blocos para aumentar o nível de segurança da imagem e minimizar o seu tempo de criptografia.

Segundo descrito neste artigo, blocos da imagem são agrupados em áreas (*precincts*), que são classificadas segundo o nível de segurança requerido. Áreas sigilosas são cifradas e permutadas com áreas não cifradas para obter a melhor degradação possível da imagem a ser transferida, respeitando os limiares de qualidade PSNR (*Peak Signal-to-Noise Ratio*) para a degradação. Os blocos de imagens são cifrados com AES (*Advanced Encryption Standard*), usando o modo CFB (*cipher feedback*) para cifragem [35].

2.8.3 Comunicação segura de imagem DICOM utilizando os protocolos IPv6/IPv4

Jianguo Zhang (et al) [36] propõe a implementação da camada de troca ou transferência de imagens, segundo o padrão DICOM, como camada superior para o protocolo TCP/IPv6, mantendo a compatibilidade com o IPv4.

Características como tempo de transferência, custo e segurança dos dados são relevantes para a transmissão de imagens médicas via Internet. As redes WANs (*Wide Area Network*) são comumente usadas para transmitir esse tipo de dado e são consideradas lentas, enquanto que WANs de alta velocidade apresentam um custo elevado. Como alternativa para prover maior velocidade com um custo aceitável, WANs vêm sendo usadas com a versão 6 do protocolo IP (IPv6) e apresenta vantagens sobre o IPv4 relacionadas a aspectos como maior capacidade de endereçamento e avanços em áreas como segurança, mobilidade e desempenho. Por exemplo, IPv6 inclui outros protocolos associados como IPSec e ICMPv6 [36].

A maioria das operações de troca de imagens médicas utiliza os serviços de comunicação definidos no DICOM para realizar transferências entre servidores PACS, Workstations, sistemas de teleradiologias, dentre outros [36].

A maioria das aplicações e protocolos de redes usadas na Internet são desenvolvidas para IPv4, inclusive soluções que implementam serviços DICOM. Neste artigo foi selecionada uma dessas soluções, baseada em IPv4 e em software livre, para adaptação ao IPv6 de modo que ambas permanecessem compatíveis e capazes de trocar imagens entre si [36].

Experimentos foram realizados sobre combinações de configurações que variavam o sistema operacional, protocolo (IPv6 ou IPv4), configurações e algoritmos de segurança (IPSec, SSL/TLS) e unidades de dados (PDU) para comparação do desempenho em cada caso. Foi constatado que o uso de bibliotecas e softwares DICOM é facilmente adaptado para o desenvolvimento de protocolos de comunicação que suportem tanto IPv4 quanto IPv6. No entanto existem alguns *overhead* na utilização do IPv6 devido às suas novas funcionalidades. Por outro lado, quando há intenção de uso do IPSec, é mais vantajoso utilizar o IPv6 devido à integração de ambos [36].

2.8.4 Autenticação de imagens médicas utilizando funções *hash* e transformada inteira de *wavelet*

Cheng-Ri Pião (et al) [37] propõe um novo algoritmo de autenticação capaz de detectar modificações nas imagens com a utilização de marcas d'água digitais frágeis. O algoritmo proposto adapta a transformada de *wavelet* (função capaz de decompor e descrever outras funções no domínio da freqüência, de forma a podermos analisar estas funções em diferentes escalas de freqüência e de tempo) para gerar números inteiros ao invés de utilizar as transformadas discretas de *wavelet*, comumente usadas para este tipo de marca d'água digital. A transformada inteira de *wavelet* mapeia inteiros para inteiros e permite, com isso, a reconstrução exata da imagem transformada, sem perda de informação [37].

Para inserção da marca d'água, a imagem é inicialmente transformada em um domínio de tempo-frequência através da transformada inteira de *wavelet*. Os valores dos bits mais significativos e o tamanho da imagem são submetidos a uma função *hash.* Os bits mais

significativos são combinados aos valores resultantes da função *hash* para gerar os coeficientes de marca d'água. Esses coeficientes são usados na transformada inversa de *wavelet* para inserir a marca d'água na imagem [37].

No processo de extração, é realizado o processo inverso e é possível detectar modificações feitas por meio de ataques à imagem através do valor dos coeficientes de marca d'água que, quando diferentes dos originais, indicam que a imagem sofreu algum tipo de alteração [37].

2.8.5 Sistema para auditoria de imagem médica de acordo com a norma HIPAA

Xiaomeng Chen (et al) [38] apresenta um sistema de auditoria baseado nas normas de privacidade e segurança especificadas pela HIPAA (*Health Insurane Portability and Accountability Act*), para sistemas de imagens médicas como PACS e RIS (*Radiology Information System*).

O sistema de auditoria proposto foi implementado sobre uma plataforma J2EE (Java2 *Platform Enterprise Edition*), JNI (*Java Native Interface*) e RMI (*Remote Method Invocation*) para troca de mensagens, XML (*Extensible Markup Language*) para formatação das mensagens e *hibernate* como camada de persistência entre a aplicação e o repositório de auditoria. Registros (l*ogs*) de eventos relativos ao uso de informações médicas protegidas gerados em *workstations* e *gateways* são capturados pelo sistema e formatados em mensagens XML. Essas mensagens são enviadas ao servidor de aplicação para armazenamento no repositório de auditoria permitindo, assim, a sua análise posterior para detecção de uso inadequado da informação [38].

2.8.6 Armazenamento e gestão de dados biomédicos cifrados em cenários reais

I. Blanquer (et al) [39] apresenta um modelo de armazenamento e gerenciamento de dados médicos criptografados, em um ambiente grid (modelo computacional capaz de alcançar uma alta taxa de processamento, dividindo os processos entre diversas máquinas), que utiliza um esquema de chaves de criptografia compartilhadas entre as diferentes organizações participantes no grid, para prover a confidencialidade e a integridade dos dados médicos manipulados por aplicações em ambiente grid.

O objetivo deste artigo é propor uma arquitetura flexível e configurável a fim de prover um esquema de compartilhamento de chaves, controle de acesso, gerenciamento de criptografia e auditoria adequado ao ambiente usado. A metodologia descrita foi adotada em um cenário real de 5 (cinco) hospitais, atendendo à necessidade de gerenciamento de objetos gerados segundo o padrão DICOM [39].

Neste artigo é levado em consideração que os dados trafegados no Grid são provenientes de um conjunto de organizações participantes de um esquema de compartilhamento, onde cada organização possui certificado digital e serviços de compartilhamento próprios no GRID. As organizações são propritárias de seus dados e, por isso, controlam-nos, são responsáveis por protegê-los de acesso não autorizado [39].

Informações necessárias para identificar as organizações são registradas no cabeçalho dos respectivos objetos, quando criptografados. Os proprietários dos objetos são capazes de identificar as organizações participantes do esquema de compartilhamento e fornecer as chaves para decifrar o objeto para as organizações interessadas. Os mecanismos de autorização utilizados são baseados em um conjunto de atributos de usuários que permitem diferenciar clientes anônimos de clientes autorizados e habilitar diferentes operações para cada perfil de cliente dentro do ambiente do Grid [39].

Os elementos principais desta metodologia são o reconhecimento e a organização das responsabilidades das organizações participantes no grid, para formar uma cadeia de responsabilidades capaz de garantir a proteção dos dados [39].

2.8.7 Sistema de segurança para troca de dados em tele-medicina

Slobodan Kovacevic (et al) [40] apresenta a implementação de um sistema para transferência segura de dados em telemedicina, recurso que provê cuidado médico remoto com o uso de telecomunicação, tecnologia e infraestrutura da informação. Utiliza uma infraestrutura de chave pública (PKI) para permitir a aquisição de chaves públicas de forma segura, conveniente e eficiente. Uma PKI consiste de um conjunto de *hardwares, softwares*, pessoas, políticas e procedimentos, necessário para criar, gerenciar, armazenar, distribuir e revogar certificados digitais baseados em criptografia assimétrica.

A arquitetura do sistema proposto utiliza web *services* no lado do servidor para realizar a autenticação dos clientes e as trocas de mensagens. No lado do cliente são realizadas a criptografia dos dados, e as funcionalidades de sinalização e de comunicação com o servidor. A comunicação entre clientes e servidor utiliza o protocolo SOAP (*Simple Object Access Protocol*) e certificados de autenticação para identificar ambas as partes. Apenas clientes com certificados válidos para o servidor terão acesso. *Smart Cards* são usados para armazenar os certificados digitais e chaves privadas utilizados para criptografia dos dados [40].

A maioria dos dados médicos transferidos consitem de imagens que são compactadas antes da transferência com o objetivo de preservar a informação. É utilizada compressão sem perda por meio de predição de coeficientes (*predictive coding*) causal. Nesse tipo de compressão, o contexto dos pixels vizinhos é utilizado para realizar a predição do pixel corrente [40].

2.8.8 Segurança do processo de distribuição de imagens médicas utilizando a tecnologia JINI

Jacek Ruminski [41] apresenta um ambiente de processamento distribuído para a troca de dados entre organizações de saúde (*Health Care Organization* - HCO), utilizando arquitetura grid baseada em JINI e JavaSpaces sobre o padrão mestre-escravo. O objetivo é prover o processamento de imagens paramétricas (*parametric image*) em ambiente distribuído onde o cálculo da imagem é subdividido em várias tarefas executadas de forma independente por processos escravos. Nesse ambiente os processos mestres são responsáveis por garantir a integração dos resultados de cada tarefa e disponibilizar o resultado final.

Métodos de segurança relativos à integridade de código, confidencialidade, autenticação cliente-servidor, autorização e auditoria são implementados para manter a segurança durante o processamento de imagens, conforme recomendações do padrão HIPAA.

Segundo a tecnologia JINI (arquitetura de rede para a construção de sistemas distribuídos), os serviços requisitados pelo cliente são executados localmente após o download do respectivo código. Para isso, o cliente deve ser autenticado e autorizado a utilizar o serviço requisitado. Todas as ações dos clientes são registradas em registros (*logs)* de auditoria e os códigos transferidos são verificados após o download para garantir que sejam os originais.

Os recursos usados para prover os aspectos de segurança relacionados foram quase todos implementados com recursos JAVA, inerentes de bibliotecas JINI, RMI, JASS (*Java Authentication and Authorization Services*) e Java *Logging*. A confidencialidade foi obtida diretamente através de SSL [41].

2.8.9 Transmissão segura de registros médicos utilizando a esteganografia de alta capacidade

Yeshwanth (et al) [42] apresenta o *Bit-Plane Complexity Segmentation* (BPCS), um esquema de esteganografia de imagens de alta capacidade. O BPCS foi introduzido para tratar pontos fracos de técnicas tradicionais para ocultar dados como a do bit menos significativo (LSB – *Last Significant Bit*) e utiliza um método onde até mesmo o plano de bits mais altos pode ser usado para ocultar informações.

Neste método, o plano de cor de uma imagem RGB é decomposto em 8 planos de bit (*bit-planes*) para cada cor, num total de 24 planos de bit por imagem. Subdividida em blocos de tamanhos variáveis, a imagem é codificada por fases, bloco a bloco, utilizando uma chave específica para cada bloco codificado, calculada em função do tamanho de cada bloco. Durante a codificação, é calculada a complexidade de distribuição de bits de cada bloco e aqueles que apresentam complexidade maior do que um limiar adotado são usados para codificar as informações em seus planos de bits mais significativos.

É proposto, também, um esquema de multicamadas que provê um nível maior de encapsulamento para registros sigilosos de pacientes e aumenta a capacidade de ocultamento por meio de um método onde imagens ocultadas em outras imagens podem ter outras informações já embutidas [42].

2.9 RELAÇÃO ENTRE TRABALHOS PUBLICADOS NA ÁREA E OS PROJETOS HUB2PACS E HUB2CONVERSOR

A tabela 2.6 demonstra, resumidamente, os principais artigos pesquisados e publicados na área desta pesquisa. Observe que os textos podem ser agrupados em dois grandes grupos. O primeiro trata de técnicas de segurança que buscam modificar a imagem, seja para armazená-

la de forma cifrada ou para criar uma assinatura digital. O segundo grupo preocupa-se com a transmissão de uma imagem em uma rede.

Tabela 2.6 – Comparação entre os trabalhos publicados na área.

Autor	Proposta	Grupo
Shaou-Gang (et al) [34]	Combinar dados heterogêneos (imagens e textos) em uma única imagem com o objetivo de esconder dados do paciente.	Criptografia da imagem independente do local de armazenamento
Brahimi-Zahia (et al) [35]	Criptografia de imagens médicas utilizando o formato JPEG 200	
Cheng-Ri Pião (et al) [37]	Autenticação de imagens médicas utilizando funções hash e transformada inteira de wavelet	
Xiaomeng Chen (et al) [38]	Sistema para auditoria de imagem médica de acordo com a norma HIPAA	
Yeshwanth (et al) [42]	Transmissão segura de registros médicos utilizando a esteganografia de alta capacidade	
I. Blanquer (et al) [39]	Armazenamento e gestão de dados biomédicos cifrados em cenários reais	
Slobodan Kovacevic (et al) [40]	Sistema de segurança para troca de dados em tele-medicina	Troca de informação e comunicação segura
Jianguo Zhang (et al) [36]	Comunicação segura de imagem DICOM utilizando os protocolos IPv6 e IPv4	
Jacek Ruminski [41]	Segurança do processo de distribuição de imagens médicas utilizando a tecnologia JINI	

Infelizmente não é possível utilizar as técnicas propostas pelo primeiro grupo de artigos da tabela 2.6. O projeto GsWeb possui limitações referentes ao visualizador de imagem que ainda não foi adaptado para ler uma imagem de forma cifrada ou assinada.

Por sua vez, os projetos hub2pacs e hub2conversor proposto nesta pesquisa buscam autenticar o equipamento médico e o servidor e garantir o sigilo do canal de comunicação. Esse comportamento possui relação direta com o segundo grupo de artigos da tabela 2.6.

No artigo de Jianguo Zhang (et al) [36], é proposto o uso do protocolo IPSec (*IP Security*) para garantir o sigilo do canal de comunicação. O problema desse método é que o

equipamento médico deve oferecer suporte para IPSec o que, na prática, ainda não foi implementado ou normalizado pelo padrão DICOM. Nesta comparação, o hub2pacs utiliza o formato DICOM-TLS constante da parte 15 do padrão DICOM e, por conseqüência, possui maior compatibilidade com os diversos fabricantes.

Já o artigo de Jacek Ruminski [41] apresenta um ambiente de processamento distribuído para troca de dados entre organizações de saúde. O hub2pacs permite que equipamentos médicos em outros hospitais possam transmitir imagens médicas de forma segura para um único servidor central. A funcionalidade de processamento de imagens pode ser centralizada neste servidor bem como a criação e a confecção de laudos médicos.

O trabalho de Slobodan Kovacevic (et al) [40] utiliza uma infraestrutura de chave públicas para a transferência de dados em um sistema de tele-medicina. O hub2pacs também faz uso de PKI afinal autentica o equipamento médico utilizando certificados digitais.

3 PROTÓTIPO DO NOVO SISTEMA

O propósito deste capítulo é expor as necessidades e funcionalidades gerais do novo servidor PACS e do conversor DICOM desenvolvidos para o Hospital Universitário de Brasília (HUB). Durante a fase de análise de requisitos, realizada em agosto de 2006, foram identificadas as seguintes necessidades:

- Construir um servidor PACS que permita a transmissão de imagens médicas de forma cifrada e autenticada utilizando certificados digitais (TLS);

- Criar, utilizando o *OpenSSL*, uma Autoridade Certificadora (CA) para o HUB e gerar dois certificados digitais, uma para o equipamento médico e outro para o servidor PACS;

- Construir um módulo para a aplicação que permita adaptar os recursos de segurança DICOM/TLS a equipamentos médicos que não possuam essa funcionalidade; e

- Garantir que a nova aplicação desenvolvida não interfira no funcionamento do sistema de laudos médicos em uso no HUB.

Os sistemas propostos neste trabalho foram desenvolvidos em java, utilizando a biblioteca dcm4che e o banco de dados postgres. Todas as informações e estruturas de tabelas foram respeitadas permitindo, dessa forma, o completo aproveitamento de todas as aplicações já desenvolvidas e disponibilizadas no servidor PACS do HUB. Para explicar o funcionamento do novo servidor PACS implementado, foi utilizado artefatos como o documento de visão que busca demonstrar os objetivos do projeto e a diagramação UML (*Unified Modeling Language*) utilizada para representar os casos de uso dos sistemas, os diagramas de classe e os diagramas de sequência dos principais métodos implementados.

3.1 PROJETO HUB2PACS

A seguir, são apresentados o documento de visão, o diagrama de caso de uso, o diagrama de classe e dois principais diagramas de sequência do projeto hub2pacs.

3.1.1 Documento de Visão

O documento de visão é um artefato utilizado para demonstrar os requisitos técnicos e formais de um software. Seu objetivo é fornecer uma visualização de alto nível que permita a equipe envolvida compreender aspectos e restrições do sistema, identificar, de forma clara, o problema e servir como base reguladora para todas as decisões futuras.

3.1.1.1 Descrição do Problema

Neste tópico, é fornecida uma descrição resumida do problema que está sendo resolvido pelo projeto. A tabela 3.1 detalha o problema que motivou a criação do projeto, quem é afetado, o impacto e a solução proposta para a correção.

Tabela 3.1 – Descrição do Problema.

O problema de	Quebra de sigilo das imagens do paciente.
Afeta	Paciente, Hospital.
Cujo impacto é	Perda da credibilidade e perda da imagem da organização devido a processos judiciais por vazamento de informações do paciente.
Solução proposta	Desenvolver um servidor PACS que permita a autenticação e o sigilo utilizando o protocolo TLS.

3.1.1.2 Sentença de Posição do Produto

A sentença de posição do produto apresenta os principais concorrentes do software proposto a ser desenvolvido e quais o fatores o diferenciam de seus concorrentes de mercado. Na tabela 3.2 são apresentadas essas diferenças.

Tabela 3.2 – Sentença de Posição do Produto.

Para	Hospital Universitário de Brasília (HUB).
Que	Necessita de um servidor PACS que garanta a autenticação do equipamento médico e a segurança das informações do paciente.
O sistema	Hub2pacs.
Que	Permite a transmissão de imagens médicas cifrando o conteúdo e autenticando o equipamento médico, utilizando certificado digital.
Diferente de	Dcm4che, MiniWebPACS.
Nosso sistema	Permite a autenticação e cifra do conteúdo da mensagem, respeitando o modelo de dados em uso no Hospital Universitário de Brasília (HUB).

3.1.1.3 Resumo dos Envolvidos

Qualquer projeto de construção de software deve identificar e considerar todas as necessidades dos usuários envolvidos durante o processo de Modelagem de Requisitos. A tabela 3.3 fornece o perfil dos usuários que integram o projeto e detalha sua visão/responsabilidades no uso da aplicação desenvolvida.

Tabela 3.3 – Resumo dos Envolvidos.

Nome	Descrição	Responsabilidades
Paciente	Cliente do Hospital.	É quem realiza os exames médicos fornecendo imagens para o sistema, auxiliando o médico na confecção de laudos.
Médico	Pessoa responsável por operar o equipamento de exame médico.	Operar o equipamento médico e realizar a coleta de imagens do paciente.
Administrador	Administrador geral do sistema.	Definir os parâmetros de configuração e inicializar o servidor Hub2pacs.

3.1.1.4 Resumo dos Usuários

Diferente da tabela 3.3, que mostra todos os usuários envolvidos, a tabela 3.4 apresenta, apenas, os elementos que irão, diretamente, utilizar a aplicação desenvolvida e suas respectivas responsabilidades.

Tabela 3.4 – Resumo dos Usuários.

Nome	Descrição	Responsabilidades no Sistema
Administrador	Administrador geral do sistema.	Definir os parâmetros de configuração e inicializar o servidor Hub2pacs.

3.1.1.5 Ambiente do Usuário

O ambiente para uso do servidor DICOM no Hospital Universitário de Brasília (HUB) é composto, basicamente, dos seguintes itens:

- Rede de comunicação ethernet utilizada para permitir a troca de informações entre o equipamento médico, o servidor PACS e o servidor de visualização de imagens.

- Servidor PACS (MiniWebPACS) que permite o recebimento e armazenamento de imagens. Esse software gratuito foi desenvolvido pelo INCOR–SP, utilizando a linguagem de programação JAVA, a biblioteca pública dcm4che e o banco de dados Postgres;

- Equipamento médico utilizado para realização de exames; e

- Software em Java (GsWeb) utilizado para visualizar imagens DICOM e permitir a confecção de laudos médicos.

O único recurso de segurança implementado pelo MiniWebPACS faz uso dos campos *AETitle* para identificar o equipamento médico e o servidor DICOM. O grande problema desse esquema de autenticação é o fato de que esses campos e os dados da imagem são trocados via rede em texto claro, permitindo, dessa forma, que um invasor possa identificá-los e enviar imagens se passando pelo equipamento médico ou remontar a imagem a partir de pacotes capturados.

3.1.1.6 Principais Necessidades dos Usuários ou dos Envolvidos

Neste tópico, são listados os principais problemas identificados e as soluções existentes conforme o ponto de vista dos envolvidos. A tabela 3.5, para cada problema identificado, objetiva explicar as causas, como ele está sendo resolvido atualmente e que soluções estão sendo propostas para o novo sistema

Tabela 3.5 – Principais necessidades dos usuários ou dos envolvidos.

Necessidade	Prioridade	Preocupações	Solução Atual	Soluções Propostas
Garantir o sigilo das informações do paciente.	Alta	Ter certeza de que as informações e imagens do paciente (problemas médicos) não serão visualizadas por pessoal não autorizado. Evitar que o vazamento de informações do paciente possa prejudicar a imagem do hospital junto à sociedade.	Uso de switch.	Cifrar todo o conteúdo transmitido utilizando TLS e certificados digitais.
Evitar que qualquer simulador possa enviar imagens, se passando pelo equipamento médico.	Alta	Um médico poderia realizar um diagnóstico baseado em imagens falsas.	Utilização do campo *AETitle* para autenticação do equipamento.	Realizar a autenticação do equipamento médico, utilizando certificado digital.

3.1.2 Especificação de Requisitos do Sistema

3.1.2.1 Requisitos do Software

Neste tópico, são definidos os principais requisitos do software (tabela 3.6). O grande objetivo é estabelecer um acordo entre envolvidos no projeto sobre o que o sistema deve realizar e o por quê. Como benefícios indiretos, esta fase permite, de forma inicial: visualizar as bases que deverão ser utilizadas para o design do sistema delimitar as funções a serem realizadas, permitir o planejamento técnico das iterações, permitir estimativas sobre custo e prazo e, também, permitir a definição da interface do sistema com seus usuários.

Tabela 3.6 – Requisitos do Software.

Requisito	Requisito / Funcionalidade	Problema Relacionado
Uso de certificado digital.	Autenticar o equipamento médico e o servido.	Qualquer simulador DICOM poderia transmitir imagens para o servidor se passando por um equipamento médico válido.
Cifrar os dados trocados entre equipamento médico e servidor.	Garantir o sigilo da imagem e dos campos AETitle.	Um atacante poderia remontar a imagem a partir dos pacotes capturados na rede.
Manter a mesma estrutura de banco de dados utilizado pelo miniWebPACS.	Garantir o uso do novo servidor DICOM sem causar impacto no ambiente atual.	Uma estrutura de dados diferente irá provocar impactos em todos os projetos DICOM em desenvolvimento no Hospital Universitário de Brasília.

3.1.2.2 Requisitos Suplementares

Para garantir a segurança dos dados, será programada uma cópia de todos os dados do dia a ser efetuado no fim do expediente.

3.1.2.3 Restrições Técnicas

Para garantir a compatibilidade com os sistemas em desenvolvimento no Hospital Universitário de Brasília, não foi implementado nenhum algoritmo que permita o armazenamento cifrado das imagens.

Todas as configurações de segurança são garantidas através do uso do certificado digital. Um atacante não poderá ter acesso físico ao equipamento médico afinal, caso tenha acesso, nada imperá que ele use o certificado e envie imagens.

3.1.3 Modelos de Casos de Uso

O Diagrama de Caso de Uso é um documento criado a partir da identificação de requisitos e que permite ao documentador descrever as funcionalidades de um sistema, bem como informar os atores e a interação entre usuário (humano ou máquina) e o sistema.

Na tabela 3.7 podem ser observados os casos de uso identificados durante a fase de levantamento de requisitos.

Tabela 3.7 – Descrição dos casos de uso.

Caso de Uso	Requisito(s)
Autentica Equipamento.	Não aplicável.
Recebe Imagem (com TLS).	Equipamento médico autenticado e suporte a sintaxe de transferência.
Inicializa Servidor.	Arquivo de parâmetros definidos e banco de dados inicializado.

3.1.3.1 Descrição dos Atores

Na tabela 3.8, podem ser observados os principais atores identificados durante a fase de levantamento de requisitos.

Tabela 3.8 – Descrição dos atores.

Ator	**Descrição**
Equipamento médico (com TLS).	Representa qualquer equipamento médico que implementa o protocolo DICOM/TLS.
Administrador.	Responsável pela inicialização do servidor e por definir os parâmetros para a transmissão de imagens.

3.1.3.2 Diagrama de Casos de Uso

No hub2pacs, foram identificados três casos de uso principais que podem ser observados na figura 3.1:

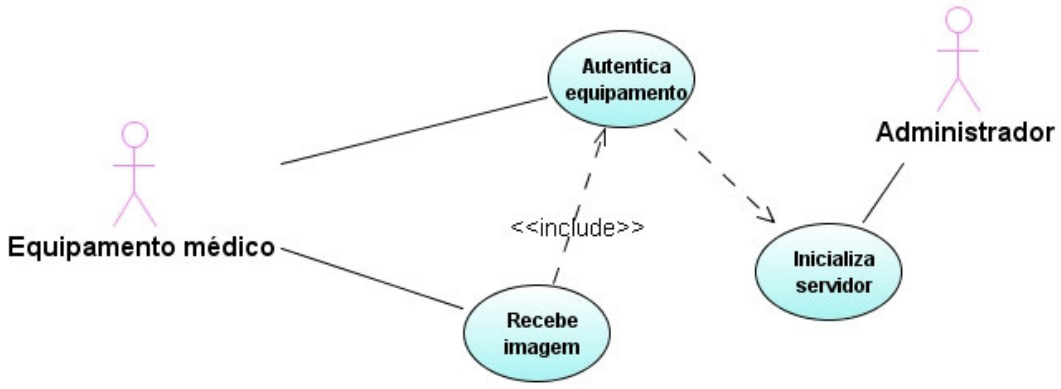

Figura 3.1 – Diagrama de caso de uso.

3.1.3.3 Caso de uso – Autentica Equipamento

Objetivo do caso de uso: Este caso de uso fornece a autenticação e a identificação do equipamento para o Sistema. Permite, também, a definição dos algoritmos criptográficos e das chaves que serão utilizadas durante a comunicação.

Atores: Equipamento Médico com TLS.

Pré-condição: Não se aplica.

Fluxo Básico (FB)

O caso de uso se inicia quando o Ator precisa criar uma conexão segura com o Sistema.

1 – O Sistema recebe do Ator o certificado digital assinado pela Autoridade Certificadora que o Sistema confia e os campos AETitle.

2 – O sistema valida o certificado digital [EXC1].

3- O Sistema valida os campos AETitle [EXC2].

4- Fim do caso de uso.

Fluxos Alternativos (AL)

Não se aplica.

Fluxos de Exceções (EX)

EXC1 – Certificado digital inválido.

Este fluxo de exceção é realizado quando o certificado digital apresentado pelo equipamento é inválido, ou seja, não é considerado confiável.

1 – O Sistema interrompe a execução e emite uma notificação.

2 – Fim do caso de uso.

EXC2 – Campo AETitle inválido.

Este fluxo de exceção é realizado quando o Sistema verifica que os campos AETitle informados pelo Ator são inválidos, ou seja, não correspondem aos campos cadastrados no servidor.

1 – O Sistema interrompe a execução e emite uma notificação.

2 – Fim do caso de uso.

3.1.3.4 Caso de uso – Recebe Imagem

Objetivo do caso de uso: Este caso de uso permite ao equipamento enviar para o Sistema o objeto DICOM que será armazenado.

Atores: Equipamento Médico com TLS.

Pré-condição: Ter realizado com sucesso o caso de uso Autentica Equipamento.

Fluxo Básico (FB)

O caso de uso se inicia quando o Ator envia a imagem para armazenamento no servidor.

1 – O Sistema lê o arquivo de configuração para determinar as regras de armazenamento que serão utilizadas [EXC1].

2- O Sistema identifica a associação de segurança utilizada e valida os campos AETitle.

3 – O Ator envia o objeto DICOM.

4- O Sistema recebe o objeto DICOM e muda seu nome, de acordo com os parâmetros de nomenclatura informados no arquivo de configuração.

5- O Sistema armazena o objeto no disco.

6- O Sistema lê as informações de identificação da imagem (cabeçalho do DICOM) e as insere no servidor de Banco de Dados [EXC2].

7 – Fim do caso de uso.

Fluxos Alternativos (AL)

Não se aplica.

Fluxos de Exceções (EX)

EXC1 – Arquivo de configuração inválido ou inexistente.

Este fluxo de exceção é realizado quando o Sistema verifica que tem algum parâmetro inválido no arquivo de configuração ou este arquivo não se encontra no mesmo diretório da aplicação.

1 – O Sistema interrompe a execução e emite uma notificação.

2 – Fim do caso de uso.

EXC2 – Erro de acesso ao banco de dados.

Este fluxo de exceção é realizado quando o servidor de banco de dados encontra-se indisponível.

1 – O Sistema emite uma notificação.

2 – Fim do caso de uso.

3.1.3.5 Caso de uso – Inicializa Servidor

Objetivo do caso de uso: Este caso de uso inicializa o servidor DICOM.

Atores: Administrador.

Pré-condição: Não aplicável.

Fluxo Básico (FB)

O caso de uso se inicia quando o Ator executa o comando de inicialização do servidor.

1 – O Sistema lê o arquivo de configuração para determinar as configurações do protocolo TCP/IP, a sintaxe de transferência suportada, etc [EXC1].

2 – Fim do caso de uso.

Fluxos Alternativos (AL)

Não se aplica.

Fluxos de Exceções (EX)

EXC1 – Arquivo de configuração inválido ou inexistente.

Este fluxo de exceção é realizado quando o Sistema verifica que tem algum parâmetro inválido no arquivo de configuração ou quando este arquivo não se encontra no mesmo diretório da aplicação.

1 – O Sistema interrompe a execução e emite uma notificação.

3.1.4 Levantamento das Classes

Neste tópico é apresentado o diagrama de classe, a descrição dos atributos e métodos. Observe que a principal classe do projeto é a Servidor e que os principais métodos são o onCStoreRQ() e o main().

3.1.4.1 Descrição das Classes

A tabela 3.9 detalhada os atributos, métodos e as classes implementadas no software:

Tabela 3.9– Descrição das classes.

Pacote	Classe	Descrição
hub2pacs	Servidor	Filha da *StorageService* tem como objetivo fornecer o suporte para o recebimento e inicialização do servidor PACS. Sua principal atividade consiste na incialização de parâmetros e na criação da associação de segurança quando da recepção da imagem.
hub2pacs	Util	Classe desenvolvida com o objetivo de nomear e armazenar o arquivo de imagem recebido de acordo com os parâmetros informados no arquivo de configuração. Possui, também, alguns métodos úteis para a aplicação.
hub2pacs	DB	Classes desenvolvidas para facilitar a conexão com o servidor de banco de dados. Nessa classe, foi implementado o *Design Partners Singleton* que, basicamente, permite a instanciação de um único objeto para conexão.
hub2pacs	DBPostgre	Classe utilizada para configurar os atributos de acesso ao banco de dados postgre (usuário, senha, servidor, etc).

Pacote	Classe	Descrição
hub2pacs	DBWriter	Utilizada como interface, seu principal objetivo é permitir que a execução e a manipulação do banco de dados possam ser realizadas em paralelo, ou seja, fazendo uso de threads. Outro objetivo refere-se ao fato de obrigar o uso padronizado caso seja desenvolvido uma nova clasa para acesso a um SGBD diferente do postgres.
hub2pacs	DBWriterAccess	Classes desenvolvidas com o objetivo de permitir a inserção de informações sobre a imagem no servidor de Banco de Dados postgres. Nesta classe, estão contidas todas as consultas SQL que serão realizadas no banco de dados.
org.dcm4che2.net.service	StorageService	Interface desenvolvida pela equipe do dcm4che. Define, resumidamente, o funcionamento de um servidor DICOM baseado no dcm4che.
org.dcm4che2.net	Device	Descreve um sistema ou dispositivo DICOM ativo em termos de seus atributos físicos (número de série, fabricante etc.), seu contexto (identificadores do paciente utilizados pelo dispositivo etc.) e suas capacidades (TLS ativado, AETitle utilizado etc).
org.dcm4che2.net	DicomServiceException	Esta classe representa todas as exceções utilizadas pelo servidor DICOM. A partir dela, existem subclasses que representam problemas específicos.
org.dcm4che2.net	NetworkApplicationEntity	Descreve e fornece todos os serviços de rede utilizados pelo DICOM.
org.dcm4che2.net	Association	Representa a associação realizada entre o SCU e o SCP. Seus principais atributos referem-se aos campos AETitle.
org.dcm4che2.net	NetworkConnection	Encapsula todas as propriedades associadas com a conexão em redes TCP/IP. Um objeto *NetworkConnection* descreve uma porta TCP e um dispositivo de rede permitindo a criação de múltiplas associações.
org.dcm4che2.net	PDVInputStream	Baseada na classe java.io.*InputStream*, oferece a funcionalidade básica para a leitura de um byte ou de uma seqüência de bytes a partir de um arquivo DICOM.
org.dcm4che2.data	BasicDicomObject	Representa características mínimas de um objeto DICOM em memória. Seu objetivo é permitir acesso aos principais atributos de um objeto DICOM sem sobrecarregar a memória.
org.dcm4che2.data	DicomObject	Representa um objeto DICOM em memória.
org.apache.commons.cli	CommandLine	A biblioteca *Apache Commons* CLI fornece uma API para o processamento de parâmetros fornecidos em linha de comando. Nesse projeto é utilizada para manipular os parâmetros fornecidos pelo administrador em linha de comando quando executa o arquivo .jar do projeto.

Pacote	Classe	Descrição
java.io	BufferedOutputStream	Representa um fluxo de saída temporário que permite escrever caracteres para um fluxo sem causar uma gravação para cada nova informação. Os dados são escritos para o buffer e só são transferidos para o fluxo real quando o buffer estiver cheio.
java.io	FileOutputStream	Permite a escrita de um determinado fluxo de dados em um arquivo ou em um *FileDescriptor.*
java.io	PrintStream	Mostra um determinado fluxo de dados.
java.io	InputStream	Oferece a funcionalidade básica para a leitura de um *byte* ou de uma sequência de *bytes* a partir de alguma fonte.
java.security	GeneralSecurityException	Esta classe representa todas as exceções de segurança estabelecidas pelo Java. A partir dela, existem subclasses que representam problemas de segurança específicos.
java.security	KeyStore	Classe que representa, na memória, as chaves e os certificados configurados para uso na aplicação.
java.util.concurrent	Executor	Permite a execução de tarefas em paralelo (tratamento de *threads*).
java.util	LinkedList	Armazena valores ordenados pela ordem de inserção. Como todos os outros tipos de *collections*, para percorrer todos os dados da *LinkedList* é necessário um *Iterator*.
java.util	Random	Gerador de seqüência (números) aleatório.
javax.rmi.CORBA	Util	Fornece utilitários e métodos que podem ser utilizados pelo processo para executar operações comuns. A classe CORBA oferece um conjunto de API para acesso a leitura da Java *Remote Method Invocation* sobre Internet *Inter-Orb Protocol* (RMI-IIOP).

3.1.4.2 Diagrama de Classe

Na figura 3.2, pode ser observado o diagrama de classe do projeto, os atributos, métodos e os relacionamentos. O diagrama de classe permite visualizar de forma simples e clara o formato utilizado no projeto e sua estrutura de recuperação de informação. O anexo B contém a descrição de cada um dos métodos e atributos das classes deste projeto.

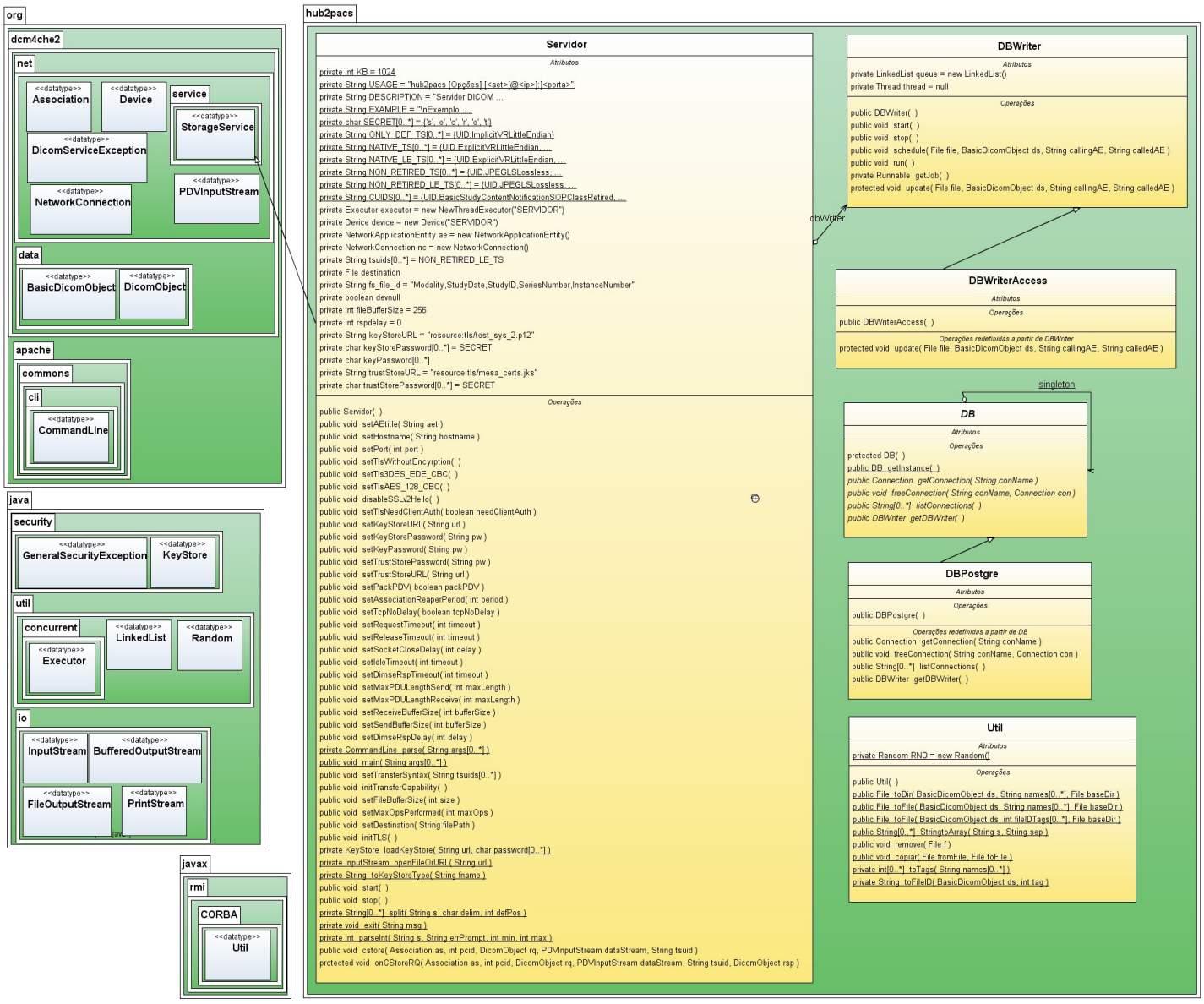

Figura 3.2 – Diagrama de Classe hub2pacs.

3.1.5 Modelos de Análise

3.1.5.1 Diagrama de sequência: método main()

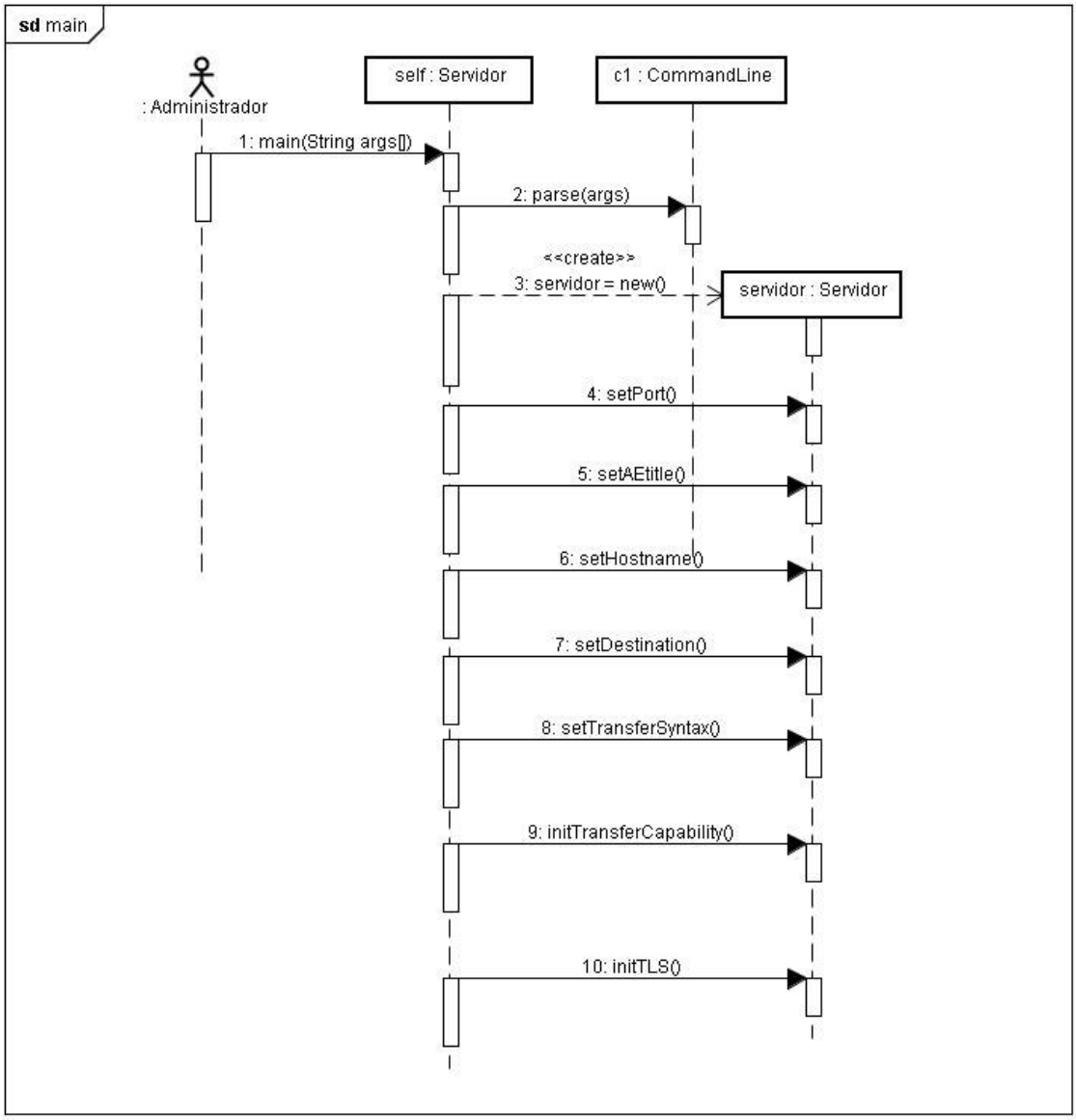

Figura 3.3 – Diagrama de Sequência método main().

A inicialização do servidor (figura 3.3) é acompanhada das seguintes atividades:

- Leitura de parâmetros e configuração da variável c1 (*ComandLine*) fornecidos pelo administrador;

- Inicialização dos parâmetros do Servidor (*new Servidor*);

- Configuração de todos os parâmetros do servidor, os principais são: *aetitle, port, hostname* e *destination;*

- Inicializa a sintaxe e a capacidade de transferência (*initTransferSintax* e *initTransferCapability*); e

- Inicialização dos algoritmos de criptografia e autenticação do processo utilizando certificados digitais (*initTLS*).

Observe que as principais atividades realizadas na inicialização do servidor hub2pacs que interessam para esse projeto são a *initTransferSintax* e a *initTLS*.

3.1.5.2 Diagrama de Sequência: método initTLS ()

Como já descrito, o método *initTLS* da classe Servidor tem como objetivo realizar a leitura de parâmetros do objeto e filiar seus valores aos requisitos mínimos para a inicialização do protocolo TLS.

Observe-se que as propriedades já foram configuradas pelo método main e pelo método construtor da classe. Esse método resume a carregar os certificados e as chaves em variáveis KeyStore e repassar o controle de fluxo de dados para a classe *StorageService* do dcm4che2.

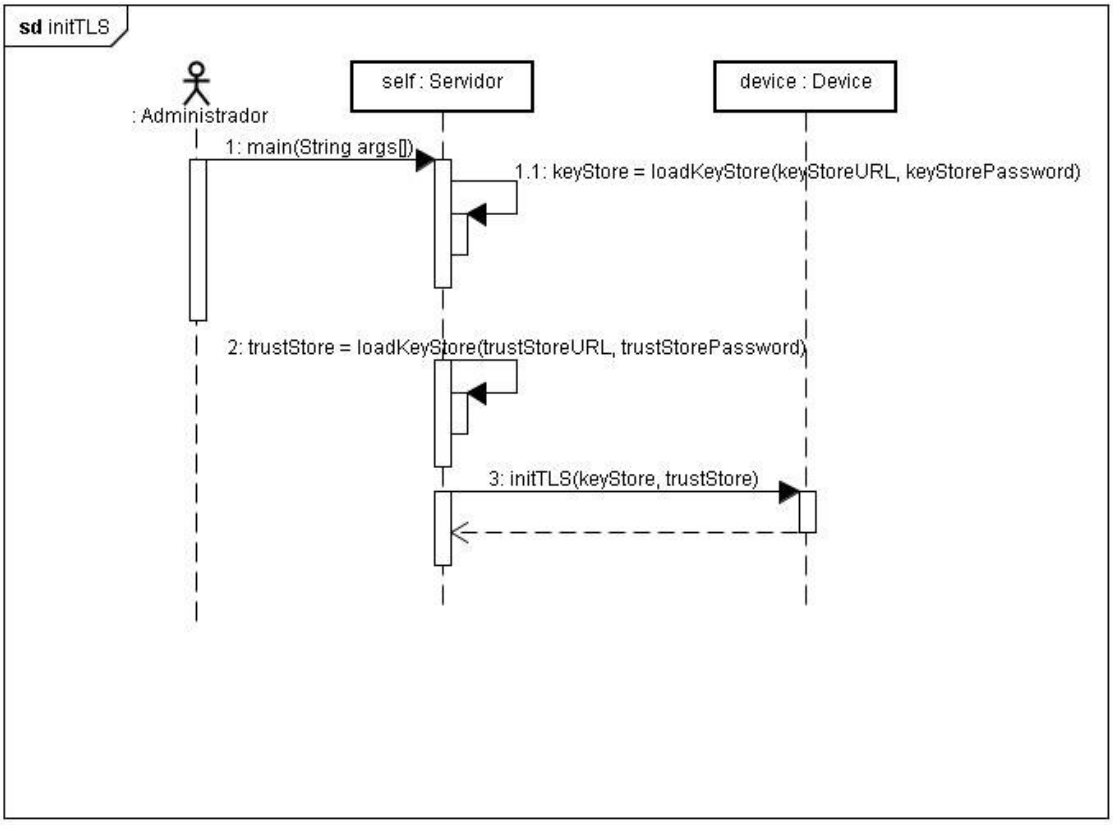

Figura 3.4 – Diagrama de Sequência método initTLS().

3.1.5.3 Diagrama de sequência: método onCStoreRQ()

O método onCStoreRQ() é um dos últimos métodos acionados sempre que um equipamento deseja transmitir uma imagem médica e todo o processo de validação já foi realizado, ou seja, a associação foi criada e os certificados validados.

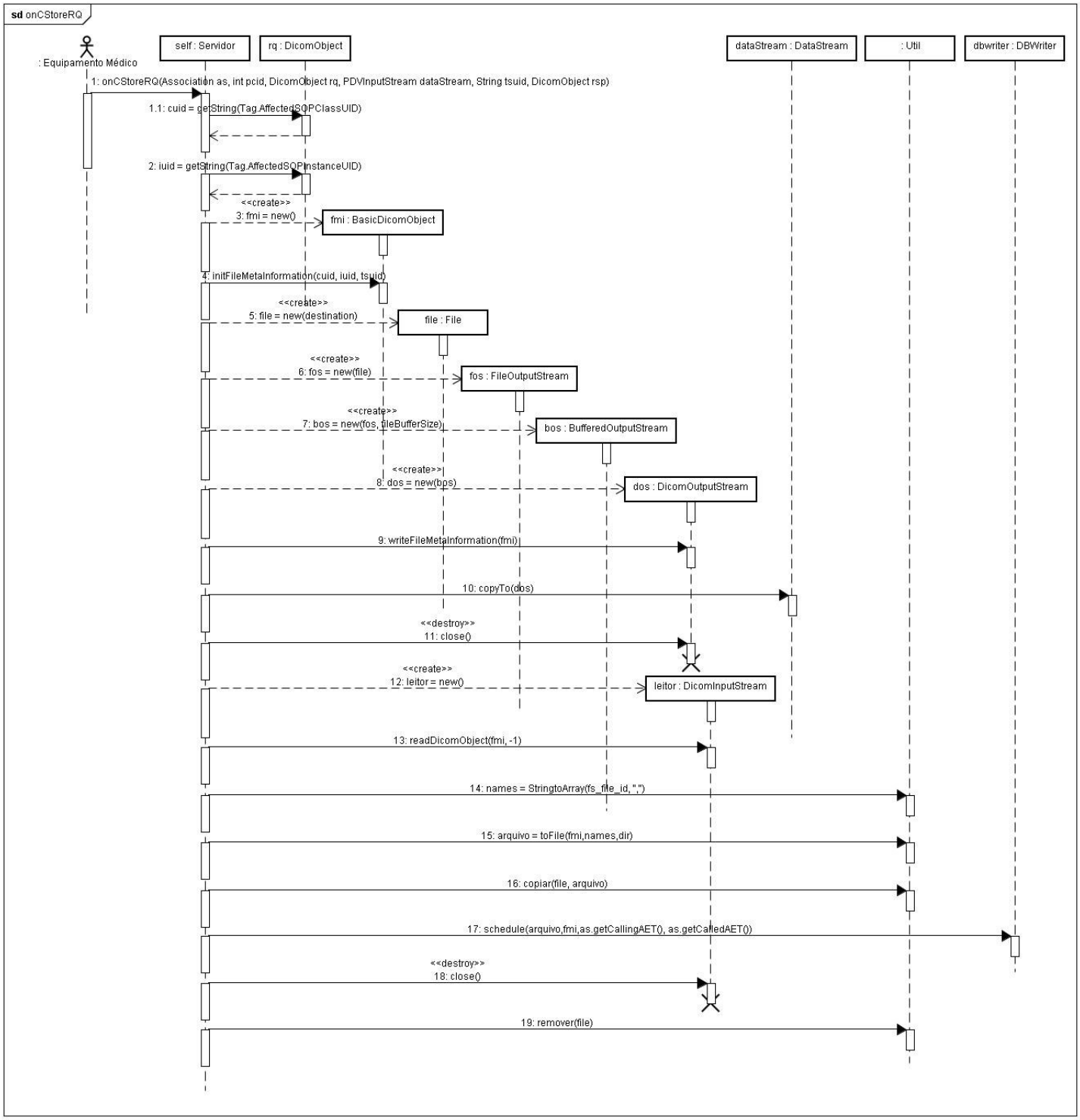

Figura 3.5 - Diagrama de Sequência método onCStoreRQ().

3.2 PROJETO – HUB2CONVERSOR

Embora o projeto hub2pacs atenda aos requisitos de compatibilidade com qualquer máquina DICOM que ofereça suporte a TLS, existe um problema quando se tratam de equipamentos médicos antigos que não oferecem este tipo de mecanismo de proteção.

O projeto hub2conversor foi desenvolvido nesse trabalho com o objetivo de criar um software que permita a um equipamento não TLS transmitir objetos de forma cifrada e protegida. Para esse fim, foi desenvolvido um conversor chamado de hub2conversor. Seu funcionamento é similar ao do miniWebPACS, ou seja, todo o segredo da transmissão está na garantia do sigilo das informações AETitle.

Por ser desenvolvido em Java, este projeto precisa de alguns requisitos de hardware. Deve estar instalado ao lado do equipamento médico antigo em um computador com duas placas de rede. Uma placa ligada diretamente ao equipamento médico (cabo *crossover*) e outra à rede de dados. Desta forma, o equipamento médico deverá estar configurado para transmitir para o hub2conversor e este, assim que receber o objeto DICOM, irá retransmitir para o servidor oficial utilizando TLS.

Com o objetivo de simplificar a documentação do projeto, logo abaixo são detalhadas as classes Configuração e Transmissor que foram inseridas no projeto original, ou seja, do hub2pacs. As demais classes já foram explicadas na seção que descreve o hub2pacs.

3.2.1 Documento de Visão

Com o objetivo de diminuir o tamanho final da documentação e evitar redundâncias de conteúdo, foi elaborado um documento de visão resumido, contendo, apenas: a descrição do problema, a sentença de posição do produto e a descrição do ambiente contemplando o uso do hub2conversor.

3.2.1.1 Descrição do Problema

Neste tópico, é fornecida uma descrição resumida do problema que está sendo resolvido pelo projeto. A tabela 3.10 detalha o problema que motivou a criação do projeto, quem é afetado, o impacto e a solução proposta para a correção.

Tabela 3.10 – Descrição do Problema hub2conversor.

O problema de	Equipamento médico antigo que não transmite imagens utilizando TLS e certificados digitais.
Afeta	Paciente, Hospital, Equipamento médico.
Cujo impacto é	Perda da credibilidade e perda da imagem da organização devido a processos judiciais por vazamento de informações do paciente.
Solução proposta	Desenvolver um conversor que permita a um equipamento médico antigo transmitir imagens utilizando o protocolo DICOM/TLS.

3.2.1.2 Sentença de Posição do Produto

A sentença de posição do produto apresenta os principais concorrentes do software proposto a ser desenvolvido e quais o fatores o diferenciam de seus concorrentes de mercado. Na tabela 3.11 são apresentadas estas diferenças.

Tabela 3.11 – Sentença de Posição do Produto hub2conversor.

Para	Hospital Universitário de Brasília (HUB).
Que	Necessita de conversor que permita a um equipamento médico antigo transmitir imagens médicas de forma cifrada e protegida em uma rede insegura.
O sistema	Hub2conversos.
Que	Permite a transmissão de imagens médicas cifrando o conteúdo e autenticando o equipamento médico, utilizando certificado digital.
Diferente de	Não aplicável.
Nosso sistema	Permite que equipamentos médicos antigos possam enviar imagens médicas de forma cifrada.

3.2.1.3 Ambiente do Usuário

O ambiente para uso do servidor DICOM no Hospital Universitário de Brasília (HUB) é composto, basicamente, dos seguintes itens:

- Rede de comunicação ethernet utilizada para permitir a troca de informações entre o equipamento médico, o servidor PACS e o servidor de visualização de imagens;

- Servidor PACS (hub2pacs) que permite o recebimento e armazenamento de imagens; e

- Equipamento médico antigo utilizado para realização de exames.

Com a implantação do hub2pacs, os equipamentos médicos antigos em uso no Hospital Universitário de Brasília passariam a não funcionar corretamente. Embora seus recursos sejam limitados e a necessidade de evolução tecnológica seja uma constante nas organizações, a aquisição de equipamentos médicos demanda grandes recursos financeiros, nem sempre disponíveis. O hub2conversor aparece como solução, evitando que os equipamentos em uso parem de funcionar e permitindo que toda a transmissão de imagens seja realizada de forma cifrada e autenticada.

3.2.2 Especificação de Requisitos do Sistema

3.2.2.1 Requisitos do Software

Neste tópico, são definidos os principais requisitos do software (tabela 3.12). O grande objetivo é estabelecer um acordo entre envolvidos no projeto sobre o que o sistema deve realizar e o porquê. Como benefícios indiretos, esta fase permite, de forma inicial: visualizar as bases que deverão ser utilizadas para o design do sistema, delimitar as funções a ser realizadas, permitir o planejamento técnico das iterações, permitir estimativas sobre custo e prazo e permitir a definição da interface do sistema com seus usuários.

Tabela 3.12 – Requisitos do Software hub2conversor.

Requisito	Requisito / Funcionalidade	Problema Relacionado
Uso de certificado digital.	Autenticar o hub2conversor e o hub2pacs.	Qualquer simulador DICOM poderia transmitir imagens para o servidor, se passando por um equipamento médico válido.

Requisito	Requisito / Funcionalidade	Problema Relacionado
Cifrar os dados trocados entre equipamento médico e servidor.	Garantir o sigilo da imagem e dos campos AETitle.	Um atacante poderia remontar a imagem a partir dos pacotes capturados na rede.
Receber a imagem médica não cifrada e retransmití-la de forma cifrada para o servidor hub2pacs.	Garantir o sigilo das informações do paciente.	Evitar que as informações do paciente sejam capturadas durante a transmissão da imagem.
Permitir que equipamentos médicos antigos possa utilizar DICOM/TLS.	Autenticar o equipamento e garantir o sigilo dos dados do paciente.	Evitar que um equipamento médico antigo deixe de transmitir imagens de forma segura.

3.2.2.2 Requisitos Suplementares

Para garantir a segurança dos dados, será programada uma cópia de todos os dados do dia a ser efetuado no fim do expediente.

É de extrema importância que seja garantido a segurança física (acesso físico) do servidor. Essa medida tem com objetivo evitar que o certificado digital utilizado pelo sistema seja copiado por um invasor.

O servidor onde o hub2conversor for instalado deve possuir duas placas de rede onde, uma deve estar conectada diretamente a interface de rede do equipamento médico e a outra a rede "insegura".

Deve ser configurado um filtro de pacotes (firewall) no servidor onde está instalado o hub2conversor. Este mecanismo irá permitir que somente o equipamento médico antigo possa enviar imagens médicas utilizando o conversor.

3.2.3 Modelos de Casos de Uso

O Diagrama de Caso de Uso é um documento criado a partir da identificação de requisitos e que permite ao documentador descrever as funcionalidades de um sistema, bem como informar os atores e a interação entre usuário (humano ou máquina) e o sistema.

Na tabela 3.13 podem ser observados os casos de uso identificados durante a fase de levantamento de requisitos.

Tabela 3.13– Descrição dos casos de uso hub2conversor.

Caso de Uso	Requisito(s)
Autentica Equipamento	Não aplicável.
Recebe imagem (sem TLS)	Equipamento médico autenticado e suporte a sintaxe de transferência.
Inicializa Servidor	Arquivos de parâmetros validos e definidos.
Retransmite imagem (com TLS)	Certificado digital valido e sistema hub2pacs ativo.

3.2.3.1 Descrição dos Atores

Na tabela 3.14 podem ser observados os principais atores identificados durante a fase de levantamento de requisitos.

Tabela 3.14 – Descrição dos atores.

Ator	Descrição
Equipamento médico sem TLS	Representa qualquer equipamento médico que não implementa o protocolo DICOM/TLS.
Administrador	Responsável pela inicialização do servidor e por definir os parâmetros para a transmissão de imagens.
Hub2conversor	Sistema PACS que aceita, apenas, transmissão de imagens médicas utilizando certificado digital válido.

3.2.3.2 Diagrama de Casos de Uso

No hub2conversor, foram identificados quatro casos de uso principais que podem ser observados na figura 3.6:

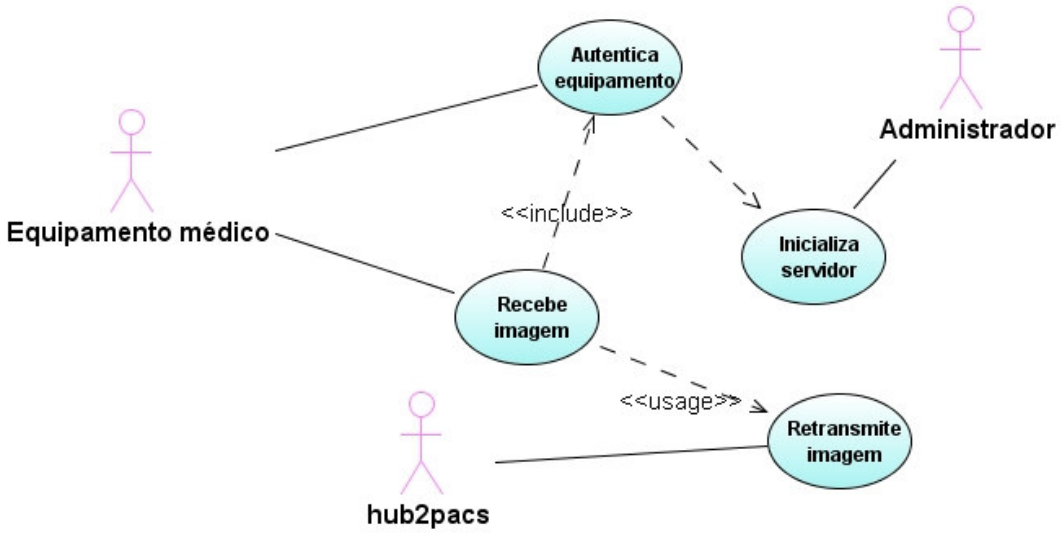

Figura 3.6 – Diagrama de caso de uso.

3.2.3.3 Descrição dos Casos de Uso

A seguir são detalhados os casos de uso Autentica equipamento e Retransmite imagem. Os demais casos de uso já foram explicados na seção do hub2pacs.

3.2.3.4 Caso de uso – Autentica Equipamento

Objetivo do caso de uso: Este caso de uso fornece a autenticação e a identificação do equipamento para o Sistema. Toda a autenticação é baseada nos campos AETitle.

Atores: Equipamento Médico sem TLS.

Pré-condição: Não se aplica.

Fluxo Básico (FB)

O caso de uso se inicia quando o Ator precisa criar uma conexão com o Sistema.

1 – O Sistema recebe do Ator os campos AETitle.

2- O Sistema valida os campos AETitle [EXC1].

3- Fim do caso de uso.

Fluxos Alternativos (AL)

Não se aplica.

Fluxos de Exceções (EX)

EXC1 – Campo AETitle inválido.

Este fluxo de exceção é realizado quando o Sistema verifica que os campos AETitle informados pelo Ator são inválidos, ou seja, não correspondem aos campos cadastrados no servidor.

1 – O Sistema interrompe a execução e emite uma notificação.

2 – Fim do caso de uso

3.2.3.4.1 Caso de uso – Retransmite imagem.

Objetivo do caso de uso: Este caso de uso permite ao equipamento médico antigo possa enviar o objeto DICOM utilizando certificado digital e TLS para o servidor hub2pacs.

Atores: Equipamento Médico sem TLS.

Pré-condição: Ter realizado com sucesso o caso de uso Autentica Equipamento e Recebe Imagem.

Fluxo Básico (FB)

O caso de uso se inicia quando o sistema confirma o armazenamento de um novo objeto recebido pelo servidor.

1 – O Sistema lê o arquivo de configuração para configurar o tranmissor com os dados referentes ao certificado digital em uso [EXC1].

2 – O Sistema envia o objeto DICOM utilizando TLS [EXC2].

3 – Fim do caso de uso.

Fluxos Alternativos (AL)

Não se aplica.

Fluxos de Exceções (EX)

EXC1 – Arquivo de configuração inválido ou inexistente.

Este fluxo de exceção é realizado quando o Sistema verifica que tem algum parâmetro inválido no arquivo de configuração ou este arquivo não se encontra no mesmo diretório da aplicação.

1 – O Sistema interrompe a execução e emite uma notificação.

2 – Fim do caso de uso.

EXC2 – Certificado(s) invalido.

Este fluxo de exceção é realizado quando o cliente SCU informa um certificado invalido ao servidor SCP.

1 – O Sistema emite uma notificação e aborta o envio do objeto.

2- Fim do caso de uso.

3.2.4 Levantamento das Classes

3.2.4.1 Descrição das Classes

Abaixo (tabela 3.15) são detalhados os atributos, métodos e as classes implementadas no *software* conversor.

Tabela 3.15– Descrição das classes.

Pacote	Classe	Descrição
hub2pacs	Configuracao	Responsável por ler o arquivo configuracao.properties e buscar informações como os certificados e as senhas utilizadas para a transmissão TLS.
Hub2pacs	Transmissor	Classe criada tendo como objetivo realizar a retransmissão do objeto DICOM recebido pelo conversor para o servidor oficial utilizando TLS.

A figura 3.6 mostra o diagrama de classe resumido com destaque para as classes: Transmissor e Configuracao. O anexo C apresenta o detalhamento sobre os métodos e atributos desse projeto.

Figura 3.7 – Diagrama de classe projeto hub2conversor.

3.2.5 Modelos de Análise

3.2.5.1 Diagrama de Sequência método onCStoreRQ()

O método onCStoreRQ() é um dos últimos métodos acionados sempre que um equipamento deseja transmitir uma imagem médica e todo o processo de validação já foi realizado, ou seja, a associação foi criada e os certificados validados.

A principal diferença entre o método onCStoreRQ do projeto hub2pacs e do hub2conversor é o fato de que aqui não existe a necessidade de armazenar as informações da imagem em um banco de dados. Neste caso, é criado um objeto do tipo Transmissor e este é utilizado apenas para retransmitir a imagem.

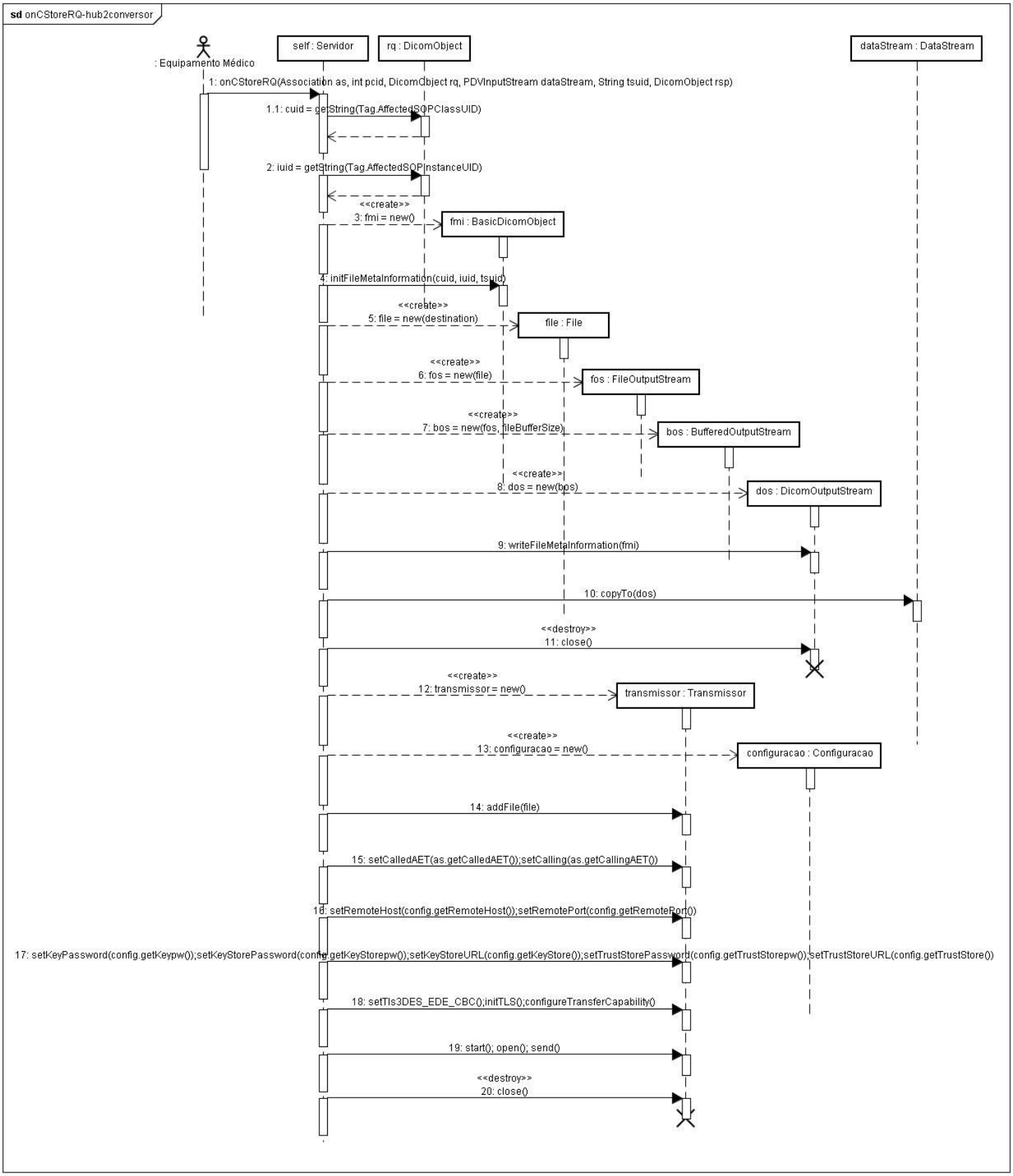

Figura 3.8 – Diagrama de sequência método onCStoreRQ do projeto hub2conversor.

3.3 DIAGRAMA DE IMPLANTAÇÃO

O diagrama de implantação descreve os elementos de configuração de processamento *run-time* e a arquitetura de um sistema onde estão interligados seus componentes. Como exemplo é a arquitetura física de hardware, processadores.

Abaixo, segue o diagrama de implantação (figura 3.9) do servidor hub2pacs/hub2conversor. Observe-se que são utilizados três equipamentos de hardware sendo: o equipamento médico/hospitalar, o servidor hub2conversor e o servidor hub2pacs.

O servidor de conversão (hub2conversor) tornou-se necessário já que o atual equipamento utilizado pelo HUB não fornece suporte à instalação e uso de certificados digitais. Esta camada de software torna-se necessária sempre que o equipamento médico antigo não consegue operar utilizando certificados digitais.

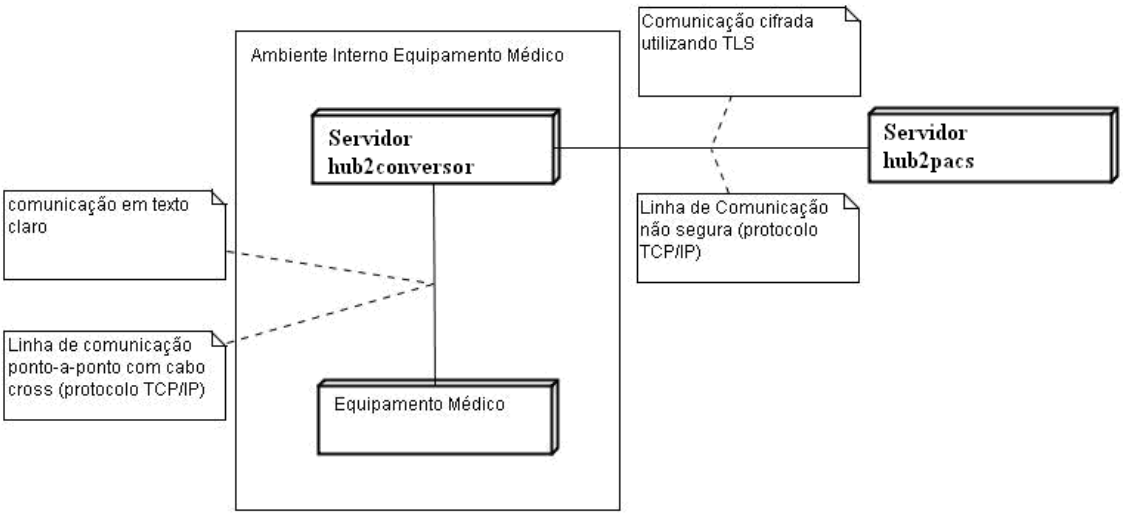

Figura 3.9 - Modelo de implementação no HUB.

4 DEMONSTRAÇÃO DO PROTÓTIPO

O capítulo de demonstração do protótipo busca verificar se os projetos hub2pacs e hub2conversor atendem aos requisitos destacados no documento de visão e verificar se todas as falhas de segurança encontradas no miniWebPACS foram corrigidas.

4.1 AMBIENTE DE TESTE UTILIZADO

O projeto hub2pacs foi testado em dois ambientes. O primeiro teste foi realizado em março de 2007 e tinha como objetivo verificar o grau de compatibilidade da nova versão com o equipamento médico utilizado no HUB (Philips, modelo Ultrasound - HDI 3500 System) e o sistema Web de visualização e manipulação de laudos médicos (GSWeb).

Nesse momento, o servidor mini foi desativado WebPACS e foi configurado o hub2pacs com todos os parâmetros utilizados pelo sistema antigo (diretório e formato para armazenamento de imagens DICOM, parâmetros de comunicação com o servidor de banco de dados postgres e os parâmetros de autenticação AETitle utilizado pelo equipamento médico).

Após a inicialização do servidor hub2pacs, foi realizado um teste de transmissão do equipamento médico para o novo servidor implementado. A transmissão ocorreu com sucesso, porém a imagem médica recebida de teste foi armazenada pelo servidor de forma errada, deslocando em, aproximadamente, 10º graus o seu conteúdo.

Tal problema foi ocasionado por um erro de programação, pois, ao abrir a conexão com o servidor postgres, o tempo levado na abertura da conexão atrasava a captura da imagem médica enviada. A solução adotada foi abrir a conexão com o banco de dados, utilizando um thread e modificando sua ordem de execução para acontecer, somente, quando a imagem tiver sido recebida completamente e com sucesso.

Como o equipamento médico utilizado pelo HUB não implementa o protocolo DICOM utilizando o TLS, a segunda bateria de testes foi realizada em um laboratório com 4 computadores conectados através de uma rede padrão ethernet (figura 4.1). Observe que o

hub2conversor foi testado apenas nesta fase, pois dependia do funcionamento do projeto hub2pacs. Em seguida são detalhados os sistemas operacionais envolvidos, os comandos utilizados para envio e os parâmetros utilizados para inicialização do servidor.

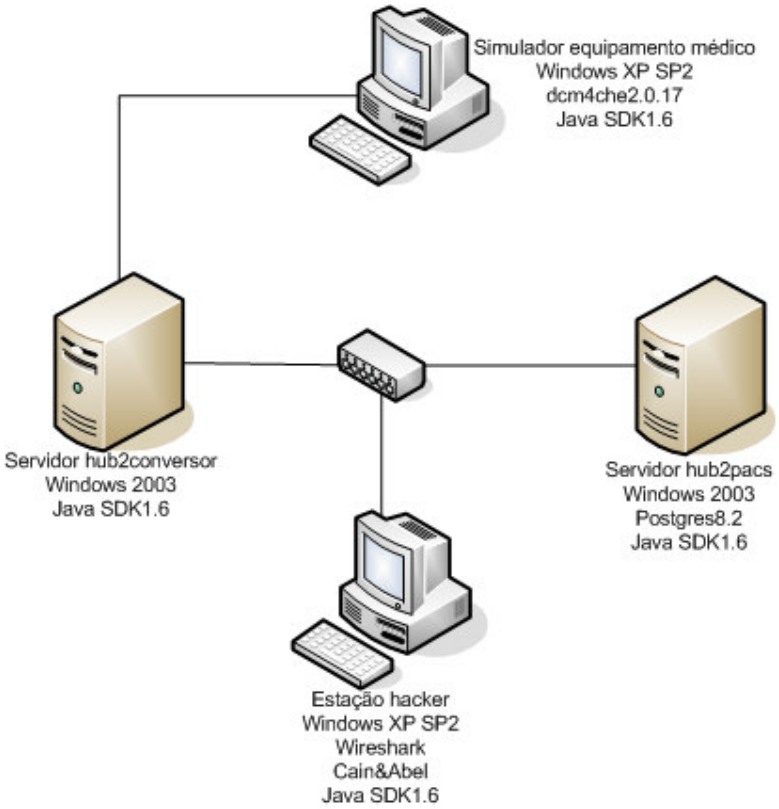

Figura 4.1 – Diagrama de rede utilizado para realização dos testes DICOM/TLS.

- Servidor hub2pacs: Sistema Operacional Windows 2003 instalação padrão e com todas as atualizações até o mês de março de 2008, Java SDK 1.6 e Servidor SGBD postgres versão 8.2. O projeto hub2pacs utiliza um arquivo texto para obter as configurações do servidor de banco de dados (postgres.properties). Foram utilizados os seguintes parâmetros para a inicialização do servidor hub2pacs:

```
C:\java -jar hub2pacs.jar SERVIDOR:3301 -dest /root/imagens1 -
tls 3DES

Onde:
     -dest = diretório de destino de todos os objetos Dicom
          recebidos;
     SERVIDOR = campo aeTitle;
     3301 = número da porta;
     -tls = ativa o suporte a TLS utilizando o certificado
          padrão;
```

```
3DES = ativa o algoritmo 3DES para cifrar o conteúdo do
     canal;
```

- Simulador Equipamento Médico: Sistema Operacional Windows XP Professional com Service Pack 2, instalação padrão e com todas as atualizações até o mês de março de 2008, Java SDK 1.6 e biblioteca dcm4che2 versão 2.0.17. Foi executada a seguinte linha de comando para transferir uma imagem:

```
C:\ java -jar dcmsnd.jar SERVIDOR@10.0.0.1:3301 c:\tmp\FR -tls
3DES

Onde:
     SERVIDOR = campo aeTitle;
     10.0.0.1 = endereço ip do servidor DICOM;
     3301 = número da porta;
     c:\tmp\FR = local onde está armazenada a imagem que será
          transmitida;
     -tls = ativa o suporte a TLS utilizando o certificado
          padrão;
     3DES = ativa o algoritmo 3DES para cifrar o conteúdo do
          canal;
```

- Servidor hub2conversor: Sistema Operacional Windows 2003 instalação padrão e com todas as atualizações até o mês de março de 2008, Java SDK 1.6. O projeto hub2conversor utiliza um arquivo texto para obter as configurações TLS que serão utilizadas durante a tranmissão (configuracao.properties). Foram utilizados os seguintes parâmetros para a inicialização do servidor hub2conversor:

```
C:\java -jar hub2conversor.jar SERVIDOR:3302 -dest
/root/imagens2

Onde:
     -dest = diretório de destino de todos os objetos Dicom
          recebidos;
     SERVIDOR = campo aeTitle;
     3302 = número da porta;
```

- Estação de Monitoração: Sistema Operacional Windows XP Professional com Service Pack 2, instalação padrão e com todas as atualizações até o mês de março de 2008, Java SDK 1.6, Winpcap Versão 4 e Wireshark versão 1.0.

Para utilizar o protocolo DICOM/TLS, foi necessário gerar uma Autoridade Certificadora (AC) para criação dos certificados digitais que seriam utilizados. Para manipulação da AC foi

utilizando o aplicativo OpenSSL e, no Anexo A, podem ser observados os passos utilizados para gerar e assinar todos os certificado envolvidos.

Conforme pode ser observada na figura 4.1, a Estação *Hacker* tinha como objetivo capturar todo o tráfego gerado entre o servidor hub2pacs e o hub2conversor/equipamento médico. Foram realizados testes de captura de tráfego de rede local que tinham como objetivo: verificar se atacante conseguia descobrir os campos AETitle; se conseguiria enviar uma imagem para o servidor se passando pelo equipamento médico ou se as informações do paciente e a imagem poderiam ser visualizadas a partir dos pacotes capturados. Para o hub2pacs o hub2conversor é um equipamento médico, desta forma, os testes realizados podem ser feitos utilizando ou não o conversor.

4.2 METODOLOGIA PARA REALIZAÇÃO DOS TESTES

No primeiro capítulo deste trabalho foi definida a metodologia utilizada para atacar a aplicação miniWebPACS. Os testes realizados envolviam a captura de tráfego de rede e a tentativa de envio de imagens se passando pelo equipamento médico. Os testes no hub2pacs e no hub2conversor foram realizados da mesma forma, afinal o objetivo principal era verificar se as falhas de segurança detectadas no projeto miniWebPACS foram realmente corrigidas.

Abaixo são demonstrados alguns testes realizados com o objetivo de verificar a eficiência dos projetos desenvolvidos neste trabalho acadêmico.

4.2.1 Tentando enviar uma imagem sem o certificado digital

Neste teste, o servidor foi inicializado com a opção de receber uma imagem somente se o equipamento médico apresentar um certificado digital válido. Os campos AETitle foram ignorados o que, sem o uso do certificado, permitiria que qualquer simulador/equipamento enviasse imagens médicas. A figura 4.2 mostra a tentativa de envio frustrada realizada pela estação hacker e o código de erro gerado pelo servidor não permitindo o recebimento.

Figura 4.2 - Tentativa de envio da estação hacker sem utilizar o certificado digital.

Como pode ser observado, o Servidor, na penúltima linha da figura 4.2, retornou um código demonstrando que o cliente não enviou o certificado (SSL_NULL_WITH_NULL_NULL). A implementação de referência de JSSE em Java 6, suporta entre outras as seguintes suítes para criptografia (tabela 4.1):

Tabela 4.1 – Mensagens de erro TLS [44].

Nome	Propriedades
SSL_NULL_WITH_NULL_NULL	Ativado o SSL, porém não é garantida a autenticidade, integridade e confidencialidade
SSL_RSA_WITH_3DES_SHA	Autenticação com RSA, confidencialidade usando 3DES e integridade via SHA-1
SSL_RSA_WITH_NULL_SHA	Difere do anterior por não oferecer confidencialidade
SSL_RSA_WITH_NULL_MD5	Difere da anterior por não oferecer confidencialidade e garantir a integridade, utilizando o algoritmo MD5
SSL_RSA_WITH_RC4_128_MD5	Autenticação com RSA, confidencialidade, usando RC4 com chave de 128 bits e integridade via MD5
SSL_RSA_WITH_RC4_128_SHA	Difere da anterior pelo uso de SHA-1 em vez de MD5
TLS_RSA_WITH_AES_128_CBC_SHA	Confidencialidade, usando AES com chave de 128 bits em CBC mode
TLS_DHE_RSA_WITH_AES_128_CBC_SHA	Acordo de chaves, usando Diffie-Hellman efêmero
TLS_DHE_DSS_WITH_AES_128_CBC_SHA	Autenticação, usando DSS
TLS_DH_anon_WITH_AES_128_CBC_SHA	Sem autenticação

4.2.2 Simulando equipamento médico enviando uma imagem para o servidor utilizando o certificado digital válido

De forma semelhante ao primeiro teste realizado, neste teste o servidor foi configurado para aceitar o recebimento de imagens apenas se o cliente enviá-la utilizando o certificado digital válido. Como pode ser observado na figura 4.3, o envio foi realizado com sucesso e o código de transmissão foi SSL_RSA_WITH_3DES_SHA, demonstrando que a transmissão foi autenticada utilizando certificados do tipo RSA, o canal de comunicação foi cifrado utilizando o algoritmo 3DES e a integridade foi garantida com o algoritmo de *hash* SHA.

Figura 4.3 - Tentativa de envio do cliente utilizando o certificado digital valido.

Com o objetivo de demonstrar o uso prático deste novo modelo de transmissão de imagens, utilizando o protocolo TLS, foi ativada uma ferramenta de captura de pacotes na estação Hacker. Na figura 4.4, pode ser observado o pacote contendo o certificado digital enviado pelo cliente para o servidor.

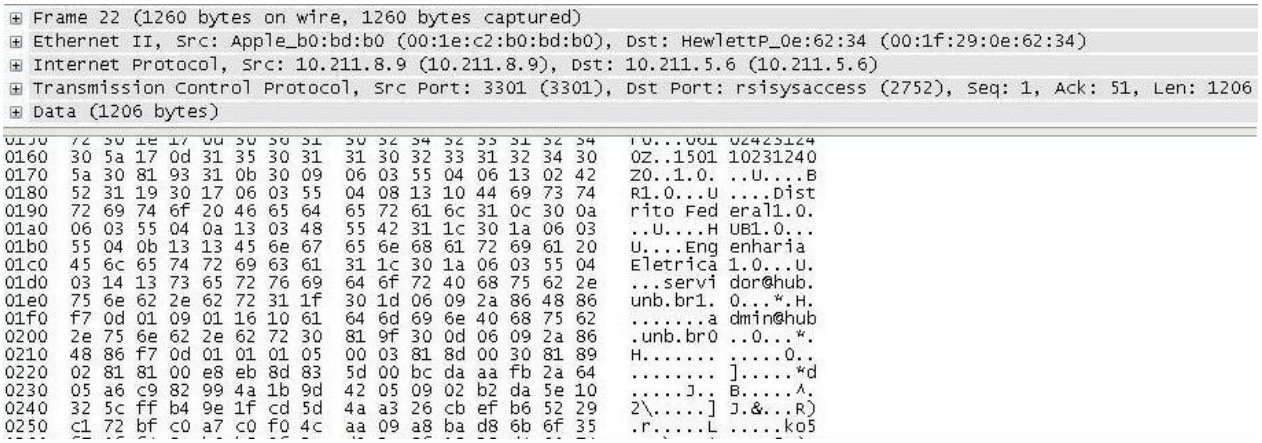

Figura 4.4 - Captura do pacote contendo o certificado digital.

Na figura 4.5 observa-se o pacote cifrado contendo a imagem transmitida entre equipamento médico e servidor. Como pode ser observado, não é possível identificar nenhuma informação sobre a imagem.

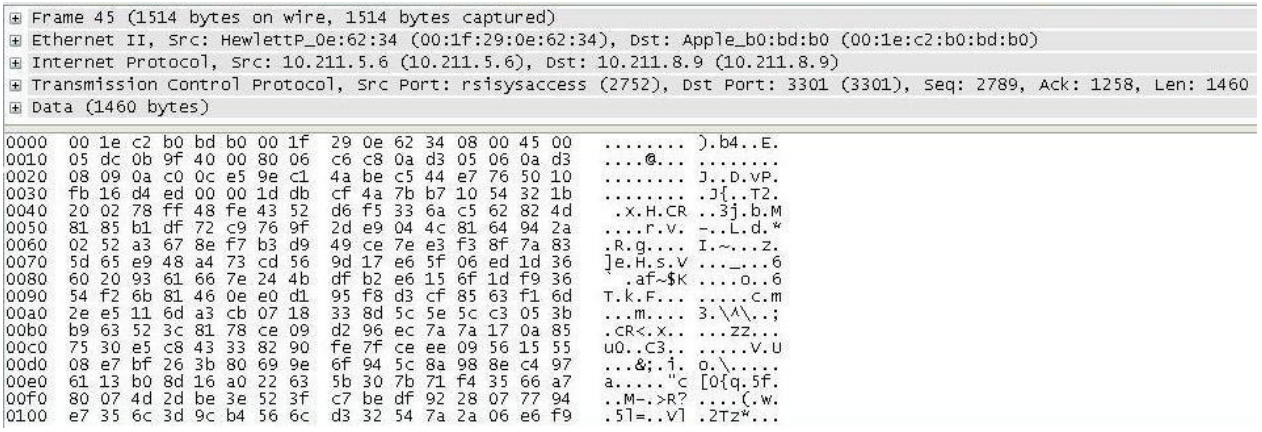

Figura 4.5 – Captura do pacote com conteúdo cifrado.

4.2.3 Tentando identificar os campos ae-called-title e ae-calling-title

Uma das vulnerabilidades detectadas no miniWebPACS é referente a possibilidade de um atacante descobrir os campos aeTitle. Para demonstrar esse problema foi transmitida uma imagem sem uso do TLS (figura 4.6) onde os campos AE são facilmente identificados.

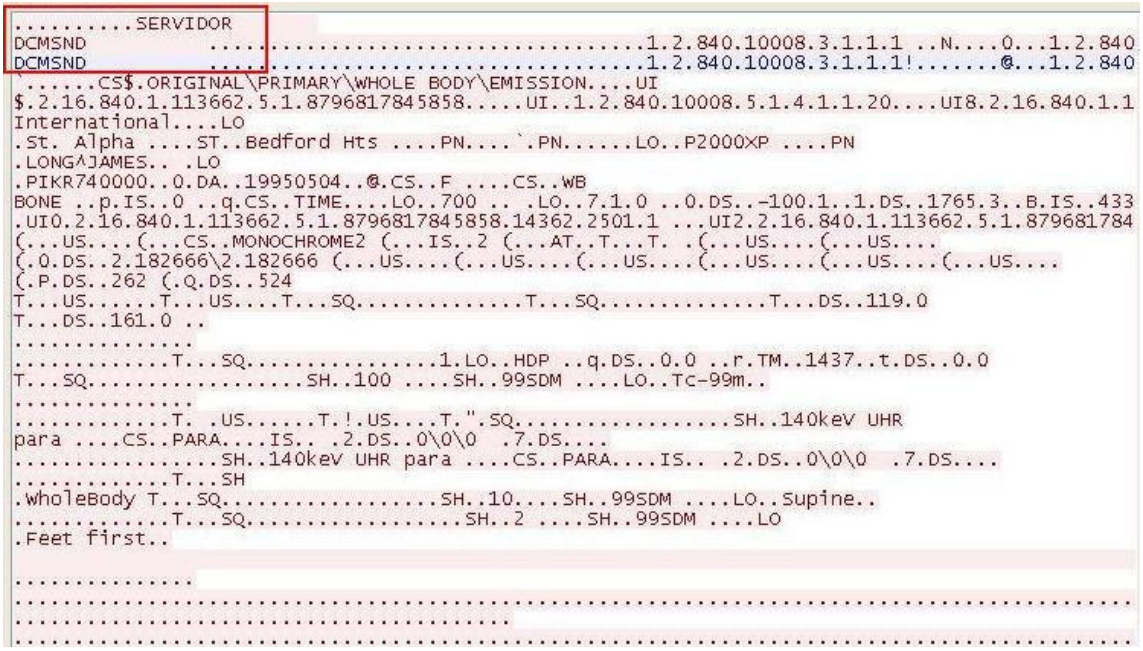

Figura 4.6 – FlowTCP Stream da transmissão sem criptografia.

Observe, também, que é possível identificar várias outras informações sobre a imagem com o tipo do exame (MONOCHROME2), o nome do paciente (LONG^JAMES), o equipamento médico (PIK740000), etc.

Na figura 4.7 mostra o mesmo tratamento de informação realizado em cima de pacotes transmitidos utilizando TLS. Observe que não é possível identificar nenhum tipo de informação sobre a imagem, apenas os dados textuais sobre o certificado digital.

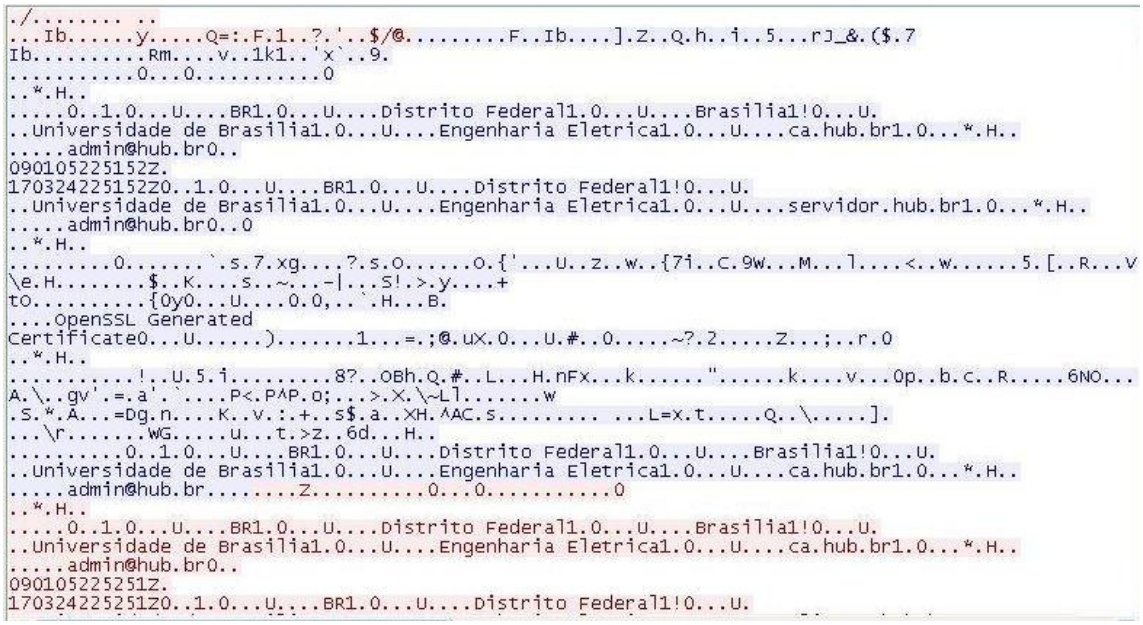

Figura 4.7 – FlowTCP Stream da transmissão com criptografia.

Com a garantia de cifra do canal de comunicação, um atacante pode até capturar o fluxo, mas não terá condições de remontar a imagem a partir destes dados.

4.2.4 Tentando transmitir uma imagem se passando pelo equipamento médico

Na figura 4.8, é possível demonstrar a tentativa frustrada de um atacante de enviar uma imagem ao servidor, tentado se passar pelo equipamento médico. É possível ver que o cliente informa os campos AETitle mas o servidor aborta a transmissão, já que não foi apresentado um certificado digital válido.

Figura 4.8 – Transmissão sem certificado digital.

4.2.5 Transmitindo uma imagem médica utilizando o hub2conversor

O projeto hub2conversor foi desenvolvido com a idéia de permitir que equipamentos médicos antigos possam transferir imagens, utilizando o protocolo TLS. Para demonstrar sua funcionalidade, foram instalados os três componentes, ou seja, equipamento médico, o hub2conversor e o hub2pacs em um único computador. Para simular o funcionamento, o

hub2conversor foi configurado para escutar na porta 3302 e o hub2pacs na porta 3301 (figura 4.9).

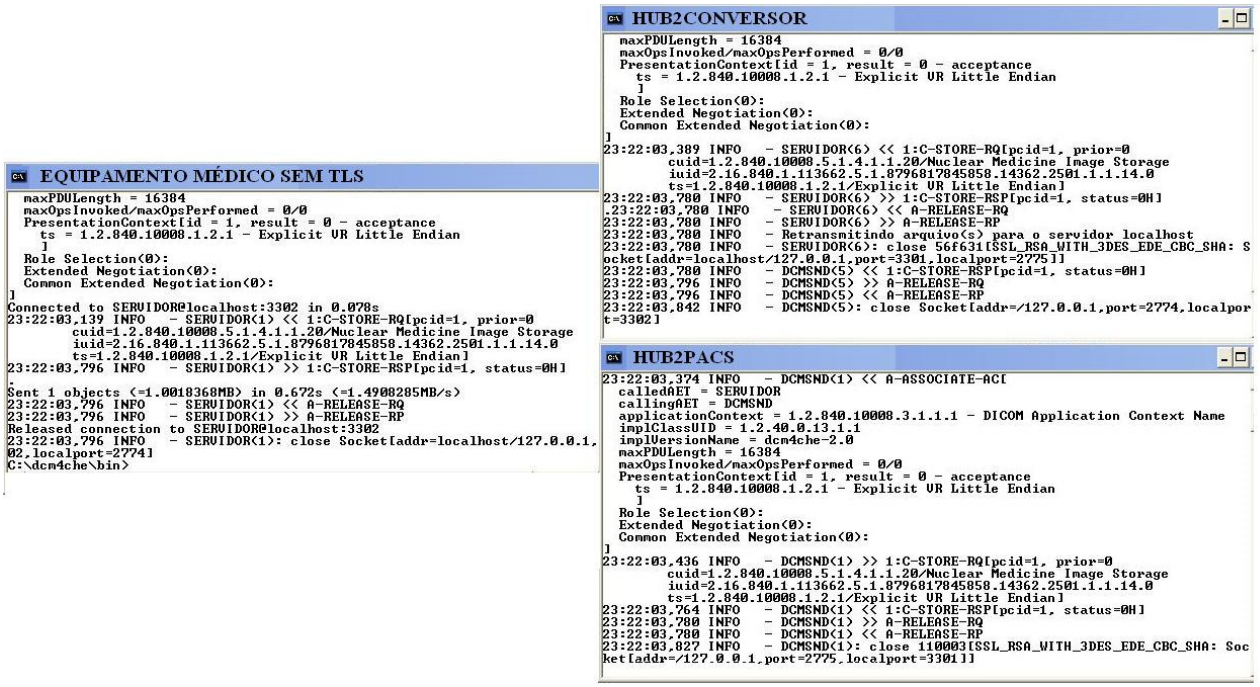

Figura 4.9 – Transmissão utilizando o hub2conversor.

Como se pode notar, o hub2conversor nada mais é do que receptor e retransmissor DICOM. Ao receber uma imagem que atenda aos requisitos AETitle, o hub2conversor a armazena e a retransmite para outro servidor, utilizando o protocolo TLS.

Para ser utilizado, portanto, faz-se necessário que este conversor seja instalado em um equipamento com duas placas de rede onde uma deverá estar ligada diretamente à interface de rede do equipamento médico antigo. É recomendado que seja instalado um filtro de pacotes (firewall) com o objetivo de controlar os acessos realizados ao hub2conversor permitindo que, somente, o equipamento médico possa utilizá-lo.

5 CONCLUSÕES E RECOMENDAÇÕES

Este trabalho teve dois grandes objetivos: demonstrar as vulnerabilidade existentes no servidor miniWebPACS em uso no Hospital Universitário de Brasília e desenvolver dois projetos: um servidor PACS que permita a transmissão de imagens médicas cifradas, utilizando certificados digitais e um conversor DICOM que permita a equipamento médicos antigos utilizarem o protocolo DICOM/TLS.

Durante a fase de demonstração de vulnerabilidades, foi possível demonstrar que o atual mecanismo utilizando pelo hospital apresenta falhas que poderiam comprometer o sigilo do paciente e denegrir a imagem da instituição. Ao todo, foram encontradas 3 (três) grandes vulnerabilidades:

- Possibilidade de enviar imagem se passando pelo equipamento médico. Quando explorada, esta vulnerabilidade permite que um invasor insira, no servidor, imagens médicas adulteradas de um determinado paciente. Isto poderá provocar erros no laudo médico o que, na prática, não pode ser aceito por um hospital;

- Possibilidade de remontar a imagem a partir de pacotes capturados. Esta falha permite que imagens médicas transmitidas possam ser visualizadas por pessoal não autorizado. O vazamento de tais informações poderia provocar perda da credibilidade do hospital bem como processos judiciais movidos por seus pacientes. As questões referentes ao sigilo do paciente são reguladas por legislação e são passíveis de multa e perda da autorização de funcionamento do hospital; e

- Possibilidade de substituir uma imagem médica válida por uma imagem forjada no servidor. Muito parecida com a primeira vulnerabilidade, esta é um pouco mais grave, pois permite a substituição da imagem médica por uma falsa. Sua exploração depende de fatores complicados e exigem do atacante um conhecimento detalhado do funcionamento do protocolo.

Todas as vulnerabilidades citadas referem-se a problemas de protocolo detectados durante a transmissão realizada entre equipamento médico e servidor. Na prática, a captura de pacotes pode ser dificultada caso um ambiente esteja utilizando, em sua rede de comunicação, equipamentos do tipo *switch.* Tal tecnologia evita que um atacante possa observar o tráfego de rede, pois a comunicação entre equipamento médico e servidor é realizada de forma isolada por meio da criação de um circuito virtual. Para que um atacante possa realizar os ataques acima citados, neste tipo de ambiente, é necessário que se efetue ataques ao *switch*, comprometendo sua tabela de endereços MAC. Tal condição faz com que o equipamento passe a não criar os circuitos virtuais, permitindo, desta forma, que todos os pacotes transmitidos na rede sejam copiados para todos os equipamentos conectados na rede.

Para solucionar os problemas detectados foi proposta a implementação de um novo servidor PACS. As vulnerabilidades encontradas foram solucionadas, utilizando conceitos de criptografia e certificação digital para a transmissão de imagens médicas. O protocolo DICOM, na sua versão 3, possui um capítulo especial sobre segurança e sugere que seja utilizado o protocolo TLS para este fim.

Como requisito fundamental, o novo Servidor PACS customizado para o Hospital de Brasília (hub2pacs) deveria utilizar o mesmo sistema de gerenciamento de banco de dados (postgres), respeitar o modelo de transmissão e formato de tabela já em uso pelo miniWebPACS. O uso da biblioteca dcm4che e da classe DBAccess foram os grandes solucionadores do problema de compatibilidade. O dcm4che permitiu manter a mesma estrutura de comunicação enquanto o DBAccess possui todas as consultas no formato adequado. Na prática, a troca do servidor antigo (miniwebPACS) para o novo servidor (hub2pacs) passou a ser, apenas, a troca do executável da aplicação.

Como já citado, foi utilizado o protocolo TLS para a transmissão das imagens. Na prática, o seu uso foi realizado para obedecer à recomendação do padrão DICOM e garantir a interoperabilidade entre sistemas de diferentes fabricantes. Todas as mensagens trocadas entre cliente e servidor passam a ser cifradas e autenticadas, utilizando certificados digitais. Como limitação, foi demonstrado que o equipamento médico precisa possuir a implementação DICOM/TLS e um local próprio para armazenamento do certificado digital. O projeto hub2conversor também desenvolvido nesta dissertação tem como objetivo vencer esta

restrição permitindo que equipamentos médicos antigos possam enviar imagens médicas de forma cifrada e autenticada.

O capítulo de revisão da literatura permitiu entender melhor os conceitos de segurança da informação e sua importância para demonstrar a necessidade de corrigir as vulnerabilidades encontradas no miniWebPACS. O detalhamento do funcionamento do protocolo DICOM permitiu compreender, de forma significativa, a implementação DICOM do dcm4che.

No capítulo de descrição funcional do protótipo, ficou demonstrado que a implementação realizada soluciona as vulnerabilidades encontradas. O novo formato de configuração do servidor permite um maior nível de customização além de facilitar a evolução do sistema sempre que uma nova versão da biblioteca dcm4che for lançada.

5.1 RECOMENDAÇÕES PARA PESQUISAS FUTURAS

Sugerem-se, para estudos futuros, os seguintes temas:

- Limitação do número de conexões realizadas entre equipamento médico e servidor. O objetivo de restringir a quantidade de conexões simultâneas é evitar ataques do tipo DOS (*Denied of Service*); e

- Guardar, de forma cifrada, todas as imagens armazenadas no servidor. Esta implementação evitará problemas de quebra da confidencialidade, caso um invasor tenha acesso físico à máquina e copie os arquivos de imagem.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] DICOM – Digital Imaging and Communication in Medicine. Disponível em: <http://medical.nema.org>. Acesso em: 31/08/2007.

[2] Martins Junior, Antonio Real (2006). "Desenvolvimento de um sistema de gerenciamento do processamento de impressão de imagens médicas digitais utilizando o protocolo dicom". Dissertação de Mestrado, Universidade de Brasília, Engenharia Elétrica, 132p.

[3] MiniWebPACS. Disponível em: <http://miniWebPACS.sourceforge.net/>, Acesso em 26/02/2008.

[4] Dcm4che. Disponível em: <http://www.dcm4che.org/>. Acesso em: 31/08/2007.

[5] Ethereal. Disponível em: <http://www.ethereal.com/>. Acesso em: 31/08/2007.

[6] Frhed. Disponível em: <http://www.kibria.de/frhed.html>. Acesso em: 31/08/2007.

[7] Adobe Photoshop. <Disponível em: http://www.adobe.com/>. Acesso em: 31/08/2007.

[8] DicomAccess. Disponível em: <http://www.desacc.com/products/da/>. Acesso em: 31/08/2007.

[9] Debian Sarge. Disponível em: <http://www.debian.org/>. Acesso em: 31/08/2007.

[10] Windows XP. Disponível em: <http://www.microsoft.com/>. Acesso em: 31/08/2007.

[11] Java SDK. Disponível em: <http://www.java.sun.com/>. Acesso em: 31/08/2007.

[12] PostgreSQL. Disponível em: <http://www.postgresql.org/>. Acesso em: 31/08/2007.

[13] EMS PostgreSQL. Disponível em: <http://www.sqlmanager.net/>. Acesso em: 31/08/2007.

[14] Sêmola, Marcos (2003). "Gestão da segurança da informação: visão executiva da segurança da informação aplicada ao Security Officer". Editora Elsevier, Rio de Janeiro.

[15] Dias, Cláudia (2000). "Segurança e Auditoria da Tecnologia da Informação". Editora Axcel Books do Brasil, Rio de Janeiro.

[16] Albuquerque, Ricardo e Ribeiro, Bruno (2002). "Segurança no Desenvolvimento de Software: Como desenvolver sistemas seguros e avaliar a segurança de aplicações desenvolvidas com base na ISO 15.408". Editora Campus, Rio de Janeiro.

[17] Beal, Adriana. (2005). "Segurança da informação: princípios e melhores práticas para a proteção dos ativos de informação nas organizações". Editora Atlas, São Paulo.

[18] Microsoft. (2006). "Guia de Gerenciamento de Risco da Microsoft". Publicado em 14/10/2006. Disponível em:

http://www.microsoft.com/brasil/security/guidance/riscos/default.mspx. Acesso em 03/11/06.

[19] Cavalcante, Sóstenes Aranha (2007). "Planejamento e implementação de melhorias na gestão". Disponível em: <http://www.abipti.org.br/tgd/ciclo2005_2006/apresentacoes/plano_melhorias/sosotenes.ppt>. Acesso em 31/08/2007.

[20] Ronald L. Krutz, Russell Dean Vine (2004). "The CISSP Prep Guide: Mastering the CISSP and ISSEP Exams", 2nd Edition. Editora Wiley.

[21] Howard, Michael e Lebland, David (2001). Writing Secure Code. Editora Microsoft Press, New York.

[22] ISO/IEC 15.408. Evaluation Criteria for IT Security.

[23] S. Vemulapalli M. Halappanavar R. Mukkamala (2002). Security in Distributed Digital Libraries: Issues and Challenges. Department of Computer Science, Old Dominion University. Publicado em: Proceedings of the International Conference on Parallel Processing Workshops (ICPPW'02) 1530-2016/02.

[24] Oosterwijk, Herman (2002). "DICOM Basics", 2a ed. Otech Inc, 2002, Estados Unidos, ISBN 0-9718867-0-9.

[25] Martins, António Cardoso (2007). "O protocolo DICOM". Disponível em: <http://dei-s1.dei.uminho.pt/outraslic/lebiom/sms/dicom.pdf>. Acesso em: 31/08/2007

[26] Digital Imaging and Communications in Medicine (2006). "Part 15: Security and System Management Profiles". PS 3.15-2006.

[27] Burnett, Steve e Paine, Stephen (2001). "RSA Security's Official Guide to Cryptography". Editora McGraw-Hill Osborne Media; 1 edition. Estados Unidos. ISBN-13: 978-0072131390.

[28] Buchmann, Johannes (2004). "Introduction to Cryptography". Editora Hardcover, Estados Unidos. ISBN: 978-0-387-21156-5.

[29] Whitfield Diffie e Martin E. Hellman (1976). "New directions in cryptography". Invited Paper. Disponível em: http://www.cs.jhu.edu/~rubin/courses/sp03/papers/diffie.hellman.pdf>. Acesso em: 01/03/2008.

[30] RFC 3280 - Internet X.509 Public Key Infrastructure Certificate.

[31] RFC 2246 - The TLS Protocol Version 1.0.

[32] Java ™ Cryptography Architecture (JCA) Reference Guide. Disponível em: <http://java.sun.com/javase/6/docs/technotes/guides/security/crypto/CryptoSpec.html> . Acesso em: 01/10/2007.

[33] RFC 4346 - The TLS Protocol Version 1.1.

[34] Miaou- Shaou-Gang, Hsu- Chin-Ming, Tsai - Yuh-Show, and Chao- Hui-Mei (2000), "A Secure Data Hiding Technique with Heterogeneous Data-Combining Capability for Electronic Patient Records", Proc. of the 22"d Annual EMBS International Conference, Chicago IL , July 23-28,2000.

[35] Brahimi-Zahia, Bessalah- Hamid, Tarabet- A., Kholladi- M. K. (2008), "A new selective encryption technique of JPEG2000 codestream for medical images transmission", Proc. of the 5th International Multi-Conference on Systems, Signals and Devices, Chicago IL.

[36] Jianguo Zhang, Fenghai Yu, Jianyong Sun, Yuanyuan Yang, Senior Member, IEEE, and Chenwen Liang (2007), "DICOM Image Secure Communications With Internet Protocols IPv6 and IPv4", IEEE Transactions on Information Technology in Biomedicine, Vol. 11, No. 1, January 2007.

[37] Cheng-Ri Piao, Dong-Min Woo, Dong-Chul Park, and Seung-Soo Han (2008), "Medical Image Authentication Using Hash Function and Integer Wavelet Transform", Congress on Image and Signal Processing, 2008.

[38] Xiaomeng Chen, Jianguo Zhang, Dongjing Wu and RuoLing Han (2005), "HIPPA's compliant Auditing System for Medical Imaging System",Proc. of the 2005 IEEE Engineering in Medicine and Biology 27th Annual Conference, Shanghai, China, September 1-4, 2005.

[39] I. Blanquer, V. Hernandez, D. Segrelles, E. Torres (2007), "Long-term storage and management of encrypted biomedical data in real scenarios",International Conference on Emerging Security Information, Systems and Technologies, IEEE Computer Society, 2007.

[40] Slobodan Kovacevic, Mario Kovac, Josip Knezovic (2007). "System for Secure Data Exchange in Telemedicine", International Conference on Telecommunications - ConTEL 2007, ISBN: 978-953-184-111-5, Zagreb, Croatia, June 13-15, 2007.

[41] Jacek Ruminski (2007), "Security of Distributed Processing of Medical Image Data Using JINI Technology", Proc. of the 29th Annual International Conference of the IEEE EMBS Cité Internationale, Lyon, France, August 23-26, 2007.

[42] Yeshwanth Srinivasan, Brian Nutter, Sunanda Mitra, Benny Phillips, and Daron Ferris (2004) “Secure Transmission of Medical Records Using High Capacity Steganography”, Proc. of the 17th IEEE Symposium on Computer-Based Medical Systems (CBMS’04), 2004.

[43] Silva, Douglas Marcos da (2001). “UML - Guia de Consulta Rápida”. Editora: Novatec.

[44] JavaTM Secure Socket Extension (JSSE) Reference Guide. Disponível em: <http://java.sun.com/javase/6/docs/technotes/guides/security/jsse/JSSERefGuide.html >. Acesso em: 01/10/2007.

[45] American National Standards Institute. Disponível em: <http://www.ansi.org>. Acesso em: 01/10/2007.

[46] Howard, Michael e Leblanc, David (2003) “Writing Secure Code - Practical strategies and techniques for secure application coding in a networked world”. 2 ed. Editora Microsoft Press, Estados Unidos. ISBN: 0735617228.

[47] Institute of Electrical and Electronics Engi. Disponível em: <http://www.ieee.org/portal/index.jsp>. Acesso em: 01/10/2007.

[48] National Electrical Manufacturers Association. Disponível em: <http://www.nema.org>. Acesso em: 01/10/2007.

ANEXO A – Criação da Autoridade Certificadora

1) Instale o OpenSSL com seu código fonte:

```
# apt-get install openssl
# apt-get source openssl
```

2) Crie o diretório onde ficará armazenado o certificado digital da CA e o Certificado do Usuário.

```
# mkdir /tmp/ca
# cd /tmp/ca
```

2) Crie o diretório onde ficará armazenado a CA.

```
# mkdir /tmp/ca/demoCA
# touch /tmp/ca/demoCA/index.txt
# mkdir /tmp/ca/demoCA/certs
# mkdir /tmp/ca/demoCA/crl
# mkdir /tmp/ca/demoCA/newcerts
# mkdir /tmp/ca/demoCA/private
# echo 01 > /tmp/ca/demoCA/serial
```

3) Criando um certificado auto-assinado para a CA

3.1) Primeiro, será necessário gerar uma nova chave privada. Certifique-se que ela não saia do servidor.

```
# openssl genrsa -des3 -out /tmp/ca/chave_privada_ca.key 2048
```

Obs.: Será solicitada uma senha para acesso a chave. Foi informado ACHub2871

3.2) Após criar a chave, o certificado para a CA pode ser criado. O parâmetro <númerodedias> especifica por quanto tempo o certificado será válido. Informe 365 ou mais dias.

```
# openssl req -new -x509 -days 3650 -key /tmp/ca/chave_privada_ca.key -out
/tmp/ca/certificado_ca.crt
```

Algumas perguntas serão feitas (veja o exemplo abaixo):

Tabela A.1 – Perguntas realizadas para a geração do certificado digital.

Country Name (2 letter code)	BR
State or Province Name (full name)	Distrito Federal
Locality Name (eg, city)	Brasilia
Organization Name (eg, company)	Universidade de Brasília
Organizational Unit Name (eg, section)	Engenharia Elétrica
Common Name (eg, YOUR name)	Nome do servidor no formato FQDN(ex. ca.hub.br)
E-mail Address	Endereço de e-mail para suporte técnico (ex. admin@hub.br)

3.3) Após estes comandos deverá ter um certificado e uma chave para a CA. Para protegê-los contra um ataque modifique as permissões de leitura e escrita somente para usuário root:

```
# chmod 600 /tmp/ca/chave_privada_ca.key /tmp/ca/certificado_ca.crt
```

O resultado é um certificado auto-assinado. Se for necessário um alto nível de segurança, pode ser utilizado uma entidade de certificação reconhecida para assinar o certificado.

4 Criar um certificado SSL para o servidor

4.1) Gerar uma chave privada com:

```
# openssl genrsa -des3 -out /tmp/ca/chave_privada_servidor.key 1024
```

Obs.: É solicitada uma senha para a chave. Foi informado ConexaoServidor2871

4.2) Em seguida criamos um *Certified Signing Request* (Requisição de assinatura para o certificado) com a chave privada RSA:

```
#openssl req -new -key /tmp/ca/chave_privada_servidor.key -out
/tmp/ca/certificado_servidor.csr
```

Mais uma vez é colocada a mesma série de questões, mas atenção: quando for perguntado o *Common Name*, informe o *Full Qualified Domain Name*, neste caso, servidor.hub.br. Esta questão é muito importante.

Obs.: É solicitado um *challenge password*. Foi deixado em branco

4.3) Assinatura da requisição

```
# openssl ca -keyfile /tmp/ca/chave_privada_ca.key -cert
/tmp/ca/certificado_ca.crt -in /tmp/ca/certificado_servidor.csr -out
/tmp/ca/certificado_servidor_ass.pen -outdir /tmp/ca -days 3000
```

Será solicitada a senha da CA. Digite-a e informe "y" (sim) para as próximas perguntas. Se tiver sucesso será mostrada a mensagem:

```
Write out database with 1 new entries
Data Base Updated
```

4.4) Conversão para formato crt

```
#openssl x509 -in /tmp/ca/certificado_servidor_ass.pen -out
/tmp/ca/certificado_servidor_ass.crt
```

5 Criar um certificado SSL para o Equipamento médico

5.1) Gerar uma chave privada com:

```
# openssl genrsa -des3 -out /tmp/ca/chave_privada_cliente.key 1024
```

Obs.: É solicitada uma senha para a chave. Foi informado ConexaoCliente2871

5.2) Em seguida criamos um *Certified Signing Request* (Requisição de assinatura para o certificado) com a chave privada RSA:

```
#openssl req -new -key /tmp/ca/chave_privada_cliente.key -out
/tmp/ca/certificado_cliente.csr
```

Mais uma vez é colocada a mesma série de questões, mas atenção: quando for perguntado o *Common Name*, informe o *Full Qualified Domain Name*, neste caso, servidor.hub.br. Esta questão é muito importante.

Obs.: É solicitado um *challenge password*. Deixar em branco

5.3) Assinatura da requisição

```
# openssl ca -keyfile /tmp/ca/chave_privada_ca.key -cert
/tmp/ca/certificado_ca.crt -in /tmp/ca/certificado_cliente.csr -out
/tmp/ca/certificado_cliente_ass.pen -outdir /tmp/ca -days 3000
```

Será solicitada a senha da CA. Digite-a e informe "y" (sim) para as próximas perguntas. Se tiver sucesso será mostrada a mensagem:

```
Write out database with 1 new entries
Data Base Updated
```

4.4) Conversão para formato crt

```
#openssl x509 -in /tmp/ca/certificado_cliente_ass.pen -out
/tmp/ca/certificado_cliente_ass.crt
```

6) Converter os certificados assinados para o formato pkcs12

```
# openssl pkcs12 -export -in /tmp/ca/certificado_cliente_ass.pen -inkey
/tmp/ca/chave_privada_cliente.key -out /tmp/ca/certificado_cliente.p12

# openssl pkcs12 -export -in /tmp/ca/certificado_servidor_ass.pen -inkey
/tmp/ca/chave_privada_servidor.key -out /tmp/ca/certificado_servidor.p12

# openssl pkcs12 -export -in /tmp/ca/certificado_ca.crt -inkey
/tmp/ca/chave_privada_ca.key -out /tmp/ca/certificado_ca.p12
```

7) Converter o certificado da AC para o formato jks. Esta atividade deve ser executada em uma máquina Windows com o JDK instalado e configurado.

```
C:\ keytool -import -trustcacerts -alias cacert -file certificado_ca.crt -
keystore ca.jks -storepass ACHub2871
```

ANEXO B – Descrição dos Atributos e Métodos do Projeto Hub2pacs

Este tópico tem como objetivo auxiliar o entendimento de cada um dos atributos e métodos propostos nas classes do sistema. Neste momento são detalhados os significados de cada atributo e de cada método, demonstrando a preocupação com a relação entre eles (tabela B.1 a B.6).

Tabela B.1 – Descrição dos atributos e métodos da classe Servidor.

Atributos	
Atributo	Descrição
int KB	Define o tamanho de 1KB.
String USAGE	Cria uma mensagem de ajuda para demonstrar o uso deste aplicativo.
String DESCRIPTION	Cria uma mensagem de ajuda que descreve este aplicativo.
String EXAMPLE	Cria uma mensagem de ajuda que mostra e comenta um exemplo de como iniciar este aplicativo.
char[] SECRET	Define a senha padrão do sistema. Uso apenas para testes.
String[] ONLY_DEF_TS	Define uma sintaxe de transferência padrão para o servidor.
String[] NATIVE_TS	Define uma sintaxe de transferência nativa para o servidor.
String[] NATIVE_LE_TS	Define uma sintaxe de transferência para o servidor.
String[] NON_RETIRED_TS	Define uma sintaxe de transferência para o servidor.
String[] NON_RETIRED_LE_TS	Define uma sintaxe de transferência para o servidor.
String[] CUIDS	Contém uma lista de UID (*unique identifier*) que podem ser manipulados pelo servidor.
Executor executor	Permite a execução da aplicação em modo *multi-thread*.
Device device	Armazena atributos referentes ao dispositivo DICOM em uso.
NetworkApplicationEntity ae	Descreve e fornece todos os serviços de rede utilizados pelo DICOM.
NetworkConnection nc	Mantém as propriedades associadas com a conexão TCP/IP.
String[] tsuids	O mesmo que o atributo NON_RETIRED_LE_TS.
File destination	Define o nome e o destino físico do arquivo que será manipulado para armazenar a objeto DICOM.
String fs_file_id	Especifica o *schema* utilizado para gerar o identificador do arquivo. O padrão é*: StudyDate, StudyID, SeriesNumber* e *InstanceNumber*.
boolean devnull	Utilizado, apenas, para direcionar resultados lógicos sem importância durante a transferência.

int fileBufferSize	Tamanho mínimo do buffer de escrita para armazenar um objeto.
int rspdelay	Atraso (*delay*) em ms para uma mensagem DIMSE-RSP. Usado apenas para teste no modo assíncrono.
String keyStoreURL	Contém o endereço físico (URL) onde está armazenado o certificado do cliente.
char[] keyStorePassword	Contém um vetor de caracteres com a senha que permite o acesso ao certificado indicado no parâmetro keyStoreURL.
char[] keyPassword	Contém um vetor de caracteres com a senha utilizada para acessar a chave privada do certificado.
String trustStoreURL	Contém o endereço físico (URL) onde está armazenado o certificado da Autoridade Certificadora.
char[] trustStorePassword	Contém um vetor de caracteres com a senha que permite o acesso ao certificado indicado no parâmetro *trustStoreURL.*
DBWriter dbWriter	Permite a manipulação do banco de dados.

Métodos		
Retorno	Método	Descrição
void	setAEtitle	Configura os campos AETitle.
void	setHostname	Configura o nome/endereço do servidor.
void	setPort	Configura a porta TCP/IP utilizada pelo servidor.
void	setTlsWithoutEncyrption	Configura o servidor para receber imagens utilizando o protocolo TLS sem criptografia do canal.
void	setTls3DES_EDE_CBC	Configura o servidor para receber imagens utilizando o protocolo TLS cifrando o canal de dados utilizando o algoritmo 3DES.
void	setTlsAES_128_CBC	Configura o servidor para receber imagens utilizando o protocolo TLS cifrando o canal de dados utilizando o algoritmo AES.
void	disableSSLv2Hello	Desativa mensagem SSLv2Hello. Se SSLv2Hello for desativado no servidor, então todas as mensagens devem obedecer aos formatos SSLv3/TLSv1.
void	setTlsNeedClientAuth	Configura o servidor obrigando ou não a autenticação utilizando o certificado digital.
void	setKeyStoreURL	Configura o endereço físico (URL) onde está armazenado o certificado do cliente.
void	setKeyStorePassword	Configura a senha que permite o acesso ao certificado indicado no parâmetro keyStoreURL.
void	setKeyPassword	Configura a senha utilizada para acessar a chave privada do certificado.
void	setTrustStorePassword	Configura a senha que permite o acesso ao certificado indicado no parâmetro trustStoreURL

void	setTrustStoreURL	Contém o endereço físico (URL) onde está armazenado o certificado da Autoridade Certificadora.
void	setPackPDV	Configura para enviar, apenas, um pacote de comandos PDV (*Protocol Data Value*) em um PDU (*Protocol Data Unit*) P-Data-TF.
void	setAssociationReaperPeriod	Configura o período que o cliente irá aguardar para solicitar uma nova associação.
void	setTcpNoDelay	Configura a opção de *socket* TCP_NODELAY. Esta opção desativa o algoritmo *Nagle* que normalmente é utilizado para reduzir o tamanho dos pacotes enviados.
void	setRequestTimeout	Configura o tempo de espera (*timeout*) em ms para receber uma mensagem ASSOCIATE-RQ (contém todas as informações que o SCU envia para o SCP durante o processo de estabelecimento de uma associação DICOM).
void	setReleaseTimeout	Configura o tempo de espera (timeout) em ms para receber uma mensagem A-RELEASE-RP(define que uma associação DICOM foi concluída com sucesso).
void	setSocketCloseDelay	Configura o tempo máximo de atraso (*delay*) para o fechamento do *socket* após o envio da mensagem A-RELEASE-RP ou A-ABORT (define que uma associação DICOM foi finalizada de forma anormal).
void	setIdleTimeout	Define o tempo máximo de espera (em ms) antes de abortar o recebimento de um objeto.
void	setDimseRspTimeout	Configura o tempo de espera (timeout) máximo antes de uma resposta DIMSE.
void	setMaxPDULengthSend	Configura o tamanho máximo em KB para envio do PDU P-DATA-TF. O P-DATA-TF é o único tipo de mensagem do protocolo responsável pela transmissão dos dados atuais.
void	setMaxPDULengthReceive	Configura o tamanho máximo para o recebimento de PDU.
void	setReceiveBufferSize	Configura o tamanho máximo do buffer para receber um objeto.
void	setSendBufferSize	Configura o tamanho máximo do buffer para enviar um objeto.
void	setDimseRspDelay	Configura o atraso (*delay*) em ms para a mensagem DIMSE-RSP. Usado apenas para teste no modo assíncrono.
CommandLine	parse	Contém os parâmetros fornecidos pelo administrador em linha de comando quando executa o arquivo.jar do projeto.

void	setTransferSyntax	Configura a sintaxe de transferência, ou seja, a codificação que será utilizada para a troca de informações.
void	initTransferCapability	Inicializa, no servidor, a capacidade de transferência de objetos.
void	setFileBufferSize	Configura o tamanho máximo do buffer para armazenar um objeto.
void	setMaxOpsPerformed	Configura o número máximo de operações pendentes assincronamente, ilimitado por padrão
void	setDestination	Configura o diretório de destino para todos os objetos DICOM recebidos.
void	initTLS	Inicia a recepção de objetos DICOM, utilizando o protocolo TLS.
KeyStore	loadKeyStore	Carregar uma chave criptográfica no formato suportado pelo Java.
InputStream	openFileOrURL	Abrir um arquivo.
String	toKeyStoreType	Configura o formato do certificado. Os formatos aceitos são: JKS ou p12
void	start	Inicia o servidor.
void	stop	Para o Servidor.
String[]	split	Divide uma string em um vetor de strings a partir de um delimitador.
void	exit	Encerra o servidor de forma indevida.
int	parseInt	Converter uma String para um Inteiro.
void	cstore	Avalia o recebimento e invoca o método apropriado para o armazenamento físico do objeto DICOM.
void	onCStoreRQ	Realiza o armazenamento físico de um objeto DICOM recebido pelo servidor. Este método insere, também, a informação do objeto no servidor de banco de dados através de uma thread.

Tabela B.2 – Descrição dos atributos e métodos da classe DBWriter.

Atributos		
Atributo	Descrição	
LinkedList queue	Cria uma lista de processos que estão em execução.	
Thread thread	Armazena um processo do tipo *Thread*.	
Métodos		
Retorno	Método	Descrição
void	start	Inicia uma *Thread*.
void	stop	Finaliza uma *Thread*.
void	schedule	Agenda a execução do comando. Este método enfileira as requisições de atualização do objeto no Banco de

		Dados.
void	run	Executa uma *Thread*.
Runnable	getJob	Busca a referência de uma *Thread* em execução.
void	update	Define a assinatura básica utilizada para atualizar um registro no banco de dados.

Tabela B.3 – Descrição dos atributos e métodos da classe Útil.

Atributos		
Atributo	Descrição	
Random RND	Gerador de número aleatório utilizado para compor o nome de uma imagem transmitida que será armazenada no diretório do servidor.	
Métodos		
Retorno	Método	Descrição
File	toDir	Criar a estrutura de subdiretórios conforme *schema* definido.
File	toFile	Cria o identificador (nome) para o objeto DICOM recebido. Uma regra interessante é que, caso um novo objeto recebido possua o mesmo identificador de um objeto já armazenado, o sistema irá "gerar" um novo identificador e evitar a sobreposição do objeto já recebido.
String[0..*]	StringtoArray	Converter uma *String* para um *Array* de acordo com um delimitador.
void	remover	Apaga, fisicamente, um determinado arquivo.
void	copiar	Copiar um arquivo para outro local.
int[0..*]	toTags	Converter um vetor de string contendo *Tags* DICOM em seus respectivos valores inteiros.
String	toFileID	Utilizado para auxiliar a criação do arquivo e do diretório. São identificadas as *Tags* utilizadas para compor o nome do arquivo.

Tabela B.4 – Descrição dos atributos e métodos da classe DB.

Atributos		
Atributo	Descrição	
Métodos		
Retorno	Método	Descrição
void	freeConnection	Define a assinatura básica para qualquer classe que deseje fechar a conexão.
Connection	getConnection	Define a assinatura básica para qualquer classe que deseje buscar a conexão ativa.

DBWriter	getDBWriter	Define a assinatura básica para qualquer classe que deseje buscar a versão ativa que será utilizada para a manipulação do banco de dados.
DB	getInstance	Buscar a instancia ativa do banco de dados.
String[]	listConections	Define a assinatura básica para qualquer classe que deseje listar as conexões ativas.

Tabela B.5 – Descrição dos atributos e métodos da classe DBPostgre.

Atributos	
Atributo	Descrição

Métodos		
Retorno	Método	Descrição
void	freeConnection	Fechar a conexão com o banco de dados postgre.
Connection	getConnection	Buscar a conexão ativa com o banco de dados postgre.
DBWriter	getDBWriter	Buscar a versão ativa que será utilizada para a manipulação do banco de dados.
DB	getInstance	Buscar a instancia ativa do banco de dados.
String[]	listConections	Listar as conexões ativas.

Tabela B.6 – Descrição dos atributos e métodos da classe DBWriterAccess.

Atributos	
Atributo	Descrição

Métodos		
Retorno	Método	Descrição
void	update	Cria a string SQL utilizada para realizar a atualização dos dados da imagem recebida pelo servidor.

ANEXO C – Descrição dos Atributos e Métodos do Projeto Hub2conversor

Este tópico tem como objetivo auxiliar o entendimento de cada um dos atributos e métodos propostos nas classes do sistema (tabela C.1 e C.2). Aqui são detalhados os significados de cada atributo e de cada método, demonstrando a preocupação com a relação entre eles.

Tabela C.1 – Descrição dos atributos e métodos da classe Transmissor.

Atributos	
Atributo	Descrição
int KB	Define o tamanho de 1KB.
int MB	Define o tamanho de 1MB.
int PEEK_LEN	Define o tamanho do buffer de leitura para objetos DICOM.
String USAGE	Cria uma mensagem de ajuda para demonstrar o uso deste aplicativo.
String DESCRIPTION	Cria uma mensagem de ajuda que descreve este aplicativo.
String EXAMPLE	Cria uma mensagem de ajuda que mostrar e comenta um exemplo de como iniciar este aplicativo
char[] SECRET	Define a senha padrão do sistema. Uso apenas para testes.
String[] ONLY_IVLE_TS	Define uma sintaxe de transferência para o transmissor.
String[] IVLE_TS	Define uma sintaxe de transferência para o transmissor.
String[] EVLE_TS	Define uma sintaxe de transferência para o transmissor.
String[] EVBE_TS	Define uma sintaxe de transferência para o transmissor.
int STG_CMT_ACTION_TYPE	Define se todas as ações realizadas pelo transmissor devem ser confirmadas.
String DCM4CHEE_URI_REFERENCED_TS_UID	Define o UID padrão para um transmissor identificando que se trata de um projeto baseado no DCM4CHE.
Executor executor	Permite a execução da aplicação em modo *multi-thread*.
Device device	Armazena atributos referentes ao dispositivo DICOM em uso.
NetworkApplicationEntity ae	Descreve e fornece todos os serviços de rede utilizados pelo DICOM. Serão armazenadas nesta variável as informações sobre a associação do lado do cliente.
NetworkApplicationEntity remoteAE	Descreve e fornece todos os serviços de rede utilizados pelo DICOM. Serão armazenadas nesta variável informações sobre a associação do lado do servidor.
NetworkApplicationEntity remoteStgcmtAE	Descreve e fornece todos os serviços de rede utilizados pelo DICOM. Serão armazenadas nesta variável informações sobre a associação do servidor/cliente quando exigido confirmação de armazenamento.

NetworkConnection remoteConn	Mantém as propriedades associadas com a conexão TCP/IP. Serão armazenadas nesta variável informações referentes à conexão do lado do servidor.
NetworkConnection remoteStgcmtConn	Mantém as propriedades associadas com a conexão TCP/IP. Serão armazenadas nesta variável informações referentes à conexão do lado do cliente/servidor quando for exigida a confirmação de armazenamento.
NetworkConnection conn	Mantém as propriedades associadas com a conexão TCP/IP. Serão armazenadas nesta variável informações referentes à conexão do lado do cliente.
Association assoc	Armazena a associação AETitle.
int transcoderBufferSize	Configura o tamanho do buffer de apoio para envio.
int filesSent	Mantém a quantidade de arquivos enviados.
long totalSize	Armazena o tamanho total do(s) objeto(s) enviado(s).
boolean fileref	Define se um arquivo será ou não referenciado.
boolean stgcmt	Controla se uma transação precisa garantir ou não a confirmação de recebimento.
long shutdownDelay	Tempo de atraso utilizado para abortar uma tentativa de transmissão sem resposta.
DicomObject stgCmtResult	Objeto utilizado para garantir o sincronismo da operação de envio.
String KeyStoreURL	Contém o endereço físico (URL) onde está armazenado o certificado do cliente.
char keyStorePassword	Contém um vetor de caracteres com a senha que permite o acesso ao certificado indicado no parâmetro keyStoreURL
char keyPassword	Contém um vetor de caracteres com a senha utilizada para acessar a chave privada do certificado.
String trustStoreURL	Contém o endereço físico (URL) onde está armazenado o certificado da Autoridade Certificadora.
char trustStorePassword	Contém um vetor de caracteres com a senha que permite o acesso ao certificado indicado no parâmetro trustStoreURL.

Métodos		
Retorno	Método	Descrição
void	setLocalHost	Configura o endereço do *host* cliente que esta enviando o objeto DICOM.
void	setLocalPort	Configura a porta que será utilizada pelo cliente para enviar o objeto DICOM.
void	setRemoteHost	Configura o endereço do *host* servidor que vai receber o objeto DICOM.
void	setRemotePort	Configura a do host servidor que vai receber o objeto DICOM.
void	setRemoteStgcmtHost	Configura o endereço do *host* servidor que vai receber o objeto DICOM. Nessa opção o servidor remoto deve confirmar o recebimento do objeto.

void	setRemoteStgcmtPort	Configura a do host servidor que vai receber o objeto DICOM. Nessa opção o servidor remoto deve confirmar o recebimento do objeto.
void	setTlsWithoutEncyrption	Configura o servidor para receber objetos, utilizando o protocolo TLS sem criptografia do canal.
void	setTls3DES_EDE_CBC	Configura o servidor para receber objetos, utilizando o protocolo TLS cifrando o canal de dados utilizando o algoritmo 3DES.
void	setTlsAES_128_CBC	Configura o servidor para receber objetos utilizando o protocolo TLS, cifrando o canal de dados utilizando o algoritmo AES.
void	disableSSLv2Hello	Desativa mensagem SSLv2Hello. Se SSLv2Hello for desativado no servidor, então todas as mensagens devem obedecer aos formatos SSLv3/TLSv1.
void	setTlsNeedClientAuth	Configura o servidor obrigando ou não a autenticação utilizando o certificado digital.
void	setKeyStoreURL	Configura o endereço físico (URL) onde está armazenado o certificado do cliente.
void	setKeyStorePassword	Configura a senha que permite o acesso ao certificado indicado no parâmetro keyStoreURL
void	setKeyPassword	Configura a senha utilizada para acessar a chave privada do certificado.
void	setTrustStorePassword	Configura a senha que permite o acesso ao certificado, indicado no parâmetro trustStoreURL
void	setTrustStoreURL	Contém o endereço físico (URL) onde está armazenado o certificado da Autoridade Certificadora.
void	setCalledAET	Configura o CalledAE (quem é chamado)
void	setCalling	Configura o CallingAE (quem está chamando)
void	setUserIdentity	Configurar a identificação do usuário. Normalmente a identificação do usuário é o campo CallingAE.
void	setOfferDefaultTransferSyntaxInSeparatePresentationContext	Oferecer transferência de sintaxe padrão em diferentes contextos de apresentação (*Presentation Context*).
void	setSendFileRef	Configura o envio de arquivos por referência.
void	setStorageCommitment	Configura a obrigação do transmissor em garantir que o armazenamento foi realizado com sucesso.
void	isStorageCommitment	Retorna se o armazenamento foi finalizado com sucesso.
void	setStgcmtCalledAETt	Configura o campo AET para o transmissor que confirma o recebimento do objeto.

void	setShutdownDelay	Configura o tempo de atraso para abortar uma transmissão pendente.
void	setConnectTimeout	Define o tempo máximo de espera (*timeout*) para uma conexão.
void	setMaxPDULengthReceive	Configura o tamanho máximo para o recebimento de PDUs (*Protocol Data Units*)
void	setMaxOpsInvoked	Define a quantidade máxima de opções que podem ser invocadas em memória.
void	setPackPDV	Configura para enviar, apenas, um pacote de comandos PDV (*Protocol Data Value*) em um PDU P-Data-TF,
void	setAssociationReaperPeriod	Configura o tempo que o transmissor irá aguardar para repetir a Associação.
void	setDimseRspTimeout	Configura o tempo de espera (timeout) máximo antes de uma resposta DIMSE.
void	setPriority	Configura a prioridade do processo de envio.
void	setTcpNoDelay	Configura a opção de *socket* TCP_NODELAY. Esta opção desativa o algoritmo *Nagle* que normalmente é utilizado para reduzir o tamanho dos pacotes enviados.
void	setReleaseTimeou	Configura o tempo de espera (*timeout*) em ms para receber até receber uma mensagem A-RELEASE-RP(define que uma associação DICOM foi concluída com sucesso).
void	setSocketCloseDelay	Configura o tempo máximo de atraso (*delay*) para o fechamento do *socket* após o envio da mensagem A-RELEASE-RP ou A-ABORT (define que uma associação DICOM foi finalizada de forma anormal).
void	setMaxPDULengthSend	Configura o tamanho máximo em KB para envio do PDU P-DATA-TF. O P-DATA-TF é o único tipo de mensagem do protocolo responsável pela transmissão dos dados atuais.
void	setReceiveBufferSize	Configura o tamanho do *buffer* para receber um objeto.
void	setSendBufferSize	Configura o tamanho do *buffer* para enviar um objeto.
void	setTranscoderBufferSize	Configura o tamanho do *buffer* de apoio para recebimento de um objeto.
int	getNumberOfFilesToSend	Pegar o número de arquivos que está sendo enviado.
int	getNumberOfFilesSent	Pegar o número de arquivos enviados.
long	getTotalSizeSent	Pegar o tamanho total de todos os arquivos enviados.
FileInfo[0..*]	getFileInfos	Pegar informações sobre todos os arquivos.

ComandLine	Parse	Contém os parâmetros fornecidos pelo administrador em linha de comando quando executa o arquivo .jar do projeto.
void	promptStgCmt	Montar mensagem para informar a confirmação do recebimento.
DicomObject	waitForStgCmtResult	Aguarda até a confirmação de recebimento do objeto DICOM por parte do servidor.
void	prompt	Montar mensagem para informar a confirmação do recebimento.
void	promptBytes	Informa a quantidade de *bytes* recebidos/enviados.
int	toPort	Informar o número da porta utilizada.
String[0..*]	Split	Dividir uma *String* em um Vetor de acordo com um delimitador.
void	Exit	Sair do programa de forma anormal.
int	parseInt	Converter uma String para Inteiro.
void	addFile	Adicionar um arquivo para ser transmitido.
void	addTransferCapability	Adicionar uma nova capacidade de transferência. Observe que existe um conjunto fixo de objetos DICOM suportados pelo dcm4che.
void	configureTransferCapability	Configurar a capacidade de transferência. Caso um objeto pertença a uma classe que não estive sido adiciona, será emitido um código de erro.
void	Start	Iniciar a transmissão.
void	Stop	Para a transmissão.
void	Open	Abrir a conexão com o servidor DICOM remoto.
void	openToStgcmtAE	Abrir a conexão aguardando uma mensagem de confirmação de recebimento.
void	Send	Enviar o(s) objeto(s).
void	Commit	Enviar mensagem para confirmar o recebimento.
String	selectTransferSyntax	Selecionar a sintaxe de transferência.
void	Close	Encerrar a conexão com o servidor remoto.
void	promptErrRSP	Imprime erros.
void	onDimseRSP	Verifica se aconteceu algum erro na resposta do servidor. Refere-se apenas a erros de DIMSE.
void	onNEventReportRSP	Verifica se aconteceu algum erro na resposta do servidor.
void	initTLS	Inicializar o transmissor para funcionar em modo TLS.
KeyStore	loadKeyStore	Carregar uma chave/certificado no formato suportado pelo Java
InputStream	openFileOrURL	Abrir um arquivo a partir do arquivo ou de sua URL.
String	toKeyStoreType	Configurar o tipo do certificado utilizado.

Tabela C.2 – Descrição dos atributos e métodos da classe Configuração.

Atributos	
Atributo	Descrição
String keypw	Senha da chave privada
String keystorepw	Senha do certificado do cliente
String keystore	URL do certificado do cliente
String truststore	URL do certificado da autoridade certificadora
String truststorepw	Senha do certificado da autoridade certificadora
String remotehost	Endereço/Nome do servidor DICOM remoto
int remoteport	Porta do servidor DICOM remoto

Métodos		
Retorno	Método	Descrição
String	getKeypw	Pegar senha da chave privada
String	getKeystorepw	Pegar senha do certificado do cliente
String	getKeystore	Pegar a URL do certificado do cliente
String	getTruststore	Pegar a URL do certificado da autoridade certificador
String	getTruststorepw	Pegar a senha do certificado da autoridade certificador
String	getRemotehost	Pegar o Endereço/Nome do servidor DICOM remoto
int	getRemoteport	Pegar a porta do servidor DICOM remoto