UNIVERSIDADE FEEVALE

SAMIR DA SILVA AVILA

PROPOSTA DE IMPLEMENTAÇÃO DE SEGURANÇA EM DADOS TRANSMITIDOS DURANTE ACESSO VIA WEB AO SISTEMA SIAP DA UFCSPA

Novo Hamburgo
2011

SAMIR DA SILVA AVILA

PROPOSTA DE IMPLEMENTAÇÃO DE SEGURANÇA EM DADOS TRANSMITIDOS DURANTE ACESSO VIA WEB AO SISTEMA SIAP DA UFCSPA

Trabalho de Conclusão de Curso
apresentado como requisito parcial
à obtenção do grau de Bacharel em
Sistemas de Informação pela
Universidade Feevale

Orientador: Profº Me. Vandersilvio da Silva
Co-Orientadora: Profª Drª. Cecília Dias Flores

Novo Hamburgo
2011

AGRADECIMENTOS

Gostaria de agradecer a todos os que, de alguma maneira, contribuíram para a realização desse trabalho de conclusão, em especial:

Primeiramente a Deus, fonte de iluminação. Minha esposa Edna e meu filho Eduardo, minha querida família, pela compreensão pelo tempo em que não pude estar junto.

Aos meus pais que sempre acreditaram em mim e aos meus irmãos queridos, amo vocês.

Ao profº Vandersilvio, pela orientação neste trabalho e ao decorrer de todo o curso. As Profª Marta e Cecília, vocês são fontes de inspiração para qualquer pessoa. A Michele, pela ajuda mais que fundamental.

Aos amigos e colegas desta caminhada, pelos bons momentos e os momentos difíceis que passamos sempre juntos. Obrigado pessoal!

Enfim, muito obrigado a todos!!!

RESUMO

É grande a utilização de sistemas informatizados na área da saúde. Isso porque a saúde é uma das áreas onde há maior necessidade de informações para a tomada de decisões. Seu uso pode ir desde um simples Prontuário Eletrônico até complexas cirurgias tele-transmitidas. Devido a esta disseminação, cresce também a preocupação com a confidencialidade dos dados dos pacientes que são transmitidos pela rede, porque podem ser alvos de pessoas mal-intencionadas e gerar grandes transtornos a uma instituição. Diante do contexto da falta de segurança em transações de dados confidencias, foram criados os protocolos criptográficos que se utilizam da criptografia para tornar sigilosos esses dados durante a sua transmissão. Esta proposta de trabalho, além de demonstrar a falta de segurança do protocolo HTTP, visa criar um canal de comunicação de dados seguro, utilizando um tipo de protocolo criptográfico, entre o Servidor (SIAP) da UFCSPA e o cliente que está acessando as informações, a fim de que sejam preservadas a confidencialidade e a integridade destas informações transmitidas durante a conexão.

Palavras-chave: Sistemas Informatizados. Segurança da Informação. Protocolos Criptográficos. Confidencialidade. Integridade.

ABSTRACT

It's large use of computerized systems in the healthcare. This is because health is one of the areas where there is greater need for information for decision-making. Its use can range from a simple Electronic Health Record to complex surgery tele-transmitted. Due to this spread, grows to the worry with the confidentiality of patient data that are transmitted over the network, because they can be targets of people malicious and generate large inconvenience to an institution. On the context of the lack of security on sensitive data transactions, were created the cryptographic protocols that use encryption to make confidential these data during the transmission. This proposed work, also demonstrate the lack of security of the HTTP protocol, aims to create a channel for secure data communication using a type of cryptographic protocol between the Server (SIAP) of the UFCSPA and the client that is accessing the information, so that they are preserved the confidentiality and integrity of information transmitted during the connection.

Key words: Computerized Systems. Information Security. Cryptographic Protocols. Confidentiality. Integrity.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AC	Authority Certification
AES	Advanced Encryption Standard
ARP	Address Resolution Protocol
ASF	Apache Software Foudation
AVI	Áudio Vídeo Interleave
CA	Autoridade Certificadora
CGI	Common Gateway Interface
CPU	Central Processing Unit
CSR	Certificate Signing Request
DES	Data Encryption Standard
DH	Diffie-Helmann
DOC	Word Processing Documents
ECC	Elliptc Curve Cryptography
ECDH	Elliptic Curve Diffie–Hellman
EDE	Encrypt Decrypt Encrypt
ENADE	Exame Nacional de Desempenho de Estudantes
EUA	Estados Unidos da América
FIPS	Federal Information Processing Standard
GIF	Graphics Interchange Format
GPL	GNU Public License
HMAC	Hash-based Message Authentication Code
HTTP	HyperText Transfer Protocol
HTTPS	Secure HyperText Transfer Protocol
IBM	International Business Machines
ICP-BRASIL	Infra-Estrutura de Chaves Publicas Brasileira
IETF	Internet Enginnering Task Force
IGC	Índice Geral de Cursos
IIS	Microsoft Internet Information Server
IP	Internet Protocol
ITI	Instituto Nacional de Tecnologia da Informação
JPEG	Joint Photographic Experts Group

LAN	Local Area Network
MAC	Message Authentication Checksun
MAC	Media Access Control
MD	Message Digest
MIT	Massachusetts Institute of Technology
MITM	Man In The Middle
MP	Medida Provisória
MPM	Multiprocessing Modules
NCSA	National Center for Computer Aplications
NIST	National Institute of Standards and Technology
NSA	National Security Agency
PDF	Portable Document Format
PHP	HyperText Preprocessor
PKI	Public Key Infrastructure
POSIX	Portable Operating System Interface for Unix
PRNG	Pseudo-randon Number Generation
PST	Personal Storage Table
RA	Regional Authorities
RES	Registro Eletrônico de Saúde
RFC	Request for Comments
RSA	Ron Rivest, Adi Shamir, Len Adleman
SHA	SecureHash Algorithm
SIAP	Sistema de Imagens Anatomopatológicas
SITS	Sistema de Imagens para teste
SSI	Server Side Interface
SSL	Secure Socket Layer
TCP	Transmission Control Protocol
TCP/IP	Tansmission Control Protocol / Internet Protocol
TIF	Tagged Image File
TLS	Transport Layer Security
UFCSPA	Universidade Federal de Ciencias da Saúde de Porto Alegre
WEP	Wired Equivalent Privacy
WWW	World Wide Web

SUMÁRIO

INTRODUÇÃO

A grande explosão da comunicação através da rede de computadores, principalmente a internet, faz crescer a necessidade de criação de mecanismos de segurança para a comunicação de dados sensíveis, pois em sua maioria, as redes não garantem a privacidade e integridade dos dados que nela trafegam (MACHADO, 2011).

Segundo Semôla (2003) e Lyra (2008), a Segurança da Informação é constituída por três princípios básicos que norteiam a sua implementação: Confidencialidade, Integridade e Disponibilidade.

- Confidencialidade: Capacidade de um sistema em permitir que informações sejam acessadas, somente por pessoas a quem são destinadas ou possua devida autorização para isso.
- Integridade: A informação precisa ser mantida nas mesmas condições quando foi disponibilizada por seu proprietário. Deve estar correta, ser verdadeira e não estar corrompida.
- Disponibilidade: A informação deve estar disponível para todos que dela necessitem.

Dentre os três princípios básicos da Segurança da Informação, a Disponibilidade, neste caso específico, não faz parte da segurança em dados transmitidos pela rede, desta forma, está fora do escopo deste projeto e não será abordada no decorrer do mesmo.

É cada vez maior o uso de sistemas informatizados em todas as áreas que se tem conhecimento. Na saúde, não poderia ser diferente. Conforme Kobayashi e Furuie (2007, p. 55), "Com a introdução das modernas tecnologias de informação e comunicação no campo médico, houve a gradual evolução de prontuários em papel para *Registros Eletrônicos de Saúde* (RES) do paciente". Esses registros, quando disponibilizados na web, trazem enormes ganhos de agilidade para a área médica, pois podem ser acessados simultaneamente e de qualquer lugar, conseqüentemente, também traz ganhos ao paciente que terá um diagnóstico mais rápido.

Esse é um exemplo de como a informática auxilia a medicina a agilizar processos e há também muitos outros exemplos que podem ser aplicados. A grande questão é, estão seguros esses registros ao serem transmitidos pela web?

A confidencialidade na medicina é um ato levado sério e está sob juramento feito pelos médicos em sua formatura da seguinte forma: "Àquilo que no exercício ou fora do

exercício da profissão e no convívio da sociedade, eu tiver visto ou ouvido, que não seja preciso divulgar, eu conservarei inteiramente secreto" (CREMESP, 2011).

A Associação Médica Mundial também se preocupa com a privacidade dos dados dos pacientes com o uso de computadores na área médica. A Declaração de Lisboa (Declaração de Lisboa, 2011), (Adotada pela 34ª Assembleia Geral Médica Mundial em Lisboa, Portugal, setembro/outubro de 1981 e emendada pela 47ª Assembleia Geral Médica Mundial em Bali, Indonésia, setembro de 1995), discorre em um dos princípios sobre o direito a confidencialidade de tudo que for identificado, sobre o estado de saúde de um paciente, deverá ser mantido em sigilo até mesmo depois de sua morte, salvo se o paciente der consentimento explicito ou se isso estiver constando na lei.

Antes disso, a Declaração de Munique (Declaração de Munique, 2011) sobre o uso do computador em medicina, (baseada em resolução adotada pela 27º Assembleia Geral da Associação Médica Mundial, República Federal da Alemanha, outubro de 1973 e emendada pela 35ª Assembleia Geral da Associação Médica Mundial em Veneza, Itália, outubro de 1983), discorre sobre os avanços e vantagens do uso do computador e dados eletrônicos que são processados no campo da saúde e também que as Associações Médicas Nacionais possam executar todos os passos para que a privacidade e a confidencialidade das informações dos pacientes sejam asseguradas.

Também existem outras declarações que tratam sobre a privacidade dos dados dos pacientes, como a de Tel Aviv (Declaração de Tel Aviv, 2011), que discorre sobre a responsabilidade e normas éticas na utilização da telemedicina. (Adotada pela 51ª Assembleia Geral da Associação Medica Mundial em Tel Aviv, Israel, em outubro de 1999), onde diz que as regras correntes do consentimento e confidencialidade do paciente também são aplicadas às situações da telemedicina, tendo o médico à obrigação de assegurar que todas as normas de medida de segurança estabelecidas para proteger a confidencialidade do paciente estejam asseguradas.

Isso deixa claro a importância da confiabilidade das informações para a área médica e os sistemas de informação precisam estar preparados para assim a garantir.

A *Universidade Federal de Ciências da Saúde de Porto Alegre* (UFCSPA) é uma Instituição Federal de Ensino Superior (UFCSPA, 2011). Fundada em 1953, mantém hoje oito cursos de graduação, seis de pós-graduação *stricto sensu*, 11 de pós-graduação *lato sensu* e 39 programas e áreas de atuação em residência médica. Todos os seus cursos são da área da saúde. Segundo "indicadores de avaliação de qualidade de ensino do Ministério da

Educação", a instituição obtém alta qualificação, apresentando resultados destacados no *Exame Nacional de Desempenho de Estudantes* (Enade) e no *Índice Geral de Cursos* (IGC) (UFCSPA, 2011).

A UFCSPA possui o *Sistema de Imagens Anatomopatológicas* (SIAP). Este foi desenvolvido com o objetivo de armazenar, organizar, otimizar o espaço físico e recuperar imagens do acervo de imagens médicas do Departamento Patológico e Medicina Legal (SPECHT, 2010).

Este sistema está disponível para acesso via web e pode ser acessado de qualquer lugar, através do protocolo *HyperText Transfer Protocol* (HTTP), permitindo assim, a disseminação e desenvolvimento do conhecimento. Em sua área restrita o acesso só é permitido utilizando um usuário e uma senha para autenticação (credenciais). Porém, os dados que estão sendo transmitidos, principalmente as credenciais para autenticação, entre o Servidor (SIAP) e o cliente que está realizando o acesso estão comprometidas. Segundo Stanger e Lane (2002), este método era seguro no início da década de 90, mas devido as sofisticadas ferramentas de rede que surgiram, assim como as pessoas que podem utilizá-las, este acesso não é confiável e com até mesmo pouco conhecimento, um agressor pode facilmente obter as informações desta conexão que estão transitando em texto puro.

Desta forma, este trabalho tem por objetivo, apresentar os riscos ao qual uma instituição médica está exposta ao transmitir dados pela rede sem o uso de criptografia. Também realizar um estudo bibliográfico sobre as principais soluções de segurança usadas atualmente que preservam a confidencialidade das informações transmitidas via rede para então implementar um canal de transferência de dados segura sobre uma rede insegura.

Este trabalho está dividido em seis capítulos, onde no primeiro capítulo será abordado o protocolo HTTP, o Servidor Web Apache e como ele se tornou o servidor mais usado no mundo.

No segundo capítulo é realizado um histórico sobre a criptografia e um estudo bibliográfico dos principais algoritmos criptográficos de chaves simétricas e assimétricas.

No terceiro capítulo é abordada a assinatura digital, um método utilizado para verificar a integridade e autenticidade de arquivos eletrônicos e também os principais algoritmos de resumos.

O quarto capítulo discorre sobre os certificados digitais, uma assinatura digital que quando assinada por uma Autoridade Certificadora garante a sua autenticidade. Também é abordada neste capítulo a infra-estrutura de chaves públicas do Brasil.

No quinto capítulo são abordados os protocolos criptográficos, uma técnica que combina algoritmos de chave simétrica e assimétrica e assinatura digital a fim de retirar de ambos as suas melhores qualidades que juntas, ajudam a proteger os dados de possíveis tentativas de espionagem e adulteração.

No sexto capítulo, em um ambiente de testes, são expostas na prática as vulnerabilidades que os dados estão sujeitos quando são transmitidos pela rede, utilizando o protocolo HTTP. Também são realizadas as configurações necessárias para a proteção destes dados.

Por último, o sétimo capítulo, mostra as configurações sendo realizadas no ambiente de produção e uma análise final sobre o nível de segurança obtido.

1 O PROTOCOLO HTTP E O SERVIDOR WEB APACHE

Neste capítulo será apresentado um histórico sobre o nascimento da web, o protocolo HTTP que constitui a base da web e suas vulnerabilidades, através de citações de outros autores, quando dados confidencias são transmitidos utilizando este protocolo. Também apresenta o Servidor Web Apache, sendo este, o Servidor onde está hospedado o Sistema de Imagens Anatomopatológicas da UFSCPA.

1.1 PROTOCOLO HTTP

A web nasceu da necessidade de colaboração, entre cientistas europeus, através de troca de relatórios, plantas, desenhos, fotos e outros documentos. Em poucos anos deixou de ser apenas um meio de transmissão de dados científicos para se tornar a aplicação usada por milhões de pessoas todos os dias, oferecendo informações sobre todos os assuntos imagináveis (TANENBAUM, 2003).

O protocolo HTTP (*HyperText Transfer Protocol*), é um protocolo da camada de aplicação, ele constitui a base para a *World Wide Web*. Quando um navegador deseja uma página web, ele envia o nome da página desejada ao servidor, utilizando o HTTP, então o servidor transmite a página de volta (TANENBAUM, 2003). A figura abaixo ilustra a comunicação cliente-servidor do protocolo HTTP.

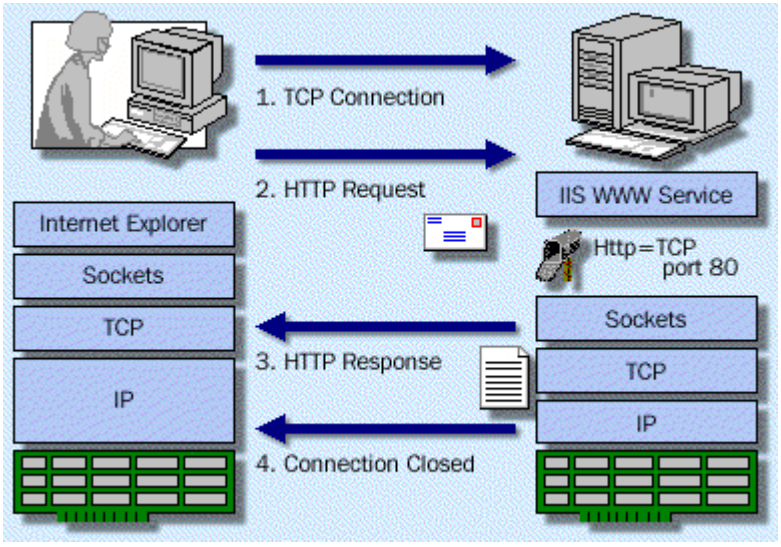

Figura 1.1: Modelo básico de funcionamento da web
Fonte: (CARNEIRO, 2011).

A tabela abaixo mostra o posicionamento do protocolo HTTP na pilha de protocolos TCP/IP (*Transmission Control Protocol/Internet Protocol*).

Tabela 1.1: Protocolos da pilha TCP/IP.

SMTP	HTTP	FTP
TCP		
IP		
PPP		
Modem, ADSL, TV a cabo		

Fonte: (TANENBAUM, 2003).

"O protocolo HTTP é um protocolo baseado em texto ASCII" (SCAMBRAY; SHEMA, 2003, p. 08). Dessa forma, as informações transmitidas entre navegador (cliente) e o Servidor Web utilizando o protocolo HTTP, a cada solicitação e resposta, estão sob o risco de terem a sua confidencialidade e integridade comprometidas. Sendo assim, informações que precisam de sigilo necessitam de algum mecanismo que possa prover segurança enquanto estão sendo transmitidas.

1.2 SERVIDOR WEB APACHE

A história do Servidor Web Apache e de seu desenvolvimento começou em 1995 quando foi criado o NCSA Web Server pela NCSA (*National Center for Computer Aplications),* tornando-se assim o Servidor HTTP mais popular da época.

Contudo, a NCSA parou de desenvolver o projeto que acabou estagnado, o que fez com que alguns desenvolvedores se desligassem da mesma e então por conta própria e ajuda de um grupo de desenvolvedores começaram a criar *patches*[1] e inovações sobre o código básico do NCSA Web Server, nascendo assim o Apache.

Assim surgiu a Fundação Apache, mantenedora do Apache, que é hoje o servidor web líder absoluto, onde estão hospedados os sites mais populares e com uma comunidade de usuários espalhados por todo o mundo, que contribuem para o desenvolvimento de novas características para o servidor ou então na correção de falhas quando possam ocorrer. Qualquer pessoa que possua o devido conhecimento para participar deste desenvolvimento é bem vindo para fazê-lo.

[1] Patch é um programa criado para atualizar ou corrigir um software.

1.2.1 CARACTERÍSTICAS DO APACHE

O Servidor Apache, segundo Kabir (2002), possui inúmeras características que fazem dele a plataforma web mais utilizada do mundo. Entre as principais estão:

- Multiplataforma: Existem versões do Apache para Linux/Unix, Windows e outros sistemas operacionais;
- Suporte ao protocolo HTTP 1.1: O Apache foi um dos primeiros servidores web a integrar o protocolo HTTP 1.1, onde foi implementada a conexão persistente, que possibilita que uma conexão seja estabelecida para enviar várias requisições em seqüência sem a necessidade de esperar por cada resposta. Desta forma, os arquivos solicitados em paralelo apareceram antes em um navegador. Anteriormente, na versão 1.0 do HTTP, as conexões entre o navegador e um servidor eram encerradas após o envio de cada requisição ou resposta, consumindo uma grande quantidade de tempo da CPU, largura de banda e de memória;
- Configuração base em um simples arquivo: O Servidor Apache não possui interface gráfica para sua administração. Sua configuração, rápida e simples, encontra-se em um poderoso arquivo chamado httpd.conf;
- Suporte para CGI's (*Common Gateway Interface*), Perl e PHP;
- Suporte a *Server Side Include* (SSI): Apache oferece um conjunto de SSI, que adicionam uma grande flexibilidade para desenvolvedores de sites web;
- Suporte a Hosts Virtuais: Possibilita hospedar diversos sites web em um único endereço IP (*Internet Protocol*);
- Suporte a *Secure Socket Layer* (SSL): Para criar canais de dados seguros.

Outra vantagem do Apache é que ele é gratuito, seu código fonte é livre e pode ser instalado em vários servidores diferentes. O apache não está coberto pela GPL (*GNU Public License*), que exige que todas as modificações feitas no código sejam tornadas públicas. O Apache está sob a Licença Apache de autoria da *Apache Software Foundation* (ASF), que não exige que estas modificações efetuadas sejam tornadas públicas por ninguém, o que acabou criando incompatibilidade entre as duas licenças.

1.2.2 APACHE PARA WINDOWS

Originalmente o NCSA foi desenvolvido para o ambiente Linux/Unix. O Apache seguiu o mesmo caminho, contudo, a versão 1.3.x do Apache funcionava na plataforma Windows, porém, não era muito popular devido às características do sistema operacional não se adequarem ao software (ausência da função fork() no windows) (MARCELO, 2005). Assim, a versão 1.3.x do Apache usava uma camada de abstração POSIX o que acabava degradando o desempenho. Foi na arquitetura 2.0 onde o Apache ganhou potência suficiente para competir com o *Microsoft Internet Information Server* (IIS) nesta área (KABIR, 2002).

De acordo com Kabir (2002, p. 61): "Embora o apache tenha sido portado para a plataforma Windows por longo tempo, o Apache 2.0 é a primeira versão que pode tirar proveito das chamadas nativas do Windows". Esse melhoramento para a plataforma Windows ocorreu principalmente com a introdução dos módulos de multiprocessamento (MPMs – *Multiprocessing modules*) em específico o MPM de winnt.

> Esse é o MPM para a plataforma Windows, incluindo o Windows 2000, o Windows NT e o Windows 9x. É um módulo de vários threads. Com esse módulo, o Apache criará um processo pai e um processo filho. O processo filho irá gerar todos os threads que atenderão à solicitação. Além disso, agora esse módulo também aproveita algumas chamadas de funções naturais apenas no Windows; isso permite que ele funcione melhor que as versões anteriores do servidor Apache na plataforma Windows (KABIR, 2002, p. 10).

Foi desta forma que o Apache tornou-se o servidor web número um. Com implementações e melhorias constantes em seu código ele é hoje utilizado em mais de 60% dos servidores web no mundo. A figura e a tabela abaixo mostram as estatísticas de uso dos principais servidores web no mundo.

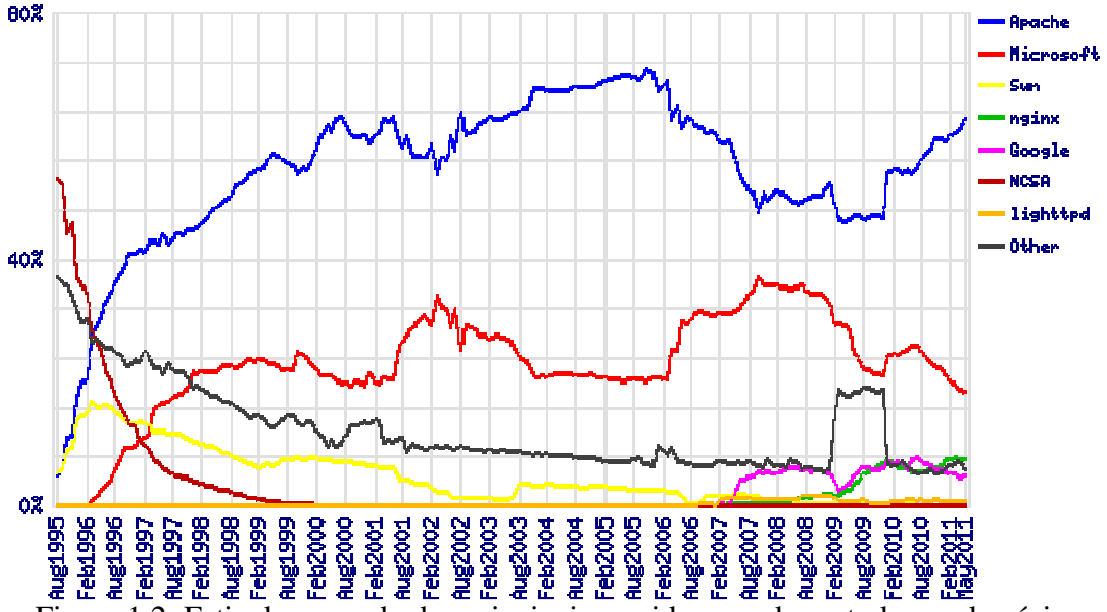

Figura 1.2: Fatia de mercado dos principais servidores web em todos os domínios
Fonte: (NETCRAFT, 2011).

Tabela 1.1: Tabela com números e percentuais de uso dos principais servidores web do mundo

Developer	April 2011	Percent	May 2011	Percent	Change
Apache	191,139,966	61.13%	203,609,890	62.71%	1.58
Microsoft	58,867,097	18.83%	59,646,778	18.37%	-0.46
nginx	23,463,669	7.50%	23,850,265	7.35%	-0.16
Google	14,690,422	4.70%	16,219,824	5.00%	0.30
lighttpd	1,862,963	0.60%	1,884,876	0.58%	-0.02

Fonte: (NETCRAFT, 2011).

2 CRIPTOGRAFIA

Este capítulo tem como objetivo apresentar o estudo realizado sobre a criptografia, uma das ferramentas mais importantes para a proteção dos dados, e alguns métodos utilizados para transformar dados legíveis em algo ilegível.

A palavra criptografia vem das palavras gregas que significa “escrita secreta” e possui uma historia de milhares de anos (TANENBAUM, 2003).

Conforme Trigo (2007, p. 92 apud MENEZES, 1997): “Criptografia é o estudo de técnicas matemáticas relacionadas aos aspectos de segurança da informação como confiabilidade, integridade dos dados, autenticação de entidade e autenticação de origem”.

O uso da criptografia é muito antigo. Os primeiros registros sobre o uso de confidencialidade apareceram no tempo dos reis, em que estes selavam as cartas com seu anel, particular de cada rei, e cera. Quando recebida à carta, o outro rei sabia que os dados continuavam sigilosos porque a carta continuava selada (TRIGO, 2007).

A criptografia também foi muito utilizada nas guerras da antiguidade. O Império Romano utilizava criptografia para enviar informações a seus aliados, porém mesmo que a mensagem fosse interceptada o inimigo não poderia saber o seu conteúdo. O Império Romano utilizava o seguinte método de criptografia, avançando quatro letras a partir da letra original (TRIGO, 2007).

A - B - C - D - E - F - G - H - I - J - K - L - M - N - O - P - Q - R - S - T - U - V - W - X - Y - Z

E - F - G - H - I - J - K - L - M - N - O - P - Q - R - S - T - U - V - W - X - Y - Z - A - B - C - D

Exemplo de uma palavra criptografada através deste método:

SEGREDO = WIKVIHS

2.1 ALGORTIMOS DE CRIPTOGRAFIA

A criptografia consiste em duas funções, a primeira cifra a mensagem e a segunda decifra. A segurança de algoritmos que utilizam este método está na sua obscuridade. Essa estratégia pode ser muito perigosa. “Imaginar que um algoritmo de criptografia é secreto quando ele não é resulta em mais prejuízo do que benefícios” (TANENBAUM, 2003, p. 772).

Para melhorar esse modelo, foi acrescentado um elemento chamado chave, onde reside exclusivamente o segredo, também chamado de princípio de Kerckhoff, em homenagem ao criptógrafo militar Auguste Kerckhoff.

"Princípio de Kerckhoff: Todos os algoritmos devem ser públicos; apenas as chaves são secretas" (TANENBAUM, 2003, p. 773).

Burnett e Paine (2002), também concordam que os algoritmos de criptografia públicos são mais seguros que algoritmos secretos, segundo estes, os invasores sempre quebram o algoritmo e é mais fácil proteger uma chave do que proteger um algoritmo.

Quando o sigilo está na chave, o tamanho desta é uma questão de extrema importância. O Tamanho da chave está diretamente relacionado ao fator de trabalho que um criptoanalista terá caso tente realizar a decodificação.

Conforme Tanenbaum (2003, p. 773): "Uma chave com tamanho de dois dígitos significa que existem 100 possibilidades, uma chave de três dígitos significa 1000 possibilidades, e uma chave de seis dígitos significa um milhão de possibilidades".

> O sigilo é decorrente de um algoritmo forte (mas público) e de uma chave longa. Para impedir que seu irmãozinho leia suas mensagens de correio eletrônico, serão necessárias chaves de 64 bits. Para uso comercial de rotina, devem usados pelo menos 128 bits. Para manter o governo de outros países à distância, são necessárias chaves de pelo menos 256 bits, de preferência maiores (TANENBAUM, 2003, p. 773).

De acordo com Tanenbaum (2003), não basta ter um algoritmo forte e uma chave fraca ou uma chave forte e um algoritmo fraco. A melhor estratégia é selecionar um algoritmo que não seja fraco e utilizar chaves longas. As chaves podem ser simétricas ou assimétricas.

2.1.1 CHAVE SIMÉTRICA

Segundo Trigo (2007), uma chave é simétrica quando a mesma chave é utilizada para codificar e decodificar, dessa forma, emissor e receptor devem conhecer a chave secreta ora em uso. O grande problema desse sistema é encontrar uma forma segura para a troca de chaves, principalmente quando emissor e receptor estão em locais separados fisicamente. Alguns algoritmos que utilizam o método simétrico são:

2.1.1.1 PRINCIPAIS ALGORITMOS DE CHAVE SIMÉTRICA

2.1.1.1.1 DES – Data Encryption Standard

Na década de 70, pesquisadores da IBM desenvolveram um algoritmo criptográfico que foi chamado de Lucifer e utilizava uma chave de 128 bits (BURNETT; PAINE, 2002).

O DES (*Data Encryption Standard*) foi baseado no Lucifer. A pedido da NSA (*National Security Agency*), responsável por proteger os dados secretos do governo dos Estados Unidos, a IBM reduziu a chave de 128 bits para 56 bits e manteve em segredo o processo segundo o qual o DES foi projetado. Segundo Tanenbaum (2003), suspeita-se que o tamanho da chave foi reduzido para que a NSA pudesse decifrar mensagens criptografadas através deste algoritmo, mas nenhuma outra organização com um orçamento menor pudesse fazê-lo.

Na década de 1990, com os computadores cada vez se tornando mais rápidos, criptógrafos sabiam que o DES não duraria muito tempo. “Em 1999, na RSA Conference, a Eletronic Frontier Foundation quebrou uma chave de DES em menos de 24 horas. O mundo precisava de um substituto” (BURNETT; PAINE, 2002, p. 40).

2.1.1.1.2 TRIPLE DES

O Triple DES é um substituto ao vulnerável DES. Como o próprio nome diz este algoritmo realiza três vezes o algoritmo DES. Segundo Tanenbaum (2003), este algoritmo utiliza três estágios e duas chaves. No primeiro estágio o texto é criptografado com K1. No segundo estágio o texto é descriptografado com K2 e por fim criptografado novamente com K1. A figura abaixo ilustra a criptografia e descriptografia utilizando o algoritmo triple DES.

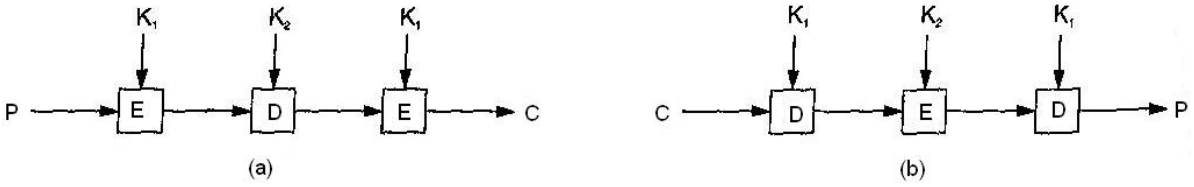

Figura 2.1: (a) Criptografia usando o triple DES. (b) Descriptografia
Fonte: (TANENBAUM, 2003).

Dessa forma o Triple DES utiliza chaves de 112 bits, mas não significa que, como o DES foi quebrado em menos de 24 horas, possa ser quebrado em 48 horas. Segundo Burnett e

Paine (2002), levam 24 horas se você souber que a quebrou, com o Triple DES se apenas uma das chaves estiver correta o texto continuará sem sentido e o invasor não saberá que uma das chaves está correta.

Um dos motivos para o EDE (*Encrypt Decrypt Encrypt*) ser usado é que ele mantém compatibilidade retroativa com os sistemas DES (TANEMBAUM, 2003).

Um dos problemas do Triple DES, segundo Burnett e Paine (2002), é a velocidade. Como o DES já era considerado um algoritmo lento para encriptar e decriptar, o Triple DES é três vezes mais lento, o que reduz de tal forma o desempenho que alguns aplicativos podem não funcionar.

2.1.1.1.3 ADVANCED ENCRYPTION STANDARD (AES)

Com as vulnerabilidades que o DES estava apresentando sua vida útil estava chegando ao fim. Mesmo com o Triple DES, o NIST (*National Institute of Standards and Technology*), órgão do departamento de comércio dos Estados Unidos encarregado de aprovar padrões para o Governo Federal dos Estados Unidos, concluiu que precisava de um novo padrão criptográfico (TANENBAUM, 2003).

Em dois de janeiro de 1997, foi anunciado e patrocinado pelo NIST um concurso onde qualquer pessoa poderia submeter propostas para o novo padrão que seria conhecido como AES (*Advanced Encryption Standard*). Segundo Tanenbaum (2003), as regras do concurso eram:

- O algoritmo teria de ser uma cifra de bloco simétrica;
- Todo o projeto teria de ser público;
- Deveriam ser admitidos tamanhos de chaves iguais a 128, 192 e 256 bits;
- Teriam de ser possíveis implementações de software e de hardware;
- O algoritmo teria de ser público ou licenciado em termos não-discriminatórios.

De todas as propostas recebidas 15 foram selecionadas, das quais os algoritmos foram analisados mundialmente através dos seguintes critérios: segurança, desempenho e tamanho. Em agosto de 1999, essa lista foi reduzida para apenas cinco candidatos que tiveram seus algoritmos testados por pesquisadores, analistas de criptografia e fornecedores de hardware e de software (BURNETT; PAINE, 2002). Depois de realizadas outras conferências e tentativas de encontrar falhas nos algoritmos, houve uma votação (TANENBAUM, 2003):

- Rijndael (de Joan Daemen e Vincent Rijmen, 86 votos);

- Serpent (de Ross Anderson, Eli Biham e Lars Knudsen, 59 votos);
- Twofish (de uma equipe liderada por Bruce Schneier, 31 votos);
- RC6 (da RSA Laboratories, 23 votos);
- MARS (da IBM, 13 votos).

Em outubro de 2000, o NIST anunciou que também votou no Rijndael e em novembro de 2001 o Rijndael tornou-se um padrão do Governo dos Estados Unidos publicado como *Federal Information Processing Standard* (FIPS 197) (TANENBAUM, 2003).

"Sob uma perspectiva matemática, o Rijndael se baseia na teoria de campo de Galois, o que proporciona ao algoritmo algumas propriedades de segurança demonstráveis" (TANENBAUM, 2003, p. 790).

2.1.1.1.4 RC4

O RC4 é um algoritmo de criptografia amplamente utilizado em aplicações do mundo real, tais como Microsoft Office, *Secure Socket Layer* (SSL), *Wired Equivalent Privacy* (WEP), etc. Devido a sua popularidade e simplicidade, o RC4 tornou-se alvo de criptoanalistas que acabaram por encontrar algumas deficiências, tornando-o inseguro para algumas aplicações (GARAY; PRISCO, 2010).

Segundo Oppliger (2009), a deficiência encontrada no RC4 é usada no WEP, protocolo utilizado para proteger redes sem fio.

2.1.1.1.5 OUTRAS CIFRAS

Além dos algoritmos criptográficos mencionados, também os mais conhecidos, existem outros que estão incorporados a vários produtos. A tabela abaixo mostra alguns desses algoritmos de chave simétrica mais comuns.

Tabela 2.1: Alguns algoritmos criptográficos de chave simétrica comuns

Cifra	**Autor**	**Comprimento da chave**	**Comentários**
Blowfish	Bruce Schneier	1 a 448 bits	Velho e lento
DES	IBM	56 bits	Muito fraco para usar agora
IDEA	Massey e Xuejia	128 bits	Bom, mas patenteado
RC4	Ronald Rivest	1 a 2.048 bits	Atenção: algumas chaves são

			fracas
RC5	Ronald Rivest	128 a 256 bits	Bom, mas patenteado
Rijndael	Daemen e Rijmen	128 a 256 bits	Melhor escolha
Serpent	Anderson, Biham, Knudsen	128 a 256 bits	Muito forte
DES Triplo	IBM	168 bits	Segunda melhor escolha
Twofish	Bruce Schneier	128 a 256 bits	Muito forte amplamente utilizado

Fonte: (TANENBAUM, 2003).

2.1.1.2 GERANDO UMA CHAVE

Em algoritmos simétricos a chave é apenas um número. Esse número precisa ter um tamanho considerável e ser aleatório. Computadores são determinísticos, portanto previsíveis. Para que bons números aleatórios sejam gerados, o computador deve ter duas coisas: um algoritmo de geração de bons números aleatórios e uma semente aleatória e imprevisível.

Segundo Burnett e Paine (2002) os algoritmos PRNGs (*Pseudo-random number generation*), geradores de números pseudo-aleatórios, são capazes de gerar números que passam em testes de aleatoriedade. Esses números são considerados pseudo-aleatórios e não-aleatórios, porque ele pode ser executado diversas vezes, até mesmo em computadores diferentes, porém o resultado é sempre o mesmo.

Para que a saída seja alterada é utilizada uma entrada conhecida como semente. Ainda segundo Burnett e Paine (2002), uma semente pode ser várias coisas, a hora do dia em milissegundos, posição do cursor do mouse, o status da memória, o volume de áudio e vários outros parâmetros. Um PRNG receberá esses números, descartando os bits de baixa entropia e os converte em números que passam em testes estatísticos de aleatoriedade, mas que poderão ser repedidos. A figura abaixo mostra um gerador de números pseudo-aleatórios coletando informações de semente e convertendo em números aleatórios.

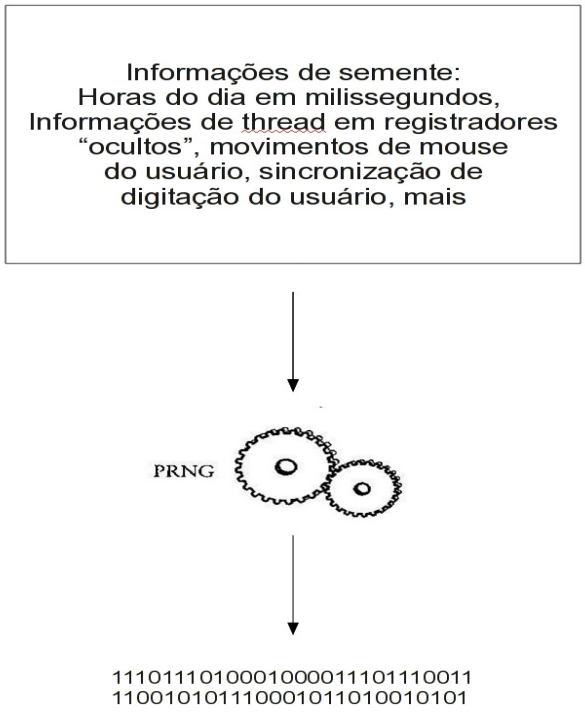

Figura 2.2: Gerador de números pseudo-aleatórios
Fonte: (BURENTT; PAINE, 2002)

Com o uso de um PRNG e um algoritmo forte, como visto anteriormente, chegamos a um alto nível de segurança para os dados que precisam ser mantidos em sigilo. Porém, segundo Tanenbaum (2002), a distribuição de chaves sempre foi o calcanhar de Aquiles da maioria dos sistemas de criptografia. Qualquer intruso que puder interceptar a chave, durante sua transmissão, poderá ler, modificar e forjar qualquer mensagem cifrada por ela. Para resolver este problema, surgiu a criptografia de chave assimétrica.

2.1.2 CHAVE ASSIMÉTRICA

Na criptografia de chave simétrica, emissor e receptor da mensagem precisam compartilhar a chave secreta sem que ninguém mais tenha conhecimento desta chave. Se estiverem em localidades fisicamente distintas, precisam encontrar um meio seguro para compartilhá-la. Se alguém, de alguma forma, interceptar a chave em trânsito, poderá ler e modificar a mensagem cifrada por ela, independente do quão sólido for o sistema criptográfico utilizado o mesmo se torna inútil.

Para resolver este problema, em 1976, o aluno Whitfield Diffie, graduado pela Stanford University e o professor Martin Hellman, propuseram um novo sistema de criptografia, que ficou conhecido como criptografia de chave pública ou chave assimétrica, onde a chave utilizada para criptografar a mensagem não era a mesma para descriptografar, permitindo que duas pessoas pudessem criar uma chave secreta compartilhada trocando informações públicas. Desde então vários pesquisadores têm se dedicado, devido às vantagens da criptografia de chave pública, ao seu estudo com alguns algoritmos sendo publicados (BURNETT; PAINE, 2002).

2.1.2.1 PRINCIPAIS ALGORITMOS DE CHAVE ASSIMÉTRICA

2.1.2.1.1 RSA

Em 1978, intrigados com a idéia de Diffie e Hellman, um grupo de pesquisadores do MIT (*Ron Rivest, Adi Shamir e Len Adleman*), criou o RSA, iniciais dos três estudiosos, um sistema de criptografia assimétrica ou de chave pública, onde cada entidade possui um par de chaves, uma pública e outra privada (TANENBAUM, 2003). A chave pública, derivada da chave privada, é compartilhada com todo mundo sem restrições, enquanto a chave privada deve ser mantida em segredo. Quando um emissor (E) tiver a necessidade de enviar uma mensagem criptografada a um receptor (R), E fará uso da chave pública de R para criptografar a mensagem e então enviá-la. Desta forma, a mensagem criptografada só poderá ser descriptografada por R através de sua chave privada.

O RSA é considerado um algoritmo muito forte, onde a segurança do método se baseia na dificuldade de fatorar números grandes. Segundo Burnett e Paine (2002), até então ninguém foi capaz de fatorar números extensos em um espaço de tempo razoavelmente curto. “Atualmente, o tamanho de chave RSA mais comumente utilizado é de 1.024 bits. O registro de fatoração (até dezembro de 2000) é de 512 bits” (BURNETT; PAINE, 2002, p. 87). Para realizar este trabalho, onde p e q tinham tamanho de 256 bits, foram utilizados 292 computadores, desktops comuns, e levou mais de cinco meses para a conclusão. A figura abaixo ilustra uma chave RSA de 512 e 1024 bits, os módulos têm 512 bits e 1024 bits, construídos pela multiplicação de números primos.

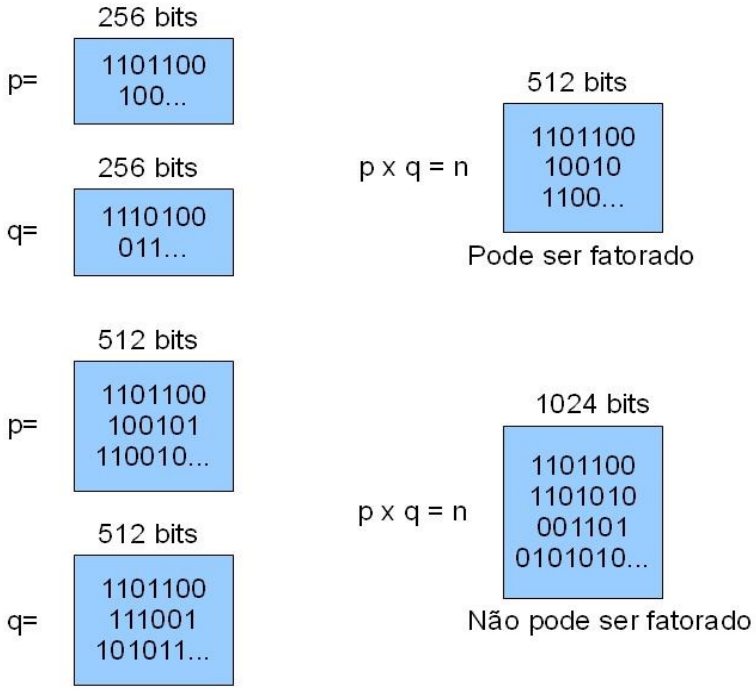

Figura 2.3: Uma chave RSA de 1024 e 512 bits, com módulos construídos pela multiplicação de dois números primos.
Fonte: (BURNETT; PAINE, 2002).

2.1.2.1.1.1 FATORANDO UMA CHAVE RSA DE 768 BITS

Em um artigo publicado por cientistas da Suíça, Japão, Alemanha, França, EUA e Holanda, onde discorre que, em 12 de dezembro de 2009, eles fatoraram uma chave RSA de 768 bits. O trabalho levou aproximadamente dois anos e meio para ser concluído. A primeira etapa levou seis meses com 80 processadores na seleção polinomial. Já a segunda etapa levou quase dois anos com um cluster de centenas de computadores. Caso fosse realizada com apenas um computador com processador 2,2 GHz AMD Opteron com 2 GB de RAM, esta etapa levaria cerca de 1500 anos (KLEINJUNG et al, 2010).

Segundo este artigo, fatorar um módulo RSA de 1024 bits é cerca de mil vezes mais difícil, enquanto fatorar um módulo RSA de 768 bits é milhares de vezes mais difícil em relação a um módulo de 512 bits. Porém, umas das conclusões dos cientistas é que, ao menos que algo dramático aconteça na fatoração, eles não serão capazes de fatorar uma chave RSA de 1024 bits dentro dos próximos cinco anos, mas alerta que seria prudente a eliminação progressiva ao uso da RSA de 1024 bits nos próximos quatro anos (KLEINJUNG et al, 2010).

2.1.2.1.2 O ALGORITMO DH

O algoritmo *Diffie-Hellman* (DH), diferentemente do RSA, não criptografa os dados. Para estabelecer uma comunicação segura, emissor e receptor geram o mesmo segredo que será utilizado como chave de sessão e será utilizada em um algoritmo simétrico. O *acordo de chaves* como é chamado este procedimento, segundo Burnett e Paine (2002), ocorre da seguinte forma: O Receptor (R), possui um par de chaves de DH, uma chave pública e uma privada. Dentro da chave pública de R, existem informações para que o Emissor (E) gere seu próprio par de chaves de DH temporário que estarão relacionados ao par de chaves de R. Agora E poderá gerar um número chamado de *valor secreto* utilizando, ao mesmo tempo, sua chave privada e a chave pública de R. Agora para R encontrar o mesmo valor secreto ele poderá gerá-lo utilizando a chave pública de E e sua chave privada. A figura baixo ilustra este procedimento.

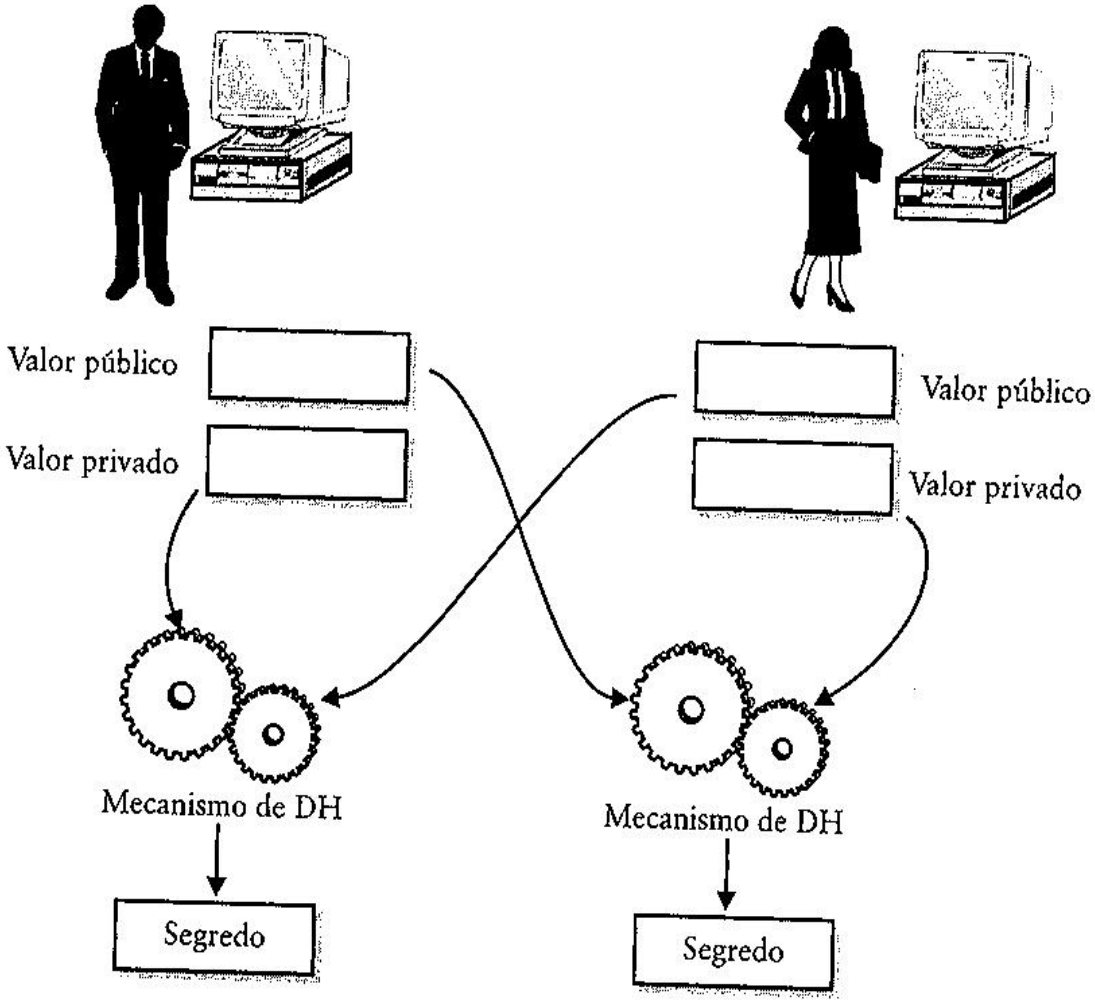

Figura 2.4: Combinação do valor privado com o valor público para criar um segredo em ambas as partes utilizando o algoritmo DH
Fonte: (BURNETT; PAINE, 2002).

A segurança do algoritmo DH advém do conhecido problema do logaritmo discreto e assim como há problemas de fatoração de números grandes está relacionado o problema de calcular logaritmos discretos. "É comumente aceito o fato de que resolvendo um deles, você resolve ambos" (BURNETT; PAINE, 2002, p. 93).

2.1.2.1.3 ELLIPTC CURVE CRYPTOGRAPHY (ECC)

Em 1985, Neal Koblitz e Victor Miller propuseram uma nova forma de implementação de um sistema de chave assimétrica em algumas das aplicações existentes, como, por exemplo, o algoritmo DH denominando-se assim ECDH (*Elliptic Curve Diffie–Hellman*), usando curvas elípticas. Assim, trabalhando no domínio das curvas elípticas, podem prover sistemas criptográficos de chave pública mais seguros e com chaves de menor

tamanho, eliminando dessa forma um dos grandes problemas dos algoritmos de chave pública, o grande tamanho de suas chaves (GOMES; RIBEIRO, 2004).

Quanto à segurança de algoritmos que se baseiam em curvas elípticas, Burnett e Paine (2002) discorrem que os algoritmos RSA e DH possuem tamanhos de chaves iguais porque se baseiam em problemas semelhantes, problema de fatoração de inteiros e problema do logaritmo discreto, respectivamente. Porém, com o ECC o tamanho de chave é diferente, porque para resolver o problema de logaritmo discreto de curvas elípticas é mais difícil que as anteriores. Diz-se que uma chave de 160 bits de ECC é tão segura quanto uma chave de 1024 bits de RSA ou de DH.

2.1.2.2 COMPARANDO OS ALGORITMOS DE CHAVE ASSIMÉTRICA

Segundo Burnett e Paine (2002), não há como chegar a uma resposta simples, para saber qual destes algoritmos de chave assimétrica é o melhor, pois cada um tem suas vantagens e desvantagens. Em questões relacionadas à segurança e desempenho não se chegou a um resultado final concreto que pudesse avaliar qual seria a melhor opção, nestes quesitos eles se equivalem. Já na questão interoperabilidade há uma vantagem ao RSA porque ele tornou-se um padrão, sendo predominante ao DH e este predominante ao ECC. Segundo Coulouris, Dollimore e Kindberg (2007), os algoritmos ECC serão, provavelmente, amplamente adotados no futuro, especialmente em dispositivos móveis, onde a capacidade de processamento é limitada, por utilizarem tamanhos de chaves menores.

3 ASSINATURA DIGITAL

Este capítulo tem por objetivo apresentar a Assinatura Digital, uma tecnologia que possibilita garantir a integridade e autenticidade a arquivos eletrônicos.

A criptografia de chave pública resolveu o problema de distribuição de chaves e também questões relacionadas à criptografia, como autenticação e não-repudio. Como visto anteriormente, quando uma mensagem é criptografada com a chave pública de alguém, somente esta pessoa poderá descriptografá-la utilizando sua chave privada. Mas o inverso também é possível. Utilizando-se do algoritmo de criptografia RSA, se uma mensagem for criptografada com a chave privada, ela poderá ser descriptografada por qualquer pessoa que tiver a sua chave pública, que está publicamente disponível. Utilizando-se deste método, não é possível manter a confidencialidade dos dados, mas é uma maneira de assegurar a identidade do autor da mensagem (autenticidade), impedindo que o mesmo retifique sua palavra eletrônica (não-repudio). "Se uma chave pública decripta os dados adequadamente, então ela deve ter sido encriptada com a chave privada. Na comunidade da criptografia, essa técnica é convencionalmente chamada de *assinatura digital*" (BURNETT; PAINE, 2002, p. 116).

Mas, dessa forma, chegamos novamente ao problema da criptografia de chave pública ser lenta, não é conveniente encriptar o texto simples inteiro. O melhor método nesse caso, segundo Burnett e Paine (2002) é encriptar apenas um resumo de mensagem.

3.1 RESUMOS DE MENSAGEM

Segundo Tanenbaum (2003), esse esquema se baseia em uma espécie de impressão digital de um documento maior, "[...] extrai um trecho qualquer do texto e a partir dele calcula uma string de bits de tamanho fixo" (TANENBAUM, 2003, p. 807). Calcular este sumário é simples e mais rápido, em relação à criptografia com algum algoritmo de chave pública, utilizando algoritmos de assinatura digital. Esta é também uma forma de garantir a integridade dos dados.

3.1.1 ALGORITMOS DE RESUMOS DE MENSAGEM

Os algoritmos de Resumos de Mensagem ou simplesmente MD (*Message Digest*) geram um resumo de um texto. Se esse mesmo texto for resumido utilizando-se o mesmo algoritmo em outro computador os resumos serão idênticos. Caso o texto sofra qualquer alteração, até mesmo a remoção ou adição de um simples espaçamento, fará com que a saída do resumo seja totalmente diferente da anterior (BURNETT; PAINE, 2002).

Segundo Burnett e Paine (2002), com base em um resumo de texto, até então, ninguém foi capaz de descobrir a mensagem, pois se trata de uma função de uma só via. Porém quando duas mensagens diferentes geram o mesmo resumo, isto é chamado de *colisão*. "As colisões existem, mas ninguém pode encontrar uma colisão por demanda (em alguns algoritmos de resumo, ninguém encontrou nenhuma colisão, nem mesmo acidentalmente)" (BURNETT; PAINE, 2002, p 124). A figura abaixo exemplifica o que seria uma possível colisão em SHA-1.

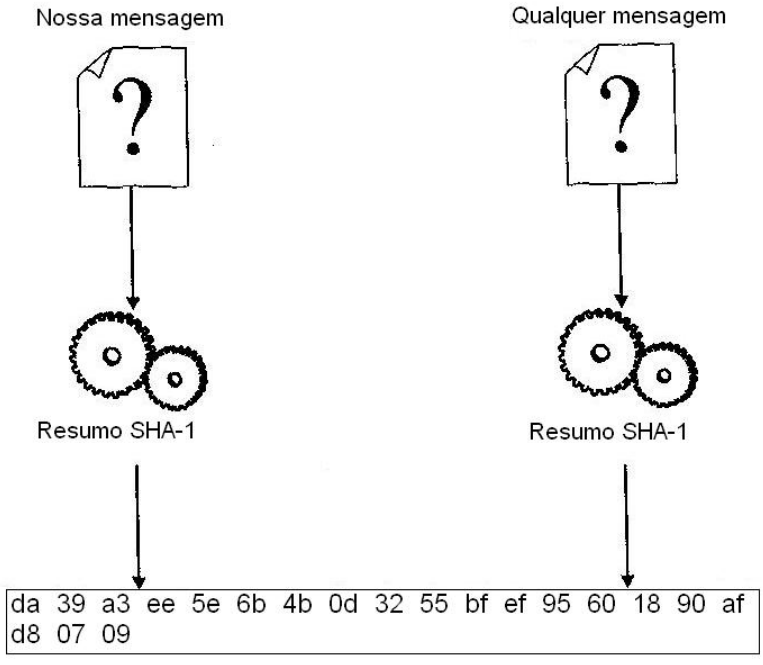

Figura 3.1: Exemplo de uma colisão em SHA-1
Fonte: (BURNETT; PAINE, 2002).

3.1.1.1 PRINCIPAIS ALGORITMOS DE RESUMO

3.1.1.1.1 MD2

Ron Rivest criou o algoritmo de resumo o qual foi chamado de MD *(Message Digest*). Logo, ele quis melhorá-lo e então começou a desenvolver a próxima geração, o MD2. O MD2 produz um resumo de 128 bits, ou seja, é capaz de produzir 2^{128} possibilidades de valores de resumo. Foi amplamente utilizado, mas com o passar dos anos foram encontrados alguns defeitos e descobertas algumas colisões (BURNETT; PAINE, 2002).

3.1.1.1.2 MD5

Quando o MD2 começou a apresentar fraquezas, Rivest, que já estava querendo um algoritmo mais rápido e mais forte, começou a criar novos algoritmos. Primeiro veio o MD3 e logo depois o MD4, mas ambos apresentaram-se um fracasso. O MD5, quinto da série de sumários de Rivest foi o mais bem-sucedido. Mais rápido e mais forte que o MD2 tornou-se dominante. Ainda hoje, apesar de algumas vulnerabilidades terem sido encontradas, continua sendo usado. Algumas pessoas argumentam que para se evitar um colapso total é melhor escolher outro algoritmo (BURNETT; PAINE, 2002).

3.1.1.1.3 SHA

A família de algoritmos SHA (*SecureHash Algorithm*), teve seu início em 1993 com a publicação de seu primeiro membro chamado de SHA, porém costuma ser chamado de SHA-0 para não confundir com seus sucessores. O SHA-0 foi retirado logo após a sua publicação devido a falhas encontradas e não divulgadas que foram corrigidas na versão SHA-1.

O algoritmo SHA-1, desenvolvido pela NSA e aprovado pelo NIST, é parecido com o MD5, Ron Rivest ajudou em seu projeto, porém é mais forte. Ele produz resumos de 160 bits, enquanto o MD5 produz 128 bits, e sobreviveu a uma análise criptográfica, sendo recomendado pela comunidade de criptografia (BURNETT; PAINE, 2002).

Em 2005, o professor Xiaoyun Wang anunciou um ataque diferencial ao hash SHA-1 esperando encontrar colisões. O NIST concordou que uma pessoa com recursos elevados poderia encontrar tais colisões. Em seminário realizado de 31 de outubro a 01 de novembro de 2005, foi revisto as implicações do ataque do professor Wang e, como medida, anunciou-se que era prudente a transição rápida para o “SHA-2” a mais forte da família de funções hash (SHA-224, SHA-256, SHA-384, e SHA-512) para aplicações de assinatura digital e que para

estes, um ataque prático é improvável para os próximos dez anos (BURR, 2006). Abaixo, a tabela com os tamanhos de SHA.

Tabela 3.1: Tamanhos de SHA

Algoritmo	Tamanho de saída (bits)	Tamanho dos blocos (bits)	Comprimento (bits)	Tamanho das palavras (bits)	Varreduras	Operações	Colisão
SHA-0	160	512	64	32	80	+,and,or,xor,rotl	Sim
SHA-1	160	512	64	32	80	+,and,or,xor,rotl	Com falhas
SHA-256/224	256/224	512	64	32	64	+,and,or,xor,shr,rotr	Não
SHA-512/384	512/384	1024	128	64	80	+,and,or,xor,shr,rotr	Não

Fonte: (WIKIPEDIA, 2011)

3.1.1.2 UMA CHAVE DE RESUMO

Foi visto anteriormente que utilizando uma assinatura digital podemos garantir a autenticidade de uma mensagem. Mas há outra forma, sem utilização da assinatura digital, de garantir essa propriedade, baseado na chave compartilhada de criptografia simétrica e utilizando uma chave resumo.

O HMAC (Hash-*based Message Authentication Code*), algoritmo de autenticação de mensagem baseado em hash, é a chave resumo mais conhecida (BURNETT; PAINE, 2002).

Um bom exemplo de como um HMAC trabalha é dado da seguinte forma: um livro razão de um contador possui uma coluna de números, no final da coluna está à soma dos números. Mais tarde, para verificar que o livro razão continua correto, não se compara os números individualmente, primeiro verifica-se se a soma da coluna corresponde com a primeira soma, se estiver correta a verificação está aprovada. Um MAC (*Message Authentication Code*) é uma maneira de detectar alterações nos dados ou na soma. As alterações nos dados podem ser baseadas em um resumo, para verificar as alterações na soma o MAC utiliza uma chave (BURNETT; PAINE, 2002).

O HMAC funciona dessa maneira: ambos, receptor e emissor, possuem a chave simétrica compartilhada de forma segura. O emissor faz um resumo da mensagem e da chave e envia para o receptor junto com a mensagem. Agora se alguém interceptar a mensagem e tentar alterá-la, não conseguirá substituir o resultado da HMAC corretamente. “O resumo depende da mensagem e da chave, dessa forma um invasor teria que saber o que a chave é para alterar a mensagem e anexar uma soma de verificação correta” (BURNETT; PAINE, 2002, p. 128).

Contudo, normalmente as HMACs são utilizadas apenas para verificar a integridade das mensagens durante o seu trânsito e não para servir como assinatura como visto neste exemplo, porque como emissor e receptor tem conhecimento da chave compartilhada uma terceira pessoa não teria como saber quem a assinou (BURNETT; PAINE, 2002). A figura abaixo ilustra a concatenação de uma mensagem mais a chave secreta através do mecanismo de HMAC.

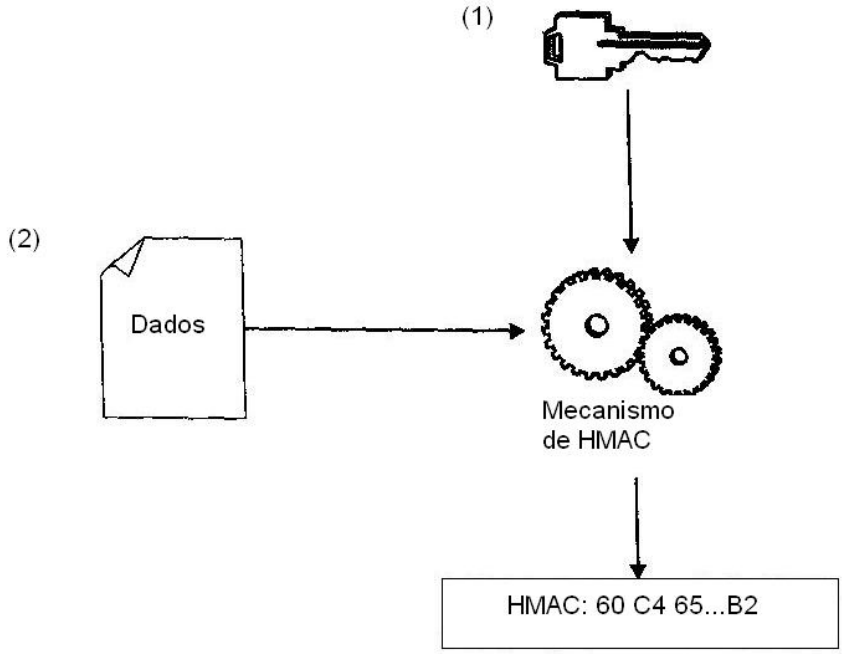

Figura 3.2: O algoritmo de HMAC resume a chave e os dados (nessa ordem) para produzir um valor

Fonte: (BURNETT; PAINE, 2002).

4 CERTIFICADOS DIGITAIS

Este capítulo visa apresentar os Certificados Digitais, uma maneira confiável de ligar uma entidade a uma chave pública, garantindo a sua autenticidade através da assinatura de uma *Autoridade Certificadora* (AC).

A criptografia de chave pública torna possível a comunicação segura para pessoas que não compartilham uma chave simétrica. Com ela também é possível realizar a assinatura de uma mensagem para garantir a autenticidade e, por fim, utilizando resumos de mensagem assinados permite verificar a integridade da mensagem (TANENBAUM, 2003).

Mas há um problema com a distribuição de chaves públicas. Em um cenário onde duas pessoas precisam se comunicar e não possuem uma chave comum previamente estabelecida, precisam de uma chave pública para que a comunicação possa ser estabelecida de forma segura. Neste cenário o receptor (R) possui em seu web site a sua chave pública disponível. O emissor (E) então pode fazer uso desta chave para criptografar a mensagem ou até mesmo criptografar outra chave simétrica e a partir desse momento toda a comunicação passa a utilizar esta chave para criptografar e descritografar as mensagens que serão trocadas. Porém um invasor (I) intercepta a comunicação no momento que E estava baixando a chave publica de R e a substitui por uma chave sua. Agora I poderá ler ou pior, modificar as mensagens enviadas entre E e R sem que eles saibam disso (TANEMBAUM, 2003).

Para resolver este problema, foi adotado um método para que seja possível certificar se uma chave pública pertence de fato a uma pessoa ou entidade de destino. Assim surgiram as organizações que certificam as chaves públicas chamadas de CA (*Certification Authority – Autoridade de Certificação*) (TANEMBAUM, 2003).

Para Burnett e Paine (2002, p.141): “A maneira mais comum de se saber se uma chave pública pertence ou não a uma entidade de destino é por meio de um certificado digital. Um *certificado digital* associa um nome a uma chave pública”.

Dessa forma, E pode ir até uma CA com sua chave pública e um documento que o identifique e solicitar a certificação. A CA combina o nome de E com a chave pública de E em uma mensagem e a assina utilizando a sua chave privada emitindo então um certificado (TANENBAUM, 2003). A tabela a seguir ilustra um certificado.

Tabela 4.1: Um certificado possível e seu hash assinado

I hebery certify that the public key 1 9836A8B030300F83737E3887837FC3587O92827262643FFA8271 03828282A

belongs to Robert John Smith 12345 University Avenue Berkeley, CA 94702 Birthday: July 4, 1958 Email: bob@superdupernet.com
SHA-1 hash of the above certificate signed with the CA's private key

Fonte: (TANENBAUM, 2003).

Agora, quando E estiver com a chave pública de R, ele terá condições de checar se esta é realmente de R, utilizando a chave pública da CA para verificar se o certificado é válido. Caso o certificado não seja validado E não deve utilizar esta chave pública (BURNETT; PAINE, 2002).

4.1 PADRÃO X.509

Segundo Burnett e Paine (2002), em um certificado de chave pública, os nomes de entidades, de emissor e sujeito precisam ser únicos. Para realizar a padronização de nomes foi adotado o padrão X.509.

A hierarquia do X.509 é baseada em uma árvore de cima para baixo, com o certificado raiz no topo, que representa a CA principal e os nós finais representam as entidades (BURNETT; PAINE, 2002). A tabela abaixo exibe um certificado X.509 com seus campos básicos.

Tabela 4.2: Campos básicos de um certificado X.509

Campo	Significado
Version	A versão do X.509
Serial Number	Este número, somado ao nome da CA, identifica de forma
Signature algorithm	O algoritmo utilizado para assinar o certificado
Issuer	Nome X.500 da CA
Validity period	A hora inicial e final do período de validade
Subject name	A entidade cuja chave está sendo sertificada
Public Key	A chave pública do assunto e a ID do algoritmo que a utiliza

Issuer ID	Um ID opcional que identifica de forma exclusiva o emissor do
Subject ID	Um ID que identifica de forma exclusiva o protagonista do
Extensions	Muitas extensões foram definidas
Signature	A assinatura do certificado (assinado pela chave privada da CA)

Fonte: (TANENBAUM, 2003).

4.2 INFRA-ESTRUTURA DE CHAVE PÚBLICA

Como não seria viável ter somente uma CA para emitir todos os certificados do mundo, foi desenvolvido um modo diferente para certificar chaves públicas chamada de PKI (*Public Key Infrastructure – Infra-Estrutura de Chave Pública*). Uma PKI possui vários componentes, como CAs, certificados, diretórios e usuários finais, e fornece um modo para estruturá-los, definindo padrões para os vários documentos e protocolos (TANENBAUM, 2003). A figura abaixo mostra a forma hierárquica de uma PKI.

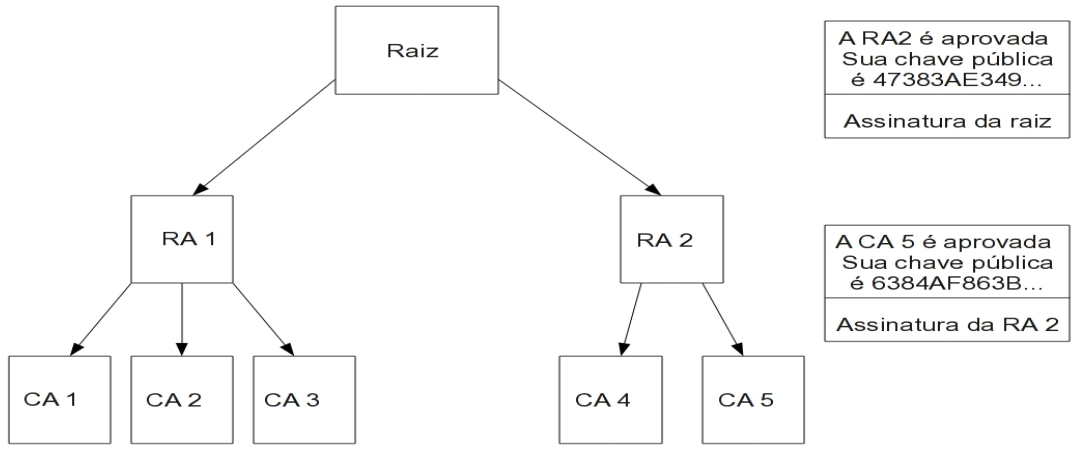

Figura 4.1: Uma PKI hierárquica e uma cadeia de certificados
Fonte: (TANENBAUM, 2003)

Segundo Tanenbaum (2003), a CA de nível superior (raiz) certifica as CAs do segundo nível, RAs (*Regional Authorities*), por sua vez certificam as CAs reais, que emitem os certificados X.509 para as organizações e indivíduos. Assim, quando há a necessidade de verificação de validade de algum certificado é necessário realizar a consulta no nó superior e

assim por diante. “Uma cadeia de certificados como essa que volta à raiz, às vezes é chamada **cadeia de confiança** ou **caminho da certificação**. A técnica é amplamente utilizada” (TANENBAUM, 2003, p. 819).

4.3 A INFRA-ESTRUTURA DE CHAVE PÚBLICA NO BRASIL

No Brasil a *Infra-Estrutura de Chaves Públicas Brasileira* (ICP-Brasil) é mantida e auditada pelo *Instituto Nacional de Tecnologia da Informação* (ITI), uma autarquia federal vinculada à casa civil, sendo a primeira autoridade da cadeia de certificação - AC Raiz. (ITI, 2011).

O sistema nacional de certificação digital da ICP-Brasil teve o início de sua implantação com base na Medida Provisória 2.200-2 de 24 de agosto de 2001, que instituiu o ICP-Brasil para garantir a autenticidade, a integridade e a validade jurídica de documentos em forma eletrônica, das aplicações de suporte e das aplicações habilitadas que utilizem certificados digitais, bem como a realização de transações eletrônicas seguras (ITIb, 2001).

Nesta *Medida Provisória* (MP) também discorre que o ITI segue as regras de funcionamento estabelecidas por um Comitê Gestor da ICP-Brasil, cujos membros são nomeados pelo Presidente da República, entre representantes dos poderes da República, segmentos da sociedade e da academia, para dar estabilidade, transparência e confiabilidade ao sistema (ITI, 2011).

Ainda de acordo com a Medida Provisória 2.200-2 de 24 de agosto de 2001, cabe a AC Raiz, emitir, expedir, distribuir, revogar e gerenciar os certificados das ACs de nível imediatamente subseqüente ao seu, porém, são as ACs a quem competem emitir, expedir, distribuir, revogar e gerenciar os certificados, vinculando pares de chaves criptográficas ao respectivo titular, bem como disponibilizar aos usuários lista de certificados revogados e outras informações pertinentes e manter registro de suas operações (ITIb, 2001). A imagem a seguir ilustra a estrutura resumida da ICP-Brasil.

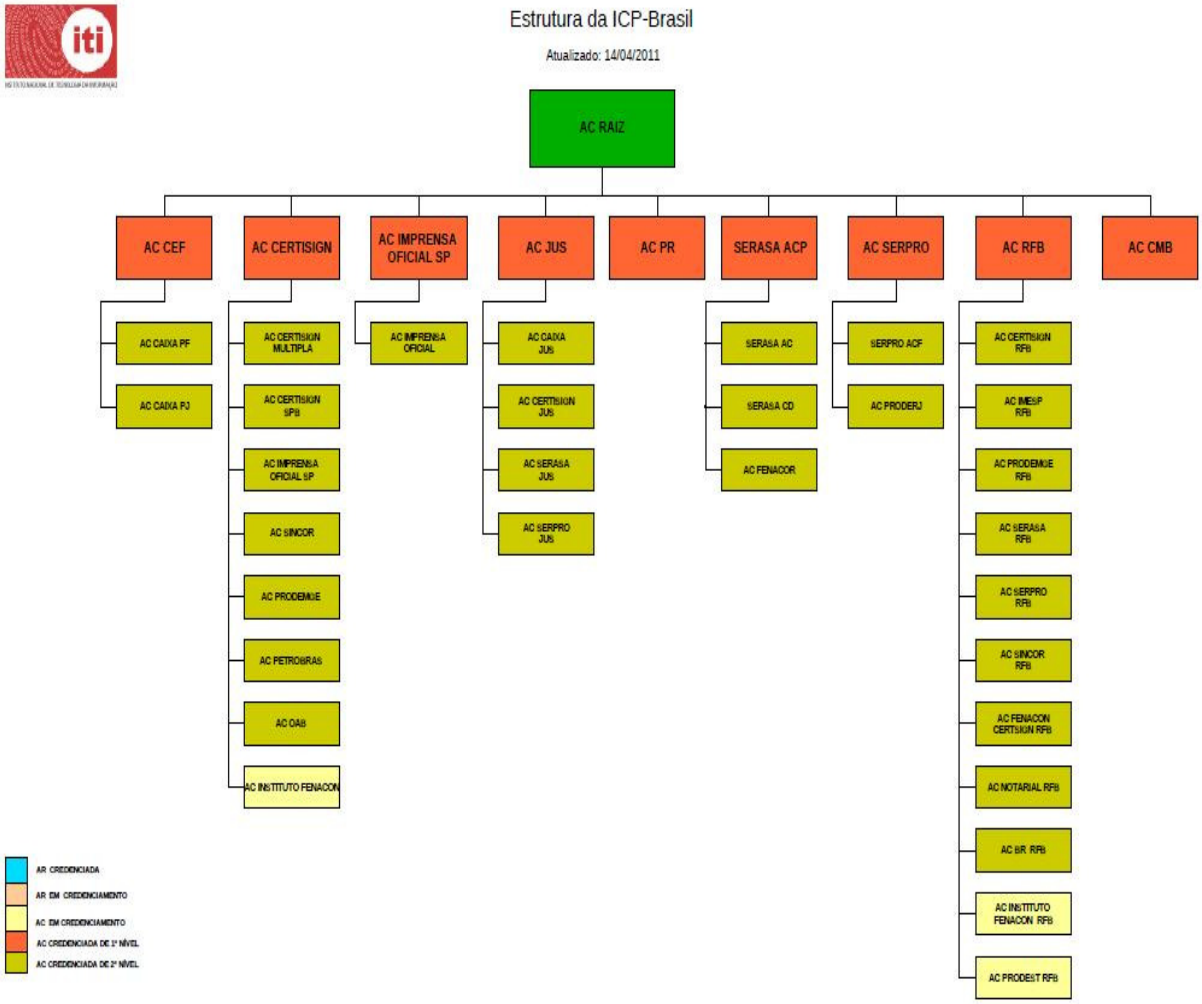

Figura 4.2: Estrutura resumida da ICP-Brasil
Fonte: (ITI, 2011).

5 PROTOCOLOS CRIPTOGRÁFICOS

Este capítulo visa apresentar os protocolos criptográficos, bem como, seu funcionamento, onde são combinados mecanismos de criptografia, requisitos necessários para transações seguras pela Internet. Será visto como são utilizadas as principais ferramentas, abordadas nos capítulos anteriores, para proporcionar segurança em transações pela internet.

Em aplicações Web baseadas no protocolo HTTP, com é o caso do SIAP, o SSL/TLS é a tecnologia mais usada e mais importante para prover segurança em informações transmitidas entre cliente e servidor.

5.1 SECURE SOCKET LAYER (SSL) E TRANSPORT LAYER SECURITY (TLS)

O SSL (*Secure Socket Layer*) e o TLS (*Transport Layer Security*) são protocolos criptográficos desenvolvidos para este fim, proteger comunicações que necessitam de confidencialidade, integridade e, ainda, possibilitam garantir a autenticidade e o não-repúdio das informações. Ambos, suportados pela maioria dos navegadores, são amplamente utilizados no comércio eletrônico e por instituições financeiras, onde o nível de segurança exigido é alto (COULOURIS; DOLLIMORE; KINDBERG, 2007).

5.1.1 A HISTÓRIA DO SSL/TLS

Logo que a web chegou ao público, viu-se uma enorme potencialidade para a realização de comércio eletrônico, transações bancárias *on-line*, permitir acesso a informações privilegiadas, porém, todas essas ações necessitavam de segurança (ARAÚJO, 2011).

Em 1994 a Netscape, que até então dominava o mercado de navegadores, desenvolveu o *Secure Sockets Layer* (SSL), um protocolo de Internet baseado em sessão, capaz de fornecer um canal de dados seguro entre cliente e servidor (BURNETT; PAINE, 2002). Desde então, o SSL tornou-se um padrão, sendo implementado pelos navegadores mais populares (Internet Explorer, Firefox, Opera, Safari, Chrome, etc) e também os servidores web mais populares (Apache, IIS, dentre outros).

A primeira versão do SSL, SSLv1.0 foi usado somente dentro da Netscape. A versão 2.0 do SSL foi fornecido nas versões 1 e 2 do Netscape Navigator. Após a publicação da

SSLv2.0 a Microsoft criou um canal seguro de dados, semelhante ao SSL, chamado de *Private Communication Technology* (PCT). O PCT superou algumas deficiências do SSLv2.0. Em 1995 foi lançada a versão 3.0 do SSL onde foram introduzidos os avanços do PCT (GARFINKEL; SPAFFORD, 2002).

Em 1996 a Netscape submeteu o SSL à Internet *Enginnering Task Force* (IETF) para sua padronização. O resultado desta padronização foi o *Transport Layer Security* (TLS), que teve a primeira versão oficial distribuída em 1999. Essa primeira versão do TLS 1.0, também é conhecida como SSLv3.1 (TANENBAUN, 2003).

Após a versão 1.0 do TLS, a IETF continuou o seu trabalho e lançou uma nova versão em 2006, denominada TLS 1.1 e especificada na RFC (*Request for Comments*) 4346, onde foram corrigidos alguns problemas de criptografia. Após dois anos de revisão, em 2008, foi lançada a versão TLS 1.2, especificada na RFC 5246. A versão 1.2 do TLS teve como processo mais significativo à incorporação de algumas extensões (OPPLIGER, 2009).

Estruturalmente o TLS é idêntico ao SSL. Segundo Oppliger (2009), a principal diferença entre esses protocolos está relacionada na forma como eles geram o material de codificação.

5.1.2 OS SUBPROTOCOLOS DE SSL E TLS

Quando o HTTP é usado sobre SSL/TLS, ele é denominado HTTPS (*Secure HTTP*), sua porta padrão passa a ser a 443, enquanto com o HTTP a porta padrão é a 80 (TANENBAUN, 2003). A figura a seguir mostra o posicionamento do protocolo SSL/TLS na pilha de protocolos TCP/IP.

Tabela 5.1: Posicionamento do SSL/TLS na pilha de protocolos TCP/IP

SMTP	HTTP	FTP
SSL/TLS		
TCP		
IP		

Fonte: (BURNETT; PAINE, 2002)

Os protocolos SSL e TLS vistos mais de perto, são compostos por duas sub-camadas e alguns sub-protocolos. A tabela a seguir ilustra os componentes dos protocolos SSL/TLS.

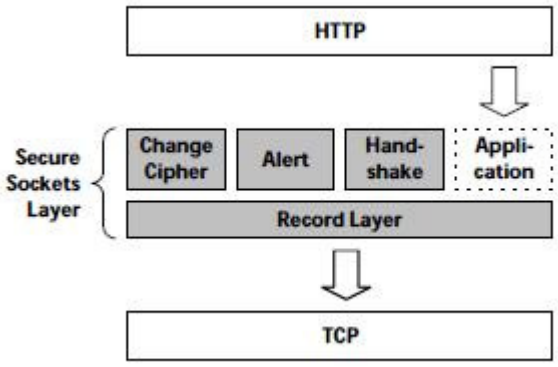

Figura 5.1: Componentes do protocolo SSL/TLS
Fonte: (THOMAS, 2000).

5.1.2.1 PROTOCOLO DE HANDSHAKE

O protocolo de handshake é responsável por estabelecer os parâmetros do estado de sessão atual. Ele permite cliente e servidor autenticar-se um ao outro, negociar algoritmos criptográficos e métodos de compressão por meio de troca de mensagens (OPPLIGER, 2009). Essas mensagens serão explicadas em detalhes no subtítulo Estabelecendo Comunicações Criptogradas.

5.1.2.2 PROTOCOLO DE ESPECIFICAÇÃO DE ALTERAÇÃO DE CIFRAGEM

Este protocolo consiste em uma única mensagem, *ChangeCipherSpec*, enviada pelo cliente e o servidor antes de terminar o protocolo de handshake, informando um ao outro que a partir deste momento os dados enviados estarão protegidos com as opções de cifragens e chaves recentemente negociadas (OPPLIGER, 2009).

5.1.2.3 PROTOCOLO DE ALERTA

Este protocolo transporta as mensagens de alerta e sua severidade para as partes em uma conexão SSL. Quando umas das partes detectarem o erro, esta envia uma mensagem para a outra parte, se a mensagem contiver um resultado fatal, ambas as partes fecham a conexão sem manter nenhuma informação quanto à conexão atual. Para erros não-fatais as partes

podem armazenar em cache as informações de resumo da conexão (BURNETT; PAINE, 2002). A tabela a seguir mostra os alertas do protocolo SSL, bem como, os alertas do protocolo TLS que adicionou novas mensagens e eliminou uma.

Tabela 5.2: Alertas do protocolo SSL/TLS

Valor	Nome	Significado
0	CloseNotify	Esta mensagem indica explicitamente que a conexão será fechada
10	UnexpectedMessage	Indica que a mensagem imprópria foi recebida; este alerta é fatal.
20	BadRecordMac	Indica que a mensagem recebida está com o código de autenticação de mensagem incorreto; este alerta é fatal.
[2] 21	DecryptionFailed	Indica que a mensagem descriptografada era inválida; este alerta é fatal.
[2] 22	RecordOverflow	Indica que a mensagem recebida era, depois de descritografa ou descompressada , maior que 2^{14} + 2048 bytes; esta mensagem é fatal.
30	DecompressionFailure	Este erro indica que os dados recebidos não puderam ser descoprimidos; este alerta é fatal.
40	HandShakeFailure	O retorno desta mensagem indica que o remetente não foi capaz de negociar um aceitável conjunto de parâmetros de segurança para esta sessão; este alerta é fatal.
[1] 41	NoCertificate	Esta mensagem indica que uma solicitação de certificado retornou sem um certificado apropriado.
42	BadCertificate	Indica que o certificado recebido está corrompido.
43	UnsupportedCertificate	Indica que o tipo de certificado recebido não é suportado.
44	CertificateRevoked	Indica que o certificado recebido foi revogado por uma Autoridade Certificadora.
45	CertificateExpired	Indica que o certificado recebido está expirado.
46	CertificateUnknown	Indica que surgiu um problema não especificado com o certificado recebido.
47	IllegalParameter	Indica que um campo do handshake estava com um parâmetro ilegal ou inconsistente com outro parâmetro.
[2] 48	UnknownCA	Indica que não pode identificar ou confiar na Autoridade

		Certificadora de uma cadeia de certificado recebido; esta mensagem é fatal.
[2] 49	AccessDenied	Indica que a parte identificada com o certificado não tem direito de acesso para continuar a negociação; este erro é fatal.
[2] 50	DecodeError	Indica que a mensagem recebida não pode ser decodificada porque o valor de um campo está fora do range permitido ou o tamanho da mensagem é inválido; este erro é fatal.
[2] 51	DecryptError	Indica que as operações criptográficas essenciais negociadas no Handshake falharam.
[2] 60	ExportRestriction	Indica que foi detectado um parâmetro de negociação não conforme com as restrições de exportação; este erro é fatal.
[2] 70	ProtocolVersion	Indica que não há suporte a versão TLS que o cliente tentou negociar não é suportada; este erro é fatal.
[2] 71	InsufficientSecurity	Indicada sempre pelo servidor informa que requer algoritmos de criptografia mais seguras que as informadas pelo cliente; esta mensagem é fatal.
[2] 80	InternalErros	Indica que um erro local a operação e independente do protocolo TLS impossibilita a continuação; este erro é fatal.
[2] 90	UserCanceled	Indica que o Handshake está sendo cancelado por algum motivo não relacionado a uma falha de protocolo. Este alerta deve ser seguido por um closeNotify. Este alerta geralmente é um aviso.
[2] 100	NoRenegotiation	Indica que a renegociação não é apropriada. Este alerta é um aviso.

Fonte: (THOMAS, 2000).

[1] Mensagem de alerta presente somente no protocolo SSL.
[2] Mensagens de alerta presentes somente no protocolo TLS.

5.1.2.4 PROTOCOLO DE REGISTRO (RECORD PROTOCOL*)*

É na camada de registro onde os dados recebidos das camadas superiores são encapsulados para transmissão a camada inferior TCP. A figura a seguir mostra uma visão geral da camada de registro com as etapas onde os dados são compactados, autenticados e encriptados.

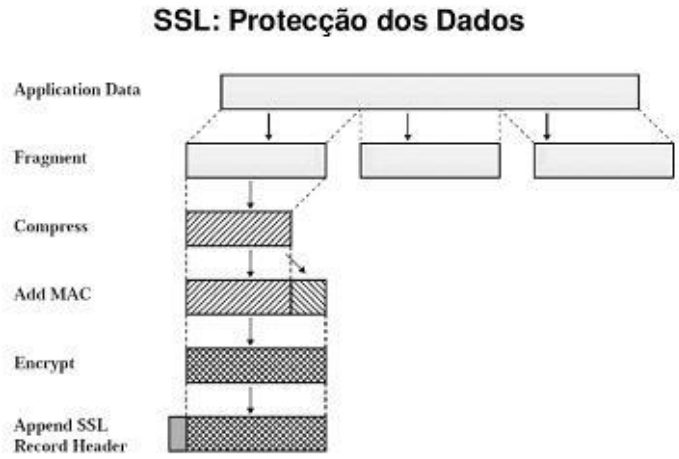

Figura 5.2: Visão geral da camada de registro SSL
Fonte: (BURNETT; PAINE, 2002).

Segundo Oppliger (2009) e Thomas (2000), estas são as etapas comentadas da camada de registro:

Na primeira etapa da camada de registro, os dados são fragmentados em pedaços gerenciáveis para então serem processados individualmente.

Na segunda etapa é onde os dados são comprimidos. Os algoritmos de compactação são definidos entre o cliente e o servidor. Por padrão, este método é opcional e vem definido como nulo. Na prática a compressão raramente é utilizada.

Na terceira etapa, um MAC (*Message Authentication Checksun*) é computado para cada registro. O MAC computado é a concatenação do resumo gerado pela chave previamente estabelecida e o resumo dos dados do registro. Aqui o SSL usa um algoritmo diferente do TLS para calcular o MAC. O algoritmo usado pelo SSL é um predecessor do HMAC, utilizado pelo TLS. Em termos de segurança o MAC SSL é comparável ao HMAC.

É na quarta etapa, onde os dados e seus MACs associados são criptografados utilizando um algoritmo de criptografia simétrica previamente acordado.

Na quinta e última etapa, mas não menos importante, é anexado um cabeçalho de registro SSL na estrutura de texto cifrado na etapa anterior. O cabeçalho de registro SSL é composto pelos seguintes campos:

- Content type (Tipo de conteúdo): referencia o protocolo SSL da camada superior utilizado para processar o registro;
- Version (Versão): este campo referencia a maior e menor versão do SSL em uso;
- Length (Tamanho): este campo indica em bytes o comprimento total do registro de texto simples.

A figura abaixo ilustra um esboço de um registro SSL.

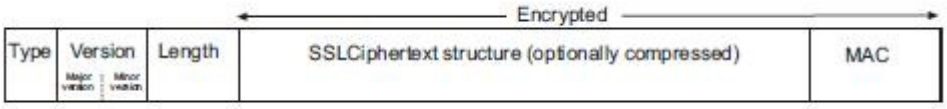

Figura 5.3: Esboço de um registro SSL
Fonte: (OPPLIGER, 2009).

Segundo Burnett e Paine (2002), a parte receptora inverte o processo realizado pela parte emissora, ou seja, convertendo os dados criptografados para o seu estado legível invertendo o processo aqui descrito.

5.1.3 ESTABELECENDO COMUNICAÇÕES CRIPTOGRAFADAS

O protocolo Handshake é o principal responsável por uma negociação de sessão SSL/TLS. A figura abaixo ilustra todas as mensagens trocadas pelo cliente e o servidor a fim de estabelecer uma conexão segura.

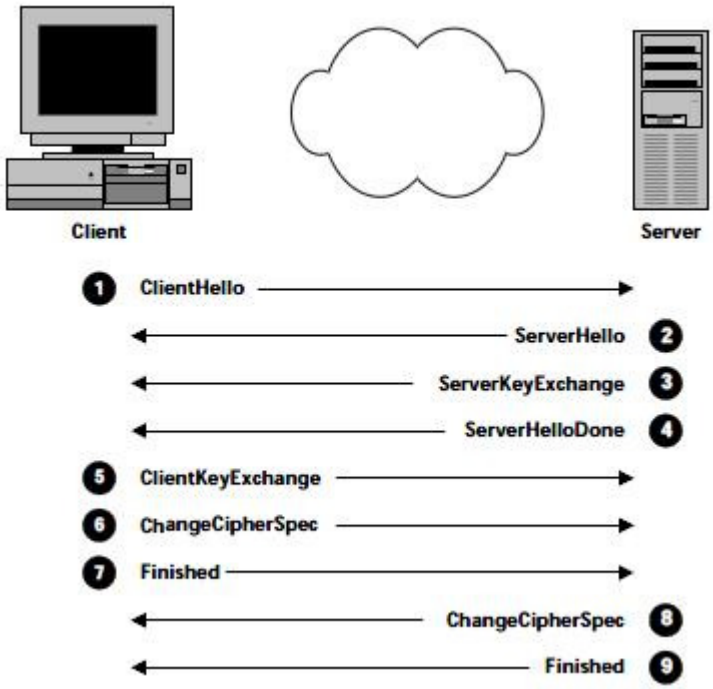

Figura 5.4: SSL usa nove mensagens para estabelecer comunicações criptografadas.
Fonte: (THOMAS, 2000).

As mensagens trocadas entre cliente e servidor são agora explicadas de uma forma detalhada, segundo Thomas (2000):

5.1.3.1 MENSAGEM HELLO DE CLIENTE

ClientHello Message é a primeira mensagem, enviada pelo cliente ao servidor, para iniciar uma comunicação. O conteúdo dessa mensagem contém os seguintes campos:

- Client_version: corresponde à versão de SSL mais alta suportada pelo cliente;
- Random: uma estrutura aleatória gerada pelo cliente, de 32 bytes, onde 4 bytes indicam data/hora e os 28 bytes restantes são gerados aleatoriamente;
- Session_id: este campo contém identificador de sessão. Se estiver vazio, novos parâmetros de segurança são gerados. Se o identificador de sessão estiver presente, ele contém um valor para identificar uma sessão previamente estabelecida, onde os parâmetros de segurança já foram acordados e o cliente deseja reutilizar;
- Cipher_suites: lista de algoritmos criptográficos suportados pelo cliente. Com base nesta lista, o servidor decidirá qual será usado. Se o servidor não escolher nenhum algoritmo da lista, um alerta de falha de handshake é gerado e a conexão é fechada;

- Compression_methodos: este campo é semelhante ao chipher_suites. O cliente envia uma lista de métodos de compactação por ele suportados. O servidor decidirá qual método será utilizado.

Depois de enviado a mensagem clientHello, o cliente espera por uma mensagem serverHello.

5.1.3.2 MENSAGEM HELLO DE SERVIDOR

Ao processar a mensagem *clientHello* e não ocorrer nenhum problema, o servidor retorna uma mensagem *serverHello* com os seguintes campos:

- Server_version: este campo contém a versão do protocolo SSL, escolhida pelo servidor, baseada na versão mais alta suportada por ambos;
- Random: estrutura aleatória semelhante à estrutura gerada pelo cliente, porém, totalmente independente;
- Session_id: este campo corresponde à identidade da sessão à conexão atual. Se este é o mesmo session_id enviado pelo cliente é realizado um handshake abreviado. Caso contrário este campo possui um valor diferente que indica uma nova sessão;
- Cipher_suite: este campo indica o algoritmo de criptografia selecionado pelo servidor, com base na lista fornecida pelo cliente;
- Compression_method: este campo indica o método de compactação selecionado pelo servidor, com base na lista fornecida pelo cliente.

5.1.3.3 CERTIFICADO

Logo após enviar a mensagem de *serverHello,* o servidor então, envia o seu certificado ou cadeia de certificados onde consta a sua identificação. Este certificado, quando assinado por uma *Autoridade Certificadora* (CA), garante ao cliente a capacidade de atestar a autenticidade deste servidor antes de prosseguir. Nesta mensagem, o servidor também disponibiliza ao cliente sua chave pública.

5.1.3.4 MENSAGEM DE TROCA DE CHAVES DO SERVIDOR

Em casos onde o certificado é usado apenas como assinatura, ou seja, a assinatura e a criptografia estão separadas, o servidor envia uma mensagem *ServerKeyExchange*, que contém a sua chave pública que será utilizada pelo cliente.

5.1.3.5 MENSAGEM HELLO DE CONCLUSÃO DO SERVIDOR

A mensagem *ServerHelloDone* informa ao cliente que o servidor finalizou a sua negociação inicial. Agora o servidor espera pela resposta do cliente.

5.1.3.6 MENSAGEM DE TROCA DE CHAVES DO CLIENTE

Esta mensagem permite que o cliente envie as informações sobre a chave ao servidor. Essas informações são necessárias para gerar a chave mestre, que será usada por um algoritmo de chave simétrica que ambas as partes utilizarão. A mensagem *ClientKeyExchange* é cifrada utilizando a chave pública do servidor antes de ser enviada pela rede.

5.1.3.7 MENSAGEM DE TROCA DE ESPECIFICAÇÃO DE CIFRAGEM

A mensagem ChangeCipherSpec serve para sinalizar uma mudança de estratégia de criptografia.

5.1.3.8 MENSAGEM DE FINALIZAÇÃO

Imediatamente após enviar a mensagem *ChangeCipherSpec*, cada lado envia uma mensagem de finalização, *Finished.* Esta mensagem permite que ambos os lados verifiquem que a negociação foi bem sucedida e a segurança não está comprometida.

Agora com a conexão segura estabelecida, os dados passam a atuar na camada de registro conforme explicado anteriormente.

5.1.4 COMPUTAÇÕES CRIPTOGRÁFICAS

Durante a execução do protocolo de Handshake, cliente e servidor trocam informações para gerar a chave criptográfica que será utilizada em conjunto com um algoritmo de chave simétrica para criptografar e descriptografar os dados que serão transmitidos durante a conexão.

O *segredo mestre* como é conhecido, é derivado do segredo pré-mestre enviado pelo cliente ao servidor através da mensagem *ClientKeyExchange,* criptografada com a chave pública do servidor.

Com ambos os lados conhecendo o segredo pré-mestre, mais os valores aleatórios gerados por ambos e combinadas as saídas de hash nas formas prescritas, cliente e servidor terão o mesmo segredo mestre (THOMAS, 2009). A imagem a seguir mostra os detalhes deste processo.

```
master_secret =
   MD5(pre_master_secret + SHA('A' + pre_master_secret
       + ClientHello.random + ServerHello.random)) +
   MD5(pre_master_secret + SHA('BB' + pre_master_secret
       + ClientHello.random + ServerHello.random)) +
   MD5(pre_master_secret + SHA('CCC' + pre_master_secret
       + ClientHello.random + ServerHello.random))
```

Figura 5.5: Gerando um segredo mestre
Fonte: (THOMAS, 2009)

5.1.5 FERRAMENTA PARA IMPLEMENTAÇÃO DOS PROTOCOLOS SSL/TLS

OpenSSL é um projeto de esforço colaborativo a fim de desenvolver um kit de ferramentas de código aberto, robusto e cheio de recursos, para implementação dos protocolos criptográficos SSL/TLS (OPENSSL, 2011).

O OpenSSL é baseado na biblioteca SSLeay, desenvolvida por Eric A. Young e Tim J. Hudson e está sob a licença estilo Apache, podendo ser usado para fins comerciais e não-comerciais (OPENSSL, 2011).

O OpenSSL é uma ferramenta de linha de comando que pode ser usado para as seguintes operações (OPENSSL, 2011):

- Criação e gestão de chaves privadas, chaves públicas e parâmetros;
- Criação de certificados X.509, CSRs e CRLs;
- Cálculo de Mensagens Digests;
- Criptografia e descriptografia com cifras;

- Testes SSL/TLS Cliente e Servidor;
- Tratamento de S/MIME assinado ou mail criptografado;
- Time Stamp pedido, geração e verificação.

Segundo Veiga, Messier e Chandra (2002), o OpenSSL é uma biblioteca de código aberto amplamente utilizada para implementação dos protocolos SSL e TLS, sendo ainda multiplataforma, trabalhando igualmente com Unix e Windows.

6 TESTES EM AMBIENTE CONTROLADO

Neste capítulo será detalhado como o ambiente para testes foi criado, bem como, as configurações realizadas para que as conexões encriptadas sejam mantidas e então com base nestes testes, as configurações sejam efetuadas no servidor da UFCSPA.

O ambiente controlado é uma forma de expor as vulnerabilidades existentes quando se utiliza uma conexão não encriptada para transmitir informações confidencias sem a necessidade de aplicá-los diretamente no sistema da UFCSPA, porém, os sistemas precisam estar configurados da forma mais semelhante possível e utilizando as mesmas tecnologias de desenvolvimento. Dessa forma, será possível demonstrar de um modo prático, os riscos que se está exposto.

Para simular o ambiente, foi desenvolvido um sistema similar ao SIAP que será chamado de SITS (*Sistema de Imagens para Testes de Segurança*). Serão utilizados três computadores: o Servidor onde o sistema ficará hospedado, um notebook para realizar a comunicação com o Servidor e visualizar as imagens e outro notebook que irá monitorar a comunicação, capturar o tráfego transmitido durante a comunicação, tendo como objetivo obter os dados sigilosos que ali trafegaram.

Inicialmente, a ideia principal era capturar as credenciais utilizadas para acesso ao sistema, como usuário e senha, para mais tarde então realizar o acesso utilizando estas credenciais. Porém, com o desenvolvimento da pesquisa percebeu-se que é possível obter todas as informações necessárias, com o uso de ferramentas que extraem arquivos do tráfego de rede, sem a necessidade de acesso ao sistema e sem deixar rastros, tornando um ataque silencioso, com base em seus cabeçalhos e rodapés. “O processo de recuperação de arquivos de mídia digital pela localização de assinatura de arquivo (cabeçalho, rodapé) e extração de dados entre esses pontos finais é conhecido como file carving” (POVAR, BHADRAN, 2010, p. 137).

Segundo Povar e Bhadran, data carving ou file carving é um termo usado no campo da forense Cibernética. “Forense cibernética é o processo de aquisição, autenticação, análise e documentação de evidências extraídas e/ou contidas em um sistema de computador, rede de computador e mídias digitais” (POVAR; BHADRAN, 2010, p. 137).

Buscando um ambiente mais próximo possível do SIAP, foram utilizados os programas abaixo relacionados com uma breve descrição de suas funcionalidades neste teste de segurança.

- SITS – Sistema WEB desenvolvido em PHP para visualização de imagens e demonstração de vulnerabilidades em transferências de dados;
- Debian Squeeze 6.0.1 – Sistema operacional Linux utilizado como plataforma operacional do Servidor onde rodará o SITS. Também será utilizado como plataforma operacional do notebook que irá monitorar a conexão cliente/servidor;
- Microsoft Windows XP Professional – Sistema operacional utilizado no cliente que irá realizar acesso ao SITS para visualização das imagens;
- Servidor Web Apache – Servidor web utilizado para disponibilizar páginas na web;
- PHP – Linguagem de programação para o desenvolvimento de páginas web dinâmicas (PHP, 2011). Utilizada no desenvolvimento do *SIAP* e *SITS*.
- Mysql – Serviço de Banco de Dados relacional utilizado o armazenamento das imagens (MYSQL, 2011);
- Wireshark – Programa utilizado para análise e captura do tráfego de rede (WIRESHARK, 2011);
- Foremost – Ferramenta para recuperar arquivos, baseados em seus cabeçalhos, rodapés e estruturas internas de dados (FOREMOST, 2011).
- Tcpxtract - Ferramenta para extrair arquivos do tráfego da rede baseado na assinatura dos arquivos (TCPXTRACT, 2011).

A figura a seguir, ilustra o ambiente de testes onde será simulada a conexão cliente/servidor, bem como, o computador equipado com ferramentas para análise e captura do tráfego.

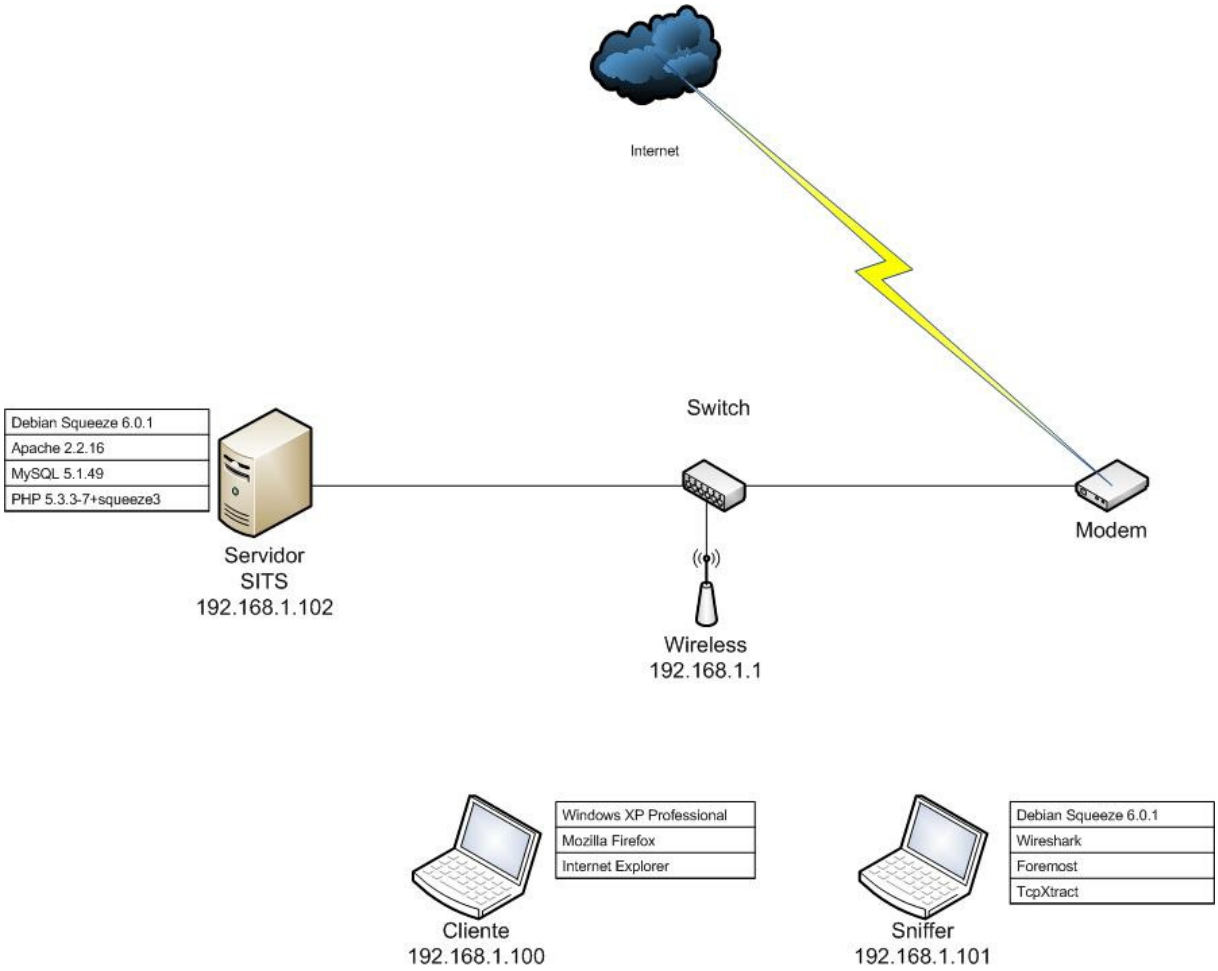

Figura 6.1: Estrutura de rede do ambiente de testes
Fonte: do Autor

Esta é a estrutura de rede onde os testes serão realizados, muito semelhante a uma rede doméstica, porém, a propagação do sinal de internet via ondas de rádio dá-se da mesma forma, muito comum nos dias de hoje, como em lugares públicos e instituições de ensino que raramente fazem uso de criptografia como mecanismo de segurança.

Segundo Kurose e Ross (2006), a segurança é um fator de muita preocupação nas redes sem fio, pois as ondas de rádio portando quadros se propagam em todas as direções podendo alcançar até centenas de metros. Dessa forma, se o cliente estiver realizando o acesso ao sistema em uma dessas redes, terá o mesmo grau de risco e facilidade que o monitoramento e captura de dados aqui demonstrados.

A segurança de um sistema é tão forte quanto seu elo mais fraco. Por si só, o problema descrito acima que ocorre nas redes wireless justificaria o uso de criptografia. Porém, para melhor fundamentar a proposta, pode-se citar alguns problemas que também ocorrem em redes cabeadas, como em redes que utilizam hubs como concentrador. Com ele os dados são propagados para todos os computadores que a ele estiverem conectados. Assim,

uma placa de rede no modo promíscuo[2] poderá capturar todo o tráfego da rede igualmente a uma rede wireless. Para resolver este problema pode-se usar um switch. Um switch é um comutador que diferentemente do hub segmenta a rede e cada porta corresponde a um domínio de colisão. Ele mantém em uma tabela interna o endereço MAC (*Media Access Control*) das estações relacionado à porta onde a estação está conectada, assim os pacotes são transmitidos diretamente ao seu destino sem que as demais estações tenham conhecimento.

Porém, mesmo redes que utilizam switchs estão suscetíveis a ataques onde uma estação pode capturar o tráfego das demais, burlando assim a sua a capacidade de criar diferentes domínios de colisão. Um exemplo de técnica deste tipo é o ARP (*Address Resolution Protocol*) Poisoning, um meio eficiente para executar o ataque conhecido por *Man-In-The-Middle* (MITM). Este ataque permite ao invasor visualizar toda a comunicação entre o computador alvo e o computador de destino. A figura a seguir mostra um ambiente onde esse tipo de ataque pode ser aplicado.

[2] Modo promíscuo é quando uma placa de rede capta informações destinadas a qualquer computador conectado no mesmo segmento de rede e não apenas pacotes endereçados ao próprio.

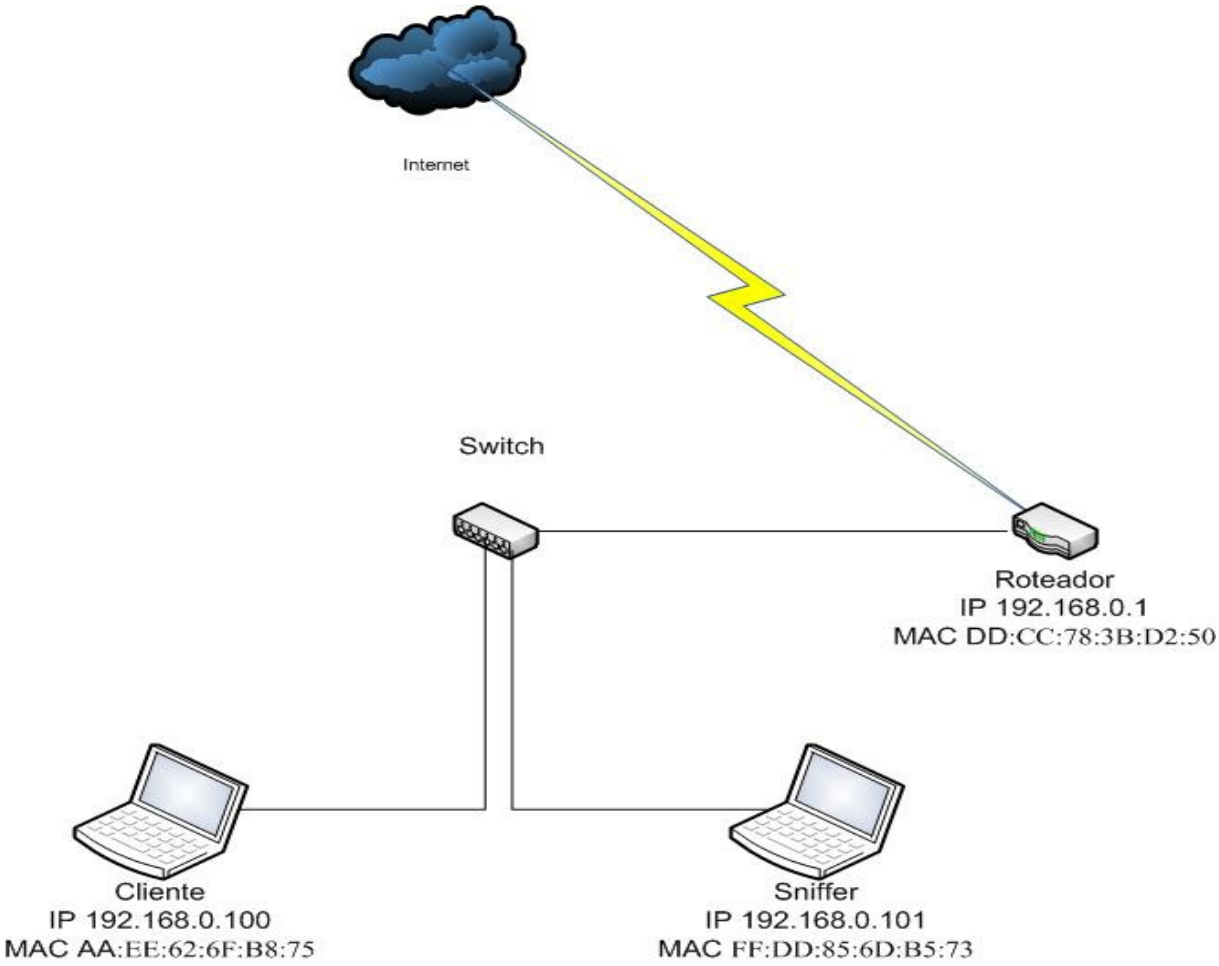

Figura 6.2: Figura de uma rede LAN utilizando um switch como comutador
Fonte: do autor

Esta imagem representa uma LAN (*Local Area Network*) e utiliza um switch como comutador. Nestas condições, o computador Sniffer não tem condições de monitorar o tráfego da estação cliente com destino a internet passando pelo Roteador. Para conseguir visualizar estes dados ele terá que realizar um ataque ARP Poisoning man-in-the-middle.

Segundo Ciampa (2009), para realizar este ataque é preciso executar os seguintes passos:

O computador Sniffer precisa enviar uma resposta ARP malicioso, mesmo sem solicitação, para o Roteador associando o seu endereço MAC FF:DD:85:6D:B5:73 ao endereço IP 192.168.0.100 que pertence ao computador Cliente;

O computador Sniffer também envia uma resposta ARP ao Cliente associando o seu endereço MAC FF:DD:85:6D:B5:73 ao endereço IP 192.168.0.1 o qual pertence ao Roteador.

Agora toda a comunicação realizada pela estação Cliente com destino a internet passando pelo Roteador, primeiramente passará pelo Sniffer que então reencaminhará os dados ao seu destino corretamente, sem que o Cliente tenha conhecimento que seus dados

estão sendo monitorados. Todos os dados que estão trafegando em texto puro, poderão agora ser facilmente capturados. A figura a seguir mostra como ficaria o fluxo dos dados com este ataque sendo executado.

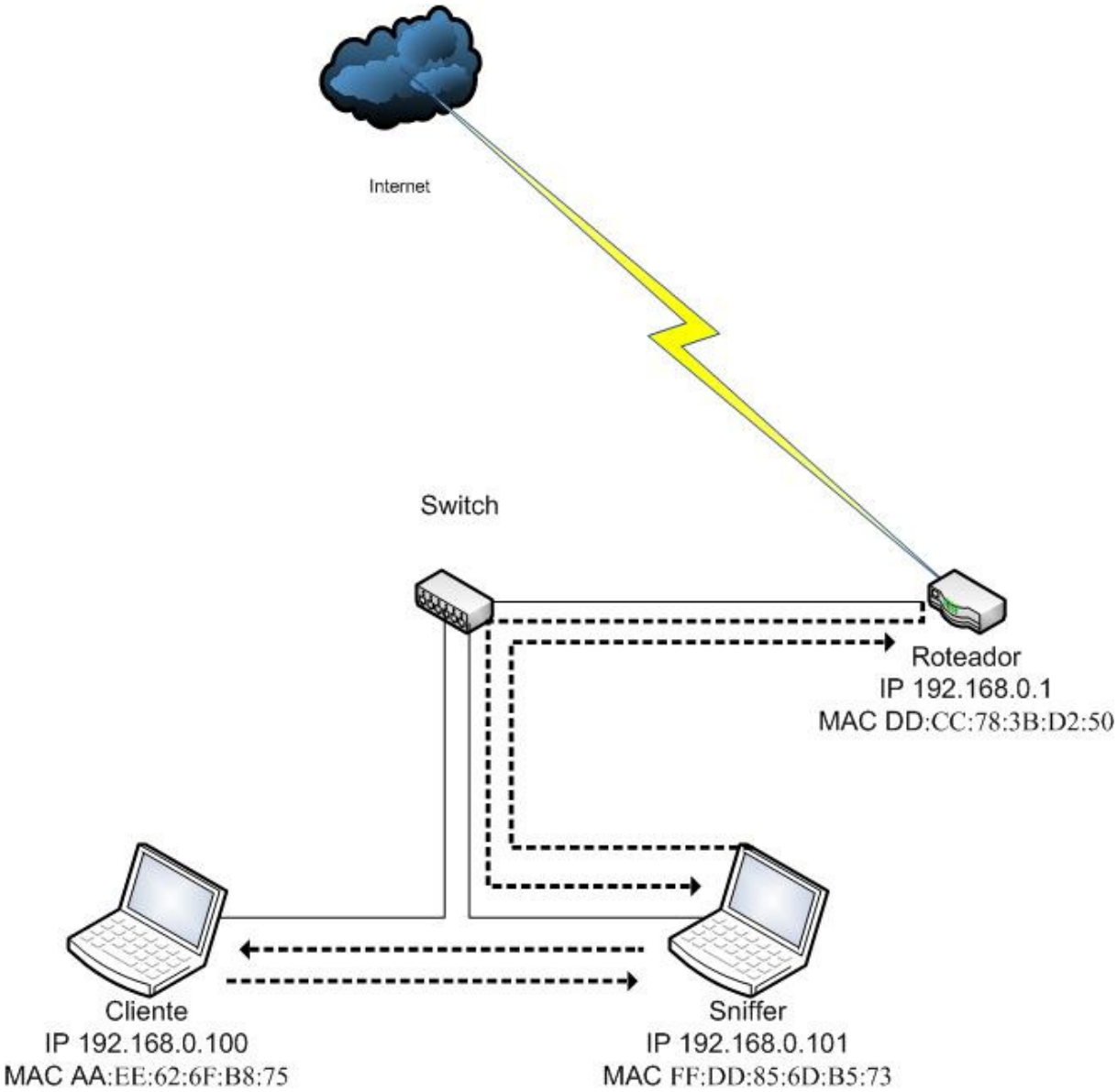

Figura 6.3: Ataque MITM: Todo o fluxo de dados passa pelo Sniffer
Fonte: do autor

Este é um exemplo de como burlar a capacidade dos switchs em criar diferentes domínios de colisão. Existem outros tipos de ataques tão perigosos quanto e também há maneiras de se proteger contra estes habilitando características, encontradas em switchs de qualidade, mas que raramente são utilizadas, como por exemplo, associando o endereço MAC da estação à porta do switch em que ela está conectada permitindo somente a alteração pelo administrador da rede.

Assim, conclui-se uma breve introdução teórica de como é possível monitorar o tráfego de estações tanto em redes wireless como em redes cabeadas, o objetivo agora é mostrar na prática como isso acontece.

6.1 CAPTURANDO O TRAFEGO

Voltando a figura 6.1, este é um exemplo do tráfego capturado neste ambiente de rede utilizando a ferramenta wireshark.

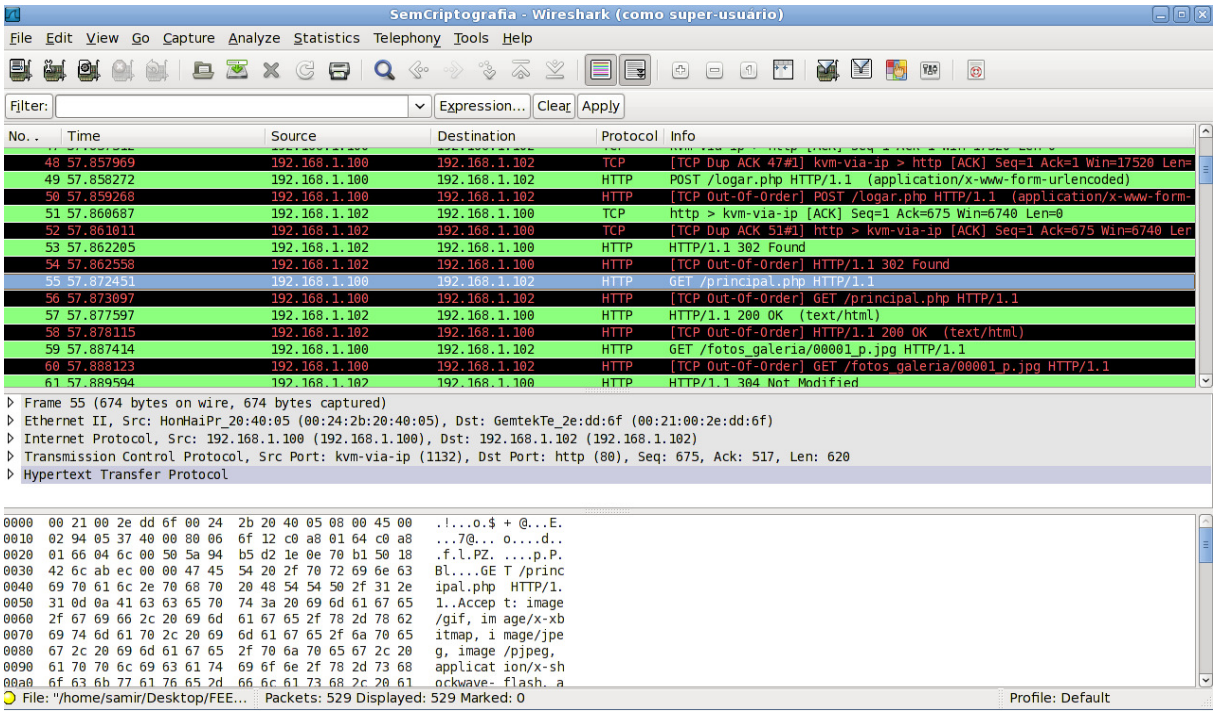

Figura 6.4: Captura de trafego com o wireshark
Fonte: do autor

Como se pode ver nesta figura, o trafego apresentado pelo wireshark mostra o endereço IP da estação Cliente e o IP do Servidor bem como o protocolo que está sendo utilizado. Em uma rede pode haver inúmeras estações transmitindo dados e pode-se capturar o tráfego de todas como também, filtrar o conteúdo especificando somente a estação alvo.

Como normalmente o tráfego capturado é volumoso, há uma opção de busca no wireshark que faz uma pesquisa no conteúdo através de strings que facilita localização de senhas ou algo do tipo, como mostra a figura a seguir.

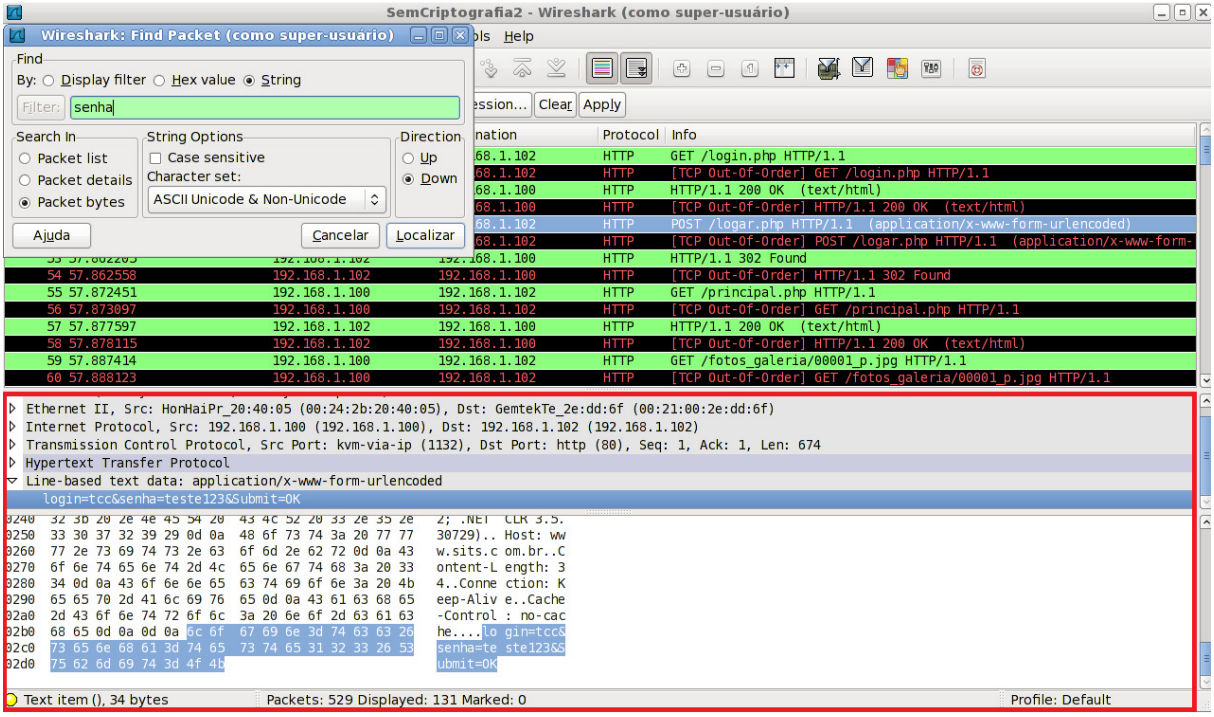

Figura 6.5: Pesquisa realizada no conteúdo do tráfego capturado em busca da string "senha" bem como a sua localização

Fonte: do autor

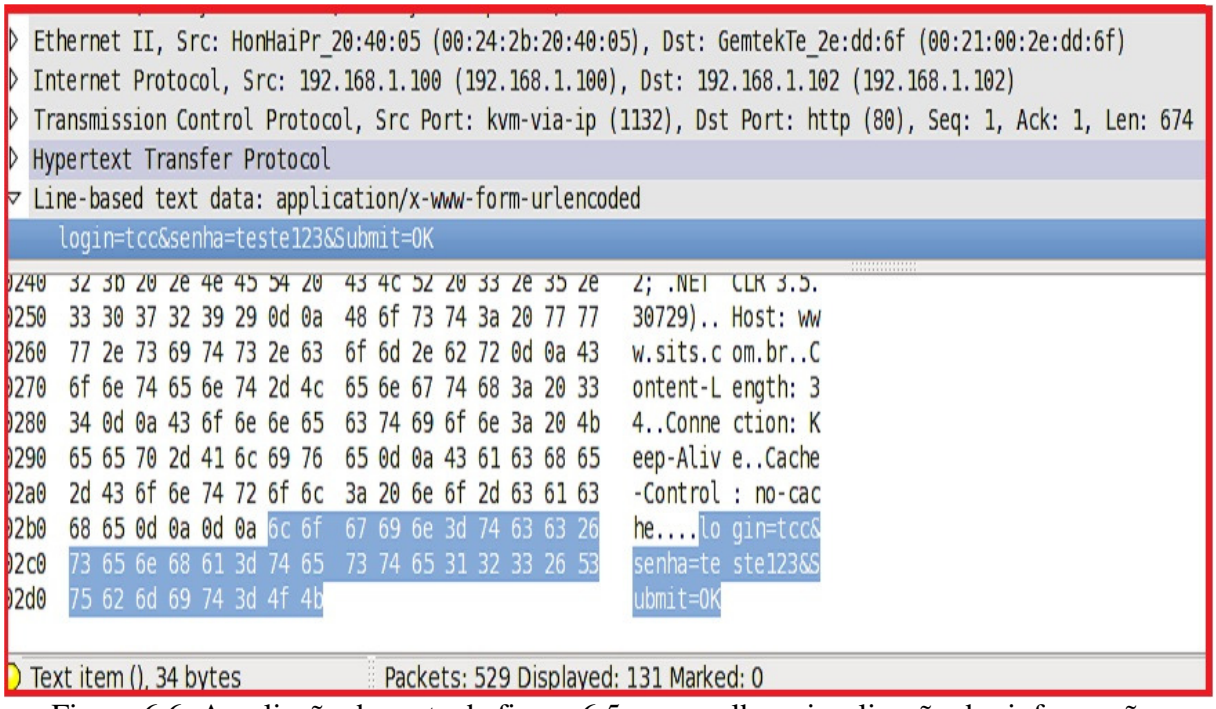

Figura 6.6: Ampliação de parte da figura 6.5 para melhor visualização das informações

Fonte: do autor

6.2 EXTRAÍNDO IMAGENS

Com o wireshark só é possível visualizar os dados em texto puro, ou seja, as imagens acessadas pelo Cliente no sistema são mostradas apenas no formato hexadecimal. Porém é possível salvar todo o tráfego capturado em um arquivo e então fazer uso de outras ferramentas para realizar a decodificação destas imagens.

O foremost e o tcpxtract são ferramentas que podem ser usadas para este fim, extrair arquivos baseados em assinaturas. A figura a seguir mostra a execução do tcpxtract, onde é possível visualizar as imagens sendo decodificadas.

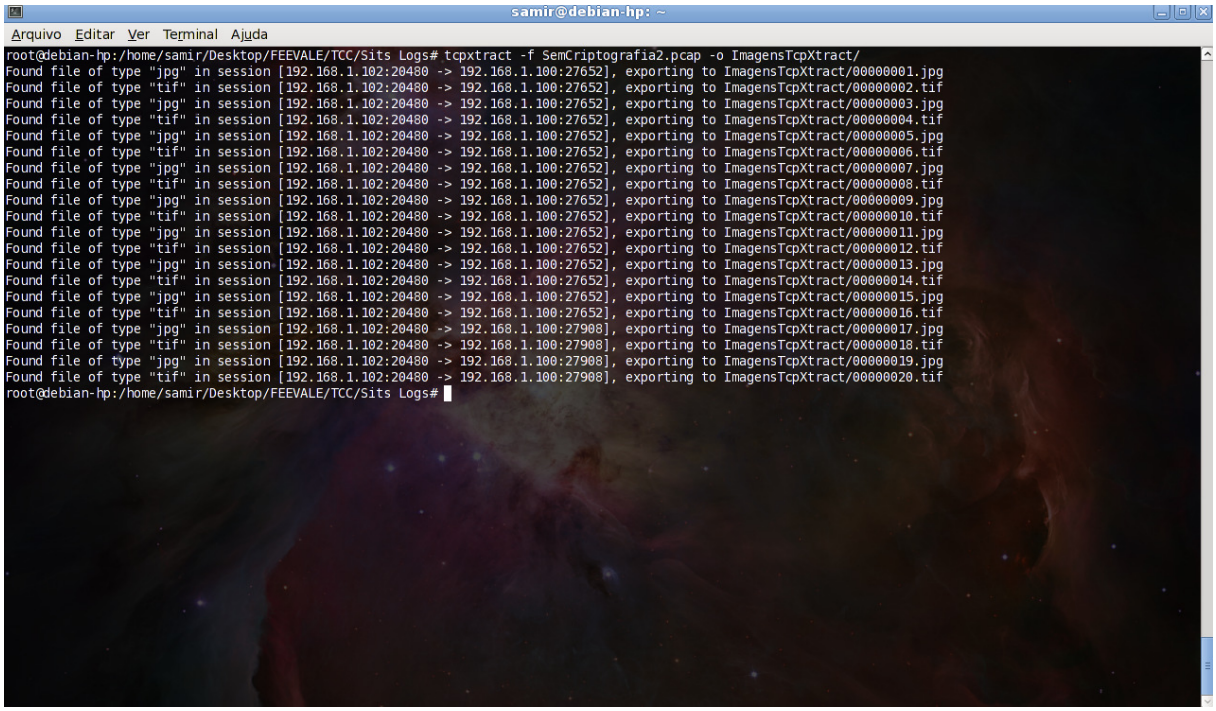

Figura 6.7: Extração das imagens com o tcpxtract diretamente do tráfego capturado
Fonte: do autor

Comando executado:

[root@debian]# tcpxtract -f SemCriptografia2.pcap -o ImagensTcpXtract

- -f: para especificar o arquivo que será analisado;
- -o: para especificar a pasta onde as imagens extraídas serão armazenadas.

Com o tcpxtract é possível extrair inúmeros tipos de arquivos, como por exemplo, das extensões a seguir: *Joint Photographic Experts Group* (JPEG), *Graphics Interchange Format* (GIF), *Tagged Image File* (TIF), *Portable Document Format* (PDF), *Áudio Vídeo Interleave* (AVI), *Word Processing Documents* (DOC), *Personal Storage Table* (PST) e

muitas outras. Também é possível adicionar novos formatos, caso estes não estejam incluídos no arquivo de configuração do mesmo (TCPXTRACT, 2011).

E assim, conclui-se na prática como é possível realizar a captura de dados em redes ethernet e também, fazendo-se o uso de algumas ferramentas de rede específicas disponíveis, pode-se obter informações que necessariamente precisam ser de conhecimento restrito. Dessa forma, o objetivo agora passa ser a proteção destas informações, de modo que os dados mesmo sendo capturados não tenham seu conteúdo revelado e ou modificado.

6.3 ATIVANDO O SSL

O primeiro passo para a criptografia dos dados é instalar o OpenSSL e habilitar o módulo SSL no apache.

Em algumas distribuições Linux o OpenSSL já vem instalado por padrão, mas caso não esteja ou se deseja a sua atualização, em distribuições Debian e suas derivadas, pode ser feita com o seguinte comando:

[root@debian]# aptitude install openssl

Segundo Pritchard et al (2007), para que o Apache possa fazer uso das bibliotecas do OpenSSL é necessário a instalação do módulo SSL que irá realizar este interfaceamento.

> Tradicionalmente, a configuração do Apache é centralizada em um único arquivo, o "httpd.conf", que pode opcionalmente incluir referências a arquivos externos (includes) que permitem segmentar a configuração. Aproveitando esta possibilidade, a equipe do Debian desenvolveu uma organização bastante prática, que é usada também no ubuntu e em outras distribuições derivadas dele (MORIMOTO, 2008).

Com esta descentralização é possível saber quais os módulos estão disponíveis para o Apache visualizando o conteúdo da pasta "mods-available" dentro do diretório "/etc/apache2". Quando um módulo já está habilitado ele é listado também dentro da pasta "mods-enable" no mesmo diretório.

A partir da versão 2.0 do Apache o mod_ssl é instalado juntamente com o pacote principal do Apache, neste caso basta apenas a sua ativação. Novamente, em distribuições Debian e suas derivadas a ativação é realizada com o seguinte comando:

[root@debian]# a2enmod ssl

6.3.1 GERANDO A CHAVE PRIVADA RSA

Com o OpenSSL instalado e o mod_ssl habilitado já é possível gerar a chave privada RSA, requerimento de certificado assinado, auto assinar um certificado e configurar o Apache para manter conexões SSL criptografadas.

A fim de manter os certificados melhores organizados será criada uma pasta onde após a sua geração ficaram armazenados.

[root@debian]# mkdir /etc/apache/certicados

Dentro da pasta "certificados" será gerada a chave privada RSA, que se chamará chave_privada.key com tamanho de 2048 bits e criptografada com o algoritmo Triplo DES.

O algoritmo de criptografia de chave assimétrica RSA é considerado um padrão para este tipo de criptografia e 2048 bits é o tamanho da chave que as Certificadoras estão exigindo para realizarem a assinatura do certificado.

[root@debian]# openssl genrsa -des3 -out chave_privada.key 2048

Após executar este comando, uma palavra-chave é requerida. Esta "senha" será solicitada toda vez que a chave privada for utilizada. Isso pode ser um problema porque toda vez que o Apache é reiniciado ele lê esta chave e então a senha deverá ser informada. Segundo Pritchard et al (2007), é recomendado a remoção da palavra-chave, embora isso implique com a segurança da chave privada porque sem a palavra-chave qualquer um de sua posse poderá se passar pelo Servidor Web nas conexões SSL. Isso requer então que a chave privada seja mantida em local seguro, fora do alcance de intrusos.

6.3.1.1 REMOVENDO A PALAVRA-CHAVE

A palavra-chave é removida com os seguintes comandos:

[root@debian]# mv chave_privada.key chave_privada.key.backup

[root@debian]# openssl rsa -in chave_privada.key.backup -out chave_privada.key

6.3.2 GERANDO O REQUERIMENTO DE CERTIFICADO ASSINADO

Agora com a chave privada sem senha, o próximo passo é criar um requerimento de certificado assinado CSR (*Certificate Signing Request*), utilizando a chave privada, que então poderá ser enviado a uma Autoridade Certificadora que o assinará com a sua chave privada para garantir a sua autenticidade.

O arquivo CSR é gerado com o seguinte comando:

[root@debian]# openssl req -new -key chave_privada.key -out requisicao_certificado.csr

Após executar o comando é necessário informar alguns dados que serão incorporados ao certificado e estarão disponíveis aos usuários que visitarem o sítio, por isso devem ser informados corretamente. A figura a seguir mostra os dados a serem informados.

```
root@debian-hp:/etc/apache2/certificados# openssl req -new -key chave_privada.key -out requisicao_certificado.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:BR
State or Province Name (full name) [Some-State]:Rio Grande do Sul
Locality Name (eg, city) []:Novo Hamburgo
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Trabalho de Conclusão
Organizational Unit Name (eg, section) []:TCII
Common Name (eg, YOUR name) []:www.sits.com.br
Email Address []:samir.avila@hotmail.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
root@debian-hp:/etc/apache2/certificados#
```

Figura 6.8: Criação de uma requisição de certificado assinado que pode ser enviada a uma Autoridade Certificadora.
Fonte: do autor

6.3.3 AUTO-ASSINANDO UM CERTIFICADO

O arquivo CSR criado pode também ser auto-assinado. O problema de um certificado auto-assinado é quando alguém acessar o site, o navegador irá informar ao usuário que não foi possível verificar se a identidade do mesmo é confiável, deixando por conta e risco do usuário a continuação. Esta mensagem pode fazer com que alguns usuários mais receosos abortem a comunicação.

Como as configurações estão sendo realizadas em um ambiente controlado, vamos assinar o certificado, para dar prosseguimento ao testes:

[root@debian]# openssl x509 -req -days 365 -sha512 -in requisicao_certificado.csr -signkey chave_privada.key -out certificado_assinado.crt

Este comando criará o arquivo certificado_assinado.crt. Ele está no padrão X.509 como foi detalhado anteriormente, terá a validade de um ano e o algoritmo de resumo de mensagem SHA-512 que faz parte da SHA-2, considerada a mais forte da família de funções hash. Para saber quais os algoritmos de hash estão disponíveis pelo OpenSSL basta executar o seguinte comando:

[root@debian]# openssl dgst -h

Agora que a chave privada e o certificado auto-assinado foram criados resta realizar as configurações no Apache para começar a servir páginas criptografadas.

6.3.4 DIRETIVAS DO MOD_SSL

Antes de iniciar a configuração do apache será feita uma descrição, de acordo com (ENGESCHALL, 2001), de algumas diretivas do mod_ssl que poderão ser utilizadas. Há diretivas que podem ser configuradas apenas para o contexto Virtual Host, ou seja, esta diretiva não afetará web sites que estiverem fora deste contexto. Porém, as diretivas configuradas no contexto global do Apache terão seu valor estendido a todos os sites ali configurados.

SSLProtocol – Esta diretiva define o protocolo SSL que será usado. Por padrão todos são habilitados. As opções são: SSLv2, SSLv3, TLSv1 e All.

SSLRandomSeed – Esta diretiva define as fontes para geração de dados aleatórios necessários para criptografar conexões SSL. Esses dados são conhecidos como semente, que irão alimentar o PRNG.

SSLRequireSSL – Quando esta diretiva está ativada, todos os pedidos são negados quando requisições HTTP não estiverem usando SSL.

SSLEngine – Esta diretiva habilita ou desabilita (on | off) o mecanismo do protocolo SSL.

SSLCertificateFile – Esta diretiva aponta para o caminho do certificado do servidor emitido por uma CA.

SSLCertificateKeyFile – Esta diretiva aponta para o caminho da chave privada do servidor.

SSLCipherSuite – Esta diretiva especifica a suíte de cifras disponíveis pelo servidor para uma negociação no momento do handshake. Por padrão, todas as cifras são

disponibilizadas, mas também é possível habilitar somente as mais seguras. A lista completa de cifras disponíveis pode ser verificada com o comando “openssl ciphers -v”.

SSLCertificateChainFile – Esta diretiva opcional, aponta para um arquivo que contem o caminho de certificados concatenados, começando pelo próprio certificado e terminando no certificado raiz da CA. Este caminho é enviado para o browser do usuário que poderá certificar a validade do certificado.

6.4 CONFIGURANDO O APACHE

O Apache é capaz de servir vários web sites em um único servidor. Cada um desses web sites é chamado de host virtual.

Como no servidor Apache já existe um web site rodando, o SITS, já existe um virtual host configurado. Com a descentralização do Apache, como já comentado, cada virtual host pode ser configurado em um arquivo diferente, o que ajuda na organização. Igualmente ao “mods-available” no diretório “/etc/apache2”, há uma pasta “sites-available” onde ficam os hosts virtuais, ou seja, é onde está o SITS que está configurado da seguinte forma:

[root@debian]# cat /etc/apache2/sites-available/sists

```
<VirtualHost *:80>
ServerAdmin admin@sits.com.br
ServerName www.sits.com.br
ServerAlias sits.com.br www.sits.com.br
DocumentRoot /var/www/sits
ErrorLog /var/log/apache2/error.log
CustomLog /var/log/apache2/access.log combined
</VirtualHost>
```

No momento em que o mod_ssl é ativado, o Apache passa a escutar também na porta 443.

Segundo Pritchard et al (2007, p.278): “A maneira mais fácil de ativar o Apache para manter conexões SSL é criar um contêiner de host virtual que mantenha todo tráfego vindo da porta 443”.

Conforme a declaração de Pritchard (2007), podemos adicionar um novo contêiner no mesmo arquivo, agora para a porta 443 e assim o SITS passaria a funcionar tanto com conexão criptografada quanto com conexões não encriptadas. Em alguns casos isso é muito interessante, por exemplo, há muitos sites que precisam do SSL em apenas uma parte de seu

conteúdo, como para cadastros e vendas e o restante do site pode-se navegar usando o protocolo HTTP não encriptado. Um dos motivos para isso é o alto custo do HTTPS em termos de processamento (MORIMOTO, 2008).

Como no caso do SITS toda conexão será encriptada, poderíamos remover o contêiner <VirtualHost *:80> deixando apenas o Virtual Host da porta 443. Mas também pode-se aproveitar este contêiner para redirecionar as conexões recebidas para porta 443. Normalmente, as pessoas não tem o costume de especificar o protocolo com o endereço do web site e por padrão quando não se especifica o protocolo, o browser utiliza o HTTP. Dessa forma a página não seria exibida e alguns usuários poderiam achar que o sistema não estivesse funcionando.

Agora, temos então esta configuração no arquivo sits:

Conforme comentado, este contêiner foi mantido para redirecionar conexões da porta 80 para a 443.

<VirtualHost *:80>
Endereço de E-mail que aparece com as mensagens de erro emitidas pelo servidor.
ServerAdmin admin@sits.com.br

Esta diretiva define o nome do hostname que o servidor utiliza para se identificar.
ServerName www.sits.com.br

Diretiva que seta nomes alternativos para o host.
ServerAlias sits.com.br

Redireciona uma solicitação de URL para outro URL permanentemente.
RedirectPermanent / https://www.sits.com.br

</VirtualHost>

<VirtualHost *:443>
ServerName www.sits.com.br
Especifica o diretório a partir do qual o Apache servirá os arquivos.
DocumentRoot /var/www/sits

Habilita o SSL/TLS
SSLEngine on

Caminho para o certificado assinado pela CA.
SSLCertificateFile /etc/apache2/certificados/certificado_assinado.crt

Caminho para a chave privada.
SSLCertificateKeyFile /etc/apache2/certificados/chave_privada.key

Cifras que o web site irá suportar.
SSLCipherSuite AES256-SHA:AES128-SHA

Arquivo onde serão registrados todos os erros que vierem a ocorrer.
ErrorLog /var/log/apache2/error.log

```
# Arquivo onde serão registrados os acessos realizados, ao site, de forma customizada.
CustomLog /var/log/apache2/access.log combined

# Instrui o Apache a criar variáveis de ambiente padrão, relacionadas à SSL/TLS, em solicitações CGI, SSI ou PHP.
#SSLOptions +FakeBasicAuth +ExportCertData +StrictRequire
    <FilesMatch "\.(cgi|shtml|phtml|php)$">
        SSLOptions +StdEnvVars
    </FilesMatch>
    <Directory /usr/lib/cgi-bin>
        SSLOptions +StdEnvVars
    </Directory>
# Estas diretivas contornam problemas que ocorrem em algumas versões do Internet Explorer nas interações entre SSL e HTTP/1.1.
    BrowserMatch "MSIE [2-6]" \
            nokeepalive ssl-unclean-shutdown \
            downgrade-1.0 force-response-1.0
    # MSIE 7 and newer should be able to use keepalive
    BrowserMatch "MSIE [17-9]" ssl-unclean-shutdown
    </VirtualHost>
```

Diretivas configuradas para o contexto global, no arquivo "/etc/apache2/apache.conf":

```
#SSL Global Options
# Habilita os protocolos SSLv3 e TLSv1, mas não o SSLv2
SSLProtocol all -SSLv2

# Fontes para geração de dados aleatórios
SSLRandomSeed startup file:/dev/urandom 1024
SSLRandomSeed connect builtin
```

Segundo a documentação do Apache, na diretiva SSLRandomSeed a variante builtin é a que consome menos ciclos de CPU (*Unidade Central de Processamento*) e está sempre disponível, mas não é recomendada a sua utilização na inicialização (startup) porque produz poucos bytes de entropia (ENGESCHALL, 2011).

Outro detalhe da configuração realizada é na diretiva SSLCipherSuite. Nesta, esta sendo habilitada somente a cifra AES, considerada uma cifra muito forte e a melhor escolha segundo a Tabela 2.1. O tamanho da chave poderá ser de 128 ou 256 bits, isso será definido pela negociação entre o navegador e o Servidor.

Com estas configurações, o apache já pode manter as conexões criptografadas, e caso as conexões sejam iniciadas pelo cliente com o protocolo HTTP não criptografada, a mesma será redirecionada para uma conexão criptografada.

6.5 CAPTURANDO OS DADOS DE UMA CONEXÃO CRIPTOGRAFADA

Agora, o tráfego será novamente capturado para que então sejam executados os mesmos testes que anteriormente demonstraram as vulnerabilidades de uma conexão não encriptada, onde foi possível capturar usuário e senha e também realizar a extração das imagens.

A figura 6.9 mostra o tráfego de dados sendo capturado no momento em que o cliente está realizando o acesso ao SITS.

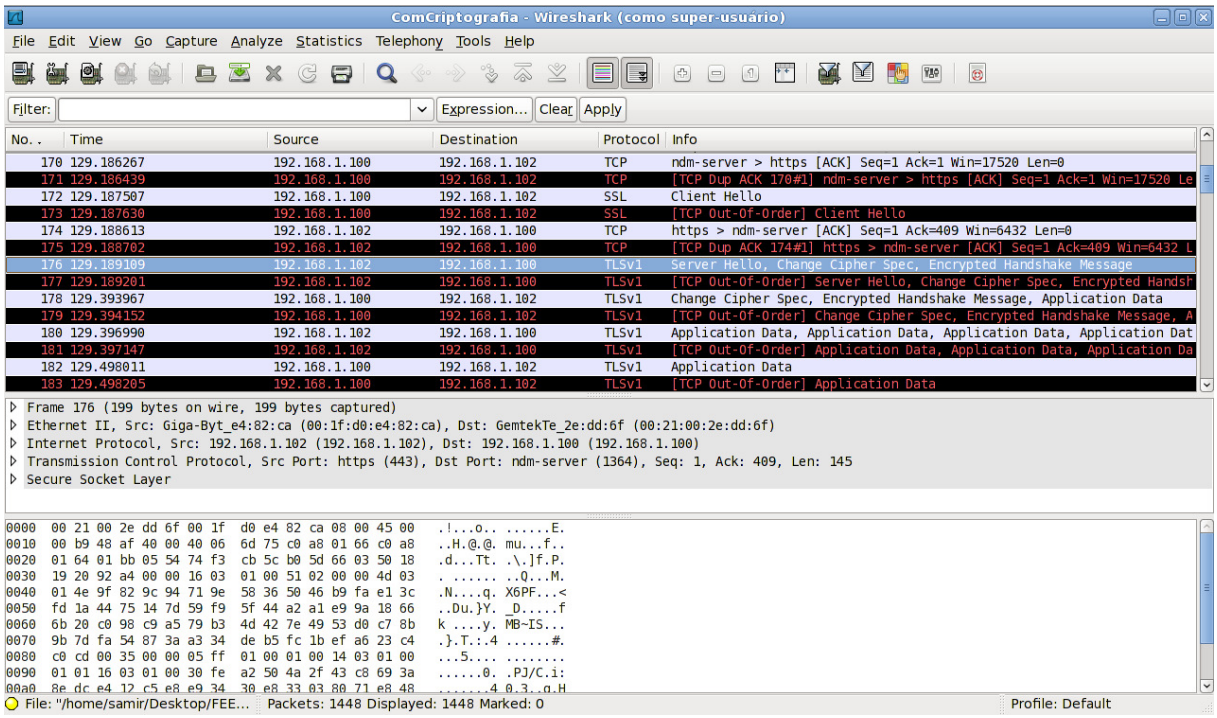

Figura 6.9: Tráfego criptografado capturado com o wireshark
Fonte: do autor

Nesta imagem é possível visualizar que o protocolo que está sendo utilizado é o TLSv1, bem como, o momento em que o handshake é estabelecido.

Com o tráfego capturado, será agora realizada uma pesquisa em seu conteúdo em busca da string “senha” que foi encontrada quando o tráfego não estava encriptado. A figura a seguir mostra a pesquisa realizada.

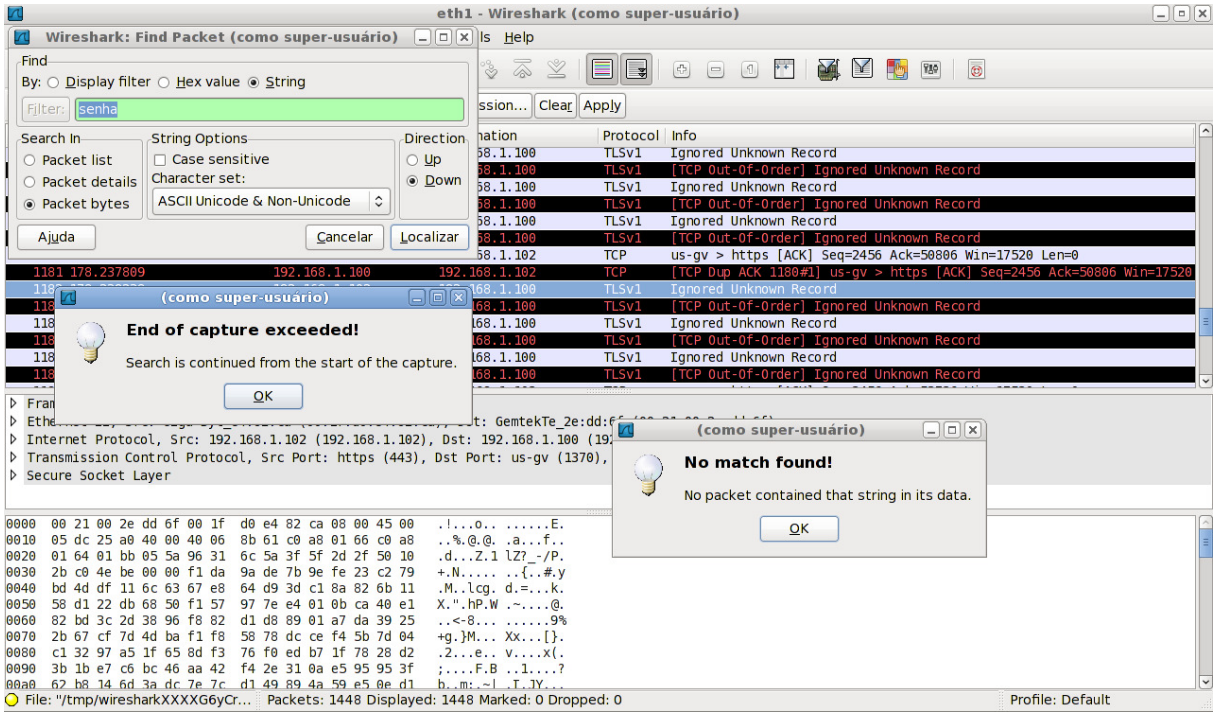

Figura 6.10: Tentativa de localização da String “senha” em conteúdo criptografado
Fonte: do autor

Como pode-se ver nesta figura, a string “senha” pesquisada não pode ser encontrada como ocorreu anteriormente. O próximo passo é salvar o tráfego capturado e então fazer uso novamente do tcpxtract, na tentativa de obter as imagens que foram visualizadas pelo cliente. A figura 6.5.3 mostra a execução do tcpxtract na tentativa frustrada de extrair imagens do tráfego capturado.

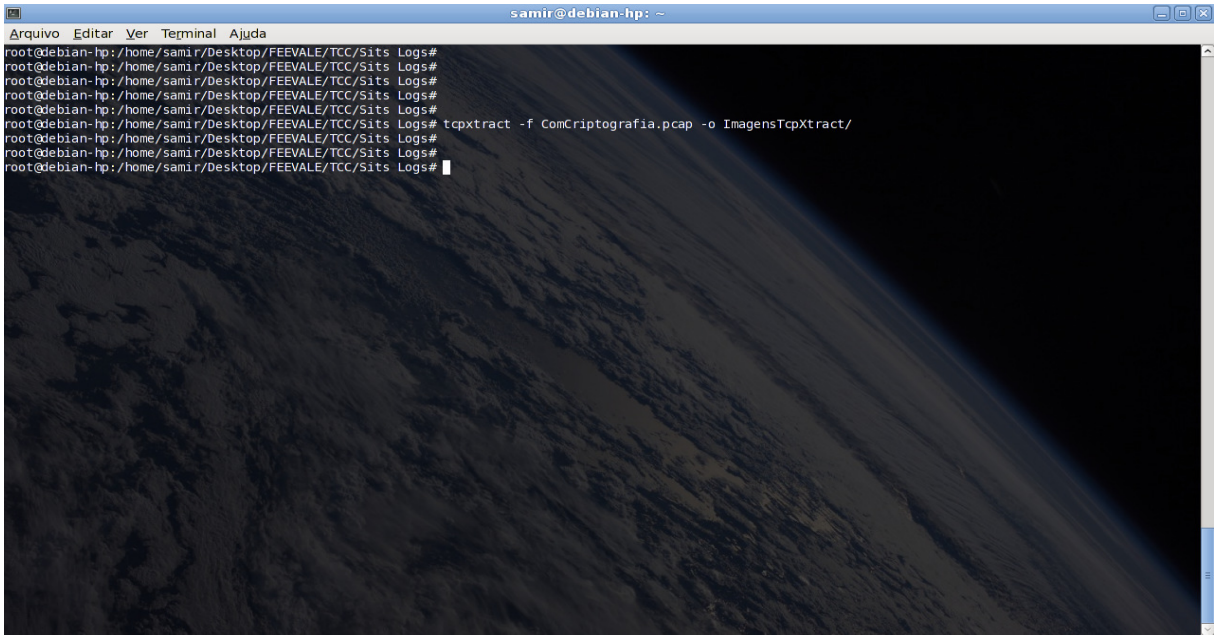

Figura 6.11: Tentativa de extrair imagens do tráfego criptografado capturado
Fonte: do autor

Assim foi demonstrado como a criptografia ajuda a proteger os dados, e mesmo quando são capturados não tem o seu conteúdo revelado. Certamente que, com esta configuração realizada, há riscos de um ataque MITM SSL, onde um atacante poderia substituir o certificado do site pelo seu próprio certificado e dessa forma não há como verificar a sua autenticidade (Anexo A).

Uma poderosa ferramenta para esse tipo de ataque é o ettercap. “Ettercap é uma suíte para ataques MITM em uma LAN” (ETTERCAP, 2011). Em ataques MITM SSL ele substitui o certificado real com o seu próprio. O falso certificado é criado no momento da conexão e os campos são preenchidos de acordo com o certificado real apresentados pelo Servidor, sendo somente o emitente diferente do original. A maneira mais eficiente de se proteger contra este ataque é tendo o certificado assinado por uma CA. Assim, caso o browser informe que não foi possível verificar a identidade do certificado porque o mesmo não foi homologado por uma autoridade reconhecida se faz necessário uma análise cuidadosa antes de continuar. Certas vezes isso poderá ocorrer mesmo o certificado tendo sido assinado por uma autoridade confiável, isso porque, o browser não tem em sua lista todas as CA’s existentes, mas estas podem então ser importadas.

Com as configurações definidas e os resultados dos testes aplicados, já é possível executar os mesmos passos no Servidor onde o Siap está hospedado para realizar esta implementação de segurança, objetivo principal deste projeto.

7 CONFIGURAÇÃO DO AMBIENTE DE PRODUÇÃO

Este capítulo tem como objetivo descrever a configuração do OpenSSL no Servidor Siap para este manter as conexões criptografadas.

A estrutura do Siap está dividida da seguinte forma: em uma parte do sistema a navegação é liberada sem a necessidade de autenticação, sendo em outra, uma área restrita, conhecida como *manager,* onde é exigido identificação dos usuários para o acesso.

Neste caso, não há a necessidade em criptografar toda a comunicação cliente/servidor, mas somente na área de acesso restrito economizando assim processamento por parte do servidor.

Os primeiros passos realizados para esta configuração foram gerar a chave privada, remover a palavra chave e gerar a requisição de certificado assinado. Até aqui, os passos executados foram idênticos aos executados nos subtítulos 6.3.1, 6.3.1.1 e 6.3.2.

Depois disso, o arquivo requisicao_certificado.csr foi enviado para a Certificadora GeoTrust para que a mesma o assinasse com a sua chave privada para assim garantir a sua autenticidade. A GeoTrust é uma certificadora reconhecida mundialmente, está na lista dos principais navegadores como Autoridade de Certificação e possui um dos valores da assinatura do certificado mais em conta.

O envio do arquivo para a Certificadora é feito através da internet em sua própria página. É necessária a informação de alguns dados os quais serão analisados para saber se a empresa ou instituição possui permissão para utilizar o domínio configurado no certificado.

Se os dados informados estiverem corretos e o valor cobrado pela assinatura pago, a Certificadora retorna por e-mail o certificado assinado e junto um certificado intermediário que contém outros certificados concatenados, que fazem parte da cadeia de certificados da CA.

Após o recebimento dos certificados, iniciou-se o processo de configuração do Servidor Web Apache. O processo foi um pouco diferente do realizado no ambiente de testes, devido às diferenças de organização dos arquivos de configuração do Apache que existem entre os sistemas operacionais do servidor de testes (Debian) e o de produção (Red Hat), porém sem grandes mudanças.

No Red Hat, os arquivos de configuração do Apache ficam no diretório "/etc/httpd". O arquivo principal de configuração é o httpd.conf localizado na pasta "/etc/httpd/conf/httpd.conf" e, para a configuração do contêiner para conexões criptografadas

o arquivo está localizado na pasta "/etc/httpd/conf.d/ssl.conf". É também no arquivo ssl.conf onde deve-se especificar o carregamento do mod_ssl da seguinte forma:

LoadModule ssl_module modules/mod_ssl.so

No arquivo httpd.conf, onde está configurado o virtual host do Siap foram adicionadas as seguintes linhas para negar pedidos a pasta manager, quando não estiver usando SSL, e redirecionar o tráfego HTTP para HTTPS aos usuários que acessarem o conteúdo da pasta manager.

Configuração inicial do arquivo httpd.conf:

```
<VirtualHost *:80>
#     ServerAdmin
DocumentRoot /var/www/html/siap
ServerName siap.ufcspa.edu.br
ServerAlias siap.ufcspa.edu.br
ErrorLog /var/www/log_siap/error.log
CustomLog /var/www/log_siap/access.lgo combined
</VirtualHost>
```

Configuração final do arquivo httpd.conf:

```
<VirtualHost *:80>
#     ServerAdmin
DocumentRoot /var/www/html/siap
ServerName siap.ufcspa.edu.br
ServerAlias siap.ufcspa.edu.br
ErrorLog /var/www/log_siap/error.log
CustomLog /var/www/log_siap/access.lgo combined
<Directory /var/www/html/siap/manager>
SSLRequireSSL
<Directory>
Redirect /manager https://siap.ufcspa.edu.br
</VirtualHost>
```

Configuração do arquivo ssl.conf:

```
# Diretivas globais
SSLRandomSeed startup file:/dev/urandom 1024
SSLRandomSeed connect builtin
SSLProtocol all –SSLv2

<VirtualHost *:443>
ServerName siap.ufcspa.edu.br
DocumentRoot /var/www/html/siap/manager
SSLEngine on
SSLCertificateFile /etc/httpd/certificados/certificado_assinado.crt
SSLCertificateKeyFile /etc/httpd/certificados/chave_privada.key
SSLCertificateChainFile /etc/httpd/certificados/certificado_intermediario.crt
SSLCipherSuite AES256-SHA:AES128-SHA
```

```
CustomLog logs/ssl_request_log \
        "%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x \"%r\" %b"
ErrorLog logs/ssl_error_log
TransferLog logs/ssl_access_log
LogLevel warn
#SSLOptions +FakeBasicAuth +ExportCertData +StrictRequire
<Files ~ "\.(cgi|shtml|phtml|php3?)$">
   SSLOptions +StdEnvVars
</Files>
<Directory "/var/www/cgi-bin">
   SSLOptions +StdEnvVars
</Directory>
SetEnvIf User-Agent ".*MSIE.*" \
        nokeepalive ssl-unclean-shutdown \
        downgrade-1.0 force-response-1.0
</VirtualHost>
```

Com esta configuração realizada bastou reiniciar o Apache para as conexões criptografadas começarem a funcionar. A figura a seguir mostra a confirmação de que a conexão está segura. Mais informações sobre o certificado (Anexo B).

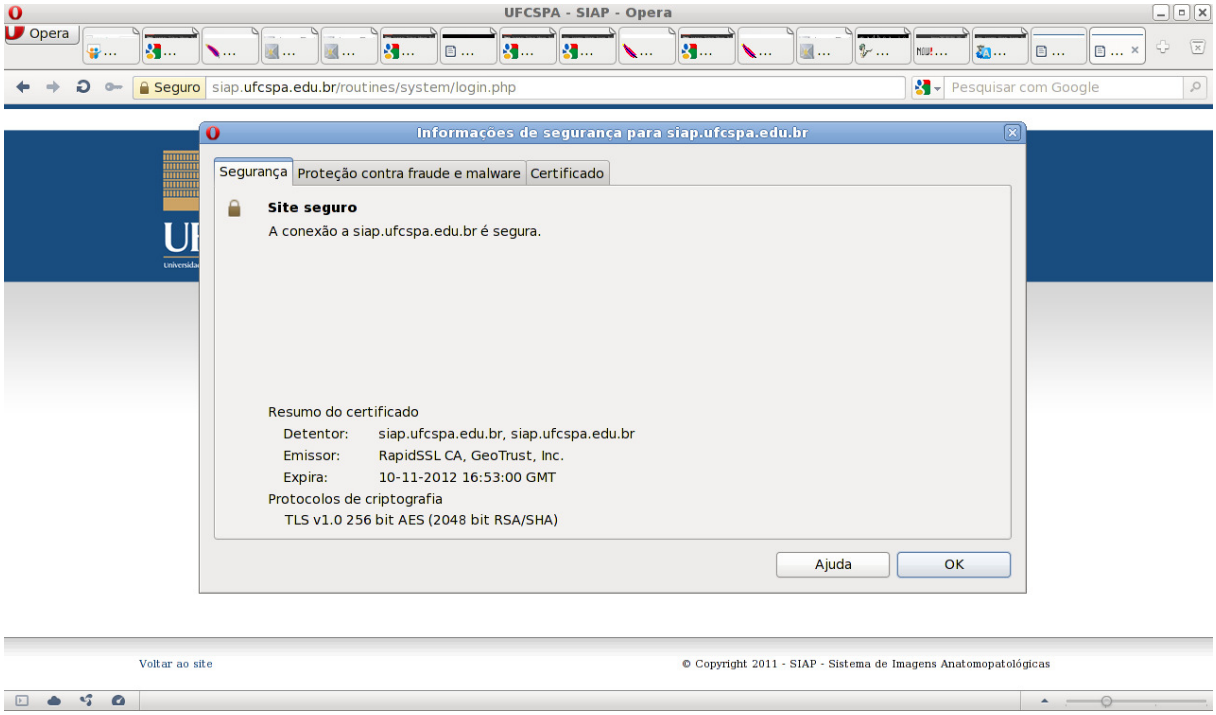

Figura 7.1: Confirmação de conexão segura
Fonte: (SIAP, 2011)

Com está confirmação é possível certificar que realmente estamos acessando o Servidor certo e a comunicação está sendo criptografada.

É extremamente importante clicar sobre o cadeado para verificar o certificado e não apenas visualizar a sua existência, pois existem ataques que retiram a criptografia da conexão, novamente em ataques MITM. Neste ataque, os dados trafegados entre o cliente e o invasor terão a criptografia removida, substituindo as requisições HTTPS por HTTP e até mesmo inserindo um favicon[3] com um cadeado para enganar o usuário. Dessa forma, o tráfego entre o usuário e o atacante seria totalmente em texto puro e entre o atacante e o servidor criptografado novamente. Este ataque é silencioso, pois o navegador não informa ao usuário de uma possível interceptação dos dados.

Este tipo de ataque foi exposto por Moxie Marlinspike, na Black Hat Briefings[4] de Washington DC em 2009. Utilizando a ferramenta sslstrip, de sua autoria, Marlinspike apresentou como capturar o tráfego SSL. Em sua página, Marlinspike disponibiliza o vídeo de sua apresentação onde mostra exemplos deste ataque (MARLINSPIKE, 2011).

7.1 AVALIAÇÃO FINAL DE SEGURANÇA DO SIAP

As tecnologias de hoje, por si só, não garantem a segurança de um sistema. O Siap está agora configurado para que mantenha conexões criptografadas em parte de seu conteúdo onde trafegam informações confidencias. Esta configuração além de proteger a confidencialidade e integridade dos dados também permite ao usuário certificar se a conexão realmente está sendo realizada com o servidor da UFCSPA. Contando com a atenção do usuário, é possível ter um alto nível de proteção contra a captura de dados na rede.

[3] Favicon são pequenas imagens no formato *.ico que podem ser vistas na barra de endereços dos navegadores.
[4] Black Hat Briefings constitui uma série de conferências altamente técnicas de segurança da informação que reúnem lideres de todas as facetas do mundo infosec.

CONCLUSÃO

Este trabalho apresentou um estudo das principais técnicas de criptografia disponíveis para proporcionar os requisitos necessários de segurança às informações transmitidas através da rede de computadores em um modelo cliente-servidor.

O foco de desenvolvimento do projeto de segurança abordado neste estudo está direcionado ao projeto SIAP (*Sistema de Imagens Anatomopatológicas*), disponível na UFCSPA. Uma análise prévia do sistema permitiu identificar que o mesmo não conta com nenhum tipo de segurança onde trafegam informações confidenciais. O sistema está disponível na web e alunos têm acesso às imagens e informações sobre as mesmas. Seguindo o que foi exposto na introdução, esta é uma grande falha em se tratando de informações médicas.

Para garantir a confidencialidade e integridade dos dados transmitidos pela rede pelo sistema SIAP, conforme apresentado, há a necessidade de uso de técnicas de criptografia combinadas para garantir o nível de segurança exigido para dados com este grau de sensibilidade.

O objetivo geral deste projeto foi propor uma solução para proteger os dados durante a sua transmissão pela rede. Com o uso do OpenSSL, que implementa os protocolos SSL/TLS, foi possível chegar a esta solução de segurança.

Quanto aos objetivos específicos do projeto, estes também foram alcançados no decorrer do trabalho da seguinte forma:

A falta de confidencialidade do Sistema de Imagens Anatomopatológicas (SIAP) foi demonstrada no subtítulo 6.1, onde o tráfego capturado em uma conexão utilizando o protocolo HTTP teve seu conteúdo exposto através da utilização de algumas ferramentas de rede.

As técnicas de proteção para envio de dados pela internet estudadas, criptografia simétrica, criptografia assimétrica e assinatura digital, isoladas não oferecem uma proteção segura para este tipo de comunicação.

Os protocolos criptográficos fazem o uso destas técnicas combinando-as e utilizando suas características únicas para prover um canal de dados seguro. Foram estudados os protocolos criptográficos SSL/TLS que são amplamente utilizados para o comércio eletrônico e transações financeiras, onde o nível que se exige de segurança é elevado.

A implementação da solução foi dividida em duas etapas. A primeira etapa foi realizada em um ambiente de testes onde foi possível realizar a configuração do módulo SSL e algumas de suas diretivas que ajudaram na compreensão da teoria que foi estudada e ainda permitiu a execução de testes de segurança e verificar que os dados estavam seguros.

Por fim, a implementação no ambiente de produção. As configurações realizadas foram praticamente idênticas ao ambiente de testes, como principal diferença o certificado sendo assinado por uma Autoridade Certificadora, o que permitiu um nível a mais de segurança obtido.

A avaliação final da segurança obtida com a utilização dos protocolos criptográficos SSL/TLS e o uso do certificado digital foi de que é possível realizar uma conexão segura neste ambiente desde que, necessariamente, o usuário certifique que a conexão foi estabelecida com o servidor intencionado. Isso vale para qualquer conexão realizada, via rede de computadores, por qualquer usuário que estiver fornecendo informações de caráter confidencial.

Como sugestão para trabalhos futuros fica a possibilidade de realização de um estudo sobre criptografia assimétrica baseada em curvas elípticas. Devido ao fato de que o tamanho das chaves assimétricas RSA estarem cada vez maiores para prover um nível de segurança adequado, as curvas elípticas podem proporcionar segurança equivalente com chaves menores o que implica em uma necessidade menor de capacidade de processamento dos dispositivos, ideal para smartphones, tablets e celulares.

REFERÊNCIAS BIBLIOGRÁFICAS

ARAÚJO, Gorgonio. **Transações Seguras na Web.** (1998) Disponível em: <http://www.rnp.br/newsgen/9803/https.html>. Acesso em 01 de abr. de 2011.

BURR, Willian E. **NIST Comments on Cryptanalytic Attacks on SHA-1**. (2006). Disponível em: <http://csrc.nist.gov/groups/ST/hash/statement.html>. Acesso em: 10 jun. 2011.

BURNETT, Steve. PAINE, Stephen. **Criptografia e Segurança: O guia oficial RSA**. Rio de Janeiro: Campus, 2002.

CARNEIRO, Tiago G. de Senna. **HTTP versus TCP.** Disponível em: <http://www.iceb.ufop.br/decom/prof/tiago/disciplinas/2006/sistDist/trabalhos/httpXtcp.htm>. Acesso em: 01 de jun. 2011.

COULOURIS, George. DOLLIMORE, Jean. KINDBERG, Tim. **Sistemas Distribuídos: Conceitos e Projeto**. 4ª ed. Porto Alegre: Artmed, 2007.

CREMESP. **Juramento de Hipócrates**. Disponível em: <http://www.cremesp.org.br/?siteAcao=Historia&esc=3>. Acesso em: 21 de março de 2011.

Declaração de Lisboa. Disponível em: <http://www.dhnet.org.br/direitos/codetica/medica/14lisboa.html>. Acesso em: 23 de mar. de 2011.

Declaração de Munique. Disponível em: <http://www.dhnet.org.br/direitos/codetica/medica/19munique.html>. Acesso em: 23 de mar. de 2011.

Declaração de Tel Aviv. Disponível em: <http://www.dhnet.org.br/direitos/codetica/medica/27telaviv.html>. Acesso em: 23 de mar. de 2011.

ENGESCHALL, Ralf S. **Apache Module mod_ssl**. Disponível em: <http://httpd.apache.org/docs/2.2/mod/mod_ssl.html>. Acesso em 14 de out. 2011.

ETTERCAP, **Short Description**. Disponível em: < http://ettercap.sourceforge.net>. Acesso em 10 de set. de 2011.

FOREMOST, **Introduction**. Disponível em: <http://foremost.sourceforge.net>. Acesso em: 29 de ago. de 2011.

GARAY, Juan A. PRISCO, Roberto De. **Security and Cryptografy for Networks: 7th International Conference, SCN 2010 Amalfi, Italy, September 2010 Proceeding.**

GARFINKEL, Simson. SPAFFORD, Gene. **Web Security, Privacy & Commerce**. Second Edition. United States of America: O'Reilly, 2002.

GOMES, Carlos F. Simões. RIBEIRO, Priscilla C. Cabral. **Gestão da Cadeia de Suprimentos a Tecnologia da Informação**. São Paulo: Pioneira Thomson Learning, 2004.

ITI. **Apresentação**. Disponível em: http://www.iti.gov.br/twiki/bin/view/ITI/Apresentacao. Aceso em: 12 de jun. de 2011.

ITIb. **Medida Provisória Nº 2.200-2, de 24 de agosto de 2001.** (2001) Disponível em: < http://www.iti.gov.br/twiki/pub/Certificacao/MedidaProvisoria/MEDIDA_PROVIS_RIA_2_200_2_D.PDF>. Acesso em: 12 de jun. de 2011.

KABIR, Mohammed J. **Apache Server 2**: **a Bíblia**. Rio de Janeiro: Campus, 2002.

KLEINJUNG, Thorsten et al. **Factoring of a 768-bit RSA modulus**. Versão 1.4 (2010). Disponível em: **<**http://eprint.iacr.org/2010/006**>.** Acesso em: 10 de jun. 2011.

KOBAYASHI, Luiz Octávio Massato. FURUIE, Sérgio Shiguemi. **Segurança em Informações Médica: Visão introdutória e panorama atual.** Revista Brasileira de Engenharia Biomédica. Volume 23, Número 1, 2007. Disponível em: <http://rbeb.ceb.unicamp.br/artigos/rev23/n1/art-f_23_1.pdf>. Acesso em: 20 de mar. de 2011.

KUROSE, James F. ROSS, Keith W. **Redes de Computadores e a Internet: Uma abordagem top-down**. 3ª edição. São Paulo: Pearson Addison Wesley, 2006.

LYRA, Maurício Rocha. **Segurança e Auditoria em Sistemas de Informação**. Rio de Janeiro: Editora Ciência Moderna, 2006.

MARLINSPIKE, Moxie**. New Tricks For Defeating SSL In Practice**. Disponível em: <http://www.thoughtcrime.org/software/sslstrip>. Acesso em: 12 de nov. de 2011.

MYSQL, **Open Source Database**. Disponível em: <http://www.mysql.com>. Acesso em: 28 de ago. de 2011.

NETCRAFT. Web Server Survey. Disponível em: <http://news.netcraft.com/archives/category/web-server-survey/>. Acesso em 21 mai. 2011.

OPENSSL. **OpenSSL: Cryptographic and SSL/TLS Toolkit**. Disponível em: <http://www.openssl.org/>. Acesso em: 21 de jun. de 2011.

OPPLIGER, Rolf. **SSL and TLS: Theory and Practice**. Massachusetts: Artech House, 2009.

PHP, **Hypertext Preprocessor**. Disponível em: <http://www.php.net>. Acesso em: 28 de ago de 2011.

SCAMBRAY, Joel. SHEMA, Mike. **Segurança Contra Hackers: Aplicações web**. São Paulo: Editora Futura, 2003.

SÊMOLA, Marcos. **Gestão de Segurança da Informação: Uma visão executiva**. Rio de janeiro: Elsevier, 2003.

SIAP, **Sistema de Imagens Anatomopatológicas**. Disponível em: <https://siap.ufcspa.edu.br>. Acesso em: 16 de nov. 2011.

SPECHT, Sandro Frazão. **Proposta de um Banco de Imagens em Código Livre para Recuperação e Conservação de Imagens Médicas da UFCSPA.** (2010) Novo Hamburgo: Universidade Feevale. Trabalho de conclusão de curso de Sistemas de Informação. Disponível em: <http://tconline.feevale.br/tc/files/0002_2323.doc>. Acesso em: 15 de mar. de 2011.

STANGER, James. LANE, Patrick T. **Rede Segura Linux: Seu guia de segurança em programas de código aberto.** Rio de janeiro: Editora Alta Books, 2002.

PRITCHARD, Steven et al. **Certificação Linux LPI: Nível 2: Exames 201 e 202**. Rio de Janeiro: Editora Alta Books, 2007.

TANENBAUM, Andrew S. **Redes de Computadores**. Tradução da 4ª edição. Rio de Janeiro: Elsevier, 2003.

TCPXTRACT, **Introduction**. Disponível em: <http://tcpxtract.sourceforge.net>. Acesso em: 29 de ago. de 2011.

THOMAS, Stephen. **SSL and TLS Essentials: Securing the Web**. New York: John Wiley & Sons, 2000.

TRIGO, Clodonil Honório. **OpenLDAP: Uma Abordagem Integrada**. São Paulo: Novatec, 2007.

UFCSPA, **Histórico**. Disponível em: <http://www.ufcspa.edu.br/institucional/historico.php>. Acesso em: 20 de mar. de 2011.

VIEGA, John. MESSIER, Matt. CHANDRA, Pravir. **Network Security with OpenSSL**. Estados Unidos da América: O´Really, 2002.

WIRESHARK, Network Protocol Analyzer. Disponível em: <http://www.wireshark.org>. Acesso em: 29 de ago. de 2011.

ANEXO A – IMAGEM DE UMA CONEXÃO COM CERTIFICADO AUTO ASSINADO

ANEXO B – INFORMAÇÕES DO CERTIFICADO ASSINADO

```
# openssl x509 -text -in certificado_assinado.crt
Certificate:
   Data:
      Version: 3 (0x2)
      Serial Number: 256996 (0x3ebe4)
      Signature Algorithm: sha1WithRSAEncryption
      Issuer: C=US, O=GeoTrust, Inc., CN=RapidSSL CA
      Validity
         Not Before: Nov  8 17:16:15 2011 GMT
         Not After : Nov 10 15:53:58 2012 GMT
      Subject: serialNumber=iDU2HZA3fTkd7xACPZtF87dbmNlz/EKf, C=BR,
O=siap.ufcspa.edu.br, OU=GT58776030, OU=See www.rapidssl.com/resources/cps (c)11,
OU=Domain Control Validated - RapidSSL(R), CN=siap.ufcspa.edu.br
      Subject Public Key Info:
         Public Key Algorithm: rsaEncryption
         RSA Public Key: (2048 bit)
            Modulus (2048 bit):
               00:b3:dc:5e:25:e5:86:8b:d8:d5:72:4c:5e:e3:a4:
               40:75:de:0d:28:b2:ce:58:cc:4a:e1:60:7d:76:d7:
               13:f7:19:8f:55:03:62:56:f5:b7:7e:d6:35:80:3c:
               82:8d:98:b3:80:5d:f4:2f:49:6c:68:72:36:d1:32:
               35:13:e3:be:f7:63:dd:93:86:42:4c:88:85:4a:4c:
               a4:7f:dc:c0:a7:ac:a8:2e:26:e6:b2:ff:09:8c:7e:
               86:71:30:ca:cf:4e:8a:9b:ba:2c:74:87:1d:91:1a:
               2b:b7:b6:c1:cf:ee:8a:87:c8:4d:24:0c:82:ed:57:
               6e:d3:1f:09:87:79:61:e7:89:ba:80:37:dc:0e:06:
               a2:49:6a:fd:54:3e:32:f7:88:48:8f:9c:6c:bb:85:
               c8:08:a6:35:76:d5:d0:d3:ba:a1:d8:8b:68:b3:34:
               e3:1c:dd:d8:23:3a:68:df:da:06:76:ea:6d:d5:03:
               18:de:cf:eb:4c:fe:6a:69:c9:4e:63:7a:ff:99:67:
               af:b7:e2:88:20:e5:a3:7d:e3:9b:a5:2a:08:8a:db:
               e7:92:2b:56:c2:de:04:74:76:0c:51:02:41:55:94:
               1b:ad:1d:53:87:bf:df:e2:95:80:54:4b:38:62:07:
               ef:f3:a7:69:5b:36:87:cf:eb:62:04:a7:97:39:de:
               ff:bb
            Exponent: 65537 (0x10001)
      X509v3 extensions:
         X509v3 Authority Key Identifier:
            keyid:6B:69:3D:6A:18:42:4A:DD:8F:02:65:39:FD:35:24:86:78:91:16:30

         X509v3 Key Usage: critical
            Digital Signature, Key Encipherment
         X509v3 Extended Key Usage:
            TLS Web Server Authentication, TLS Web Client Authentication
         X509v3 Subject Alternative Name:
            DNS:siap.ufcspa.edu.br
```

X509v3 CRL Distribution Points:
URI:http://rapidssl-crl.geotrust.com/crls/rapidssl.crl

X509v3 Subject Key Identifier:
E2:C2:EB:CE:57:8F:65:F2:4E:20:67:62:B6:06:7B:43:57:63:D5:1B
X509v3 Basic Constraints: critical
CA:FALSE
Authority Information Access:
CA Issuers - URI:http://rapidssl-aia.geotrust.com/rapidssl.crt

Signature Algorithm: sha1WithRSAEncryption
96:bb:fd:19:01:a6:39:06:88:14:f7:38:f4:ea:c6:bb:8b:af:
fb:3c:ce:9b:7d:d2:78:aa:59:7e:35:b6:c5:57:b0:8c:89:1d:
e8:ac:0d:83:1e:92:b7:11:17:e7:e6:b2:cf:8b:a5:17:89:d1:
9c:16:cc:5c:e0:d4:02:88:9a:8e:3c:8e:9f:50:be:32:50:43:
46:bc:03:13:4e:48:65:80:94:6f:c4:1c:e2:c1:81:b9:3b:2e:
70:b8:42:1a:33:ac:2a:04:34:fa:98:64:9d:00:04:9b:42:ea:
d1:91:29:af:11:0b:0a:42:99:0c:0e:3e:b6:9d:e4:25:e5:31:
ce:6d:d2:b2:a3:6f:36:e4:a8:a6:ea:34:75:40:27:e7:28:32:
e7:f5:99:45:f4:f8:68:a6:5f:99:d5:41:92:8c:0c:2f:5f:32:
52:ea:74:2e:8f:10:33:fc:39:a8:27:7d:c3:b5:bc:ef:f7:f9:
49:7a:23:71:48:ea:a7:71:59:b1:09:49:af:d5:b5:15:76:30:
67:ad:b9:1c:98:3b:d2:c1:23:5a:14:99:42:7f:21:31:ae:e1:
05:2d:b8:73:28:ff:2d:ae:f2:64:84:d3:0d:47:51:c3:a1:59:
a9:43:bc:ab:e7:b4:2d:b9:c9:ab:e5:f4:08:8f:90:87:7f:fe:
c1:27:86:d9

-----BEGIN CERTIFICATE-----
MIIE1zCCA7+gAwIBAgIDA+vkMA0GCSqGSIb3DQEBBQUAMDwxCzAJBgNVBAYTAlVT
MRcwFQYDVQQKEw5HZW9UcnVzdCwgSW5jLjEUMBIGA1UEAxMLUmFwaWRTU0wgQ0Ew
HhcNMTExMTA4MTcxNjE1WhcNMTIxMTEwMTU1MzU4WjCB6zEpMCcGA1UEBRMgaURV
MkhaQTNmVGtkN3hBQ1BadEY4N2RibU5sei9FS2YxCzAJBgNVBAYTAkJSMRswGQYD
VQQKExJzaWFwLnVmY3NwYS5lZHUuYnIxEzARBgNVBAsTCkdUNTg3NzYwMzAxMTAv
BgNVBAsTKFNlZSB3d3cucmFwaWRzc2wuY29tL3Jlc291cmNlcy9jcHMgKGMpMTEx
LzAtBgNVBAsTJkRvbWFpbiBDb250cm9sIFZhbGlkYXRlZCAtIFJhcGlkU1NMKFIp
MRswGQYDVQQDExJzaWFwLnVmY3NwYS5lZHUuYnIwggEiMA0GCSqGSIb3DQEBAQUA
A4IBDwAwggEKAoIBAQCz3F4l5YaL2NVyTF7jpEB13g0oss5YzErhYH121xP3GY9V
A2JW9bd+1jWAPIKNmLOAXfQvSWxocjbRMjUT4773Y92ThkJMiIVKTKR/3MCnrKgu
Juay/wmMfoZxMMrPToqbuix0hx2RGiu3tsHP7oqHyE0kDILtV27THwmHeWHnibqA
N9wOBqJJav1UPjL3iEiPnGy7hcgIpjV21dDTuqHYi2izNOMc3dgjOmjf2gZ26m3V
Axjez+tM/mppyU5jev+ZZ6+34ogg5aN945ulKgiK2+eSK1bC3gR0dgxRAkFVlBut
HVOHv9/ilYBUSzhiB+/zp2lbNofP62IEp5c53v+7AgMBAAGjggEwMIIBLDAfBgNV
HSMEGDAWgBRraT1qGEJK3Y8CZTn9NSSGeJEWMDAOBgNVHQ8BAf8EBAMCBaAwHQYD
VR0lBBYwFAYIKwYBBQUHAwEGCCsGAQUFBwMCMB0GA1UdEQQWMBSCEnNpYXAudWZj
c3BhLmVkdS5icjBDBgNVHR8EPDA6MDigNqA0hjJodHRwOi8vcmFwaWRzc2wtY3Js
Lmdlb3RydXN0LmNvbS9jcmxzL3JhcGlkc3NsLmNybDAdBgNVHQ4EFgQU4sLrzleP
ZfJOIGditgZ7Q1dj1RswDAYDVR0TAQH/BAIwADBJBggrBgEFBQcBAQQ9MDswOQYI
KwYBBQUHMAKGLWh0dHA6Ly9yYXBpZHNzbC1haWEuZ2VvdHJ1c3QuY29tL3JhcGlk
c3NsLmNydDANBgkqhkiG9w0BAQUFAAOCAQEAlrv9GQGmOQaIFPc49OrGu4uv+zzO
m33SeKpZfjW2xVewjIkd6KwNgx6StxEX5+ayz4ulF4nRnBbMXODUAoiajjyOn1C+
MlBDRrwDE05IZYCUb8Qc4sGBuTsucLhCGjOsKgQ0+phknQAEm0Lq0ZEprxELCkKZ
DA4+tp3kJeUxzm3SsqNvNuSopuo0dUAn5ygy5/WZRfT4aKZfmdVBkowML18yUup0
Lo8QM/w5qCd9w7W87/f5SXojcUjqp3FZsQlJr9W1FXYwZ625HJg70sEjWhSZQn8h
Ma7hBS24cyj/La7yZITTDUdRw6FZqUO8q+e0LbnJq+X0CI+Qh3/+wSeG2Q==
-----END CERTIFICATE-----