ALVARO ROGÉRIO CANTIERI

ANÁLISE DE SEGURANÇA DO PROTOCOLO SSL - SECURE SOCKET LAYER

Dissertação de Mestrado apresentada como requisito parcial para a obtenção do título de Mestre em Engenharia Elétrica no Programa de Pós-Graduação em Engenharia Elétrica - PPGEE da Universidade Federal do Paraná.

Orientador: Prof. Dr. Eduardo Parente

CURITIBA

2002

UFPR

UNIVERSIDADE FEDERAL DO PARANÁ
Programa de Pós-Graduação em Engenharia Elétrica
Área de Concentração Telecomunicações
Setor de Tecnologia

RELATÓRIO DE DEFESA DE MESTRADO

Aos vinte e cinco dias do mês de junho de 2002, no Laboratório didático do Departamento de Engenharia Elétrica - DELT, foi instalada pelo Prof. José Ricardo Descardeci, coordenador do Programa de Pós-Graduação em Engenharia Elétrica, a Banca Examinadora para a quarta Dissertação de Mestrado área de concentração: TELECOMUNICAÇÕES . Estiveram presentes no Ato, além do coordenador do Curso de Pós-Graduação, professores, alunos e visitantes.

A Banca Examinadora, atendendo determinação do Colegiado do Programa de Pós-Graduação em Engenharia Elétrica, ficou constituída pelos professores doutores **Eduardo Parente Ribeiro** (UFPR), **Marcus Vinícius Lamar** (UFPR), **Aloysio Nogueira Salgado**(UTP).

Às 14:00 horas, a banca iniciou os trabalhos, convidando o(a) candidato(a) **Álvaro Rogério Cantieri** a fazer a apresentação da dissertação intitulada " Análise de Segurança do Protocolo SSL – *Secure Socket Layer*". Encerrada a apresentação, iniciou-se a fase de argüição pelos membros participantes.

Tendo em vista a dissertação e a argüição, a banca atribuiu as seguintes notas: Prof. Dr. Marcus Vinícius Lamar Nota: 9,0, Prof. Dr. Aloysio Nogueira Salgado , Nota: 9,0. Prof. Dr. Eduardo Parente Ribeiro Nota: 9,0. A média obtida: 9,0, resulta na Aprovação do candidato, (de acordo com a determinação dos Artigos 61,62,63,64 da Resolução 38/96 de 14.06.96), e corresponde ao conceito A/B/C/D.

Curitiba, 25 de junho de 2002.

Prof. Dr. Aloysio Nogueira Salgado

Prof. Dr. Marcus Vinícius Lamar

Prof. Dr. Eduardo Parente Ribeiro

PROGRAMA DE ... UNIVERSIDADE FEDERAL DO PARANÁ ... ENGENHARIA ELÉTRICA

PRPTC - DELT

AGRADECIMENTOS

Agradeço a todos os que me apoiaram e acreditaram no meu trabalho, em especial aos meus familiares, e a minha querida companheira Elisa, que me fez acreditar ainda mais em meu potencial. Agradeço também em especial aos professores Eduardo Parente, meu orientador, pelo apoio essencial prestado, ao professor José Ricardo Descardeci pela ajuda na resolução dos problemas burocráticos que tanto incomodaram, e ao professor Horácio Tertuliano Filho, meu amigo pessoal e a pessoa que me levou a iniciar este mestrado.

SUMÁRIO

LISTA DE FIGURAS E TABELAS

LISTA DE SIGLAS

3DES - DES Triplo, ou seja, a aplicação do DES três vezes sobre a mesma mensagem.

AES – Advanced Encription Algorithm

ANSI - American National Standards Institute

CA – Certification Autority

CERT – Computer Emergency Response Team

CGI – Common Gateway Interface

DAS - Digital Signature Algorithm

DES - Data Encryption Standard

DH - Algoritmo Diffie-Hellman

DNS – Domain Name Sistem

DSS - Digital Signature Standard

EDH – Ephemeral Diffie-Hellman

FIPS PUBS - Federal Information Processing Standard Publications

FTP – File Transfer Protocol

HTTP – Hyper Text Transfer Protocol

IDEA - International Data Encryption Algorithm

IE – Internet Protocol

KDE – K Desktop Environment

MAC – Message Autentication Code

MD5 - Message Digest 5

NIST - National Institute of Standards and Tecnologies

NSA - National Security Agency

RC2 – Algoritmo de criptografia de chave privada desenvolvido pela RSA Data Security.

RC4– Algoritmo de criptografia de chave privada desenvolvido pela RSA Data Security .

RC5– Algoritmo de criptografia de chave privada desenvolvido pela RSA Data Security.

RFC – Request for Comments

RSA – Rivest, Shamir & Adleman

SHA-1 - Secure Hash Algorithm –1

SMTP – Simple Mail Transfer Protocol

TCP – Transport Control Protocol

TI – Tecnologia da Informação

TLS – Transport Layer Secure

VPN – Virtual Private Network

RESUMO

Este trabalho realiza uma análise teórica de segurança do protocolo para comunicações seguras SSL – Secure Sockets Layer. Inicialmente o protocolo é descrito, bem como os algoritmos utilizados em sua construção. Logo após, situações que possam gerar falhas na segurança do protocolo são analisadas e as conclusões são feitas. Finalizando, sugestões para a melhoria do nível de segurança das implementações do SSL são propostas.

ABSTRACT

This work realize a theoretical security analizis of the secure communications protocol SSL – Secure Sockets Layer. First, the protocol is described, as well the algorithms used in its construction. After that, situations that can generate security flaws of the protocol are analyzed and conclusions are made. Ending, suggestions to make better the security level fo implementations of SSL are proposed.

Capítulo 1 – Introdução

1.1 Introdução

O SSL, Secure Socket Layer, é um protocolo criptográfico utilizado em sistemas de computação, para a criação de um túnel de comunicação seguro entre duas máquinas. Os objetivos primários do protocolo são a garantia de privacidade nas comunicações, confiabilidade das informações, a verificação de identidade dos envolvidos, a garantia dos conteúdos das mensagens e não repúdio.

O SSL é provavelmente o protocolo mais utilizado para desenvolvimento de sistemas de comunicação seguros em aplicativos comerciais, estes para os mais diversos fins, principalmente os relacionados ao comércio eletrônico. O protocolo foi desenvolvido visando independência de tecnologia e flexibilidade de implementação, garantindo que possa ser implementado em qualquer plataforma, e ainda assim interagir com outros aplicativos de plataformas diferentes, o que o torna bastante versátil. Este provavelmente foi um dos motivos que o ajudou a se difundir tão amplamente.

A revisão dos aspectos de segurança do protocolo tem uma grande relevância prática, uma vez que este já existe há alguns anos, e que as tecnologias evoluíram consideravelmente neste tempo, de forma que torna-se bastante interessante a verificação da eficiência do mesmo diante dessa nova realidade. Isto, aliado ao fato de que atualmente o SSL vem sendo amplamente aplicado em diversos sistemas de comércio eletrônico baseados na Internet, mostram a necessidade de constante pesquisa e evolução do mesmo, buscando meios de garantir o cumprimento de seus objetivos fundamentais.

1.2 Motivação

O SSL é provavelmente o protocolo de comunicação segura mais utilizado nos esquemas de comércio eletrônico via Internet na atualidade, devido à sua versatilidade. Além dessa grande utilização, podemos notar também que este vem despontando como um padrão de fato para este tipo de aplicação.

Podemos notar um grande interesse na utilização do protocolo neste ano através de uma pesquisa realizada pela Computerworld e a J.P. Morgan com 174 gerentes de TI durante as duas últimas semanas de outubro de 2001 [1]. Nesta pesquisa 53% dos entrevistados disseram que em 2002 irão reservar uma porção maior do orçamento para segurança, em relação ao investimento dedicado neste ano. Apenas 5% responderam que os gastos com a proteção dos sistemas irão diminuir. Dentre estes investimentos, os relacionados à produtos *Secure Sockets Layer* seriam realizados por 47% dos entrevistados, enquanto os programas antivírus seriam realizados por 44% destes e os *firewalls* por 41%.

Assim podemos notar a relevância que este protocolo vem tomando no cenário da comunicação segura, mundialmente. Dessa forma, os estudos que realizem avaliações de segurança para o mesmo são sem dúvida uma ferramenta importante para o pleno desenvolvimento desta tecnologia, garantindo sua plena eficiência enquanto padrão para comunicações seguras escolhido informalmente pela comunidade mundial, mas de fato totalmente presente em nossas vidas.

1.3 Objetivo

O objetivo deste trabalho é realizar uma avaliação do protocolo SSL, no sentido de verificar sua eficácia em garantir os elementos de segurança por ele propostos perante o estado atual da tecnologia. Os objetivos são:

a) Realizar a revisão de segurança com relação à sua construção lógica, buscando a possível existência de pontos vulneráveis à quaisquer tipos de ataques que possam comprometer um dos elementos de segurança que este se propõe a garantir.

b) Verificar citações de ocorrências de falhas em aplicativos comerciais, buscando identificar o motivo de sua ocorrência e o nível de comprometimento que estas trazem ao funcionamento do protocolo.

c) Sugerir formas de aperfeiçoamento e correções para os pontos sensíveis encontrados na análise, de forma a gerar melhorias para o SSL ou para os aplicativos e esquemas práticos que o utilizam.

1.4 Metodologia Utilizada

Para a obtenção dos objetivos citados, inicialmente foram estudados os documentos descritivos do SSL e dos protocolos que este utiliza para sua construção, no sentido de entender os detalhes de funcionamento do protocolo de forma integral.

O segundo passo foi a verificação da utilização prática do SSL, desde as ferramentas de implementação até *softwares* que o utilizam comercialmente, com o intuito de avaliar como os detalhes de utilização do protocolo são trabalhados nos aplicativos comerciais.

Em seguida, foram levantadas citações de falhas do SSL, dos protocolos que o compõe ou de aplicativos que o utilizam, em artigos, revistas e sites de divulgação de falhas de segurança, especialmente no CERT – *Computer Emergency Response Team*, de forma a levantar os possíveis tipos de ataques aos quais o protocolo pode ser vulnerável. Todos as citações encontradas foram analisadas e as relevantes foram descritas e comentadas no trabalho.

No último passo foi realizada a análise de segurança do protocolo sob o foco de sua utilização em aplicativos comerciais, especialmente os voltados ao comércio eletrônico, de forma a verificar se este cumpre efetivamente as proposições de segurança descritas inicialmente em sua documentação.

1.5 Estrutura do Trabalho

A estruturação do documento foi feita da seguinte forma: No capítulo 2 são descritos os conceitos fundamentais da criptografia e dos mecanismos criptográficos utilizados no SSL. Em seguida, no capítulo 3, o protocolo é descrito e suas principais características analisadas. No capítulo 4 é realizada a análise de segurança do mesmo. No capítulo 5 são realizadas a análise e a discussão dos resultados obtidos. No capítulo 6 são sugeridos cuidados de implementação e ações que trariam possíveis melhorias ao funcionamento do protocolo ou dos aplicativos que dele fazem uso. Finalmente, no capítulo 7 são realizadas as conclusões do estudo.

Capítulo 2 – Segurança Computacional, Criptografia e Mecanismos Criptográficos

2.1 Considerações Gerais sobre Segurança de Sistemas Computacionais

Para que possamos desenvolver sistemas computacionais seguros, é necessária primeiramente uma definição formal dos conceitos de segurança computacional. A seguir serão realizadas as definições relevantes neste caso.

2.2 Definição de Segurança em um Sistema Computadorizado e Segurança em Sistemas de Comunicação Eletrônica.

Um sistema computacional pode ser considerado seguro se pudermos garantir que o mesmo se comportará de uma forma previamente esperada em qualquer situação, ou se em situações específicas conhecidas possamos prever, evitar ou corrigir uma possível falha, além de podermos confiar nos dados gerados por ele [2]. Dessa forma, o trabalho de segurança em computadores pode ser resumido na garantia dos seguintes itens:

- Disponibilidade - Garantia que o mesmo seja plenamente capaz de processar ou realizar um pedido em qualquer tempo de seu funcionamento.
- Confiabilidade - Garantia que os resultados de qualquer operação realizada pelo sistema sejam corretos.

- Privacidade - Garantia que o sistema e suas informações só possam ser acessadas por usuários que possuam autorização previamente definida.

Existem várias falhas que podem comprometer a confiabilidade de um sistema, mas todas podem ser resumidas em alguns pontos fundamentais:

- Erros de concepção – O erro de concepção ocorre quando da concepção lógica do sistema, ou seja, durante seu projeto teórico. Pode vir do desconhecimento de uma variável cuja influência seja relevante no problema, ou pode aparecer como um ponto de falha que tenha escapado à revisão dos envolvidos no desenvolvimento.
- Erros lógicos de implementação – Ocorre na transição do sistema teórico para o prático, ou seja, na sua implementação. Muitas vezes, principalmente em sistemas computacionais, existe a necessidade da inserção de aproximações ou a criação de novos algoritmos lógicos, para possibilitar a efetivação de algumas operações exigidas pelo sistema. O mesmo pode, dessa forma, vir a possuir uma incongruência lógica proveniente de uma implementação não totalmente eficiente, que se manifesta em situações específicas e gera um retorno que compromete uma parte ou a totalidade do mesmo. A possibilidade de um erro deste tipo ocorrer aumenta com o tamanho do sistema, com o número de variáveis envolvidas, com o nível de abstração da tecnologia utilizada para seu desenvolvimento e com o nível de complexidade matemática exigido para seu funcionamento.

2.3 Definição de Sistema de Comunicação Eletrônico Seguro

Para que um sistema de comunicação seja seguro, esse deve prover, no mínimo, os seguintes serviços [3]:

- Privacidade – Garantia de que somente o transmissor e receptor da mensagem possam ter acesso aos seus conteúdos.
- Identificação – Garantia de que os comunicantes sejam realmente quem dizem ser.
- Garantia de Conteúdo – Garantia de que as mensagens não sofreram alteração de seu conteúdo original durante o trânsito.

Atualmente a melhor forma de provermos tais serviços é através de técnicas criptográficas, assunto que será comentado no capítulo seguinte.

2.4 Definição de Criptografia e Conceitos Gerais

A criptografia é a melhor forma encontrada atualmente para a implementação de uma série de elementos necessários à criação de um sistema de comunicação computacional seguro. Apesar de ser uma técnica milenar, nunca antes a criptografia teve uma influência tão direta na vida das pessoas comuns, pois a usamos em praticamente todos os sistemas comerciais que utilizam computadores. Dessa forma, existe uma necessidade real de que todos possam compreender seus conceitos fundamentais, para que dela possamos nos utilizar de forma consciente e eficaz em nosso dia-a-dia.

De acordo com Lucchesi [4], "Criptografia ou grafia secreta é um conjunto de técnicas que permitem tornar incompreensível uma mensagem originalmente escrita com clareza de forma que apenas o destinatário a decifre e compreenda". Dessa forma podemos dizer que a criptografia "embaralha" uma mensagem através de um método que utiliza um determinado segredo ou "chave", tornando-a ininteligível, sendo que a mesma somente poderá ser recuperada através de um algoritmo que inverta tal transformação, utilizando-se para isto da "chave" em questão. A Figura 2.1 exemplifica o sistema.

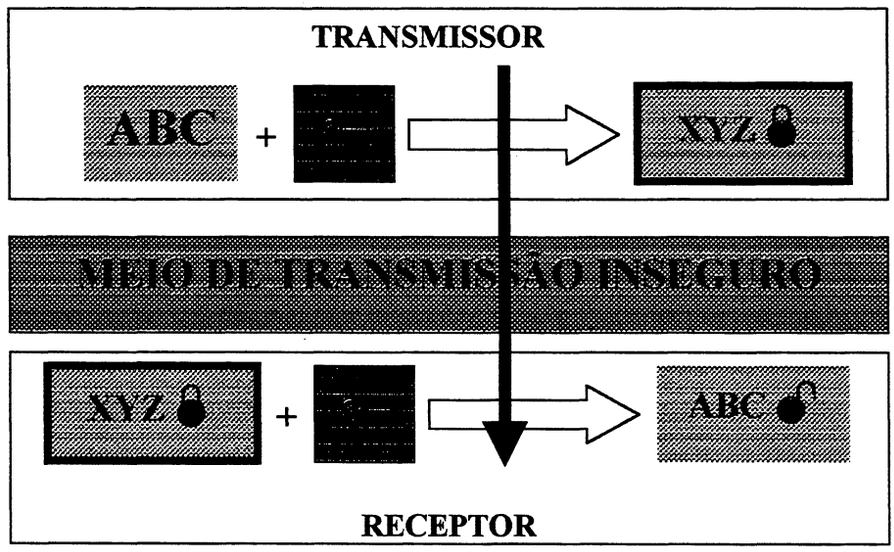


FIGURA 2.1 – Esquema Geral de Criptografia

Atualmente a criptografia é a ferramenta mais efetiva para garantir a privacidade em qualquer tipo de transmissão de dados através de sistemas abertos, pois garante que mesmo que qualquer pessoa não autorizada possa interceptar a transmissão, esta não poderá retirar dela nenhuma informação relevante.

Tecnicamente assume-se que um sistema criptográfico será considerado seguro se o mesmo obedecer os seguintes critérios [5]:

1) A segurança da mensagem não depende do conhecimento do método criptográfico utilizado.
2) A chave é grande o suficiente para que seja impossível descobri-la através de tentativas exaustivas.
3) O conhecimento de partes de texto cifrado com seu respectivo original não diminuem consideravelmente a dificuldade de recuperação do texto completo.

Existem dois tipos possíveis de algoritmos de encriptação de mensagens, a criptografia de chave simétrica ou privativa e a de chave assimétrica ou pública. No primeiro, a chave de

transmissão e recepção é a mesma, de forma que é necessário um método seguro para que os usuários realizem sua troca. No segundo, as duas chaves são diferentes, e somente a chave de desencriptação deve permanecer secreta, o que o torna mais facilmente administrável.

2.5 Algoritmos de Chave Simétrica ou Privada

No algoritmo de chave simétrica as chaves de encriptação e de desencriptação são as mesmas, de forma que para mantermos a mensagem em segurança devemos garantir que tal chave somente será conhecida pelo receptor e pelo transmissor. Esse tipo de criptografia é computacionalmente mais leve, mas possui o inconveniente da dificuldade de manuseio da chave utilizada [4].

Existem dois tipos possíveis de ciframento para algoritmos de chave simétrica, o ciframento em blocos e o ciframento bit a bit [5].

No caso do ciframento em blocos, a mensagem é fragmentada em blocos de tamanho fixo e cada bloco é cifrado separadamente com a chave secreta da sessão criptográfica. A saída é também um bloco de tamanho fixo, maior ou igual ao tamanho do bloco de entrada. Para este tipo de mecanismo de ciframento deve-se fazer o tamanho dos blocos o maior possível, para evitar a possibilidade de análise por exaustão.

No caso do ciframento bit a bit, utiliza-se um gerador de bits, que por sua vez gera uma cadeia criptográfica que será combinada através de uma operação ou-exclusivo bit a bit com a cadeia de bits da mensagem. Geralmente esta mensagem também é fragmentada em blocos de n bits para efeito de aplicação do algoritmo, reduzindo assim a dificuldade computacional.

Para dificultar os ataques que exploram a substituição de blocos de texto criptografados selecionados, os algoritmos criptográficos utilizam o encadeamento dos blocos das mensagens, ou seja, a realimentação do sistema com o resultado dos criptogramas gerados nos blocos anteriores.

Os algoritmos de chave simétrica padrões do SSL são o DES (*Data Encryption Standard*), o IDEA (*International Data Encryption Algorithm*), o RC2, RC4 e RC5 [6].

2.5.1 O DES - Padrão de Criptografia de Dados

O DES, (Data Encryption Standart) ou Padrão de Criptografia de Dados, foi adotado como padrão pelo governo dos EUA em 1977 para atividades não ligadas à segurança nacional. Em 1981 foi aceito como padrão também pela ANSI (American National Standards Institute) e atualmente está presente em muitos protocolos de comunicação seguros, como por exemplo o SSL [3].

O DES é um ciframento composto que cifra blocos de 64 bits, mediante uma chave de 64 bits dos quais 8 são de paridade [7]. Assim, para cada chave, o DES é uma substituição monoalfabética sobre o alfabeto de $2^{64} = 1{,}8x10^{19}$ letras. O processo de deciframento é realizado reaplicando-se o algoritmo à mensagem cifrada, com a inversão da ordem das chaves utilizadas.

A segurança do algoritmo DES vem sendo constantemente desafiada pelos pesquisadores da criptografia, e apesar de seus 20 anos de utilização e testes, nunca apresentou alguma falha lógica capaz de comprometê-lo. Apesar disto, em 1999, a Distribute.Net e a Electronic Frontier Foundation realizando um trabalho em conjunto, implementaram a um custo relativamente baixo, um sistema computacional denominado DES

Cracker que foi capaz de quebrar uma chave do DES, em menos de 24 horas efetuando uma busca exaustiva [8]. Este sistema utilizava cerca de 100.000 computadores pessoais, o que provou entre outras coisas, que sistemas de computação distribuída podem ser eficientes na quebra de chaves criptográficas através de força bruta. Além disso este trabalho demonstrou que o tamanho da chave do DES já não era mais adequado e que seria necessário um sucessor para este. Seu sucessor escolhido é o AES - Advanced Encryption Algorithm [9]. O AES utiliza chaves de 160 ou 224 bits, o que o torna extremamente robusto à ataques de força bruta. Como o SSL foi projetado para poder facilmente ser adequado a um novo tipo de algoritmo, não deverão haver maiores problemas para a substituição do DES pelo AES.

2.5.2 DES Triplo - 3DES

Outro esquema criptográfico utilizado é o DES Triplo, ou seja, a aplicação do DES três vezes sobre a mesma mensagem. Geralmente utiliza-se apenas duas chaves diferentes para as três encriptações, o que nos garante uma chave efetiva de 112 bits, o que já é suficientemente grande para a maioria das aplicações, mas pode-se utilizar três chaves diferentes se necessário, gerando uma chave efetiva de 168 bits [3].

2.5.3 IDEA – International Data Encryption Advanced

O IDEA foi projetado por dois pesquisadores na Suíça, como uma opção para a substituição do DES [7]. Ele utiliza uma chave de 128 bits, o que o torna bastante seguro em relação aos ataques de força bruta. Não foram encontradas indicações nas referências bibliográficas sobre qualquer fraqueza ou inconsistência do algoritmo do IDEA. Sua chave criptográfica de 128 bits torna sua quebra através de força bruta muito difícil, garantindo à este um bom nível de segurança para as aplicações a que se propõe [10].

2.5.4 Os algoritmos RC2 e RC4

O RC2 e o RC4 são algoritmos criptográficos desenvolvidos na década de 80 por Ronald Rivest para serem utilizados pela RSA Data Security. São protegidos por segredo industrial e patentes, de forma que sua utilização precisa ser licenciada. Como são algoritmos de código secreto, não existem publicações oficiais dos mesmos disponíveis para análise sendo portanto difícil o levantamento de parâmetros técnicos adequados para medi-la [3].

O RC2 é um algoritmo de cifra em blocos, que utiliza blocos de tamanho variável [11]. De acordo com a RSA Security, é mais rápido que o DES e pode ser mais seguro de acordo com o tamanho da chave utilizada.

O RC4 é um cifrador de cadeia, com chave de tamanho variável dependendo do tipo de operação [12]. Seu funcionamento é baseado na utilização de permutações randômicas. De acordo com a RSA, é mais rápido que o DES e pode ser mais seguro dependendo do tamanho da chave utilizada.

2.5.5 O algoritmo RC5

O RC5 foi projetado para qualquer computador de 16 ou 32 ou 64 bits. Possui uma descrição compacta e é adequado para implementações em *software* ou hardware. Como o DES, RC5 possui várias iterações e as várias subchaves são utilizadas em uma função de iteração. Ao contrário do DES, o número de iterações e o número de bytes na chave são variáveis. Baseia-se na operação de rotação (deslocamento circular) de um número variável de posições, e esse número depende de quase todos os bits resultantes da iteração anterior e do valor da subchave em cada iteração [10].

Após alguns anos de análise dos especialistas sabe-se que, depois de oito iterações, todo bit de entrada legível afeta pelo menos uma rotação. Há uma criptanálise diferencial para o RC5 com 64 bits de entrada que necessita 2^{24} textos legíveis escolhidos para cinco iterações, e 2^{68} para 15 iterações. E contra a criptanálise linear o RC5 é considerado seguro após seis iterações [10].

2.6 Algoritmos de Chave Assimétrica ou Pública

Os algoritmos de chave assimétrica ou pública trabalham baseados em uma idéia fundamental, de utilizar-se duas chaves distintas, uma para encriptar a mensagem e outra para desencriptar. O método funciona da seguinte forma:

- O receptor da mensagem deve ter gerado anteriormente as duas chaves da sessão, a primeira, que será enviada ao transmissor, e que não exige uma confidencialidade, e a segunda, secreta, que fica em sua posse.

- Para que o transmissor possa enviar a mensagem secretamente, esse usa a chave pública do receptor para encriptá-la.
- De posse da mensagem encriptada, o receptor utiliza sua chave secreta e a desencripta, recuperando assim a mensagem original. Como é impossível desencriptar a mensagem a partir da mesma chave utilizada para encriptá-la, não existe problema em torná-la pública. A Figura 2.2 mostra o esquema de chave pública.

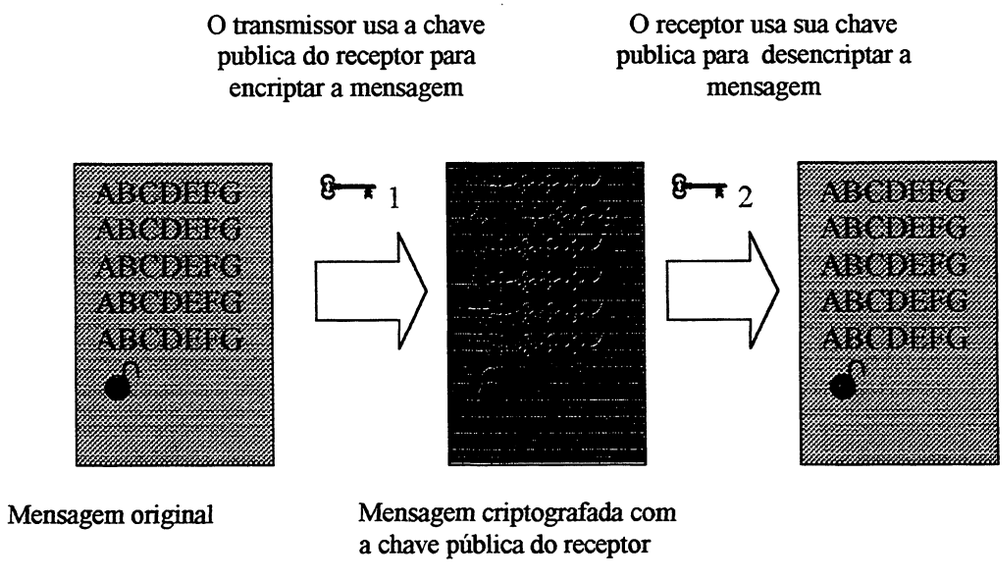


FIGURA 2.2 - Esquema de Criptografia de Chave Pública

O algoritmo de criptografia de chave pública deve atender a três requisitos básicos [5]:

1. **D (E(P)) = P,** onde **D** é o algoritmo de desencriptação, **E** o de encriptação e **P** a mensagem original.
2. É excessivamente difícil deduzir **D** de **E**
3. **E** não pode ser decifrado através do ataque do texto simples escolhido, onde **D** é o algoritmo de criptografia, **E** o algoritmo da desencriptação e **P** é a mensagem original.

Os dois algoritmos de chave pública utilizados pelo SSL são o algoritmo Diffie - Hellman e o algoritmo RSA (Rivest, Shamir e Adleman), descritos em seguida.

2.6.1 O Algoritmo Diffie – Hellman

O método Diffie – Hellman de troca de chaves não é um algoritmo criptográfico em si, mas sim um algoritmo de chave pública específico para a troca de chaves de uma sessão criptográfica. Com ele podemos enviar uma chave criptográfica simétrica através de uma conexão em aberto sem que um interceptador possa ter acesso à ela [3]. O funcionamento do método segue os seguintes passos:

- O transmissor gera dois números públicos, o número **p**, primo, e o número **g**, inteiro menor que ***p***, e os envia abertamente ao receptor.
- O transmissor calcula o valor $\mathbf{Kt} = \mathbf{g}^{\mathbf{x}} \bmod \mathbf{p}$, onde **x** é um valor secreto gerado randomicamente, e envia **Kt** ao receptor.
- O receptor calcula o valor $\mathbf{Kr} = \mathbf{g}^{\mathbf{y}} \bmod \mathbf{p}$, onde **y** é um valor secreto gerado randomicamente, e envia ao transmissor **Kr**.
- O transmissor recebe **Kr** e calcula $\mathbf{K} = (\mathbf{g}^{\mathbf{y}})^{\mathbf{x}} \bmod \mathbf{p}$, onde **K** é a chave a ser utilizada por ambos.
- O receptor recebe **Kt** e calcula $\mathbf{K} = (\mathbf{g}^{\mathbf{x}})^{\mathbf{y}} \bmod \mathbf{p}$, obtendo também a chave.

Como nenhuma outra pessoa conhece **x** e **y**, mesmo que intercepte todas as mensagens transmitidas, não será capaz de decifrá-las e encontrar **K**. Dessa forma somente o transmissor e o receptor terão acesso a esta chave, podendo comunicar-se com segurança.

2.6.2 O algoritmo RSA – (Rivest, Shamir & Adleman)

O RSA, cujo nome é uma homenagem aos seus criadores, é um algoritmo de chave pública capaz de oferecer tanto criptografia quanto assinaturas digitais. Foi desenvolvido em 1977, sendo ainda hoje o algoritmo desse tipo mais utilizado nas aplicações comerciais. Utiliza como base a intratabilidade do problema da fatoração de inteiros, pois de acordo com Coutinho [13] não se conhece, ao menos publicamente, um algoritmo eficaz para a fatoração de números inteiro longos sendo que, por outro lado, a geração de dois números primos e o posterior produto dos mesmos é uma tarefa computacional relativamente fácil.

O método RSA funciona basicamente da seguinte forma [14]:

1 – Geração das chaves criptográficas:

- Escolhem-se dois números primos extensos, **p** e **q**, (geralmente maiores que 10^{100}).
- Calcula-se $\mathbf{n = p \cdot q}$ e $\mathbf{z = (p-1) \cdot (q-1)}$
- Escolhe-se um número relativamente primo em relação a **z**, chamado de **d**.
- Encontra-se **e** de forma que $\mathbf{e \cdot d = 1 \bmod z}$.

O par **(n,e)** é a chave pública e o par **(n,d)** a chave privada. Os valores **p** e **q** podem ser descartados.

2 – Encriptação de uma mensagem **"m":**

- O transmissor recebe a chave pública **(n,e)** do receptor de forma não secreta.
- Divide-se o texto simples (considerado como um *string* de bits) em blocos de tamanho fixo **(m1, m2...)**, que serão encriptados separadamente.
- De posse da chave, ele forma o cipertexto **"c"** calculando para cada bloco $\mathbf{m_n}$ o valor $\mathbf{c = m^e \bmod n}$

3 – Desencriptação da mensagem ***"c"***:

- O receptor toma **"c"**, a divide em seus blocos e calcula $\mathbf{m = c^d \bmod n}$, reconstruindo a mensagem original.

A segurança do RSA se baseia na dificuldade de encontrar-se a chave privada de alguma forma. Conforme Coutinho [13] e a própria RSA [15], qualquer tentativa de obtenção da chave privada equivaleria a fatorar **n**, o que atualmente é inviável pela falta de algoritmos de fatoração eficientes. Deve-se porém considerar a necessidade de geração de números primos grandes o suficiente, conforme citado inclusive no protocolo. Chaves de 155 bits já foram quebradas em 1999 através de computação distribuída, em aproximadamente 5,2 meses de trabalho [16], a própria RSA lançou um desafio para a quebra de chaves 576 bits e 2048 bits, aberto à comunidade internacional, como forma de verificar a segurança proporcionada por estas chaves.

2.7 Algoritmos de Resumo de Mensagens (hash)

Os algoritmos de resumo de mensagem, ou de *hash*, são utilizados para gerar um resumo numérico correspondente a uma mensagem qualquer, que deve ser único e que pode ser utilizado para verificação de alteração de seu conteúdo. Tais algoritmos funcionam através da aplicação de um processo matemático que gera um valor numérico de tamanho geralmente fixo, único, dependendo da entrada. Os algoritmos utilizados no SSL são o *Secure Hash Algorithm –1* (SHA-1) e o *Message Digest 5* (MD5).

2.7.1 Secure Hash Algorithm – 1 (SHA – 1)

O *Secure Hash Algorithm*, SHA-1, é um algoritmo definido na *Federal Information Publications* 180-1 para ser utilizado no padrão DSS (*Digital Signature Standard*) [17]. Quando uma mensagem qualquer de tamanho menor que 2^{64} bits é introduzida no algoritmo, este gera uma saída de 160 bits chamada *message digest* (resumo da mensagem). Esse resumo pode ser incorporado à mensagem original e transmitido num bloco, formando assim uma mensagem assinada. Quando essa mensagem chega ao receptor, este retira a assinatura do bloco, reaplica nessa o algoritmo e compara o resultado com a assinatura recebida. Caso estas sejam iguais, pode-se garantir que a mensagem não sofreu alterações no seu trajeto.

2.7.2 O Algoritmo Message Digest 5 (MD5)

O Message Digest 5 ou MD5 é um algoritmo definido na *Request for Comments 1321* para ser utilizado no padrão DSS (*Digital Signature Standard*). Assim como o SHA, a função deste algoritmo é gerar um resumo de uma mensagem de tamanho fixo, que seja único e inviolável. Quando uma mensagem de qualquer comprimento é introduzida no algoritmo, este gera uma saída de 128 bits correspondente à *Message Digest* [18]. O MD5 foi desenvolvido pelo professor Ronald Rivest do *MIT Laboratory for Computer Science* e *RSA Data Security*.

2.8 Assinaturas Digitais

A assinatura digital é um valor numérico que somente pode ser gerado pelo emitente de um documento, incorporado no corpo do mesmo e que garante sua autenticidade. As assinaturas digitais devem ter as mesmas propriedades que as que são assinadas à mão, ou seja, devem ser únicas, facilmente autenticáveis, não repudiáveis, baratas e fáceis de gerar.

Um algoritmo de assinatura é um algoritmo que transforma uma mensagem de qualquer comprimento e uma chave privada em uma assinatura. Como tal, a assinatura digital permite que qualquer pessoa que leia uma mensagem se certifique que ela realmente foi assinada pelo seu originador (garantia de origem) e de que a mensagem não foi modificada (garantia de integridade). Além disso, assinaturas digitais não podem ser repudiadas, isto é, o originador da assinatura não pode, após assiná-la, deixar de cumprir as obrigações determinadas pela mesma utilizando a afirmação de que esta teria sido falsificada, uma vez que somente este possui a chave necessária para criá-la. [3] As Figuras 2.3(a) e (b) mostram o esquema geral de assinatura digital.

A assinatura digital baseada em chave pública utiliza um algoritmo criptográfico de chave pública para assinar a mensagem transmitida. Isso é possível pelo fato de que alguns algoritmos de chave pública são bidirecionais, ou seja, suas chaves poderem ser aplicadas em qualquer ordem. Uma assinatura digital de chave pública funciona da seguinte forma:

- O transmissor realiza uma algoritmo envolvendo simultaneamente sua chave secreta e a mensagem propriamente dita, transmitindo a mensagem e o seu resultado ao receptor.
- O receptor, para verificar a assinatura, realiza um segundo algoritmo envolvendo a mensagem recebida, a assinatura e a chave pública do transmissor. Se o resultado do

algoritmo aplicado tiver um valor esperado, pode-se considerar que a assinatura é verdadeira, caso contrário, a mesma será falsa.

Os padrões de assinatura digital utilizados pelo SSL são o Digital Signature Standard (DSS) definido pelo NIST, e o RSA, da RSA Security [6].

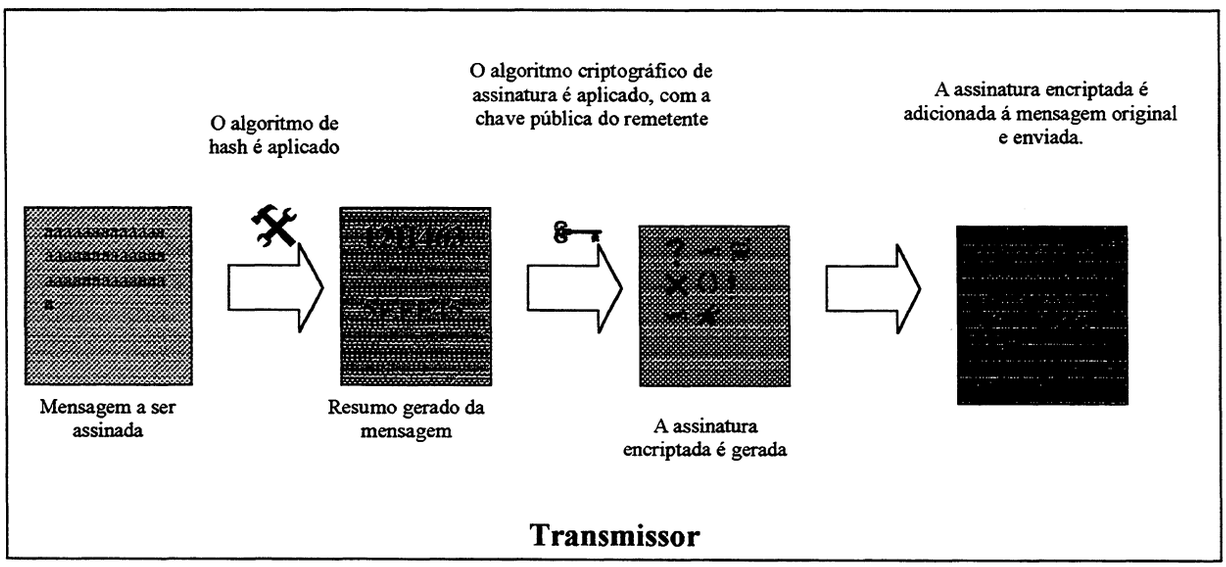


FIGURA 2.3 (a) - Esquema de Assinatura Digital - Transmissão

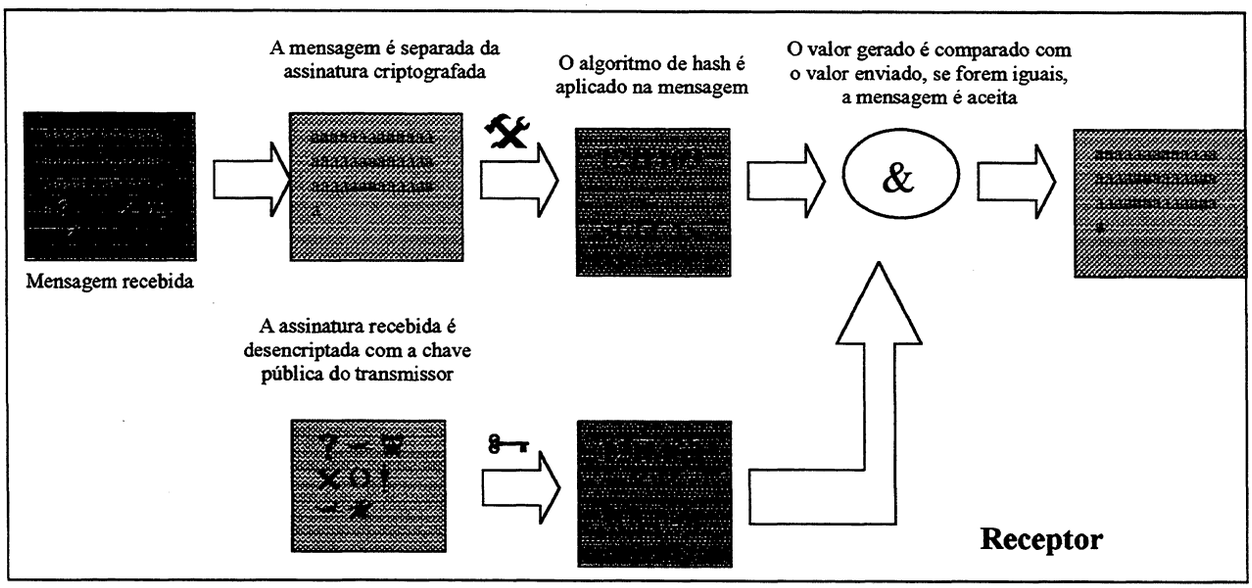


FIGURA 2.3 (b) - Esquema de Assinatura Digital - Recepção

2.8.1 Assinatura Através do RSA

O algoritmo RSA pode ser utilizado para a geração e verificação de assinaturas digitais, uma vez que o mesmo possui a propriedade de ser bidirecional, ou seja, uma mensagem pode ser encriptada com a chave privada e desencriptada com a chave pública.

Para assinarmos um documento digital com o RSA procedemos da seguinte forma [19]:

1 – Geração das chaves

O processo de geração das chaves criptográficas é exatamente o mesmo do método de criptografia RSA visto no item 2.6.2.

2 – Assinatura do documento

Se o transmissor deseja enviar um documento assinado ***M*** para um receptor, ele cria a assinatura ***S*** fazendo a operação $\mathbf{S = M^{d} \bmod n}$, e envia então **S** e **M** para o receptor.

3 – Verificação da assinatura

Para verificar a assinatura, o receptor realiza a operação $\mathbf{S^{e} \bmod n}$. Se o resultado for **M** a mensagem é autêntica.

Antes da mensagem **M** ser assinada, geralmente essa é passada por um algoritmo de ***hash***, que gerará um resumo da mesma e será adicionado a ela. Dessa forma, mesmo se a mensagem original sofrer modificações no meio do caminho, isto pode ser descoberto com a verificação desse ***hash***, aumentando bastante o nível de segurança do esquema.

Vale notar que apenas o remetente é capaz de assinar a mensagem corretamente, uma vez que somente este possui o valor secreto **d**, o que garante ao receptor que a mensagem que este recebeu é realmente proveniente do transmissor que a assinou.

A possibilidade de gerarmos uma assinatura RSA falsa é tão remota quanto a de quebrarmos uma chave RSA, uma vez que para gerarmos tal assinatura deveremos possuir uma chave privada igual à pertencente ao transmissor.

Todas as considerações de segurança já mencionadas anteriormente para o algoritmo RSA de chave pública também valem neste caso, devendo portanto ser consideradas.

2.8.2 DSS - Digital Signature Standard

Em agosto de 1991 o NIST, guiado pela National Security Agency, lançou a proposta de um padrão de assinatura digital, o Digital Signature Standard (DSS). Este foi aprovado pela Secretaria do Comércio Americana como padrão para utilização em aplicações de comércio eletrônico, e publicado na Federal Information Processing Standard Publications 186-2 (FIPS PUBS186-2). Sua função é especificar algoritmos apropriados para aplicações que requerem uma assinatura digital. Para isso o mesmo utiliza um algoritmo específico, chamado Digital Signature Algorithm (DSA) [20].

O DSA é um algoritmo desenvolvido específicamente para ser aplicado no DSS. Utiliza os seguintes parâmetros no seu processo:

- **p** = um módulo primo, onde $2^{L-1} < p < 2^{L}$ para $512 < L < 1024$ e **L** multiplo de 64.
- **q** = um primo divisor de **p-1**, onde $2^{159} < q < 2^{160}$.
- $g = h^{(p-1)/q} \bmod p$, onde **h** é qualquer inteiro positivo menor que **p** tal que $h^{(p-1)/q} \bmod p > 1$.
- **x** = um inteiro gerado por função geradora randômica ou pseudorandômica com $0<x<q$.
- $y = g^{x} \bmod p$
- **k** = um inteiro gerado por função geradora randômica com $0<k<(p-1)$.
- **H** = uma função de hashing unidirecional.

Os inteiros **p,q** e **g** podem ser públicos e comuns a um grupo de usuários. A chave privada e a chave pública são **x** e **y**, respectivamente. Os valores de **x** e **k** precisam ser mantidos secretos e **k** deve ser trocado a cada assinatura.

A assinatura de uma mensagem **M** é o par de números **f** e **s**, computados de acordo com as equações abaixo:

$$\mathbf{r = (g^k \bmod p) \bmod q}$$

$$\mathbf{s = (k^{-1} (SHA\text{-}1(m) + xr)) \bmod q.}$$

O valor **k-1** é o inverso multiplicativo de **k mod q** , isto é:

$$\mathbf{(k\text{-}1)k \bmod q = 1 \ , \ e \ 0 < k\text{-}1 < q}$$

O valor de SHA-1 é uma string de 160-bits resultante do Secure Hash Algorithm, que para ser usado computacionalmente deverá ser transformado em inteiro.

Os valores ***r*** e ***s*** são a assinatura da mensagem, sendo transmitidos junto com **M**.

Antes de verificar a mensagem assinada, o receptor primeiro verifica se:

$$\mathbf{0 < r' < q, \ e}$$

$$\mathbf{0 < s' < q}$$

Se alguma dessas condições foi violada, a assinatura é rejeitada. Se a assinatura for aprovada, o receptor calcula:

$$\mathbf{w = (s')^{-1} \bmod q}$$

$$\mathbf{u1 = ((H(m'))w) \bmod q}$$

$$\mathbf{u2 = ((r')w) \bmod q}$$

$$\mathbf{v = (((g)u1 \ (y)u2 \ ) \bmod p) \bmod q}$$

Se $\mathbf{v} = \mathbf{r}'$, a assinatura foi verificada e o receptor pode ter uma confiança alta de que a mensagem recebida foi enviada pela entidade que possui a chave privada **x** correspondente a **y**. Caso contrário, a mensagem foi comprometida e deve ser considerada inválida.

2.9 Certificados Digitais

Os certificados digitais são conjuntos de informações de identificação, resumidas e assinadas criptograficamente, que são divulgadas na Internet como uma forma de comprovação de identidade daqueles que o possuem. Para que os certificados digitais possam ser utilizados é necessária também a existência de uma série de órgãos de certificação, reconhecidos legal e publicamente, que possuam técnicas adequadas para a criação e distribuição dos mesmos.

O SSL utiliza o padrão X.509 versões 2 e 3 de certificados digitais como ferramenta para identificação digital. Todos os órgãos de certificação que trabalham com o SSL devem seguir o formato do X.509 para que seus certificados possam ser aceitos em servidores ou navegadores que utilizam o protocolo.

2.9.1 Padrão X.509 de Certificados Digitais

O X.509 é um padrão utilizado para a geração de certificados digitais, inicialmente publicado em 1988 como parte do diretório X.500. A norma descreve o formato dos certificados digitais X.509, sua sintaxe e aplicabilidade [21].

O certificado é constituído por três campos distintos, o campo **tbsCertificate**, que contém elementos gerais de identificação e informação do certificado, o campo

signatureAlgorithm, que define o algoritmo de assinatura utilizado pelo certificado e o campo **signatureValue**, um valor numérico correspondente à assinatura do certificado.

O campo **signatureAlgorithm** contém um valor identificador do algoritmo utilizado no certificado. O certificado X.509 aceita como algoritmos de assinatura o RSA e o DSS, com algoritmos de hash MD5 e SHA-1.

O campo **signatureValue** contém a sequência numérica correspondente à assinatura gerada através desse algoritmo.

O campo **tbsCertificate** é o mais complexo e contém todas as informações necessárias à identificação do certificado e sua verificação. Seus subcampos são os seguintes:

Version – Versão do certificado. Atualmente a versão mais recente do padrão X.509 é a de número 3.

Serial number – Número serial do certificado, atribuído pela autoridade certificadora.

Signature – Identificador do algoritmo de assinatura utilizado no certificado.

Issuer – Identificador da entidade que emitiu o certificado.

Validity – Período de validade do certificado, com data inicial e final.

Subject – Identifica a entidade associada à chave pública armazenada no certificado.

Subject Public Key Info – Contém a chave pública e a identificação do algoritmo de verificação da mesma, atribuída à entidade portadora do certificado.

Unique Identifiers – Este campo contém a identificação única do emissor e do portador, com a função de evitar que uma empresa possa ter vários certificados independentes.

Extensions – Este campo está presente apenas na versão 3. Descreve uma série de informações adicionais possíveis para o certificado, com o intuito de prover uma flexibilidade de uso maior para o padrão do certificado. Nesse campo podem estar presentes informações

particulares como nomes alternativos para o usuário do certificado, período de validade das chaves, entre outras.

O valor do **signatureValue** é gerado através da realização de um hash sobre os valores presentes no campo **tbsCertificate**, seguido de uma posterior aplicação de um algoritmo de assinatura. Essa assinatura garante a integridade e autenticidade das informações presentes no certificado, que caso tenham sidas alteradas, gerarão um erro na mesma o que invalidará o certificado no momento de sua verificação pelo receptor.

As figuras 2.4 e 2.5 mostram os detalhes de um certificado recebido apresentados na tela do Internet Explorer®, com seus campos definidos.

As informações descritas na tela inicial são as mais simples, como a entidade portadora do certificado, sua validade e a entidade certificadora que o emitiu. Podemos verificar que este é um certificado emitido pela Thawte Server CA para a RSA Security, em 05/10/01 e válido até 5/10/02, ou seja, por um ano. Caso o usuário necessite de informações adicionais, este pode clicar em detalhes, conforme a figura 2.5

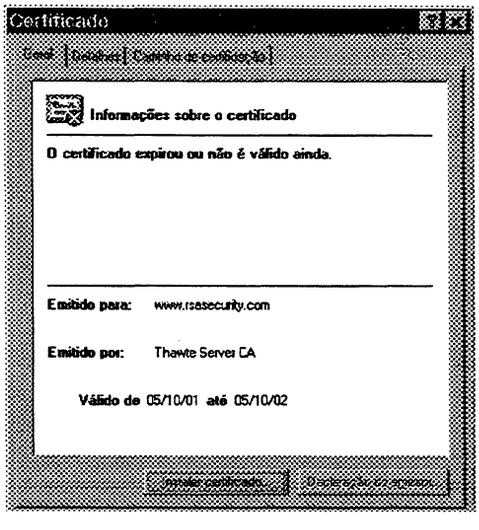
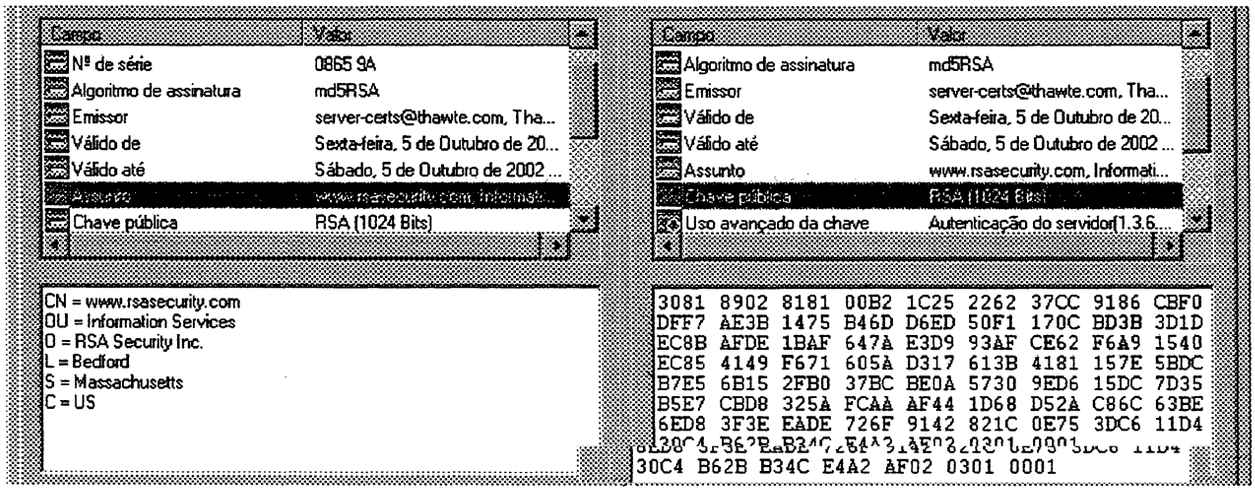


FIGURA 2.4 - Exemplo de Certificado Digital



(a) (b)

FIGURA 2.5 - Detalhes de certificados digitais:
(a) Informações sobre o Retentor do Certificado
(b) Detalhe da Chave Pública do Certificado.

Conforme podemos verificar, o certificado contém as informações da empresa para o qual foi emitido. Na figura 2.5 (b) podemos encontrar o valor da chave pública para verificação de assinatura, que utiliza o RSA em conjunto com o MD5. Ao receber este certificado, o computador utiliza esta chave para verificar a validade do mesmo, recalculando o valor do *hash* MD5 aplicado aos campos descritivos do certificado e comparando o resultado com o valor resultante do algoritmo de assinatura.

2.9.2 Atribuição de Certificados e Órgãos de Certificação

Para que seja possível a existência de um mecanismo de certificação digital, é necessário que exista um organismo responsável pela implementação das técnicas de criação, manejo e distribuição dos mesmos. Estes organismos são chamados Órgãos de Certificação. Atualmente algumas empresas realizam tal serviço, mas a discussão legal sobre sua validade

ainda está no início, de forma que não existe uma padronização dos serviços prestados ou normalização dos métodos utilizados.

O padrão X.509 não faz qualquer exigência sobre formas de implementação do sistema de certificação, exceto no caso das listas de revogação, deixando uma grande abertura para uma variedade de certificados e entidades de certificação. Fica sob responsabilidade do aplicativo a verificação e aceitação de cada tipo de certificado em questão. A figura abaixo demonstra a árvore de certificação de um certificado emitido para a RSA. A Autoridade Certificadora da RSA é a empresa chamada Thawte Server CA. A Figura 2.6 mostra um exemplo de caminho de certificação.

FIGURA 2.6 - Exemplo de Caminho ou Árvore de Certificação

2.9.3 Listas de Revogação de Certificados

O padrão X.509 define os parâmetros básicos para a implementação das listas de revogação de certificados, necessárias para informar os usuários sobre revogações da validade de algum destes. Mais uma vez porém, o padrão deixa sob responsabilidade do órgão de certificação o manejo de tais listas, o que cria uma diversidade de formas de implementação. Em todos os casos porém, deverão existir as listas de revogação, colocadas em locais padrão,

com redundância e com atualização periódica, de forma que um usuário possa verificar a validade de um certificado em questão através do acesso à tais listas [21].

2.10 Considerações Finais do Capítulo

Os mecanismos de criptografia, resumo, autenticação e assinatura de mensagens fornecem uma base técnica bastante forte e viável para a criação de sistemas de comunicação segura via Internet, constituindo atualmente da forma mais eficaz existente para este fim. Pelo fato de serem de simples aplicação e economicamente viáveis, estes mecanismos vêm se estabelecendo como o grande foco das atenções dos desenvolvedores de sistemas computacionais. Apesar da existência de mecanismos mais robustos, como por exemplo a biometria para identificação pessoal, estes ainda esbarram em aspectos técnico-econômicos que devem garantir que a utilização das técnicas anteriormente descritas deverá perdurar por algum tempo, de forma prática, principalmente em sistemas comerciais via rede.

Capítulo 3 – SSL - Secure Sockets Layer

3.1 Introdução

O Secure Sockets Layer é um protocolo de comunicação que implementa um túnel seguro para comunicação de aplicações na Internet, de forma transparente e independente da plataforma. Foi desenvolvido pela Netscape Communications em sua versão inicial em julho de 1994. Em abril de 1995 foi lançada a referência para implementação da versão 2, sendo distribuído junto os Browsers Netscape Navigator® e Internet Explorer® e os servidores Web mais comuns - Apache, NCSA httpd, IIS, Netscape Server etc, apoiado pela Verisign e Sun Microsistens, e transformando-se em um padrão em e-commerce. Sua publicação foi realizada na Internet-Draft 302 [6].

O objetivo primário do protocolo é prover privacidade e autenticação entre duas aplicações que se comunicam entre si. Os objetivos gerais podem ser descritos como abaixo: [6]

- Prover segurança criptográfica para estabelecer uma conexão segura entre dois pontos.
- Prover interoperabilidade de forma que programadores independentes sejam aptos a desenvolver aplicações capazes de trocar dados com outros códigos que também o utilizem.
- Extensibilidade, ou seja, possibilitar de forma flexível a implementação de novos métodos criptográficos ou protocolos sem a necessidade de mudança das suas bases estruturais.

- Relativa eficiência, implementando recursos para reduzir o número de processos de hardware bem como o número de conexões.

O SSL foi desenvolvido de forma a maximizar a flexibilidade de implementação, ou seja, poder comunicar os *softwares* que o utilizem independentemente da plataforma computacional utilizada. Além disso, sua aplicabilidade é bastante extensa, podendo servir para o desenvolvimento de quaisquer tipo de sistema de comunicação segura, como serviços de comércio eletrônico, e-mail seguro, Virtual Private Networks (VPN's), entre outros.

3.2 Funcionamento do SSL

O protocolo funciona criando um túnel de comunicação segura entre dois pontos através de uma rede insegura, seguindo o modelo cliente servidor. A comunicação pode ser realizada entre qualquer cliente e servidor SSL sem a necessidade prévia da existência de qualquer tipo de autenticação, uma vez que isto é realizado no início da conexão, através do chamado "*handshake*". Cada nova conexão exige a troca dos parâmetros necessários para a comunicação segura entre as máquinas, além da autenticação dos usuários.

O cliente inicia a conexão através de uma requisição ao servidor. Nesta mensagem de requisição, o cliente indica ao servidor quais são os algoritmos criptográficos que está apto a utilizar para a comunicação segura.

O servidor responde com uma mensagem de conexão aceita, onde ele define quais parâmetros criptográficos foram escolhidos para a comunicação, dentre aqueles que o cliente indicou.

O cliente e o servidor trocam as chaves criptográficas simétricas que serão utilizadas na comunicação, como auxílio de um algoritmo de criptografia assimétrica.

A comunicação protegida começa neste momento. Todas as mensagens daí por diante serão criptografadas com a chave simétrica escolhida pelos dois comunicantes no momento do *handshake*.

3.3 SSL - Descrição Geral

O SSL é um protocolo implementado em duas camadas distintas, a *Handshake Layer* e a *Record Layer*. Para enviar uma mensagem, o *Record Protocol,* situado na *Record Layer* toma-a, fragmenta-a em blocos, comprime a informação, aplica um código de autenticação, criptografa-a e transmite o resultado. A mensagem recebida é descriptografada, verificada, descomprimida e recomposta, sendo repassada para a camada de aplicação. A camada de *Handshake* é responsável por realizar a troca de parâmetros criptográficos no início da sessão, além de realizar manipulação e informação de eventos de erro.

O SSL atua entre as camadas transporte (TCP) e aplicação, sendo independente do protocolo de alto nível, podendo rodar sob HTTP, Telnet, FTP, SMTP, entre outras, de forma relativamente transparente. Tal transparência é relativa por que o protocolo pode gerar uma diminuição na velocidade de comunicação e, em alguns casos, pode causar falhas na conexão. A figura 3.1 mostra o posicionamento do SSL entre as camadas.

O SSL é formado por duas camadas com protocolos diferentes, a superior, chamada ***Handshake Layer***, onde se encontra o ***Handshake protocol***, o ***Alert protocol*** e o ***Change Cipher Spec protocol*** e a inferior, a ***Record Layer***, onde se encontra o ***Record protocol***.

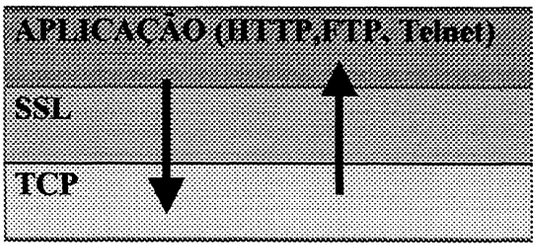


FIGURA 3.1 - Localização do SSL nas Camadas

A função do ***Handshake protocol*** é iniciar a negociação dos parâmetros necessários para que a comunicação segura possa ocorrer, como chaves criptográficas e MAC's (*Message Autentication Codes*).

O ***Alert protocol*** é responsável pela troca de informações de alerta entre os comunicantes, no caso de algum tipo de erro.

O ***Change Cipher Spec protocol*** é utilizado para mudar o estado da sessão quando os parâmetros criptográficos foram negociados, ou seja, quando a conexão já possui os parâmetros necessários para se tornar segura. É composto de apenas uma mensagem, a ***Change Cipher Spec.***

O ***Record protocol*** tem a função de gerar todos os blocos de informação protegidos e transmiti-los através da conexão através dos parâmetros criptográficos previamente definidos. Os outros protocolos são clientes do ***Record protocol***, ou seja, este é utilizado para o encapsulamento das mensagens geradas pelos protocolos superiores durante toda conexão.

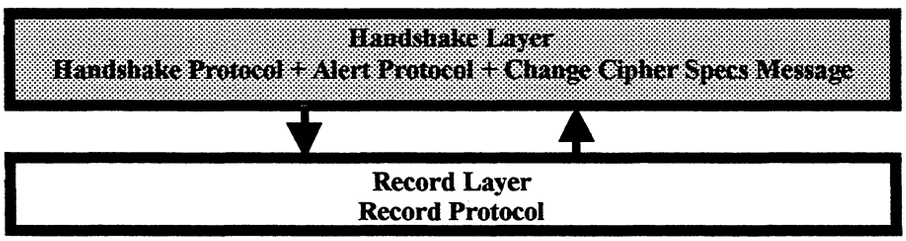


FIGURA 3.2 - Camadas do SSL

3.3.1. Handshake e Estabelecimento de uma Sessão Criptográfica

A comunicação é iniciada pelo estabelecimento de uma sessão. Chamamos de sessão a associação entre o cliente e o servidor, com todos os parâmetros relativos à comunicação já definidos e ativos. A sessão inicial possui todos os seus atributos com valores iniciais padrões, geralmente nulos, e assim que os parâmetros criptográficos são negociados entre cliente e servidor, a sessão é mudada para uma sessão segura através de uma mensagem chamada ***Change Cipher Spec***. Para iniciar uma sessão, o ***Handshake protocol*** troca mensagens de inicialização entre cliente e servidor definindo os parâmetros necessários para o início da comunicação segura, como suíte criptográfica, MAC's, algoritmos de compressão etc. Uma vez que o *handshake* esteja completo e ambas as partes possuam os segredos criptográficos necessários, o ***Record protocol*** pode iniciar a transmissão das mensagens provenientes dos aplicativos, devidamente criptografadas.

A figura 3.3 demonstra de forma mais clara a troca de mensagens entre cliente e servidor desde o *handshake* até o momento em que os pacotes vindos da camada de aplicação são enviados. As mensagens trocadas são as seguintes:

Client Hello - Primeira mensagem enviada pelo cliente ao servidor, com função de iniciar a conexão e gerar os parâmetros iniciais de segurança. Nesta mensagem o cliente envia ao servidor uma série de suítes criptográficas às quais ele está apto a manipular.

A tabela 3.1 mostra três das possíveis combinações de suítes criptográficas com as quais o cliente pode estar apto a trabalhar durante uma conexão. Tais combinações dependerão de quais algoritmos criptográficos o cliente possuir em seu aplicativo. Geralmente o cliente define tais combinações de forma a colocar as suítes mais "fortes" no início, pois o servidor deverá escolher obrigatoriamente a primeira que estiver capacitado a manipular. No

caso da última suíte, onde todos os parâmetros estão definidos como **Null**, não existirá qualquer tipo de proteção à comunicação.

- **Server Hello** - Mensagem enviada ao cliente em resposta ao **Client Hello**. Nesta mensagem, o servidor envia ao cliente a definição da suíte criptográfica escolhida para a comunicação, dentre as enviadas na mensagem **Client Hello**.

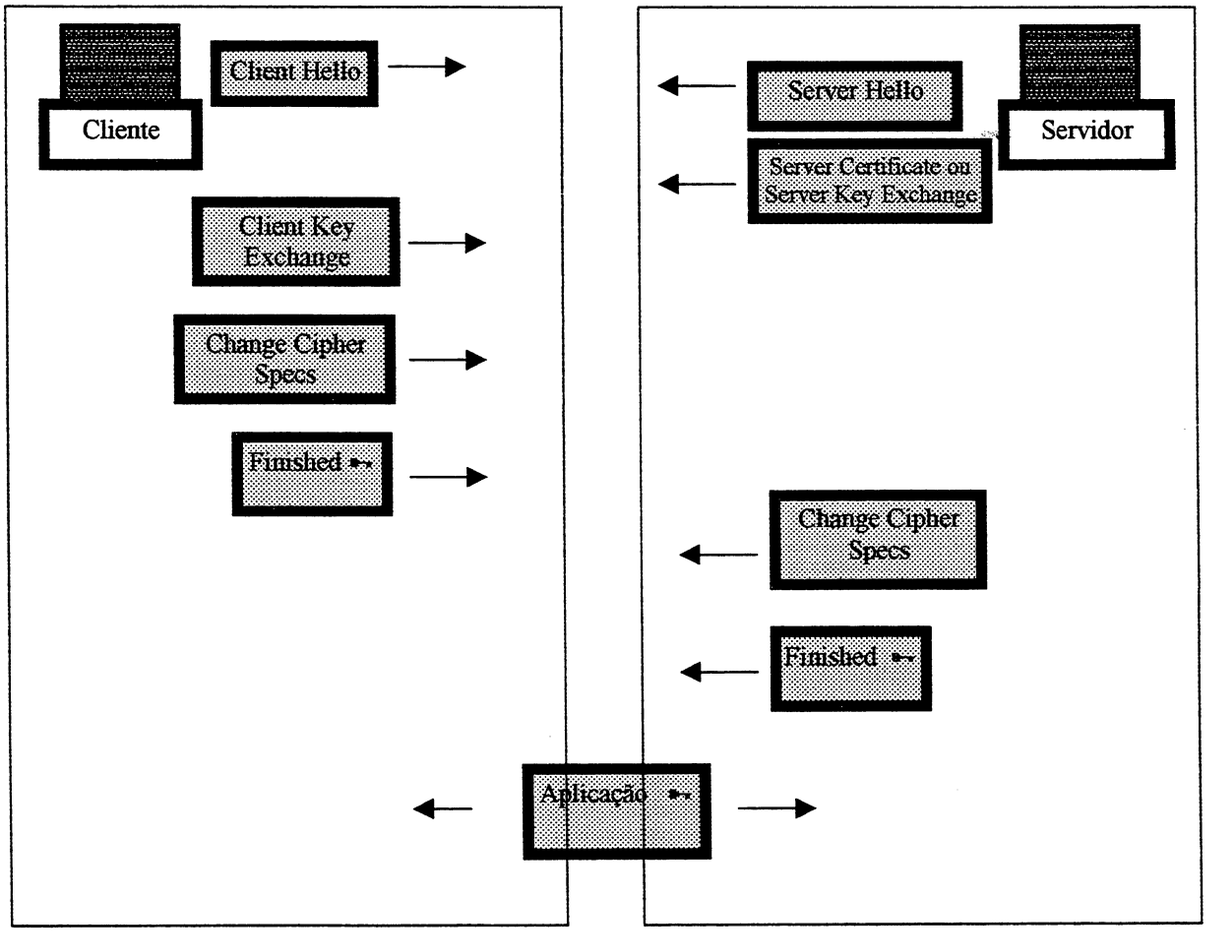


FIGURA 3.3 - Exemplo de Comunicação Através do SSL

Algoritmo de Troca de Chaves	Algoritmo de Assinatura Digital	Algoritmo de Chave Simétrica	Algoritmo de Hash
RSA	**RSA**	**DES 40**	**MD5**
DHKE	**DSS**	**RC4**	**SHA**
Null	**Null**	**Null**	**Null**

TABELA 3.1- Exemplo de Suítes Criptográficas

Algoritmo de Troca de Chaves	Algoritmo de Assinatura Digital	Algoritmo de Chave Simétrica	Algoritmo de Hash
RSA	RSA	DES 40	MD5

TABELA 3.2 - Exemplo de Suíte Criptográfica Escolhida para a Comunicação

- **Server Certificate** - Após o servidor mandar o **Server Hello** ele imediatamente envia o **Server Certificate**, que tem a função de passar os parâmetros do seu certificado digital para o cliente. No caso de um servidor sem certificado ou uma conexão anônima, esta mensagem não será enviada. Caso este certificado possua parâmetros para troca de chaves simétricas através de um algoritmo de chave pública, tais valores serão utilizados para que o cliente e o servidor possam negociar, através da mensagem **Client Key Exchange**, valores que serão utilizados por ambos para a criação das chaves privadas da sessão criptográfica. Neste caso a mensagem seguinte não será enviada.

- **Server Key Exchange** - Essa mensagem será enviada pelo servidor ao cliente com o intuito fornecer ao mesmo uma chave assimétrica que possa ser utilizada para a troca das chaves simétricas que deverão ser utilizadas na sessão. Esta mensagem somente será enviada no caso do certificado não possuir os parâmetros criptográficos necessários para a geração da chave pública ou no caso de uma conexão anônima.

- **Certificate Request** - Tem a função de requisitar ao cliente um certificado para a conexão. Esta mensagem só será enviada em caso de conexões onde o servidor exija um certificado de cliente para estabelecer uma sessão. Esta mensagem deverá ser respondida pelo cliente com uma mensagem de **Client Certificate**.

- **Server Hello Done** - Esta mensagem será enviada pelo servidor para indicar para o cliente que as mensagens de *Hello* terminaram. Após o envio dessa mensagem, o servidor aguarda a resposta do cliente.

- **Client Key Exchange** - Será enviada pelo cliente logo após o recebimento da **Server Hello Done**, e tem a função de transmitir o **premaster_secret** criptografado com a chave pública enviada pelo servidor. O **premaster_secret** é um valor utilizado para gerar as chaves secretas da sessão.

- **Client Change Cipher Spec** - Apesar de não ser uma mensagem do **Handshake Protocol**, esta mensagem normalmente é enviada durante o *handshake* indicando que o cliente passará a utilizar os parâmetros criptográficos recém-gerados para proteger as mensagens enviadas a partir desse momento. Ao receber essa mensagem o servidor também deve mudar os parâmetros de sua conexão e enviar sua **Server Change Cipher Spec** para o cliente.

- **Client Finished** - Imediatamente após o cliente enviar a sua **Change Cipher Spec**, ele enviará uma mensagem **Finished**. Essa mensagem já será criptografada com a chave secreta da sessão e será protegida com todos os outros elementos de segurança gerados. Após receber esta mensagem, o servidor saberá que o cliente não enviará mais nenhuma mensagem relacionada com o *handshake*, e que a partir desse momento ele passará a enviar os dados provenientes da camada de aplicação, protegidos por criptografia.

- **Server Change Cipher Spec** - Esta mensagem é enviada pelo servidor indicando que ele também passará a utilizar os parâmetros criptográficos recém-gerados para proteger as mensagens enviadas a partir desse momento.

- **Server Hello Finished** - O servidor também deverá enviar sua **Server Finished** da mesma forma que o cliente.

Qualquer erro ocorrido durante o *handshake* invalida automaticamente a conexão, que deverá ser iniciada novamente. Após as mensagens de *handshake*, o cliente e o servidor passam a trocar as mensagens criptografadas, conforme descrição a seguir.

3.3.2 Troca de Mensagens Através do Record Protocol

A *SSL Record Layer* é a camada responsável pela transmissão dos blocos de informação, chamados de *SSL Plaintext Records* (ou Registro de Textos SSL), gerados nas camadas superiores. A *SSL Record Layer* recebe informações não interpretadas das camadas superiores, fragmenta-as, comprime-as e as encripta, aplicando um MAC no resultado, realizando a operação inversa quando recebe uma informação da camada de transporte. Esta camada possuí um campo para a identificação do tipo da mensagem enviada e não preserva os limites das mensagens do mesmo tipo, concatenando-as quando necessário. É responsabilidade das camadas superiores restabelecerem esses limites.

A quantidade máxima de informação transmitida num registro deve ser 2^{14} bytes, sendo que qualquer valor superior a este gerará um erro fatal. As operações geradas nesta camada são as seguintes:

- **Fragmentação** - O *SSL Record Protocol* recebe as informações provenientes das camadas superiores e as fragmenta montando os chamados *SSL Plaintext Records*, de tamanho máximo igual a 2^{14} bytes, com as informações necessárias para sua correta interpretação. Cada pedaço de informação da camada de aplicação fracionado é chamado de *SSL Plaintext*. As informações contidas em um *SSL Plaintext* são:

type: O tipo de informação enviado no campo de dados de aplicação.

version: Versão do protocolo SSL utilizado (SSL 2.0 ou SSL 3.0)

length: O tamanho do fragmento de dados transmitido, que não deve ultrapassar 2^{14} bytes.

fragment: O fragmento da informação em questão.

- **Compressão e Descompressão dos Dados Transmitidos** - Todos os Registros são comprimidos utilizando o algoritmo de compressão corrente, definido no *handshake*, antes de serem criptografados. Sempre existe um algoritmo definido para a compressão, mesmo que este seja do tipo *NULL*, onde nenhuma modificação é realizada. O algoritmo de compressão transforma o *SSLPlaintext.Fragment* em um *SSLCompressed.Fragment*. Este algoritmo não age sobre os campos de informação da mensagem, apenas sobre seu conteúdo, ou seja, o *fragment*. A compressão deve ser sem perdas, e o tamanho total do registro após a compressão não deve ser maior que 2^{14} bytes. O protocolo não define formalmente os algoritmos de compressão a serem utilizados, ficando sua escolha a critério do aplicativo. Atualmente a compressão dos dados não vem sendo utilizada na prática pelos aplicativos comuns que utilizam o SSL [10].
- **Aplicação de Algoritmo de Autenticação (MAC)** - Todo registro é protegido através da aplicação de um algoritmo de MAC, que deve ser definido na *Cipher Suíte*. Este algoritmo é aplicado antes da encriptação da mensagem, sobre todo o registro, inclusive sobre os campos de comprimento e tipo da do *SSLCompressed.Fragment*. O valor gerado pelo MAC é adicionado à mensagem.
- **Encriptação da Mensagem** - A encriptação da mensagem será feita sobre todo o bloco de informações, incluindo o MAC, mas excetuando os campos de tipo e comprimento. O algoritmo transforma o *SSLCompressed.Fragment* em um *SSLCiphertext.Fragment*. O

algoritmo de encriptação é definido na *Cipher Suíte* e pode ser do tipo *Stream Cipher* ou *Block Cipher*.

3.3.3 Mensagens de Alerta

As mensagens de alerta são criadas e manipuladas pelo *Alert Protocol. O Alert Protocol* tem três tipos de mensagens, mensagens de erro fatal, de erro não fatal e de encerramento. A forma que o protocolo utiliza para gerenciar essas mensagens é simples. No caso de uma mensagem de erro fatal e de uma mensagem de encerramento a conexão deve ser imediatamente desfeita. Já no caso de uma mensagem não fatal um aviso é apresentado ao cliente ou ao servidor, dependendo de quem originou a mesma, cabendo ao aplicativo decidir o que fazer quando recebe uma mensagem desse tipo. A mensagem de encerramento serve para indicar que o cliente ou o servidor estão encerrando a conexão e, quando recebida por qualquer um dos lados, nenhuma outra mensagem posterior deve ser enviada na mesma sessão. As mensagens são as seguintes:

- **close_notify** - Notificação de fim de conexão, fecha imediatamente a conexão
- **unexpected_message** - Mensagem Inesperada (fatal)
- **bad_record_mac** - MAC não confere
- **decompression_failure** - Falha de descompressão
- **handshake_failure** - Falha de *handshake* (fatal)
- **no_certificate** - Sem certificado
- **bad_certificate** - Certificado danificado
- **unsupported_certificate** - Certificado não suportado
- **certificate_revoked** - Certificado revogado

- **certificate_expired** - Certificado expirado
- **certificate_unknown** - Certificado não conhecido
- **illegal_parameter** - Parâmetro ilegal (fatal)

Tanto o *Record protocol* quanto o *Handshake protocol* serão clientes do *Alert protocol* durante a conexão, ou seja, este receberá qualquer ocorrência de erro dos outros protocolos e providenciará o seu correto envio ao outro ponto da conexão.

3.4 Utilização do SSL em Aplicativos Comerciais

Diversos aplicativos atuais possuem funções capazes de realizar conexões através do SSL. Conforme foi comentado anteriormente, servidores como o Apache e navegadores como o Netscape Navigator ® e o Internet Explorer ® já possuem há algum tempo recursos para a realização de conexões SSL. Além disso, muitos aplicativos proprietários são especialmente desenvolvidos para bancos e instituições de comércio eletrônico, através de ferramentas de desenvolvimento específicas.

A conexão SSL é realizada através da portas padrão, como as listadas abaixo, mas os aplicativos podem escolher outras portas específicas para esta conexão.

Protocolo	Porta TCP	Descrição
HTTPs	443	HTTP protegido por SSL
Ssmtp	465	SMTP protegido por SSL
Snews	563	Usenet News protegido por SSL
Spop3	995	Recuperação de Correio protegido por SSL

TABELA 3.3 – Esquema de Portas TCP utilizadas pelo SSL

A conexão sempre será gerada pelo cliente SSL. Existem duas possibilidades para que este inicie uma comunicação protegida:

a) O usuário deseja conectar-se, por exemplo, a um servidor Web através do SSL. Neste caso o usuário pode gerar uma conexão segura trocando, na barra de endereços do navegador, o **http** por **https**, que por padrão é o formato mnemônico que define a utilização do SSL na conexão. Caso o servidor que está sendo contatado seja capaz de realizar uma conexão SSL, este responderá normalmente e as definições do protocolo serão transparentes ao usuário. A figura 3.4 mostra um exemplo de página acessada através deste esquema.

b) A conexão é iniciada pelo usuário cliente ao clicar sobre um hiperlink para uma página segura. Neste caso, o próprio formato HTML da página, (ASP, PHP etc) gerarão a conexão através do SSL também de forma transparente. No caso de formulário CGI, por exemplo, basta substituir o "action = http://www. . .com/cgi-bin" por "action = https://www. . .com/cgi-bin" e o navegador se encarregará do restante da tarefa. A figura 3.5 mostra um exemplo do uso desta técnica.

Estando conectado através do SSL, o navegador exibirá um pequeno cadeado na barra de tarefas, que se receber um clique, exibirá as configurações utilizadas na conexão em questão. A figura 3.6 mostra a indicação de cadeado em uma página do Internet Explorer.

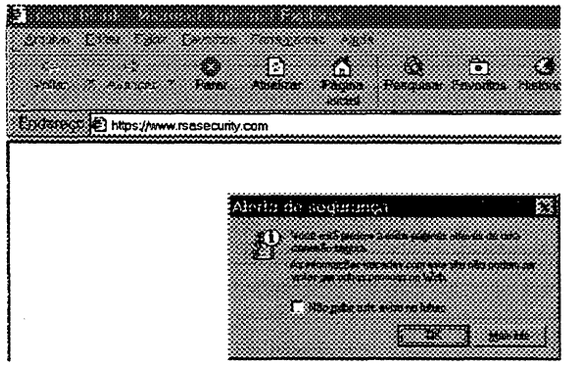


FIGURA 3.4 – Exemplo de Página Acessada Através do SSL

```
</select>
<input type="text" name="HD_PALAVRAS" size="10">
<img src="/imgc/icones/pesqrap2.gif" width="48" height="19" align="absmiddle">
<input type="image" src="/imgc/icones/pesqra.gif" border="0">
  <a href="/megapesq/megapesq.dll/avancada"><img src="/imgc/icones/pesqavanc.
<img src="/imgc/icones/pesqavanc2.gif" width="53" height="19" align="absmiddle">&r
<a href="/scripts/megasite.dll/cesta"><img src="/imgc/icones/suacompra.gif" borde
<img src="/imgc/icones/compra2.gif" width="42" height="19" align="absmiddle">&nbs
<a href="https://www.livrariasaraiva.com.br/megatrak/megatrak.dll"><img src="/img
```

FIGURA 3.5 - Exemplo de Código Fonte de Página com Link SSL

FIGURA 3.6 - Ícone do Cadeado Demonstrando Conexão Segura

3.4.1 A Conexão Vista pelo Lado do Usuário

O SSL é um protocolo complexo na sua forma de trabalho, uma vez que utiliza-se de conceitos de criptografia que são desconhecidos da maioria dos usuários. Provavelmente os maiores problemas de segurança relacionados ao protocolo venham da interação com este usuário. O sistema pode acabar tendo a sua segurança quebrada no momento em que coloca como responsabilidade do mesmo uma decisão técnica. Este é o caso dos certificados digitais no Internet Explorer® e no Netscape Navigator®, onde a recepção de um certificado inválido gera uma mensagem ao usuário que deve decidir se aceita ou não a conexão. Se este for o caso, estará perdendo qualquer garantia de identificação do servidor ao qual está se conectando. A figura 3.7 mostra um exemplo deste esquema.

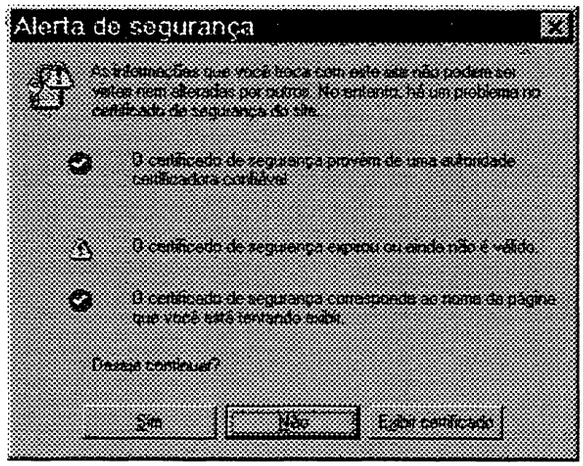


FIGURA 3.7 - Mensagem de Erro do Internet Explorer® para Certificado Inválido

3.4.2 Configurações do SSL em Aplicativos Comerciais e Sistemas Operacionais

Em alguns casos, o usuário pode desejar configurar o SSL em seu sistema de forma personalizada. No caso do sistema operacional Windows®, as configurações de segurança relacionadas ao SSL são tratadas ao nível de aplicativos. Alguns desses aplicativos, como os navegadores Internet Explorer® e Netscape Navigator® permitem configurações deste tipo. Já o sistema operacional LINUX toma um caminho diferente, pois coloca o SSL como componente do sistema, permitindo configurações bem mais detalhadas e que, uma vez realizadas, servirão para todos os aplicativos que façam uso do SSL.

Utilizaremos como exemplo as versões 6.0 do Netscape, 6.1 do IE e 6.0 do Conectiva Linux.

3.4.2.1 Configurações do SSL no Navegador Internet Explorer®

As configurações de segurança relativas ao SSL podem ser realizadas no Internet Explorer®, no menu "Opções da Internet" & "Avançado", como mostra a figura 3.8. Nesta figura estão marcadas as opções padrões do navegador. No caso, os itens que poderemos configurar são:

- Avisar ao alterar o modo de segurança – Aqui podemos escolher se o navegador vai nos apresentar uma mensagem cada vez que entrarmos ou sairmos de uma conexão segura.
- Avisar quando o certificado do site for inválido – Podemos escolher se o navegador vai nos avisar ou não de um certificado inválido.
- Usar Fortezza – Esta opção permite que definamos se o navegador vai utilizar esta técnica de criptografia baseada em hardware.
- Usar SSL 2.0 – Permitir conexões com servidores que ainda utilizem *softwares* baseados no SSL 2.0.
- Usar SSL 3.0 – Permitir conexões com servidores que utilizem *softwares* baseados no SSL 3.0, versão mais atual do protocolo.
- Verificar revogação de certificados de servidor – Aqui podemos definir que o navegador pesquise em uma lista on-line se o certificado recebido foi revogado por algum motivo. Vale a pena notar que o padrão para esta opção é inicialmente desmarcada.

A figura 3.8 mostra uma tela do navegador Internet Explorer® mostrando a opção de verificação de revogação on-line, inicialmente desmarcada.

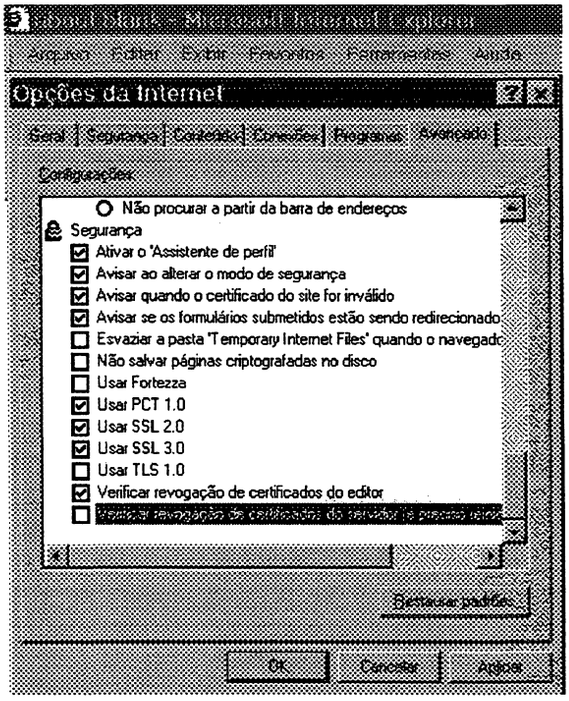


FIGURA 3.8 – Configurações de Segurança no Internet Explorer®

3.4.2.2 Configuração do SSL no Navegador Netscape Navigator® for Windows

No caso do Netscape Navigator® , as configurações possíveis são um pouco maiores, conforme mostrado nas figuras 3.11 e 3.12. Neste caso, as configurações possibilitam as mesmas configurações descritas anteriormente no Internet Explorer® , além de possibilitar ao usuário escolher os algoritmos de criptografia que deseja utilizar.

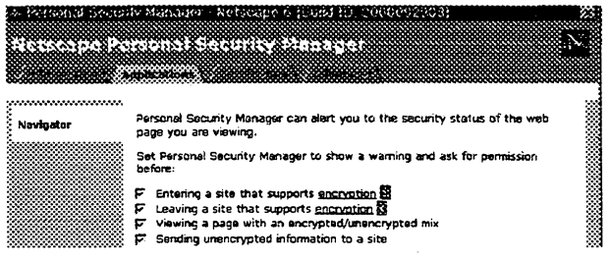


FIGURA 3.9- Configurações do Netscape sobre Entrada e Saída de Sites Protegidos

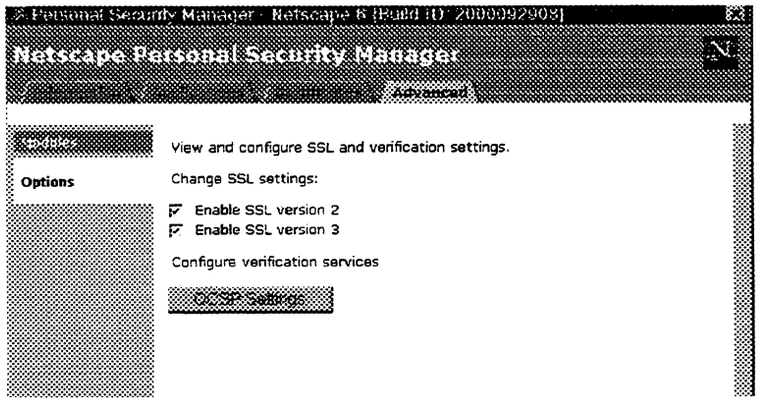


FIGURA 3.10 - Configurações do SSL no Netscape Navigator®

3.4.2.3 Configuração do SSL no Sistema Operacional LINUX

O sistema operacional Linux apresenta uma abordagem mais interessante ao lidar com o protocolo SSL. Basicamente, o sistema permite que o usuário configure todos os itens importantes, desde versão do protocolo até as suítes criptográficas que devem ser utilizadas. Como existem questões de segurança que dependem dessas características mais avançadas, o sistema nos fornece uma oportunidade de personalização muito interessante, tendo em vista os diversos tipos de aplicações que um usuário do sistema pode ter para o protocolo.

Como o protocolo é trabalhado pelo sistema operacional diretamente, a segurança dada pelo Linux na utilização de conexões SSL pode ser muito maior, dependendo das configurações. Outra característica interessante é que o sistema torna-se mais rápido, uma vez que recursos de memória serão alocados mais eficientemente durante sua utilização, bem como o tempo do processador. A figura 3.13 mostra as configurações do Linux Conectiva 6.0, através do centro de controle KDE.

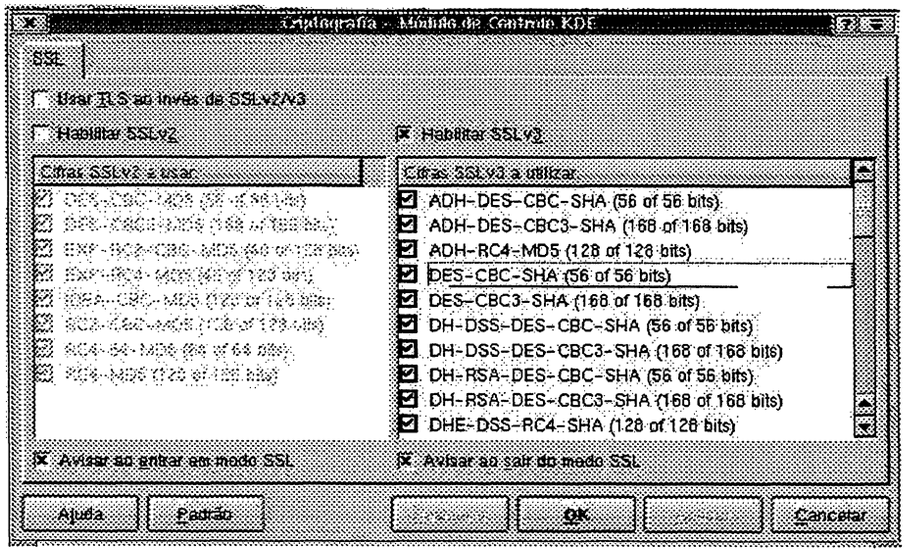


FIGURA 3. 11- Configurações do SSL no Linux

3.4.3 Construção de Sistemas SSL Através de Ferramentas de Desenvolvimento

O desenvolvimento de sistemas aplicativos proprietários que utilizam o SSL seria bastante difícil e complexo para a maioria dos programadores, se estes tivessem que realizá-lo através das ferramentas de programação comuns. Além da dificuldade técnica da programação, a quantidade de conhecimento de assuntos relacionados ao protocolo, como criptografia, assinaturas digitais etc, seria simplesmente enorme, o que causaria um grande desconforto ao programador. Além disso, a quantidade de detalhes de implementação dos algoritmos criptográficos necessários aumentaria bastante a possibilidade de falhas. Dessa forma, algumas empresas como a RSA Security, e também alguns grupos de trabalho independentes, criaram bibliotecas especiais para linguagens como o C ou Java que

implementam as funções realizadas no protocolo. Como exemplo, daremos uma breve descrição do projeto OpenSSL [22], um projeto de fonte aberta (*open source*), desenvolvido por um grupo de pesquisadores voluntários, baseado na livraria do projeto SSLeay, um outro projeto, mais antigo e também de fonte aberta. O OpenSSL pode ser utilizado para a criação de sistemas para qualquer tipo de necessidade, comercial ou não.

O OpenSSL *toolkit* permite a implementação de aplicativos baseados no SSL versões 2 e 3, através de comandos modulares. As funções da lista de comandos podem ser utilizadas para a criação e manuseio dos diversos algoritmos criptográficos e mecanismos utilizados no SSL, como:

- Criação de parâmetros RSA, DH.
- Criação de certificados X.509, além de CRL´s.
- Calculo de resumos de mensagens e assinaturas.
- Encriptação e desencriptação de mensagens.
- Criação e testes de clientes e servidores SSL.

Abaixo estão alguns exemplos de comandos característicos do OpenSSL:

dgst – Cálculo da Message Digest

dsaparam – Criação de parâmetros DAS

enc - Codificação através de algoritmos criptográficos.

rand – Geração de pseudo-randômicos.

s_client – Implementação de um cliente SSL/TLS genérico que pode estabelecer uma conexão com um servidor remoto.

s_server – Implementa um servidor genérico SSLTLS.

O projeto OpenSSL oferece ainda uma grande quantidade de documentação referente à implementação do protocolo, bem como códigos fonte das funções que este suporta. Dessa

forma um programador experiente pode ter um controle maior sobre o código que deseja implementar, o que é a grande vantagem dos sistemas *open source*. Além disso, a comunidade envolvida com o assunto pode verificar e avaliar tais códigos, sugerindo modificações que visem melhorar a segurança dos sistemas desenvolvidos por ele.

Vale notar aqui que existe uma necessidade, por parte dos programadores, do conhecimento da teoria envolvida. Isto é um grande problema prático, pois a quantidade estudo necessário para que o programador sinta-se a vontade na discussão de aspectos práticos do protocolo é bastante grande, o que deve limitar bastante o número de pessoas capacitadas nesse sentido. Dessa forma, é sempre interessante que sejam feitas revisões de códigos em implementações do SSL, independentemente da ferramenta de programação utilizada, para que se verifiquem possíveis falhas na segurança, principalmente em aplicativos comerciais.

3.5 Recomendações para Implementação

Comentaremos sobre as recomendações de implementação descritas no documento do protocolo. Essas recomendações visam minimizar a possibilidade de erros de implementação gerados por desconhecimento de detalhes técnicos referentes aos sistemas que o protocolo utiliza para seu funcionamento. De acordo com o próprio protocolo, "o SSL não pode prever muitos tipos comuns de erros de segurança" [6], referindo-se à alguns detalhes de implementação que devem ser levados em consideração pelos programadores, sob pena do comprometimento de todo o esquema de segurança proposto pelo mesmo.

3.5.1 Utilização de Certificados com Chaves RSA de 512 bits

Devido às leis de exportação dos EUA, muitos *softwares* que utilizam o SSL possuiam uma restrição quanto ao tamanho da chave RSA utilizada para troca de mensagens de *handshake*, ou seja, para troca dos segredos criptográficos da sessão. Esta restrição limitava o tamanho da chave RSA em 512 bits, o que é considerado "relativamente inseguro" de acordo com o próprio protocolo. Dessa forma, o documento sugere que o servidor que utilize esse tipo de chave, a troque a cada dia ou à cada 500 operações, em sistemas comerciais, para garantir um nível de segurança adequado. Deve-se notar que no caso do servidor prover esta chave através de um certificado digital de sua posse, não há como realizar tal troca no período indicado sem a troca do próprio certificado. Assim sendo, os aplicativos desse servidor deveriam ser capazes de utilizar as chaves desse tamanho apenas para assinatura, ainda assim com um certo cuidado se levado em conta o fato de que teoricamente uma chave RSA pequena poderia ser quebrada em um tempo viável utilizando-se de computação distribuída. De acordo com o artigo " Cryptoanalysis: A Survey of Recent Results" [23], as recentes descobertas sobre algoritmos de fatoração, somadas às possibilidades de computação distribuída podem gerar mecanismos eficientes para a quebra de chaves RSA de tamanho pequeno. Existe inclusive um desafio da própria RSA Security para a quebra de chaves de 576 bits e 2048 bits. Caso um atacante pudesse, através desse tipo de técnica, quebrar uma chave dessa ordem em um tempo menor que um ano, que é o tempo de validade geralmente atribuído aos certificados digitais, este poderia utilizar a mesma para ataques, gerando um terrível transtorno. Atualmente a Verisign, empresa que mais certifica no mundo, atribui certificados com chaves de 1024 e 2048 bits, o que parece adequado para aplicações comerciais.

3.5.2 Geração de Números Randômicos

O SSL requer um gerador de números pseudo-randômicos seguro para seu trabalho. O protocolo somente faz alusão ao fato de que um gerador baseado no MD5 ou SHA-1 [6] seriam recomendáveis, mas é de responsabilidade do programador a escolha do algoritmo utilizado pela implementação. O fato é que este é um problema bastante relevante, pois devemos imaginar que o conhecimento necessário para a correta avaliação do nível de confiabilidade de um algoritmo de geração seguro estaria nas mãos de um matemático e não de um programador, e que se por algum motivo este algoritmo for escolhido de forma arbitrária (por questões de facilidade de implementação, por exemplo), teríamos aí uma brecha considerável na segurança. Além disso, de acordo com Schneier [24], não existem bons padrões para a geração de números randômicos computacionais, o que nos coloca em uma situação perigosa, uma vez que esses números são utilizados na criação de todas as chaves criptográficas necessárias ao SSL. O fato de um gerador de número randômicos gerar valores previsíveis pode facilitar bastante o processo de busca das chaves criptográficas, uma vez que restringe o universo de análise por diminuir o número de bits realmente aleatórios desta chave.

Podemos citar como exemplo um problema ocorrido em 1995 no navegador Netscape Navigator® [25], proveniente da forma que este navegador gerava seus números randômicos. A falha existiu por que o sistema de geração buscava uma série de valores relativamente fáceis de prever nos sistemas da máquina do usuário, como por exemplo o tempo presente no relógio em milissegundos. Dessa forma o atacante precisaria procurar, através de tentativas exaustivas, em um universo menor de chaves que o inicialmente previsto, pois o

conhecimento desses valores possibilitava a exclusão de uma série conjuntos de valores possíveis, e o ataque poderia ser realizado em apenas alguns minutos.

Tal fato nos mostra que a escolha arbitrária de valores randômicos pode gerar grandes problemas nos algoritmos criptográficos, comprometendo todo um sistema seguro. Dessa forma, podemos considerar a falta da definição formal de um algoritmo padrão para geração dos números pseudo-randômicos como uma fonte potencial de falhas de segurança no protocolo. A inclusão de uma instrução específica indicando algoritmos padrões para a geração dessas chaves no documento que descreve o SSL poderia, além de obrigar que as implementações utilizassem um sistema comprovadamente seguro, concentrar os esforços da comunidade mundial de criptógrafos na verificação constante de possibilidades de quebra do mesmo, garantindo constantemente sua confiabilidade.

3.5.3 Sessões Reassumidas

O SSL possibilita que uma sessão entre um cliente e um servidor possa ser reassumida em um tempo posterior variável, dependendo do tipo da implementação, sem a necessidade de um processo de *handshake* completo, ou seja, reaproveitando os parâmetros criptográficos da sessão anterior. Isto é interessante pelo fato de diminuir a necessidade de processamento do servidor. O cuidado maior neste caso é verificar um tempo de permanência não muito grande dos parâmetros da sessão no *cache* do computador servidor, para diminuir ao máximo a possibilidade de um ataque de replay. Um valor aceitável para a troca, de acordo com o protocolo, é de 24 h.

3.6 Considerações Finais sobre o Capítulo

O SSL, além de sua flexibilidade, é um protocolo bastante robusto em sua concepção, o que o torna atrativo para o desenvolvimento de sistemas de comunicação segura. A existência de diversas ferramentas para desenvolvimento, inclusive as de uso aberto, constituem-se em uma grande facilidade para os desenvolvedores, o que provavelmente foi um dos motivos para sua disseminação. Apesar disso, o protocolo é complexo em seu funcionamento e nas técnicas utilizadas, possuindo uma série de detalhes importantes que lhe são característicos e que podem confundir os programadores. Dessa forma é interessante notar que, apesar da robustez do protocolo, é possível que aplicativos que utilizem o SSL possam apresentar falhas de segurança provenientes de implementação, o que é confirmado pelos diversos incidentes de segurança relacionados ao protocolo, divulgados na literatura. A avaliação constante dos *softwares* que implementam o SSL é o mecanismo mais eficiente para a manutenção da segurança por ele provida.

Capítulo 4 - Análise de Segurança

4.1 Considerações Iniciais

O SSL tem como objetivo assegurar a transferência de informações de forma segura entre um cliente e um servidor, ou seja, seu campo de ação se restringe à segurança das mensagens em trânsito durante a comunicação, mais especificamente a partir da camada de transporte. Conforme Schneier [24], os possíveis ataques aplicativos que utilizam protocolos seguros poderiam ser divididos em alguns pontos fundamentais:

a) Ataques ao design do protocolo, baseados em falhas, erros de lógica, falta de definições mais rígidas sobre algoritmos e até mesmo definições com escrita dúbia nos documentos relacionados ao mesmo.

b) Ataques baseados em falhas de implementação devidas à erros de programação, falhas de avaliação do *software*, falta de conhecimento sobre o assunto ou mesmo códigos desenvolvidos por programadores relapsos.

c) Ataques aos criptosistemas, baseados em erros de lógica matemática ou lógica de construção, falhas de geração de randômicos, primos ou quaisquer outras classes de números especiais necessários.

d) Ataques baseados na falta de conhecimento de usuários, o que provavelmente é o maior ponto fraco de qualquer sistema de segurança atualmente, uma vez que uma grande parte destes podem possibilitar, através de ações inseguras, a quebra da segurança.

Serão realizadas a seguir as descrições de alguns detalhes encontrados no protocolo, que poderiam de alguma forma gerar situações de risco à segurança por ele proporcionada, tomando como referência os pontos anteriores.

4.2 Ataques aos Algoritmos de Criptografia

É provável que a primeira forma de ataque a um protocolo criptográfico seja através da quebra dos algoritmos criptográficos, principalmente do segredo de suas chaves. Cada vez que um novo algoritmo de criptografia é proposto, imediatamente é "bombardeado" por tentativas de ataques de todos os tipos, como forma de testar sua confiabilidade. Além disso, os pesquisadores procuram incessantemente novas formas de ataque, de forma que não podemos dizer que exista um algoritmo realmente seguro, ou inquebrável.

Comentaremos a seguir as citações encontradas na bibliografia sobre os protocolos criptográficos e traçaremos uma ligação com as possibilidades práticas de quebra de segurança do protocolo SSL atualmente.

4.2.1 Ataques ao Algoritmo DES – Data Encryption Standard.

Conforme citado na sessão 2.5.1, o DES já foi quebrado através de um ataque de força bruta que utilizava computação distribuída, chamado *DES Cracker*, em menos de 24 horas [8]. O sistema possuia uma taxa média de testes de $90x10^9$ chaves/segundo [10]. Dessa forma, um substituto para o DES já foi avaliado e padronizado, e deve ser implementado em todos os novos sistemas que utilizam o SSL. Este é chamado de AES - Advanced Encryption Algorithm [26]. O AES utiliza chaves de 160 ou 224 bits, o que o torna extremamente

robusto à ataques de força bruta. Como o SSL foi projetado para poder facilmente ser adequado a um novo tipo de algoritmo, não deverão haver maiores problemas para a substituição do DES pelo AES.

Como o DES foi quebrado devido ao tamanho da sua chave, em um tempo relativamente pequeno, seria de esperarmos que os sistemas deixassem de utilizá-lo. Porem podemos notar que uma grande parte dos sistemas comerciais atuais o utilizam normalmente. Mesmo trabalhando com a possibilidade de que um sistema de quebra do DES custe relativamente caro para a maioria das pessoas, não podemos esquecer de que existem empresas que poderiam estar interessadas em investir em um sistema desse tipo, para utilizá-lo contra seus concorrentes. Fica aqui o alerta para que pelo menos os servidores de sistemas mais críticos, como sistemas bancários por exemplo, desabilitem o uso do DES simples como garantia da segurança proveniente da utilização do mesmo.

Com relação à utilização do 3DES, esta ainda pode ser considerada como adequada tendo em vista que sua chave pode ser de 112 ou 164 bits. Com chaves de 112 bits, uma máquina como o DES Craker levaria um tempo médio de $1{,}8 \times 10^{15}$ anos para testar todas as possibilidades [10].

4.2.2 Ataque ao Algoritmo IDEA

O IDEA é conhecido publicamente desde 1991 e até o momento não foram encontradas indicações de inconsistências no mesmo, ou uma forma eficaz de atacá-lo que não fosse através de força bruta [28]. Como sua chave criptográfica possui 128 bits, a sua quebra através desta técnica ainda é muito improvável, o que deve garantir à este algoritmo um bom nível de segurança para as aplicações à que se propõe durante algum tempo.

4.2.3 Segurança do RC2 e RC4

Conforme comentado anteriormente, não existem publicações oficiais abertas ao público que descrevam o funcionamento destes dois algoritmos, por serem classificados como propriedade industrial. Desta forma qualquer análise sobre os mesmos torna-se inviável.

A utilização de algoritmos de código fechado é bastante criticada pela comunidade criptográfica mundial, como podemos ver no artigo de Schneier [24]. Conforme a opinião do autor, a utilização de algoritmos de código fechado como esses pode ser perigosa, pois a falta de testes oficiais realizados por organismos independentes pode deixar passar alguma fraqueza que os comprometeria. Se um atacante tivesse recursos para realizar a engenharia reversa dos códigos e vontade para estudá-los, poderia encontrar tais fraquezas e utilizá-las anonimamente. Dessa forma seria recomendável que os aplicativos desenvolvidos para utilizações comerciais que trabalhem com valores financeiros ou informações cujo sigilo fosse importante, evitassem sua utilização.

4.2.4 Segurança do Algorítmo Diffie – Hellman

O algoritmo de troca de chaves Diffe-Hellman é utilizado em duas formas diferentes no SSL, o algoritmo tradicional e o algoritmo efêmero EDH. O algoritmo EDH é susceptível á ataques do tipo *man-in-the-middle* [28], de forma que devemos utilizar uma forma de autenticação das mensagens enviadas para evitar esse tipo de ataque, o que é tradicionalmente feito em suas aplicações.

Com relação à segurança proporcionada pela sua chave, foi demonstrado que a quebra de seu segredo equivale à computar logaritmos discretos, o que ainda é inviável tecnicamente sob condições normais [28].

4.2.5 Segurança do Algoritmo de Chave Pública RSA

Segundo Brickell e Olkyzko [23], o algoritmo RSA é o melhor sistema de chave pública conhecido. Este algoritmo tem sua segurança baseada na dificuldade proveniente da fatoração de números extensos. Com a exceção de algumas situações particulares, não foram encontradas formas de quebrar o algoritmo, além da possibilidade de quebra direta da chave criptográfica. Conforme Coutinho [13], qualquer tentativa de obtenção da chave privada equivaleria a fatorar n, o que atualmente é inviável pela falta de algoritmos de fatoração eficientes.

Algumas situações específicas devem ser consideradas, principalmente com relação às chaves criptográficas RSA. Em primeiro lugar, existe a necessidade de geração de números primos grandes o suficiente, conforme citado inclusive na referência de implementação do algoritmo publicada pela RSA Security [19]. Chaves de 155 bits já foram quebradas em 99 através de computação distribuída, em aproximadamente 5,2 meses de trabalho [16]. Particularmente, em 1999, Adi Shamir apresentou um computador específico para a fatoração, chamado de TWINKLE [29], que utiliza os dois melhores sistemas de fatoração atuais, o Quadractive Sieve (QS) [30] e o Number Field Sieve (NFS) [31]. Este seria capaz de analisar 100 milhões de inteiros e determinar, em menos de 10 milissegundos, quais desses são fatoráveis sobre uma base dos primeiros 200 mil primos, o que tornaria viável a fatoração de primos na ordem de 650 bits em um tempo razoável. Atualmente o valor recomendado para o

tamanho da chave RSA é de 1024 bits, mas muitos sistemas já utilizam chaves de 2048 bits. Apesar disto, fica difícil considerar por quanto tempo estes valores ainda serão adequados, devido ao aumento da capacidade computacional e à possibilidade do surgimento de uma nova técnica de fatoração de primos [10].

Outro detalhe importante à considerar é a qualidade dos algoritmos para geração dos números primos utilizados, pois de acordo com Coutinho [13], não existe ainda um algoritmo capaz de testar, de forma garantida, a primalidade de dois números. Consideraremos a falta de citações sobre procedimentos formais para essa geração como uma deficiência no protocolo, uma vez que grande quantidade de estudo já existe à respeito. Uma definição neste sentido minimizaria a possibilidade de erros de implementação.

Algumas particularidades relacionadas aos valores p e q devem ser levadas em consideração durante a implementação dos algoritmos de geração desses valores. Conforme Coutinho [13], se a diferença entre os números for pequena, é possível através do algoritmo de Fermat a quebra da chave em poucas interações computacionais. Da mesma forma, é citado por Brickell e Olkyzko [23] que se $e<n$ e se $d<n^{1/4}$, então d pode ser facilmente determinado e utilizado para a fatoração de n. A RFC 2313 [19] não faz nenhuma menção à estes fatos.

Podemos então concluir que a informação sobre estas importantes características de implementação do RSA não é realizada de forma suficientemente eficiente, o que pode fornecer espaço para erros de implementação graves. Tais informações deveriam estar mais claramente descritas na documentação de forma a evitar esta possibilidade.

4.3 Falhas de Implementação do Protocolo

As falhas de implementação relacionadas ao protocolo são provavelmente uma das maiores potencialidades de quebra de segurança encontradas atualmente com relação ao SSL. Tais falhas são, na maioria das vezes, provenientes de uma avaliação mal executada dos parâmetros de segurança necessários ao perfeito funcionamento do protocolo.

Algumas falhas reportadas nos avisos do CERT (*Computer Emergency Response Team*) servirão como exemplo para demonstrarmos os problemas provenientes de uma implementação descuidada dos protocolos de segurança. As falhas descritas estão relacionadas ao processo de validação de certificados digitais durante a conexão segura. De acordo com o protocolo X509 [21], a validação dos certificados deve passar por três estágios:

1) Verificação do CA do certificado, ou seja, o *software* deve verificar se o certificado que recebeu foi atribuído por uma autoridade de certificação confiável, de acordo com a lista de CA´s confiáveis presente no computador do usuário.
2) Verificação de tempo de validade do certificado, de suas informações essenciais e verificação de revogação através da consulta de uma lista de revogação.
3) Verificação de compatibilidade entre o nome do servidor presente no certificado e o endereço presente na barra de endereços do navegador, para que se possa garantir que o servidor ao qual se está conectando é o mesmo para o qual o certificado foi atribuído.

4.3.1 Falha de Validação de Endereços nos Navegadores Netscape Comunicator e Internet Explorer

Este problema, ocorrido com os dois navegadores [32,33], é resultado da falha dos *softwares* ao considerar conexões iniciadas através de hiperlinks presentes nas páginas baixadas durante conexões seguras como sendo implicitamente seguras. Nestes casos específicos os *softwares* não realizavam as verificações dos endereços contidos na barra de endereços dos navegadores, referentes às páginas apontadas pelos hiperlinks, com os endereços dos certificados baixados destas páginas. Uma idéia básica de como o ataque funcionaria está mostrada a seguir:

a) Inicialmente o atacante utiliza alguma técnica para colocar um link, apontando para sua página, dentro da página do servidor pelo qual desejasse se passar. No caso de um site de compra, por exemplo, o atacante poderia colocar seu link dentro da figura que aponta para o formulário de compra, onde o usuário digitaria as informações de pagamento, como número do cartão de crédito.

b) Quando o cliente clicasse neste hiperlink, presente na página do servidor seguro, seria redirecionado para a página do atacante, recebendo deste um certificado digital pertencente a ele, e não ao servidor seguro original.

c) Como o *software* não realiza a verificação do endereço do certificado, a conexão continuaria normalmente, e o cliente passaria a se comunicar com o atacante. O cliente não perceberia nada, pois o mecanismo seria transparente para ele e o navegador não apresentaria nenhuma mensagem indicando que o servidor conectado não era confiável, conforme deveria.

Para que este ataque possa funcionar, o atacante deveria possuir um certificado válido em seu nome, pois quando a vítima se conectava ao seu servidor, o navegador realizava a verificação do certificado normalmente, exceto pela validação do endereço. De acordo com as empresas que responsáveis pelos navegadores, as correções necessárias já foram realizadas, bem como disponibilizados remendos (*patches*) para os *softwares* antigos que são passíveis desse tipo de ataque.

4.3.2 Falha de Validação On-Line de Certificados no Internet Explorer

O navegador Internet Explorer® possui uma opção onde podemos escolher se desejamos que a validade dos certificados recebidos no momento da conexão segura sejam avaliados nas listas de revogação on-line.

O fato divulgado pelo CERT [34] indicava que o navegador falhava ao realizar a verificação on-line da validade do certificado, deixando de exibir mensagens de erro para certificados revogados. Dessa forma, a verificação da lista de revogação era inútil, pois o usuário não recebia nenhuma mensagem de aviso caso o certificado não fosse válido.

4.3.3 Validação Automática de Certificados no Navegador Netscape Comunicator

Quando um usuário se conecta à um site que apresenta um certificado inválido por qualquer motivo, o Netscape Navigator® apresenta uma mensagem ao mesmo avisando sobre o problema e questionando se o mesmo deseja estabelecer a conexão apesar do problema [35]. No caso em questão, se o usuário não quisesse prosseguir, não havia problema pois a conexão

era desfeita, mas caso este aceitasse a conexão, o navegador passava a considerar automaticamente que qualquer outra conexão que apresentasse certificados inválidos deveria também ser aceita, e nenhuma mensagem de erro era apresentada ao usuário. Dessa forma, qualquer conexão segura efetuada pelo cliente posteriormente, enquanto o navegador não fosse reiniciado, seria considerada correta sem a verificação do certificado recebido, o que acabava com a possibilidade de garantia de identificação do servidor acessado.

4.4 Ataques a Arquitetura do Protocolo

Serão comentados a seguir alguns possíveis ataques que se utilizam de características intrínsecas do protocolo SSL ou de outros que o compõe. Desses, os mais potencialmente perigosos são baseados em interações dos protocolos em diversos níveis, ataques de difícil prevenção e alta complexidade.

4.4.1 Ataques de Interação de Protocolos

Os ataques baseados em interações de protocolos fazem uso de uma particularidade muito comum dos protocolos computacionais existentes atualmente, sua interoperabilidade. Os protocolos são desenvolvidos para poderem trabalhar uns com os outros de forma harmoniosa, além de utilizarem-se na maioria das vezes das mesmas técnicas em sua construção. Eles são desenvolvidos e testados individualmente, de forma que seus erros são normalmente corrigidos, mas no momento da implementação em sistemas computacionais complexos, o papel da interação entre diferentes protocolos gera situações de erros não originalmente previstas. No artigo "*Protocol Interactions and the Chosen Protocol Attack*"

[36], são descritos alguns possíveis ataques teóricos baseados nessa técnica, partindo do princípio que segurança não necessariamente se soma (*security does not necessarily compose*).

Um ataque prático perigoso, baseado na interação de protocolos, foi descoberto por um pesquisador da Bell Labs. Em seu artigo [37], o pesquisador descreve que as chaves privadas de mecanismos que faziam uso do RSA baseadas no PKCS#1, podiam ser quebradas se uma série de mensagens cuidadosamente escolhidas fossem enviadas ao servidor durante a conexão, e se as respostas desses erros fossem analisadas adequadamente. O ataque é dinâmico e adaptativo e somente funciona em protocolos que possuam uma grande troca de mensagens durante o *handshake*, como é o caso do SSL.

O ataque é implementado enviando-se mensagens construídas através de uma mensagem inicial criptografada, interceptada durante uma conexão qualquer. Apesar de não poder recuperar a mensagem original, o atacante pode tentar criar uma mensagem criptografada aleatória e enviá-la ao receptor. As mensagens do PKCS#1 têm um formato particular que pode ser reproduzido para este ataque. Para uma mensagem **M**, antes da criptografia, o formato padrão do PKCS#1 será o seguinte [19]:

$M = 00 \,||\, BT \,||\, PS \,||\, 00 \,||D$, onde o primeiro valor, 00, indica a versão do protocolo,

BT indica o tipo de operação, que no caso de criptografia será 02,

PS é um valor pseudorandômico de *padding* para fazer o bloco ***M*** ter o comprimento igual ao comprimento do módulo ***n***.

00 é um valor colocado para garantir que o bloco de encriptação tenha um valor numérico menor que o módulo.

D é o bloco de informações a serem criptografadas.

Conforme podemos ver, a mensagem sempre iniciará com os valores 00 e 02, no caso de uma troca de chaves. Para que o atacante possa recuperar a chave privada do servidor, ele tentará criar mensagens que pareçam ter sido criptografadas corretamente com a chave pública do mesmo. Tais mensagens seriam do tipo

$$Ci = C\, r_i^e \bmod n,$$

onde ***Ci*** é a mensagem falsa, ***C*** é uma mensagem criptografada com a chave pública do servidor, r_i são valores quaisquer entre 1 e ***n***-1 e ***(n,e)*** são os valores da chave pública do receptor.

De posse dessas mensagens, o atacante faz repetidas tentativas, onde ele as envia ao servidor para que este as desencripte e verifique. Se o servidor não for capaz de desencriptar a mensagem, ele simplesmente vai abandoná-la, mas caso a mensagem tenha sido bem construída, o servidor realizará a desencriptação da mensagem falsa, verificará seu conteúdo, e como não será capaz de entendê-la, retornará ao atacante uma mensagem de erro. Neste momento o atacante será capaz de retirar dessas respostas informações que o auxiliarão a descobrir qual é a chave privada do servidor, utilizada na sessão, através da diminuição do universo de busca.

Para que tal ataque seja efetivo, o servidor deve responder à um número considerável de mensagens falsas, na ordem de milhões, pois o ataque , na busca da chave, realiza aproximações sucessivas utilizando informações estatísticas retiradas da análise. Apesar deste número parecer elevado, tal ataque é muito mais eficiente que uma busca exaustiva da chave criptográfica, sendo portanto viável na prática. Para evitar tal tipo de ataque poderíamos empreender certas ações simples. Primeiramente, os servidores deveriam ser configurados de forma a não responder a um grande número de mensagens incorretas em um tempo pequeno. Isso seria eficiente para evitar qualquer ataque deste tipo, e também seria facilmente

implementado. Em segundo lugar, a chave pública do servidor, utilizada na criptografia, deveria ser mudada com freqüência, não dando tempo para que um atacante pudesse quebrá-la. O ideal seria que cada sessão fosse realizada com uma chave pública diferente, gerada no momento da conexão e descartada no fim do *handshake*. O SSL permite esse tipo de ação através do uso de certificados para assinatura apenas, onde as chaves públicas para criptografia são geradas e transmitidas no *handshake* entre este e o cliente.

4.4.2 Ataque de Algorítmos Null

Um tipo de ataque simples, mas eficiente pode ser implementado contra o SSL utilizando-se de uma particularidade do estágio de *handshake*. No início da conexão, o cliente envia ao servidor uma mensagem chamada "**client hello**" , onde estão descritas uma série de possíveis suítes criptográficas que o servidor poderá escolher para realizar a conexão. Esta mensagem inicial não possui nenhum tipo de proteção, pois nesse momento nenhum algoritmo criptográfico está ativo [6].

O protocolo define como possíveis uma série de suítes criptográficas, com o seguinte conjunto de parâmetros; Protocolo, Algoritmo de Troca de Chaves, Algoritmo de Criptografia e Algoritmo de Assinatura. Entre as possíveis configurações das suítes criptográficas estão três que são extremamente perigosas em seu uso, pois não fornecem proteção criptogáfica à comunicação. Estas suítes foram incluídas no protocolo para dar ao mesmo mais flexibilidade de utilização. Este é o caso, por exemplo, quando desejamos estabelecer uma conexão sem proteção criptográfica mas com garantia de conteúdo de mensagens, o que pode ser gerada através da utilização de uma suíte que somente realize a assinatura digital. Essas suítes são as seguintes [6]:

SSL_NULL_whith_NULL_NULL – (Sem criptografia, sem assinatura digital)

SSL_RSA_whith_NULL_MD5 – (Sem criptografia, com assinatura MD5)

SSL_RSA_whith_NULL_SHA - (Sem criptografia, com assinatura SHA)

Como essas suítes não oferecem nenhuma confidencialidade à comunicação, qualquer conexão que se utilizar das mesmas poderá ser abertamente examinada. Dessa forma seria esperado que qualquer servidor repudiasse conexões com esse tipo de características, salvo nos casos em que esta configuração fosse desejada por motivos específicos.

Existe porém a possibilidade de que este fato não seja verdadeiro, ou seja, que o servidor esteja configurado para aceitar tais conexões. Apesar do mesmo dever escolher necessariamente a suite mais segura proposta pelo cliente, um atacante poderia tentar enganá-lo para que este iniciasse uma sessão através dessas suítes inseguras. Isto poderia ser realizado da seguinte forma.

1) O atacante intercepta a mensagem **Client_Hello** enviada ao servidor, substituindo-a por outra que indica que o cliente pode trabalhar apenas com uma das três anteriormente citadas.
2) O servidor responde normalmente à mensagem, iniciando uma conexão com o cliente utilizando-se da suíte não segura, pois ele irá pensar que esta é a única que cliente está capacitado a usar.
3) O cliente continua o processo de *handshake* normalmente, imaginando que a conexão está sendo realizada da forma esperada.
4) O cliente e o servidor passam a comunicar-se e trocar suas mensagens confidenciais, sem nenhuma proteção criptográfica. Neste momento o atacante pode verificar o conteúdo das mensagens livremente.

Este ataque pode ser evitado simplesmente através de uma configuração correta do servidor, onde deverão ser desabilitadas as suítes com característica insegura, para que este repudie este tipo de conexão.

4.4.3 Falta de Padrões para Geração de Números Randômicos

A falta de definição de um mecanismo específico para a geração dos números randômicos utilizados no protocolo é outro ponto importante a ser analisado. Conforme citado anteriormente, muitos problemas de segurança relacionados ao protocolo foram criados por falhas na geração desses números. Devemos lembrar que a melhor forma de criação de sementes randômicas em sistemas computacionais comuns é um problema ainda em discussão. Apesar disso, vários mecanismos de geração de pseudorandômicos vêm sendo propostos e analisados. A escolha de alguns destes algoritmos para a formação de uma suíte de geradores pseudorandômicos seria uma iniciativa interessante para o momento, pois criaria um nível de confiabilidade maior para o protocolo, uma vez tais algoritmos seriam constantemente testados e avaliados pelo meio científico.

A utilização das funções de *hash*, como o MD5 e o SHA-1 para a criação de funções de geração de pseudorandômicos é uma alternativa interessante, adotada pelo padrão TLS – Tranport Layer Security. As duas funções de *hash* são capazes de gerar números pseudo-aleatórios criptograficamente fortes desde que sua entrada seja uma semente aleatória de tamanho adequado. Chamamos de números pseudo-aleatórios criptograficamente fortes aos números gerados através de uma função ou algoritmo que seja capaz de gerar valores distintos com baixa correlação estatística, a partir de uma semente qualquer. O motivo de serem chamados de pseudo-aleatórios é que podemos reproduzi-los desde que possuamos a semente correta, o que não aconteceria a um gerador realmente aleatório.

As sementes randômicas são valores binários necessários para servirem de ponto de partida para a geração de pseudorandômicos em qualquer função geradora. A geração das sementes randômicas é um problema relativamente complexo, pois o trabalho consiste em "encontrar" um número relativamente grande de bits aleatórios dentro do sistema computacional. Para que os bits sejam aleatórios, devemos implementar um gerador capaz de gerar os bits 0 e 1 ambos com uma probabilidade de 50%. Apesar de parecer uma tarefa simples, dentro de uma máquina computacional os sistemas são geralmente previsíveis pela forma que foram construídos, ou seja, é difícil encontrarmos fontes adequadas para estes bits. Uma maneira possível para isto é a utilização de sistemas onde a entrada seja o movimento aleatório do mouse, criado pelo usuário. Apesar disto, a implementação desse sistema deve ser cuidadosa.

O SSL deveria possuir em sua documentação a referência a um algoritmo de geração de números aleatórios fortes, de forma a diminuir a possibilidade de falhas neste sentido. Como isto não é realizado atualmente, o implementador deve ser cuidadoso na avaliação da previsibilidade dos valores gerados pelo seu sistema, sob pena de programar um sistema inseguro que pode ser facilmente quebrado por um criptoanalista habilidoso.

4.5 Manipulação de Certificados Digitais

Os certificados digitais são, atualmente, a única ferramenta efetiva para identificação de um servidor na Internet. Através destes certificados podemos confiar que estamos realmente conectados à uma empresa ou entidade à qual acreditamos, e assim realizar com segurança nossa transação. Dessa forma, a manipulação correta dos certificados digitais é de suma importância na implementação de um processo de comunicação seguro via Internet.

Atualmente a responsabilidade de decisão sobre algumas situações relacionadas à manipulação destes certificados têm sido colocada a cargo do usuário, geralmente leigo, o que vem gerando situações de risco para a segurança dos sistemas, principalmente comerciais.

O SSL utiliza como modelo para certificados digitais o padrão X.509. O padrão [20] descreve formalmente os formatos dos campos de um certificado digital, bem como os algoritmos utilizados para resumo de mensagens e a geração e verificação das assinaturas. A assinatura é realizada gerando-se um *hash* sobre os valores do campo tbsCertificate, que depois será transformado em assinatura através do RSA Signature Algorithm ou do DSS, garantindo que ninguém possa alterar os campos de identificação do certificado sem invalidar a mesma. Dessa forma, um usuário pode ter certeza de que estará se comunicando com quem realmente deseja, se realizar uma correta verificação do certificado recebido.

O padrão X509 define o processo de validação dos certificados, conforme citado anteriormente, verificando três itens [21]:

1) Verificação do CA do certificado, ou seja, o *software* deve verificar se o certificado que recebeu foi atribuído por uma autoridade de certificação confiável, de acordo com a lista de CA´s confiáveis presente no computador do usuário.
2) Verificação de tempo de validade do certificado e de uma possível revogação do mesmo através da consulta de uma lista de revogação.
3) Verificação de compatibilidade entre o nome do servidor presente no certificado e o endereço presente na barra de endereços do navegador, para que se possa garantir que o servidor ao qual está se conectando é o mesmo que possui o referido certificado.

O padrão X.509 não define um roteiro formal para esta validação, delimitando apenas de forma geral o processo. Assim, muitos detalhes técnicos de implementação ficam a cargo dos desenvolvedores, que podem não considerá-los de forma correta. Um exemplo deste tipo de problema é a consulta às listas de revogação de certificados, que apesar de encorajada, é colocada como opcional para o processo de validação. Não existe um algoritmo formal que defina a forma com que tal verificação ocorrerá, o período de tempo mínimo entre as atualizações das listas de revogação, ou mesmo as ações tomadas pelo usuário da conexão quando da verificação da validade de um certificado. Tais ações são chamadas genericamente de "políticas de certificação digital", e a discussão à seu respeito ainda possui muitos pontos que deverão ser esclarecidos. Dessa forma, cada empresa possui uma forma diferente de tratar o problema, o que vem gerando os sistemas mais diversificados.

No caso específico do SSL, o certificado digital do servidor garante o cumprimento de um dos três pilares fundamentais propostos pelo protocolo, a autenticação. Dessa forma, o rompimento da segurança dos certificados quebra automaticamente qualquer nível de segurança proposto pelos sistemas que utilizam o SSL, pois uma vez que não podemos identificar o outro ponto da conexão, estamos sujeitos a ataques de identidade falsa. Dessa forma, estudaremos a seguir a política adotada pelo protocolo para lidar com a validação destes certificados, verificando sua efetividade tanto com relação à documentação como com relação à alguns exemplos de *softwares* que implementam esta mesma política.

4.5.1 Verificação do Período de Validade do Certificado

O próximo ponto crítico da tecnologia dos certificados digitais provém do período de validade do mesmo. Este período de validade é claramente definido no corpo do certificado, e

é verificado através da comparação simples com o relógio da máquina cliente. O fato é que, na prática e principalmente nos computadores pessoais, tal relógio nem sempre está correto, o que geralmente invalida certificados válidos e pode validar certificados expirados. No caso no Internet Explorer® e do Netscape Navigator®, quando um certificado é verificado e sua validade não é confirmada, ambos apresentam uma mensagem ao usuário avisando sobre o fato e perguntando se a conexão deve ou não ser aceita. Duas situações são possíveis neste caso:

1) Um usuário desconfiado poderia ser levado a não aceitar uma conexão correta. Dessa forma este usuário pode deixar de realizar suas tarefas através do mecanismo seguro, gerando uma perda econômica e um marketing negativo para a empresa com a qual ele tentou se conectar.

2) O usuário pode aceitar a conexão mesmo com a mensagem de falta de validade apresentada. Isto já representa uma possibilidade gigantesca de quebra da segurança, pois a veracidade da identificação do servidor proveniente deste certificado se perderá, o que abre todas as possibilidades para ataques de identidade falsa.

Em ambos os casos grandes perdas podem ser impostas às empresas cujos certificados estão sendo colocados em dúvida, por causa de um erro de configuração da máquina do cliente, o que provavelmente passará desapercebido.

Tal falha pode parecer simples para a comunidade acostumada com os termos de segurança computacional, mas devemos reiterar que o SSL é um protocolo que têm em sua essência a possibilidade de aplicação em um grande número de situações, que podem envolver usuários leigos, mas que mesmo assim devem ser protegidos.

4.5.2 Política de Adição de Autoridades Certificadoras Confiáveis

Um problema grave gerado pelo esquema de certificação digital atual é derivado de um dos fundamentos desta tecnologia, a necessidade de uma entidade certificadora confiável. Atualmente os *softwares* realizam uma verificação em uma lista própria de entidades certificadoras para concluir se o certificado recebido em uma conexão foi atribuído por uma CA confiável. O problema surge justamente neste ponto, como podemos definir um CA confiável? Se analisarmos o modelo presente nos *softwares* navegadores atualmente, podemos dizer que um CA confiável é aquele que possui um registro no arquivo de configurações de certificados deste navegador, onde o mesmo realiza a verificação de entidades às quais ele está apto a aceitar como confiáveis.

Para implementar um ataque baseado neste tipo de modelo, um atacante pode tentar instalar, nos arquivos de configurações de certificados da vítima, uma entidade certificadora falsa, por ele criada, e a partir daí gerar seus próprios certificados digitais que passariam como certificados válidos pela máquina da vítima em questão. Para isso, o atacante poderia utilizar o processo de instalação automática presente nos navegadores. Este processo é disparado quando um certificado atribuído por uma entidade certificadora desconhecida gera uma mensagem de alerta ao usuário e questiona se o mesmo deseja que esta entidade seja adicionada à lista do navegador. As telas do IE referente à esta ação são apresentada na figura 4.3.

Caso o atacante consiga convencer a vítima a realizar esta instalação, estará apto a realizar seus ataques contra a mesma, pois este poderá gerar qualquer certificado necessário para este fim, sendo que estes serão validados na conexão.

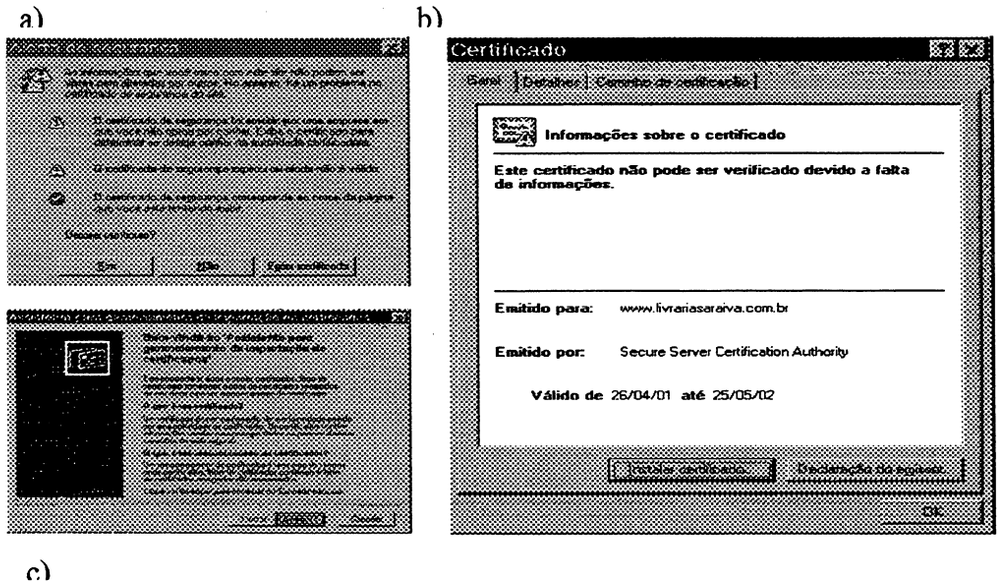


FIGURA 4.1. (a), (b), e (c) - Mecanismo de Auto Instalação de Certificados do IE.

Vejamos um exemplo interessante. A figura 4.4. mostra a mensagem de alerta apresentada pelo IE ao receber um certificado do SIAPE – Sistema Integrado de Administração de Recursos Humanos, do Governo Federal. Este site utiliza um certificado atribuído por um CA chamado "Autoridade Certificadora do SERPRO – Serviço de Processamento de Dados do Governo Federal", que não é uma entidade certificadora padrão presente na lista do navegador. Dessa forma, ao acessar o site através de uma conexão segura, o navegador alerta o usuário, que pode pedir para apresentar o certificado, conforme figura 4. 5. O procedimento padrão do navegador neste caso é verificar com o usuário se este deseja instalar o certificado, conforme descrito anteriormente. A figura 4.6. mostra o resultado desta instalação. Em todo caso, para que o usuário possa acessar o site de maneira segura, este deve decidir se o CA descrito é confiável ou não.

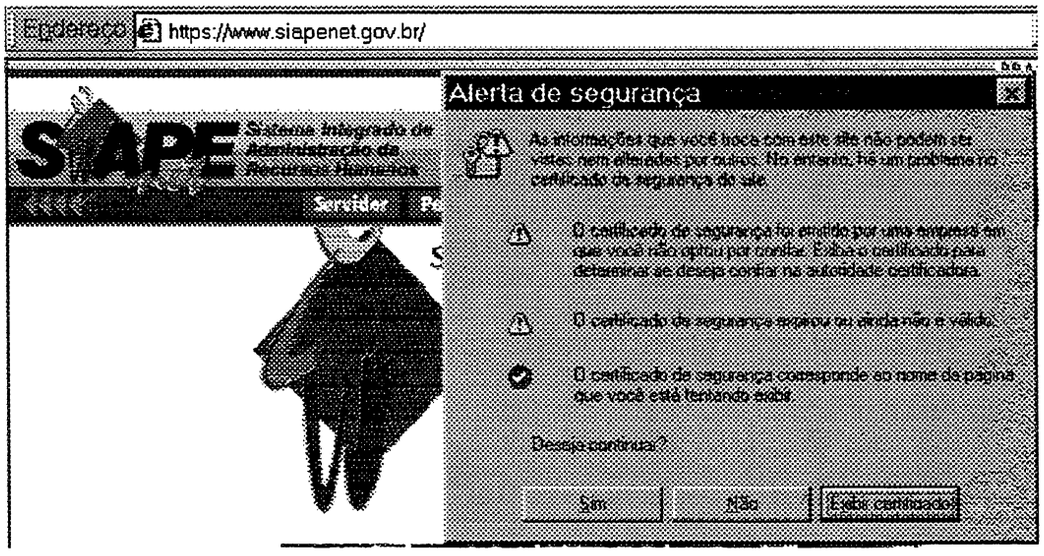


FIGURA 4.2 - Alerta de Segurança do IE ao Receber Certificado do SIAPE

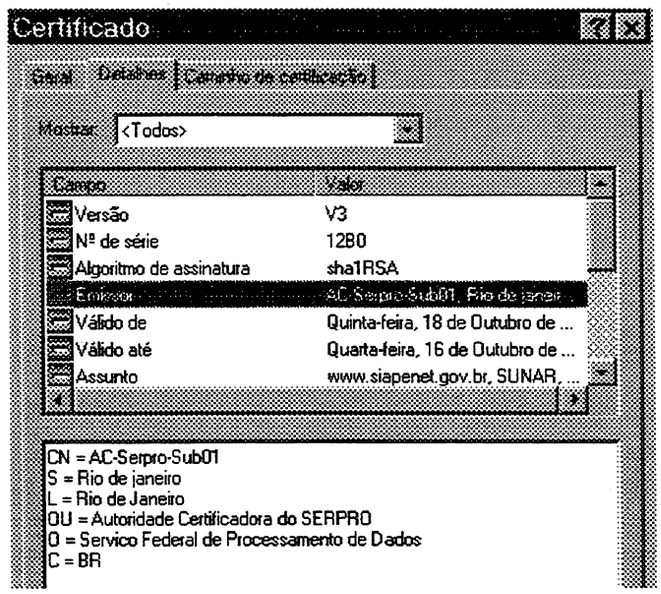


FIGURA 4.3 - Detalhes do Certificado do SIAPE

Outra possibilidade de ataque, baseada neste tipo de problema, é a utilização de um navegador cuja procedência não seja totalmente confiável. Um atacante pode, por exemplo, alterar a lista de certificados digitais presente na versão de instalação de um navegador como o IE, colocando nesta um CA a ele pertencente, e a partir daí realizar a distribuição deste via

Web, através de seu site, por exemplo. Neste caso, todos os usuários que fizerem uso desta cópia de instalação do *software* estarão vulneráveis.

Capítulo 5 - Resultados da Análise e Discussão

Por ser um sistema complexo, formado por diversas tecnologias, e por não possuir uma especificação formal tão rígida, o SSL está sujeito à detalhes técnicos que podem possibilitar falhas de segurança em situações específicas durante sua utilização. Essa especificação flexível do protocolo foi possivelmente, um dos motivos para sua grande disseminação nos sistemas computacionais comuns. Comentaremos em seguida os pontos importantes levantados na análise de segurança.

5.1 Considerações sobre Algoritmos Criptográficos Utilizados no Protocolo

Os perigos provenientes da utilização de algoritmos criptográficos "quebráveis" não são necessariamente uma imperfeição do protocolo, uma vez que a evolução das técnicas criptográficas vão sempre gerar novas formas de ataque aos algoritmos existentes. Um grande ponto positivo do protocolo é justamente sua flexibilidade no momento da inclusão ou substituição destes algoritmos, o que garante que este sempre esteja trabalhando com o que há de mais moderno nesta área.

Excetuando-se situações específicas, a análise da bibliografia demonstrou que o protocolo faz uso de algoritmos criptográficos robustos o suficiente para a maioria das aplicações comerciais atuais. Dessa forma, com relação a este item, e salvo os detalhes anteriores citados na análise de segurança, podemos afirmar que o protocolo cumpre

eficientemente sua proposta de garantia da privacidade através da utilização destes algoritmos na forma proposta pelo mesmo.

5.2 Considerações sobre Falhas de Implementação

A maioria dos problemas de segurança relacionados ao protocolo foram provenientes de implementações mal realizadas. Isto provavelmente se deve ao fato de que tais protocolos possuírem um nível de complexidade razoavelmente grande, o que torna a correta avaliação de todas as suas necessidades e particularidades, no momento da implementação, algo não muito simples. Dessa forma, um implementador sem conhecimento adequado do protocolo pode, através de uma análise ruim, gerar um grande problema ao criar um *software* que em princípio deveria ser confiável, mas na realidade não o é.

A solução provável para este problema deve ser a avaliação constante dos produtos presentes no mercado, realizada pelo maior número de pessoas com conhecimento adequado, de preferência de forma independente. Isto, infelizmente, pode não ser viável na maioria dos casos, por causa dos segredos comerciais envolvidos nestes produtos, o que impossibilita a abertura dos códigos para análise. Dessa forma, talvez o problema somente possa ser minimizado quando realmente ocorrer. Assim sendo, é muito importante que todos os cuidados sejam tomados pelos usuários ao utilizar um novo produto, e ao confiar nele, pois esta é provavelmente a única arma efetiva no momento para combater este tipo de problema.

Apesar da gravidade destes problemas, devemos notar que o protocolo não teve sua confiabilidade afetada, uma vez que esses problemas não são decorrentes de falhas do protocolo, mas sim dos erros dos *softwares* que o utilizam.

5.3 Conclusões Sobre o Sistema de Manipulação de Certificados Digitais

Apesar da utilização de certificados digitais para identificação *on-line* ser eficiente tecnicamente, o fato do processo ainda depender da influência do fator humano em algumas situações específicas pode tornar o sistema implementado vulnerável à alguns tipos de ataques. As implementações desses sistemas deveriam ser extremamente cuidadosas ao delegar responsabilidades aos seus usuários, tendo em vista que não podemos garantir que estes possuirão o conhecimento técnico necessário para um correto discernimento das questões de segurança a eles apresentadas.

Sob o ponto de vista da segurança proposta pelo protocolo, deveríamos esperar que este definisse de forma rígida os pontos necessários ao processo de validação de um certificado, bem como as ações tomadas nos casos em que estes não forem considerados válidos. A consulta às listas de revogação, realizadas *on-line* a cada conexão, e a atualização diária destas listas nos servidores deveriam ser obrigatórias, ou no mínimo fortemente recomendadas. Isto não acontece na prática, e dessa forma surgem falhas que podem ser aproveitadas para algum tipo de ataque.

Exceto por isso, a tecnologia de certificados digitais se apresenta como uma forte ferramenta para utilização em processos de verificação de identidades *on-line*, principalmente pela sua simplicidade e flexibilidade. Dessa forma, podemos considerar que o protocolo SSL cumpre eficientemente com seu objetivo primário de prover identificação aos participantes da conexão, pelo uso dessa tecnologia, desde que cuidados adequados no momento da implementação sejam tomados.

5.4 Considerações Sobre os Ataques a Arquitetura do SSL

Os ataques a arquitetura do protocolo são possivelmente o maior desafio à segurança do SSL, uma vez que são extremamente difíceis de serem identificados. Tais ataques são baseados em detalhes técnicos de alto grau de complexidade, de forma que sua ocorrência é bastante pequena em relação aos outros problemas encontrados. Infelizmente, de acordo com a filosofia deste tipo de ataque, a única maneira de preveni-los é encontrando-os através de um estudo profundo e contínuo. Parece também possível afirmarmos que um protocolo nunca poderá garantir total segurança em seu esquema, pois mesmo que tal protocolo não possua erros comuns, o mesmo não pode ser afirmado com falhas não previstas na sua concepção inicial. Uma vez que o protocolo é desenvolvido por humanos, é perfeitamente normal que o mesmo seja falível, e que estas falhas sejam encontradas um dia. Apesar disto, os métodos de ataque descritos até agora são pontuais, ou seja, somente afetam o protocolo em situações específicas. Além disso esses problemas são relativamente simples de resolver, bastando para isso alguma pequena alteração no algoritmo dos *softwares* vulneráveis. Dessa forma, o SSL pode ser considerado seguro com relação aos ataques relatados até o momento, no caso das aplicações comerciais, onde o interesse em sua aplicação é maior.

Concluindo, ataques do tipo interação de protocolos são perigosos e pouco previsíveis, principalmente pela complexidade e diversidade dos sistemas computacionais existentes atualmente. Para que tais tipos de ataque sejam evitados, é necessário um estudo constante das possibilidades, e uma busca das correções necessárias, em todos os níveis envolvidos.

Capítulo 6 - Sugestões para as Implementações do SSL

Serão sugeridas a seguir algumas ações que poderiam ajudar a evitar algumas das possíveis quebras de segurança do protocolo, baseadas nos problemas descritos anteriormente, além de garantir uma utilização mais eficiente do mesmo.

6.1 Especificação de uma Política de Manipulação de Certificados Digitais Adequada

Uma política de Manipulação de certificados digitais mais adequada seria suficiente para evitar uma série de incidentes de segurança. A manipulação, verificação e aceitação de certificados digitais deve ser deixada a cargo do *software*, evitando colocar sobre os ombros dos usuários a responsabilidades de decisões técnicas. Algumas medidas tomadas neste sentido durante a implementação do protocolo seriam interessantes, como por exemplo:

1) Verificação de todos os parâmetros dos certificados digitais a cada nova conexão, seja ela requisitada a outro servidor ou ao mesmo. Esta ação evitaria ataques baseados em erros de implementação como os descritos anteriormente, ocorridos nos navegadores comuns. Uma indicação mais direta e detalhada sobre este ponto seria suficiente par colocar fim a estes problemas.
2) Rejeição incondicional de certificados com parâmetros incorretos, com conseqüente cancelamento da conexão. Para isso seria suficiente que os sistemas passassem a considerar as mensagens de erro de certificados como fatais, o que garantiria o fechamento automático da conexão sempre que um erro de certificado

ocorresse. As mensagens de erro de certificado, de acordo com a seção 3.3.3 são as seguintes: *bad_certificate* - Certificado danificado, *unsupported_certificate* - Certificado não suportado, *certificate_revoked* - Certificado revogado, *certificate_expired* - Certificado expirado, *certificate_unknown* - Certificado não conhecido [6].

Está claro que a adoção de uma política como esta poderia gerar um certo transtorno em alguns casos, como por exemplo quando um usuário estiver com o relógio de sua máquina com a data incorreta, fato que causaria a invalidação das conexões SSL, até que a data deste relógio fosse acertada. Apesar disto, o fato de evitarmos conexões inseguras tem suficiente precedência neste caso, de forma que o usuário em questão, apesar de ter que corrigir as configurações de sua máquina para resolver seu problema, estaria seguro. Em último caso, poderíamos sugerir que os navegadores colocassem a opção de não aceitação de certificados com parâmetros incorretos como *default* no seus navegadores, deixando a possibilidade de modificação desta condição dentro das suas possíveis configurações padrões.

Caso estes itens fossem adotados, representariam um avanço considerável no tratamento dado aos sistema de identificação do protocolo, garantindo uma grande redução nos possíveis ataques que se utilizam desse artifício.

6.2. Servidor de Data/Hora

A verificação dos certificados digitais é baseada em uma série de parâmetros necessários ao seu funcionamento, conforme as descrições anteriores. Um desses parâmetros é o tempo no relógio da máquina do usuário que recebe o certificado, que é utilizado para a verificação de período de validade deste. O fato é que nem sempre podemos garantir que um

cliente esteja com o relógio de sua máquina correto, o que gerará uma mensagem de certificado expirado ou não válido.

Tal problema poderia ser resolvido através da criação de um sistema de verificação de datas nos certificados digitais. Tal sistema poderia ser implementado juntamente com listas de revogação automáticas, que seriam acessadas no início da conexão. Atualmente existem listas deste tipo implementadas através de um protocolo chamado OCRP (On-line Revocation Certification Protocol) [38], esquema utilizado pelo Netscape Navigator®. Este protocolo poderia ser alterado ou mesmo complementado por um protocolo que definissem os parâmetros para implementação do sistema Data/Hora.

Atualmente, a verificação dos certificados é realizada somente através do seu número, que somente estará presente nesta lista quando houver algum problema que cause sua revogação. Este sistema não leva em conta o período de validade do certificado verificado.

No caso em questão, a proposta é que fosse implementado juntamente com um sistema de verificação de validade on-line automático, um sistema auxiliar que verificasse o período de validade do certificado recebido. Assim, mesmo que a data presente na máquina do usuário estivesse incorreta, um certificado expirado não teria como se passar como válido, e vice-versa, resolvendo o problema de verificação incorreta de datas de certificados recebidos.

Para que tal mecanismo fosse eficaz, a verificação automática de certificados recebidos em conexões via SSL deveria ser obrigatória, e realizada à cada nova conexão. Apesar de parecer que tal mecanismo geraria um acréscimo de tempo indesejável na conexão segura, devido ao aumento de passos necessários para validar uma conexão, vale a pena notar que a utilização de conexões protegidas é rara, sendo portanto aceitável tal acréscimo de tempo necessário garantir a segurança do usuário.

6.3 Criação de uma Documentação Auxiliar para a Avaliação Correta dos Parâmetros de Segurança nas Implementações de Aplicativos Baseados no SSL

Outro mecanismo interessante para reduzir o número de erros de implementação e possibilidade de falhas nos sistemas que utilizam o SSL seria a criação de uma documentação de referência para implementações do protocolo. Tal documentação poderia indicar um conjunto de regras adequadas à implementação em questão, conforme o nível de complexidade e segurança do sistema fossem crescendo. Podemos exemplificar o fato utilizando um sistema de comércio eletrônico que realiza transações de crédito pela Internet, cujo sistema deveria possuir regras muito rígidas para a validação de certificados, senhas maiores e uma política de gerenciamento das conexões eficiente, e onde todas as considerações de segurança possíveis deveriam ser realizadas durante seu desenvolvimento

Uma proposta que fica aqui registrada é a criação de tais documentos oficiais de orientação às implementações, que definam os parâmetros de segurança aplicáveis a cada tipo de situação ou sistema, a serem seguidos durante o desenvolvimento de *softwares* comerciais. Tais parâmetros poderiam passar por uma ampla discussão, e uma vez que estivessem claros, poderiam garantir uma efetividade maior para a segurança oferecida pelo protocolo nas situações específicas de seu uso. Além disso, o fato da divisão de níveis de segurança para o mesmo garantiria a possibilidade de desenvolvimento de aplicativos adequados para cada tipo de utilização, melhorando sua *desempenho*.

Capítulo 7 – Conclusões

O SSL é um protocolo muito valioso para a criação de sistemas de comunicação segura para uso geral. Sua flexibilidade provê um valioso ponto que garante que este possa ser utilizado de forma eficiente em diversos tipos de sistemas, principalmente relacionados ao comércio eletrônico. Provavelmente é este o motivo que levou o SSL a se tornar tão popular dentro dos mecanismos que implementam este tipo de comércio.

Com relação à segurança, pode-se comprovar que o protocolo atende adequadamente os pontos necessários à criação de sistemas robustos e eficientes. No estudo não foram encontradas formas efetivas de atacá-lo, exceto em situações específicas de implementações mal realizadas, o que garante que o mesmo pode ser utilizado sem receio na maioria dos casos. Apesar disso, torna-se claro que o desenvolvimento das regras que compõe tais sistemas é um ponto crítico na segurança dos mesmos, e que o protocolo não pode garantir a segurança por ele propostos se tais regras não forem corretamente implementadas. Este é o caso, por exemplo, dos sistemas de certificação digital, onde o esquema proposto nem sempre cumpre efetivamente as regras descritas na documentação, o que pode gerar um meio de ataque ao sistema. Deve-se lembrar que o cumprimento destes pontos são de responsabilidade dos desenvolvedores, e que o estudo profundo do protocolo é um elemento de suma importância para este fim.

De forma geral, pode-se concluir que o protocolo SSL é uma importante e eficiente ferramenta para o desenvolvimento de sistemas de comunicação computacional seguros, e que sua segurança deve permanecer adequada para tal utilização, principalmente se revisões e verificações constantes destes aspectos forem realizadas, o que garantirá sua efetividade no cumprimento dos seus objetivos por um longo tempo.

Como propostas para trabalhos futuros, pode-se sugerir o estudo aprofundado e individual das vulnerabilidades citadas no trabalho, especialmente de ataques de algoritmo *null*, análises relacionadas ao sistema de certificação digital atualmente em utilização, análise comparativas entre *softwares* comerciais que se utilizam do SSL, nos aspectos de segurança e *desempenho*, verificação de *desempenho* das suítes criptográficas utilizadas pelo protocolo, desenvolvimento de documentações específicas para apoio a implementações de sistemas baseados no SSL e também implementação de sistemas específicos que se utilizem do protocolo, como sistemas de distribuição de informação empresarial ou afins.

GLOSSÁRIO

3DES - DES Triplo, ou seja, a aplicação do DES três vezes sobre a mesma mensagem.

AES – Advanced Encription Algorithm

ANSI - American National Standards Institute

CA – Certification Autority

CERT – *Computer Emergency Response Team*

CGI – Common Gateway Interface

Chave criptográfica – Valor, geralmente numérico, utilizado para a geração do texto criptografado.

Cifragem – Aplicação de um algoritmo criptografico em um texto.

Cliente – Máquina iniciante da comunicação.

Criptosistema – Sistema de criptografia.

DAS - Digital Signature Algorithm

Decifragem – Processo inverso à cifragem.

DES *Data Encryption Standard*

Desencriptação – O mesmo que decifragem.

Diffie – Hellman - Algoritmo de chave pública específico para a troca de chaves simétricas.

DNS – Domain Name Sistem

DSS - Digital Signature Standard

EDH – Ephemeral Diffie-Hellman

Encriptação – O mesmo que cifragem.

FIPS PUBS - Federal Information Processing Standard Publications

FTP – File Transfer Protocol

Hash – Resumo numérico de mensagem, utilizado para verificação de alterações de conteúdo da mesma.

Help – Arquivo disponibilizado pelos *softwares* para prover informações para seus usuários.

HTTP – Hyper Text Transfer Protocol

IDEA *International Data Encryption Algorithm*

IE – Internet Protocol

Interação – Aplicação de um determinado processo em um conjunto de valores durante uma vez.

KDE – K Desktop Environment

Link/Hiperlink – Disparador de comunicação colocado geralmente em texto ou figura para facilitar o acesso do usuário de um *software* a um arquivo ou a um ponto específico de um arquivo.

MAC – Message Autentication Code

Man-in-the-Middle – Ataque baseado na interceptação e substituição de mensagens em trânsito.

MD5 - *Message Digest 5*

NIST - National Institute of Standards and Tecnologies

NSA - National Security Agency

Opção Default – Parâmetro previamente definido em um *software*, mas que geralmente permite modificação pelo usuário.

Open Source – Sistemas de código fonte aberto para análise.

Patches – "Remendos". Programas desenvolvidos para correção de problemas ocorridos em *softwares* comerciais.

Quebra da Chave Criptográfica – Ato de encontrar a chave utilizada na criptografia, sem o conhecimento prévio da mesma.

Quebra de Segurança – Ato de burlar um método ou sistema de segurança através de formas escusas.

Randômico – Valor que não pode ser previsto por função matemática.

RC2 – Algoritmo de criptografia de chave privada desenvolvido pela RSA Data Security.

RC4– Algoritmo de criptografia de chave privada desenvolvido pela RSA Data Security .

RC5– Algoritmo de criptografia de chave privada desenvolvido pela RSA Data Security.

RFC – Request for Comments

RSA – Rivest, Shamir & Adleman

Semente Randômica – Valor de bits randômicos usados para a geração de chaves criptograficas.

Servidor – Máquina receptora de pedidos de comunicação.

Sessão criptográfica – Comunicação com o conjunto de valores necessários ao funcionamento do sistema criptográfico previamente definidos.

SHA-1 - *Secure Hash Algorithm –1*

Sistema Criptograficamente Forte - Sistema que obedece às regras de segurança de construção de algoritmos de criptografia.

SMTP – Simple Mail Transfer Protocol

TCP – Transport Control Protocol

TLS – Transport Layer Secure

VPN – Virtual Private Network

Referência Bibliográfica

[1] MÓDULO E-SECURITY " Pesquisa: Orçamento dedicado à segurança será maior em 2002" [on-line] MÓDULO E-SECURITY mai, 2002, visitado em 05/2002 Disponível: http://www.modulo.com/index.jsp

[2] RUSSEL, D. & GANGEMI, G.T. **"Computer Security Bases"** 1 ed USA: O'Reilly & Associates, Inc, 1991.

[3] GARFINKEL, S. & SPAFFORD,G. **"Comércio & Segurança na Web"** 1 ed. São Paulo: Market Press, 1999.

[4] LUCCHESI, C. L. **"Introdução à Criptografia Computacional"** 1 ed. Campinas: Unicamp, 1986.

[5] MENEZES, A.; VAN OORSCHOT, P.; VANSTONE, S. **"Handbook of Applied Cryptography"** [on line]. CRC Press, 1997 visitado em 11/2000. Disponível: http://cacr.math.waterloo.ca/hac/about

[6] NETSCAPE COMMUNICATIONS CORPORATION **"SSL version 3.0"** [on line]. Internet Engineering Task Force, mar, 1996, visitado em 03/2000 Disponínel: http://home.netscape.com/eng/ssl3/ssl-toc.html

[7] TANENBAUM, A. S. **"Redes de Computadores"** 3 ed. Rio de Janeiro: Campus, 1997.

[8] RABELLO, L. **"O fim inevitável do DES""** [on-line] Mólulo e-Security, jan 1999, visitado em 06/2001. Disponível: http://www.modulo.com/co.../docs

[9] NIST NATIONAL INSTITUTE OF STANDARDS AND TECNOLOGIES "**AES - The Early Years (1997-98)"** [on-line] visitado em 06/2000 Disponível: http://csrc.nist.gov/encryption/aes/pre-round1/earlyaes.htm

[10] TERADA, R. **" Segurança de Dados Criptografia em Redes de Computador"** 1 ed São Paulo: Editora Edgard Blücher LTDA, 2000.

[11] RSA SECURITY **"What is the RC2 Cryptosistem?"** [on line] RSA Security Labs, visitado em 08/2000. Disponível: http://rsasecurity.com/rsalabs/faq/3-1-3.html

[12] RSA SECURITY **"What is the RC4 Cryptosistem?"** [on line] RSA Security Labs, visitado em 08/2000. Disponível: http://rsasecurity.com/rsalabs/faq/3-1-4.html

[13] COUTINHO, S.C. "**Números Inteiros e Criptografia RSA"** Rio de Janeiro: IMPA/SBM, 1997

[14] RSA SECURITY **"What is the RSA Cryptosistem?"** [on line] RSA Security Labs, visitado em 08/2000. Disponível: http://rsasecurity.com/rsalabs/faq/3-1-1.html

[15] RSA SECURITY **"What would it take to break the RSA cryptosistem?"** [on line] RSA Security Labs, visitado em 08/2001. Disponível: http://rsasecurity.com/rsalabs/faq/3-1-3.html

[16] RSA SECURITY "**Factorization of RSA-155"** [on-line] RSA Security Labs, visitado em 09/2001 Disponível: http://www.rsasecurity.com/rsalabs/challenges/factoring/rsa155.html

[17] COMPUTER SYSTEMS LABORATORY **"FIPS PUB 186-2 –Digital Signature Standard (DSS)"** [on line] U.S. DEPARTMENT OF COMMERCE/NATIONAL INSTITUTE OF STANDARDS AND TECNOLOGY, jan, 2000, visitado em 06/2000 Disponível: http://csrc.nist.gov/publication/fips/fips186-2/fips186-2.pdf

[18] RIVEST, R. **"RFC 1321 –The MD5 Message-Digest Algorithm"** [on line] Massachusets: MIT Laboratory for Computers Science, abr, 1992, visitado em 06/2000 Disponível: http://www.ietf.rnp.br/ftp/rfc/rfc1321.txt

[19] KALISKI, B. **"RFC 2313 PKCS#1: RSA Encryption Version 1.5"** [on-line] The Internet Society, 1998., visitado em 05/2001 Disponível: www.ietf.org/rfc/rfc2313.txt

[20] COMPUTER SYSTEMS LABORATORY **"FIPS PUB 180-1 – Secure Hash Standard"** [on line]. U.S. DEPARTMENT OF COMMERCE/NATIONAL INSTITUTE OF STANDARDS AND TECNOLOGY, abr, 1995, visitado em 06/2000 Disponível: http://csrc.nist.gov/cryptval/shs.html

[21] Housley, et. al. **"RFC: 2459 - Internet X.509 Public Key Infrastructure Certificate and CRL Profile"** [on-line] Network Working Group, Jan 1999 visitado em 07/2001 Disponível: www.ietf.org/rfc/rfc**2459**.txt

[22] **"The OpenSSL Project"**, página da Web, visitada em 01/2002 Disponível: http://www.openssl.org/

[23] BRICKELL, E. F. ODLYZKO, A. M **"Cryptanalysis: A survey of recent results",** pp. 501-540 in Contemporary Cryptology, G. J. Simmons (ed.), IEEE Press (1991). Preliminary version in Proc. IEEE 76, 1988, pp. 578-593.

[24] SCHNEIER, B. **"Cryptographic Design Vulnerabilities",** IEEE Computer, v. 31, n. 9, Sep 1998, pp. 29-33.

[25] "**Security Flaw Is Discovered In Software Used in Shopping By John Markoff San Francisco, Sept. 95"** [on-line] The New York Times, September 19, 1995, pp. A1, D21, visitado em 08/2001 Disponível: http://www.demailly.com/~dl/netscapesec/nyt.txt

[26] DAEMEN, J. RIJMEN, V. "**AES Proposal: Rijndael"** [on-line] set. 1999, visitado em 06/2001 Disponível: http://csrc.nist.gov/encryption/aes

[27] CARVALHO, D. B. **"Segurança de Dados com Criptografia"** 2 ed. Rio de Janeiro: Book Express 2001.

[28] RSA SECURITY **"What is Diffie-Hellman"** [on line] RSA Security Labs, visitado em 08/2001 Disponível: http://rsasecurity.com/rsalabs/faq/3-6-1.html

[29] SHAMIR, A. **" Factoring large numbers with the TWINKLE device"** Proceedings of EUROCRYPT'99, Lectures Notes in Computer Science, Springer – Verlag, 1999

[30] POMERANCE, C. **" The Quadratic Sieve factoring algorithm"** Eurocrypt 84, Lecture Notes im Computer Science, number 209 pp 168-182, 1985.

[31] LENSTRA, K. LENSTRA, H,W. MANASSE, M.S. POLLARD, J.M. **" The number field sieve"** Lecture Notes im Mathematics, vol. 1554 pp 11-12 Springer – Verlag, 1993

[32] COMPUTER EMERGENCY RESPONSE TEAM **"CERT Advisory CA-2000-10 Inconsistent Warning Messages in Internet Explorer"** [on-line] CERT® Coordination Center 2001, visitado em 10/2001 Disponível: http://www.cert.org/advisories/CA-2000-10.html

[33] COMPUTER EMERGENCY RESPONSE TEAM **"CERT Advisory CA-2000-08 Inconsistent Warning Messages Netscape Navigator"** [on-line] CERT® Coordination Center 2001, visitado em 10/2001 Disponível: http://www.cert.org/advisories/CA-2000-08.html

[34] COMPUTER EMERGENCY RESPONSE TEAM **"Vulnerability Note VU#399087 Internet Explorer incorrectly validates certificates when CRL checking is enabled"** [on-line] CERT® Coordination Center 2001, visitado em 10/2001 Disponível: http://www.cert.org

[35] COMPUTER EMERGENCY RESPONSE TEAM **"Vulnerability Note VU#37526 Netscape fails to revalidate certificates if a user has previously acknowledged a certificate to be non-matching Overview"** [on-line] CERT® Coordination Center 2001, visitado em 10/2001 Disponível: http://www.cert.org

[36] KELSEY, J. SCHNEIER, B. WAGNER, D. **"Protocol Interactions and the Chosen Protocol Attack",** Security Protocols, 5th International Workshop April 1997 Proceedings , Springer-Verlag, 1998.

[37] BLEICHENBACHER, D. **"Chosen Ciphertext Attacks against Protocols Based on RSA Encryption Standard PKCS #1",** *Advances in Cryptology -- CRYPTO'98,* LNCS vol. 1462, pages: 1-12, 1998.

[38] COMPUTER SYSTEMS LABORATORY **"FIPS PUB 46-2 – Data Encryption Standard (DES)"** [on line]. U.S. DEPARTMENT OF COMMERCE/National Institute of Standards and Tecnology, dez 93, visitado em 01/2000. Disponível: http://csrc.nist.gov/publication/fips/fips46-2

[21] Myers, et. al. **"RFC: 2560 – On-Line Certificate Status Protocol - OCSP"** [on-line] PKI Working Group, dez 2001 visitado em 05/2002 Disponível: www.ietf.org/rfc/rfc**2560**.txt